INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

High-level Synthesis and Its Application in the Design of Reed-Solomon Decoders

Shadia Hijazie

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial fulfillment of the Requirements
for the degree of Masters of Applied Science at
Concordia University
Montreal, Quebec, Canada

July 1997

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

0-612-40212-6

Canada

ABSTRACT

High-level Synthesis and Its Application in the Design of Reed-Solomon Decoders

Shadia Hijazie

Most of the improvements in the performance of digital systems have been brought about by recent advanced in VLSI technology. High-level Synthesis (HLS) is a major part of VLSI design process. Its techniques can be used to yield an optimal result in the design. Yet, major applications of HLS methods are on digital filters or DSP circuits. Arithmetic in these circuits are real, integer or complex. The major theme of this thesis is to show that HLS methodologies can be applied in the design of arithmetic circuits dealing with finite field elements, such as Galois Field GF(2^m).

A case study for such a design is versatile time-domain Reed-Solomon RS(n,k) decoders. The structure of the time-domain RS decoder is simple and modular which makes it fit for VLSI implementation. The first part of the thesis studies HLS and introduces a Computer-Aided-Design (CAD) tool which can be used for designing an arithmetic based circuitry such as digital filters and algebraic codes. The RS(n,k) decoding algorithm, dealing with GF(2^m) [7], is modified by applying some HLS techniques as transformation, clustering and loop unfolding. The transformed algorithm is then coded in VHDL and synthesized using Synopsys tools. The area and speed of RS(n,k) decoders are estimated and simulated for different values of m. The results proved to yield an efficient, fast with maximal throughput design.

Dedicated to My Parents

ACKNOWLEDGEMENT

I would like to thank God for giving me the ability and strength to accomplish this work and to be the way I am. I also would like to express my gratitude and appreciation to Dr. Youssef Shayan for his guidance and patience for directing the thesis process in every aspect. Special thanks to Dr. Asim Al-Khalili for his support and encouragement in submitting my work. I would like to thank Dr. Baher Haroun for his supervision in my early phases of the thesis.

I would like to express my gratitude and love to my parents who gave the best they could to raise me up and educate me. Special thanks to my kind husband and best friend, Ehsan, for his love and moral support in writing my thesis.

Finally, I would like to thank my friends and the staff in the VLSI LAb, especially Amal Khailtash, for their help and the friendly environment they provided.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

α	The Primitive Element in Galois Field 2^m
$\underline{c}$	Code Word Vector
$\underline{e}$	Error Vector
$g(x)$	Generator Polynomial of Reed-Solomon Codes
k	Number of Information Symbols in a code word
$\underline{\lambda}$	Error Locator Vector
m	Size of each symbol in bits
n	Number of Symbols in Reed-Solomon Code Word
t	Error Correction Capability of the Code
r	Iteration number
ρ	Number of Erasures in Code Word
$\underline{v}$	Received Code Word Vector
ALAP	As Late As Possible
ASAP	As Soon As Possible
ASIC	Application Specific Integrated Circuit
BCH	Bose, Chauduri and Hocquenghem
BiCMOS	Bipolar -Complementary Metal Oxide Semiconductor
CAD	Computer-Aided Design
CDFG	Control and Data Flow Graph
CMOS	Complementary Metal Oxide Semiconductor
FU	Functional Unit
GF	Galois Field
HLS	High-Level Synthesis
ILP	Integer Linear Programming

OSTA	Optimal Synthesis Tool for Architecture
RS	Reed-Solomon
RTL	Register-Transfer Level
VHDL	Very High Hardware Design Language
VLSI	Very Large Scale Integration
XOR	Exclusive OR Gate

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The influence of integrated-circuits in the past few years has been significant, in areas ranging from consumer products to manufacturing control. The driving force behind this is that the designers of modern integrated circuitry have continually attempted to provide more computational speed with less dissipated electrical power and less circuit board area, while maintaining a low failure rate and a minimal cost.

1.1 VLSI Implementation Techniques:

The tendency to use VLSI implementation techniques can be aroused from many reasons. The first reason is that the number of high volume product application for logic hardware has not grown at the same pace as the technological capability. The second reason is that the expense and the quantity of resource required to verify the design prior to manufacturing and testing the manufactured product have increased tremendously. Another reason is that the complexity of current designs is becoming considerably more difficult to manage conceptually among the design group. The design group must have the tools available that will permit the total integrated circuit functions to be partitioned for manageability and then verified collectively.

There are several rules that are established in the use of VLSI as an implementation medium[25]. One of them is that the correctness of the design is of paramount importance. Debugging a flawed chip design is both difficult and time consuming. The second rule is the degree of flexibility in the design which should be incredibly high. The designers can specify the system organization, the partitioning and even the details driving logic and gate transistors. Despite this flexibility, there are limitations in the ability of one level of

the design to compensate for shortcomings at higher levels. These limitations come from both inherent constraints in the technology as well as the need to limit the addition of new complexity at lower levels of the design.

It is important to understand how VLSI design are implemented. One can implement a design using full custom gate arrays, macro-cell arrays, or standard cells. Gate arrays and standard cells are generally called application-specific integrated circuits (ASIC).

The ASIC approach has new capabilities and benefits with the advent of high performance technology. These are figured in the following[27]:

- higher-speed circuits are achievable by the use of advanced CMOS/BiCMOS technology.
- higher levels of integration and complexity are possible with the availability of very high number of gates.
- architectural changes, coupling memory to on-chip CPU elements are possible nowadays.
- power dissipation is reduced and off-chip electrical performance improved by the use of lower power supply voltages.
- smaller chip sizes and shorter interconnection length are possible.
- interconnections are shortened by using multi-chip modules.

Obviously, the key goal implementation is to provide the fastest hardware for the architecture; this translates into two rules[26]:

(1) Minimize the clock cycle of the system which implies organizing the hardware to minimize the delays in each clock cycle.

(2) Minimize the number of cycles needed to perform each instruction.

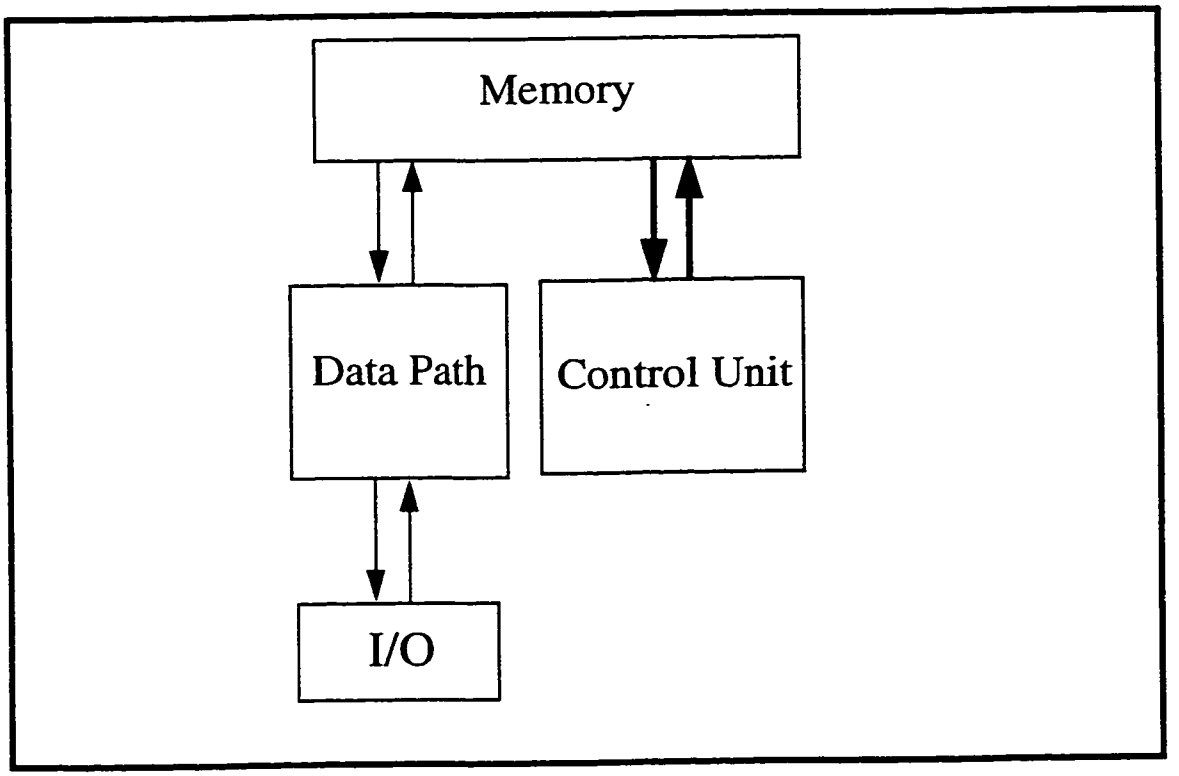


Fig. 1.1: Organization of a system design

The hardware design is divided into several parts, as shown in Fig.1.1:

a) Data path unit that shows the flow of data through the whole cycles.

b) I/O unit that contains the inputs and the outputs.

c) Control Unit that controls the data flow according to the schedule. This is implemented by a finite state machine.

d) Memory and Busses that contain the data for a specific period of time.

The interaction between the processor architecture and its organization has always had a profound influence on the cost - performance ratios attainable for an architecture[27].

The approach of optimizing the design is to go to the high-level synthesis where the specification of the algorithm is translated into architectural primitives in such a way that

the resulting implementation optimizes a certain function: the area, the speed or the throughput.

1.2 The Approach for high Level Synthesis (HLS)

Recently, automatic data-path synthesis of a digital system from a behavioral description has gained much attention in the Computer-Aided-Design (CAD) research community. If more of the design process is automated, the design time is shorter and the cost is minimized significantly. A higher level of abstraction reduces the number of objects that a designer needs to consider, which in return, allows the design and manufacturing of larger systems in shorter periods of time. High-level Synthesis (HLS) thus is the natural step in the design methodology of VLSI systems.

Another reason for the emphasis on high-level design methodologies is that high-level abstractions are closer to a designer's way of thinking. It is difficult to imagine a designer specifying, documenting and communicating a chip design in terms of circuit schematic with hundreds of thousands of gates. With increasing design complexity, it becomes impossible for a designer to comprehend the functionality of a chip or a system completely specified with circuit or logic schematics.

A major characteristic for high-level synthesis is that it provides a shorter design cycle. HLS also provides fewer errors since there will be a great assurance that the final design will correspond to the initial specification. Also, a good synthesis system can produce several design options from the same global specification in a reasonable amount of time[9]. This allows the developer to explore different trades-off between cost, speed, power etc., or to take an existing design and produce functionally an equivalent one that is faster or less expensive.

The synthesis task starts with a behavioral description of a digital system and a set of

time and/or resource constraints. The goal is to produce a structure of the digital system that satisfies the constraints. It includes four subtasks[2]. The first subtask is to describe the behavior of the digital system using a hardware description language (HDL). This step is usually followed by a translation of the description into a graph-based representation called Control-Data-Flow-Graph (CDFG). The next subtask is operation scheduling, where each operation in the CDFG is assigned to a control step. The third subtask allocates the resource for the digital system. Here, functional units are assigned to execute the operations, storage units are assigned to store the values, and wires are allocated to interconnect them using the data transfer information derived from the CDFG. At this point, a data path is completed. Finally, based on the schedule graph and the data path, a control unit is synthesized to synchronize the executions of the operations.

The benefits of the HLS approach in the design process are tremendous, yet it has only been applied to DSP applications. There are other types of data being manipulated other than floating-point and integer arithmetic, e.g. algebraic codes, specifically linear block codes. These types of codes has special characteristics. The operations applied on algebraic codes, such as addition or multiplication, are not the same as the one applied on integers or real numbers.

1.3 Linear Block Codes:

In the recent years, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. This demand has been accelerated by the emergence of large-scale, high speed data networks for the exchange, processing and storage of digital information in the military, governmental and private spheres. A merging of communications and computer technology is required in the design of these systems. One major concern of the designer is the control of errors so that reliable reproduction of data

can be obtained.

The theory of error detecting and correcting codes is a branch of engineering that deals with reliable transmission and storage of data. Information media are not 100% reliable in practice, in the sense that noise frequently causes some data to be distorted. To deal with this undesirable situation, some form of redundancy is incorporated in the original data. With this redundancy, even if errors are introduced, the original information can be recovered, or at least the presence of errors can be detected.

Any communication system is basically composed of the following components: source, encoder, modulator, channel, receiver, demodulator and decoder.The channel induces noise that corrupts the information passing through it. The transmitting channels include telephone lines, high frequency radio links, telemetry links, microwave links, and satellite links. Storage media include core and semiconductors memories, magnetic tapes and so on. Each of these examples is subject to various types of noise disturbances. On a telephone line, the disturbance may come from thermal noise, cross-talk from other lines or lightening. On a magnetic tape, surface defects are regarded as noise disturbances.

The decoder corrects the received sequence. The decoding strategy is based on the rules with which the information was encoded. Unfortunately, most channels are not entirely reliable and there is a certain probability that the decoder receives a symbol different from what was sent. Hence error control codes have taken a prominent position in obtaining reliable digital communication systems.

A block code consists of a set of fixed-length vectors called *code words*. The length of a code word is the number of elements in the vector and is denoted by n. The elements of a code are selected from an alphabet of q elements. When the alphabet consists of two elements, 0 and 1, the code is a binary code and the elements of any code word are called bits. When the elements of a code word are selected from an alphabet having q elements ($q>2$),

the code is nonbinary. It is worth noticing that when q is a power of 2, i.e. $q=2^b$ where b is a positive integer, each q-ary element has an equivalent binary representation consisting of b bits and, thus, a nonbinary code of block length n can be mapped into a binary code of block length $N= bn$.

It is important to observe that the encoding and the decoding functions involve arithmetic operations of addition and multiplication performed on code words. These arithmetic operations are performed according to the conventions of the algebraic field which has as its elements the symbols contained in the alphabet as will be discussed later in detail.

First block codes, introduced by Hamming, were a class of single - error- correcting block codes. Hamming codes were weak compared with far stronger codes promised by Shannon. The major advance in block codes was made by Bose - and -Ray Chaudhuri and Hocquenghem who introduced BCH codes which is a large class of multiple - error correcting codes. Reed and Solomon then found a related class of codes for bursty channels.

The discovery of BCH codes led to a search for practical methods for implementing the encoder and the decoder. Basically the choice of the code depends on the characteristics of the channel, for example, for burst error, non - binary BCH codes are suitable. A subclass of non-binary BCH codes are Reed Solomon RS codes which have a very large burst error correcting capability and correct a large number of random errors[3]. RS codes are used in many applications such as in satellite, spread spectrum and mobile communications due to their error correcting capabilities and optimum structure.

RS codes provide a prominent place in the theory and practice of error correction for three reasons:

1. They are good codes provided that the block length is not excessive. In long codes, the complexit of the design increases significantly.

2. Encoding and decoding techniques are simple and instrumentally known.

3. RS codes have certain optimal properties and well understood distance structure.

An RS code is a block of symbols where each symbol is an element of Galois Field 2^m ($GF(2^m)$). Galois Field elements are finite field elements. Each symbol is represented in m-bits. There are 2^m different possible symbols in a code word.

The set of 2^k codewords of length n is called an (n, k) code block with $n-k$ redundant symbols which are added to each message to form a codeword. An RS (n, k) code has the following properties:

$n = 2^m - 1$: length of codewords in symbols.

m: number of bits/symbol.

t: maximum number of error symbols that can be corrected.

d: number of check symbols $= 2t$.

$k = n - 2t$: number of information symbols.

The symbols of an RS (n, k) code are taken from finite field elements of $GF(2^m)$ Each pair of distinct n-symbol codewords differs in at least $2t + 1$ symbols. Thus an RS code has a minimum distance of $2t + 1$ and can correct t - symbol errors. Erroneous symbols of a received word confined to a region of t symbols or less are correctable. A received word with any combination of t or fewer symbols in error will be correctly decoded.

The hardware implementation of RS decoders are complex and time consuming, resulting in a large high power consumption and unreliable design that could be a disadvantage in space communications.

1.4 Contributions and Contents of the thesis:

HLS is becoming more important in practical design environments to meet the new

system requirements. A Computer-Aided-Design tool that applies the methods of HLS is implemented showing the advantages of such techniques. As a case study for adopting HLS techniques, the versatile time-domain RS(n,k) decoder is taken. The RS(n,k) decoding algorithm, developed in [7], is changed to meet the methods of HLS. The modified algorithm is coded in VHDL and synthesized to optimize its speed. The decoder that employs GF(2^m) is simulated and its speed and area are estimated for different values of m.

In this thesis, Chapter 2 deals in details with the techniques of High-level Synthesis. This chapter discusses the distinct and inter-dependent subtasks taken in HLS: scheduling and binding, Control-Data-Flow-Graphs, transformation and clustering. A CAD tool is developed adopting some of the HLS subtasks. This tool uses the Integer Linear Programming (ILP) formulation of some subtasks to the second order elliptic filter and atomizes the formulation to be used in the design of circuit having the basic operations: addition, subtraction, multiplication and delay.

Chapter 3 studies the characteristic of Galois Field elements and their operations. It is shown that the HLS techniques could be applied to circuits that operates on algebraic codes as well as real numbers or integers. Our case study in this thesis is the versatile time domain Reed-Solomon decoders. The algorithm of the versatile RS decoder is analyzed and modified after applying the methods of HLS discussed in the previous chapter.

Chapter 4 demonstrates the details of the synthesis of the decoding algorithm The complexities of the algorithm were resolved using the following synthesis subtasks: clustering, retiming and loop unfolding. The complexities were due to the presence of decision making in the middle of the loop and multiple choices of data operations in each iteration.

Chapter 5 deals with the coding and synthesis. The design is coded using VHDL, an abstract modelling language that can describe the temporal behavior and structure of a

system from the overall block diagram level down to the gate level. The synthesis of the design was performed using Synopsys emphasizing speed as a constraint for the synthesis. The decoding algorithm uses GF(2^m) elements as data. The area and speed of the decoder are simulated and estimated for different values of m. It will be shown that the HLS approach in the design process reveals interesting results and that HLS methods could be applicable for optimizing designs of digital circuits that perform algebraic operations as well as DSP filters.

CHAPTER 2

High-level Synthesis

Computer-Aided-Design (CAD) industry has been very successful and has grown exceptionally in parallel with the advances in IC fabrication. It started to deal with problems like circuit simulation, placement, routing and floor planning and then logic simulation and synthesis. The use of architecture synthesis tools to automate the search for an optimal VLSI architecture from a graph or language description of a digital system have been a considerable emphasis in the past few year. Usually there are many different structures that can be used to realize a given behavior. One of the synthesis tasks is to find the structure that best meets the constraints, such as limitations on cycle time, area, or power, while minimizing other costs[16].

There are several reasons for the fact that logic synthesis is gaining acceptance in industry, which are[9]:

Shorter design cycle: If more of the design process is automated, the design can be completed faster. Much of the cost of the chip is in design development, and hence automating the process can lower the cost significantly.

Fewer errors: If the synthesis process can be verified to be correct, there is a greater assurance that the final design corresponds to the initial specification. This results in fewer errors and less debugging time for new chips.

The ability to search the design space: A good synthesis system can produce several designs from the same specification in a reasonable amount of time. This can result in exploring different trade-off between speed, cost, power etc.

Availability of IC technology to more people: As more design expertise is moved into the synthesis system, it is easier for a non expert to produce a chip that meets a given

set of specification.

The synthesis task is to take the specification of the required behavior of a system and a set of constraints and goals to be satisfied, and to find a structure that implements the behavior while satisfying the goals and constraints. By behavior, we mean the way the system or its components interact with their environment[6]. Structure refers to the set of interconnected components that make up the system, typically described by netlist.

2.1 High-level Synthesis (HLS)

High-level synthesis means going from an algorithmic level specifications of the behavior of a digital system to a register-transfer level structure that implements that behavior[9]. An RTL design is specified by modules, storage units, and their interconnections, and can be used by a standard physical design automation tool such as silicon compiler to produce a chip layout.

The synthesis system produces a description of *data path*, that is, a network of registers, functional units, multiplexers and buses. If the control unit is not integrated into the data path, and usually is not, the synthesis system must also produce the specification of the control unit part. The system to be designed is usually represented at the algorithmic level by a hardware description language that is similar to a programming language. Most high-level synthesis approaches have used *procedural* languages. That is, they describe data manipulation in terms of assignments to variables that keep their values until they are overwritten.

The first step in high-level synthesis is usually the compilation of the formal language into an internal representation[15]. Then several high-level synthesis tasks such as scheduling, unit selection, functional storage and interconnection binding, and control generation are performed.

The goal of high level synthesis process for real time applications is to minimize the implementation cost, while still satisfying all timing constraints. Yet the resource utilization may not obviously get balanced over time. The resource utilization is defined as [7]

$$resource\ utilization = \frac{number\ of\ cycles\ a\ resource\ exploited}{number\ of\ available\ cycles}$$

2.1.1 Control-Flow and Data-Flow Graph

Most approaches use graph-based representations that contain both the data flow and the control flow, Control-Data-Flow-Graph, (CDFG) as implied by the specification. This graph captures sequencing, conditional branching and looping constructs in the behavioral description. The data flow graph shows the essential ordering of operations implied by the data dependencies in the specification.

In the data flow scheme, we use circles to denote operations, arcs to denote data-flow, rectangles to denote reading or writing of data, and inverted triangles to denote selection of data based on the value of the control line. Like control-flow representation, the data-flow representation also explicitly shows concurrence due to mutual exclusion of different conditional paths.

However, since data-flow representation evaluates all branches of a conditional test in parallel, it makes larger segment of graph available for refinement and optimization than the control-flow representation. Hence, the data-flow representation is better suited for the task of scheduling straight-line code than control-flow representation [16]. The timing information represented in CDFG is used as constraints for scheduling, unit-selection and unit-binding as well as performance of design implementation.

2.1.2 Scheduling and Allocation

Scheduling involves assigning the operations to so-called control step while preserving control and data dependencies between these operations. The aim of scheduling is to assign operations to control steps so as to minimize a given objective function while meeting constraints. The objective function may include the number of control steps, power consumption and hardware resources. Scheduling is an important task in HLS because it impacts the trades-off between design cost and performance. A control step is the fundamental unit in synchronous systems for it corresponds to a clock cycle.

To speed up the computation of a specific graph, the operations in the control graph can be scheduled with maximal parallelism, packing them into control steps as tightly as possible and observing only the essential dependencies required by the data flow graph[2].

Allocation involves assigning the operations and values to hardware i.e. providing functional units, storage and communication paths and specifying their usage. In allocation, the problem is to minimize the hardware needed. The hardware consists of functional units, memory elements, and interconnections. If more operations are scheduled into each control step, more functional units are necessary, resulting in fewer control steps for the design. On the other hand, if fewer operations are scheduled into each control step, fewer functional units are sufficient, but more control steps are needed.

In storage allocation, the values that are generated in one control step and used in another steps must be assigned to registers. Values may be assigned to the same register when their lifetimes are not overlapping. Storage assignment should be done in a way that not only minimizes the number of registers, but also simplifies the communication paths[2].

Communications paths, including buses and multiplexers, must be chosen so that the functional units and registers are connected as necessary to support the data transfers

required by the specification and the schedule. Multiplexers are considered to be the simplest type of communication path allocation compared to buses [8].

In the case where the design specification contains conditional branching or loop, the situation is more complicated. In conditional branching, we will have several branches that are mutually exclusive. During executing one path of the design, only one branch gets executed based on the outcome of the evaluated condition. Scheduling a loop body differs from scheduling a pure data flow graph, in that we need to consider potential parallelism across different loop iterations. Loop unfolding is one scheduling technique where a certain number of loop iterations are unrolled. This action results in a larger loop body with a fewer iterations. This loop body provides a greater flexibility for compacting the schedule[2].

2.1.3 Transformations

Transformations alter the organization of a computation in such a way that the user specified input/output relationship is maintained. They are often used as an effective approach for the implementing the computations. Transformations are the best way to overcome these resource utilization bounds. Three transformations that are particularly effective in achieving this goal are retiming, associativity, and commutativity[8] as shown in the Fig.2.1

Retiming has been successfully used in several areas of design synthesis and automation. It has been exclusively used in reducing the critical path in a graph, to minimize the number of delays or to optimize sequential networks with the aid of combinational logic tools by temporary moving the delays to a periphery of a network[12].

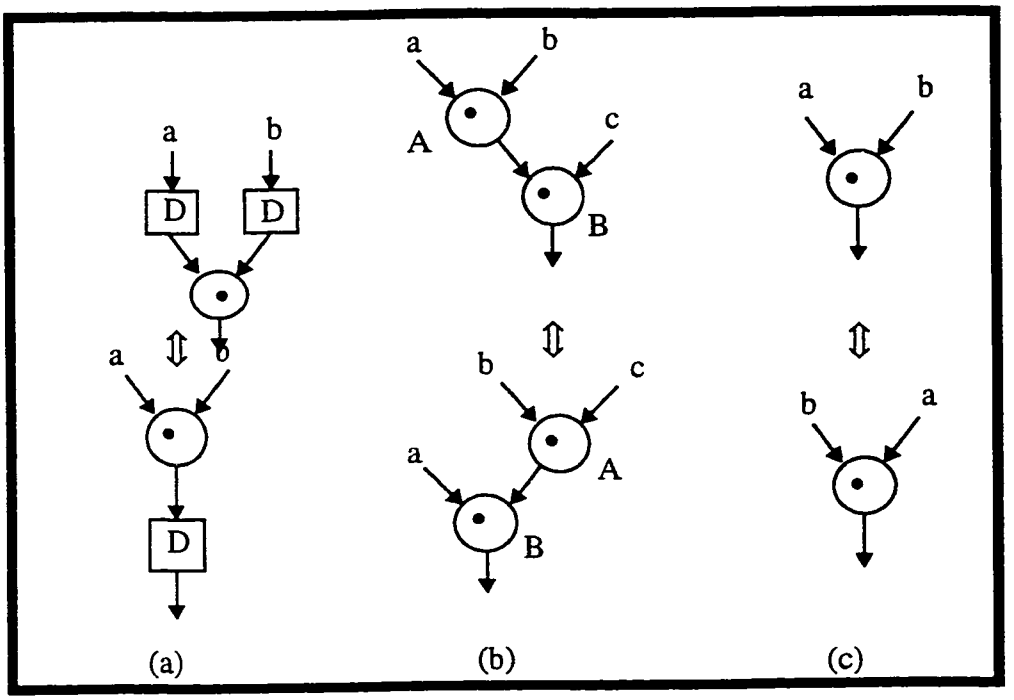


Fig. 2.1: Basic (a) retiming, (b) associativity and (c) commutativity moves

As an example, in Fig.2.2, after applying associativity on the a chain adder, the critical path is reduced to four cycles and one adder and one multiplier are sufficient for implementation. Tree-height reduction is the most common type used in associativity.In the real world, many variations on how the operations are implemented and the different structures of the data flow graph have to be considered during the scheduling phase.

Since the specification has been written for human readability and not for direct translation into hardware, it is desirable to do some initial optimization of the internal representation. These high-level transformations include many aspects, one of them is loop-unrolling. Loop unrolling can be done in the case of small number of iterations. Unfolding allows simultaneous processes to run reducing the critical path but not reducing data dependencies[7].

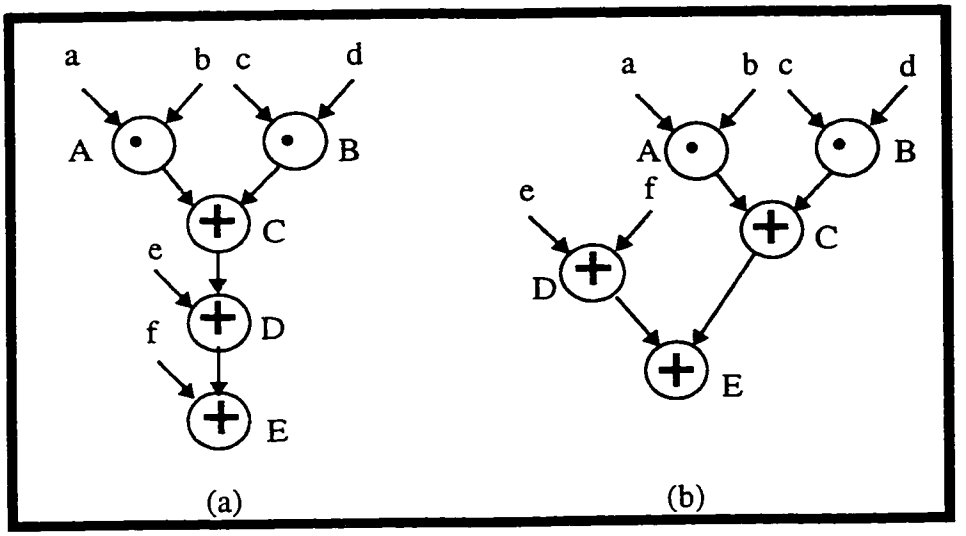


Fig. 2.2: Applying associativity to improve resource utilization: Flow graph (a) before and (b) after

2.1.4 Partitioning

Partitioning is used in HLS for scheduling, allocation, unit selection, and chip and system partitioning. Partitioning can be used to cluster variables and operations into groups so that each group is mapped to a storage element, functional unit or an interconnection unit of the real design.

The most frequent application of partitioning algorithms to HLS is in unit selection, which defines the functional units to be used for scheduling and binding. Operators in the CDFG are partitioned into clusters according to their functional similarity, where each cluster defines a functional unit executing all the operations in the cluster.

Another application of partitioning to HLS is decomposing a behavioral description into a set of interconnected processes. These processes can be mapped into a Finite State Machine (FSM). Such partitioning is similar to the scheduling of processes onto different processors in a multiprocessor. Partitioning algorithms group these processes into clusters so that each cluster is executed sequentially on one of the processors. The main goal of

this grouping is to execute the entire cluster under one control unit while satisfying design constraints such as chip size, clock rate and power consumption[15].

2.2 ILP Tools for High-level Synthesis

Automated design tools at higher levels of design provide the designer with the capability of reducing the design time, which is very crucial given the competitiveness of the current market. Furthermore, automatic tools are less likely to produce incorrect designs. Another benefit of designing at high-level is the ability to search a larger part of the design space for a satisfactory implementation.

These high-level tools will reduce not only the time for designing but also the number of iterations required for generating a satisfactory design by guiding the designers towards a better result in an earlier part of the design process.

The main goal of the synthesizers is to transform an input algorithm or behavior into a hardware architecture that minimizes a cost function and satisfies a set of constraints. The architectural synthesis problem involves several interdependent subtasks including scheduling, and the allocation of functional units, registers, and interconnects. Integer linear programming can be an efficient method to solve the scheduling and binding problems. Binding refers to allocating each operation to one specific functional unit. Operation scheduling determines the cost speed trades-off of the design.

The simplest scheduling technique is *As Soon As Possible* (ASAP) scheduling where the operations in the CDFG are scheduled step by step from the first control step to the last. An operation is said to be ready if all of its *predecessors* are scheduled. This procedure repeatedly schedules ready operations to the next control step until all the operations are scheduled.

As Late As Possible (ALAP) scheduling performs a very similar procedure as ASAP.

In contrast to ASAP, ALAP scheduling schedules the operations from the last control step toward the first. An operation is scheduled to the control step as all its *successors* are scheduled.

A third technique, the *list scheduling,* similar to ASAP, the operations in the CDFG are assigned to control steps from the first control step to the last. The ready operations are given a priority according to heuristic rules and are scheduled into the next control step according to this predefined priority. When the number of scheduled operations exceeds the number of resources, the remaining operations are delayed[2].

Another type of scheduling is that it selects the next operation to be scheduled then decides the control step in which to put it, this is called *force-directed scheduling.* In force-directed scheduling, the pairing of operation and control step that has the most attractive force is selected and assigned. After the assignment, the forces of the unscheduled operations are re-evaluated. Assignment and evaluation are iterated until all operations are assigned. In this scheduling technique, the maximum number of control steps must be specified[2].

Since the above scheduling methods mentioned above assign operations to control steps one at a time, their results depend strongly on the order of the assignments. If the design is subject to speed constraint, the scheduling algorithm will attempt to parallelize the operations to meet the time constraint. Conversely, if there is a limit on the cost (area or resources), the scheduler will serialize the operations to meet the resource constraint[9]. Once the operations are scheduled, the number and the types of functional units, the life-times of variables and the timing constraint are fixed.

A tool OSTA[14] was designed to perform architectural synthesis to digital filters and produce optimal data paths favored to other synthesis methods. In this tool, we state the scheduling problem by a mathematical description and then solve it using an Integer Lin-

ear Programming (ILP) method. ILP is an efficient method to solve the scheduling and allocation problems. This is because formulation with linear cost function can be solved much more easily than that of nonlinear cost function.

By carefully arranging the data dependency relationships in the formulation, it is possible to formulate the cost function as linear. We can reduce the solution space of the scheduling algorithm by restricting the range of control steps for each operation. This can be done by using both the ASAP and ALAP scheduling together with setting an upper and a lower limit to the number of functional units of each type. By this procedure, unnecessary searches can be easily avoided.

The ILP program was made to reduce the number of functional units, registers and buses by balancing the concurrence of operations assigned to them but without lengthening the total execution time[14]. Concurrence balancing helps to achieve high utilization - or low idle time - of the structural units, which in turn minimizes the number of units required[10].

The tool has a graphics user interface that enables the user to draw the flow graph of any digital design consisting of adders, multipliers, and delay elements. This step is usually followed by a translation of the description into a graph based representation called the control data flow graph (CDFG). The CDFG is represented in a netlist file generated automatically as the graph is drawn. The netlist file produced describes the type of edges (input, output or direct, recursive) and node operations (adders, multiplier, subtracter and ALU) used in the graph. An intermediate program is written to prepare data by determining the ASAP and ALAP of each operation, numbering edges and operations and specifying their types and the data dependency of the operations.

A database file is created containing all the information about each node and its type and each edge and its type, written to be used for the next stage. The node, as mentioned

previously, could be an adder, a multiplier, a subtracter or an ALU. The edge is either of type direct, that is a connection from one FU to another FU, or recursive, a connection between a delay element and an FU. Also the edge could be an input/output port.

After the ASAP and ALAP scheduling, the next is to assign a propagation delay value to the operations and partition them into specific control steps.The netlist file produced from the interface is used as input to an intermediate program to calculate the asap and alap of each operation and to determine the critical path of the graph. These values with the netlist are used as parameters to generate the code of the ILP for scheduling and binding using the mathematical formulations written in [12].

The ILP formulation is achieved by arranging the data dependency relationship in the graph and then restricting the range of control steps for each operation. The restriction is done to reduce the search space for each operation. The second phase is done using the ASAP and ALAP scheduling. This ILP program allocates the resources for the digital system and assigns the control step in which the operation is scheduled. This formulation results in lower interconnections and multiplexer inputs. As a whole the first ILP program sets the following goals: minimization of FUs, minimization of control steps, minimization of registers, and /or minimization of the number of parallel transfers.

When the ILP, written in OSTA, is used to do the scheduling and binding, the user can choose the number of each type of the functional unit types. To obtain an optimal design, scheduling and binding are done simultaneously. However, this tool allows the user to have choices in either tightening the time or the resources or a combination of both.

(a) Given the maximum number of control steps, find a minimum cost schedule that satisfies the given set of constraints. The maximum number of control steps should be greater than or equal to the number of control steps in the critical path. In other words, given the ALAP and ASAP of each operation, minimize the cost of the resources.

(b) Given the maximum number of resources, find the fastest schedule that satisfies the given set of constraints. In general, the resources are the number of functional units such as adders, multipliers, and ALUs. Registers and interconnections are difficult to specify as resource constraints.

(c) Given a fixed amount of resources and a specified number of control steps decide if there is a schedule that satisfies all the constraint. Produce the solution if it exists. This approach is useful since it allows the user or an expert system to control the speed-time trades-off. Since a set of optimal solutions is generated and the selection of best time/area implementation is left to the user.

Another ILP program was taken from [12] to perform bus transfer scheduling and bus allocation and binding, altogether with storage minimization. In this synthesis technique, the input/output ports are considered as functional units or registers. The register binding is done after scheduling and binding the operations to their functional units. The register binding is done concurrently with the placement and routing phase. This enforces to take into account for interconnection and storage cost at the same time when doing the scheduling and binding. The ultimate for this stage is do the following: minimize the number of busses, schedule the bus transfers, and/or minimize the destination registers connected to each bus.

This synthesis methodology performs the register binding after scheduling and binding the operations. This approach makes it essential to take into consideration the interconnection and storage cost together with the operation scheduling and binding. Minimizing the interconnects is done by minimizing the number of motifs. A motif is a representation of all edges that have the same FU bound to its source operation and the same the same FU to its destination operation.

As the number of motifs decreases, the utilizations interconnects utilization becomes

maximum leading to a minimum structural complexity. The result is: reduction of local interconnections done by optimal operations to functional units binding, and high throughput in the busses by the increase of data transfers to these busses. The Graphical User Interface and the automation techniques for the two ILP programs were written in Prolog language.

At this stage, the functional units are assigned to execute the operations, storage units are assigned to store the values, and wires are allocated to interconnect them using the data transfer information derived from the CFDG. Here the data path is completed. Finally, based on the schedule graph and the data path, a control unit is synthesized to synchronize the executions of the operations.

2.2.1 Synthesizing A Digital filter using OSTA:

The following is an example of how the tool works. The following circuit, in Fig.2.3, is drawn using the graphical user interface. The circuit elements are mapped to a CDFG representation.The CDFG will be present as a database file that informs the type of each element and the type of connection of each edge.

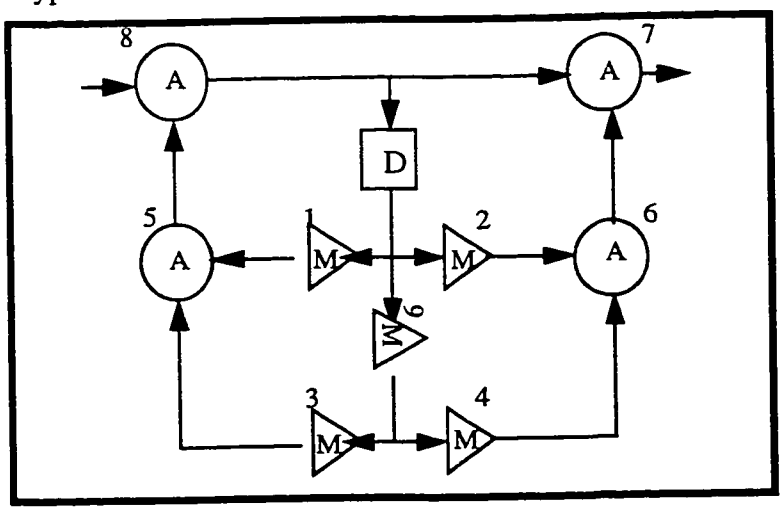


Fig. 2.3: The structure of the Biquadratic filter.[9]

Once the database file is prepared, the program calculates the ASAP and ALAP of each operation and given the time delay of each functional unit, the critical path is found.

The data in the files explains the connections and precedences of each operation. Each connection to a delay unit is cut and related to a register. As the critical path is found, the upper limit of the control step cycles is determined.

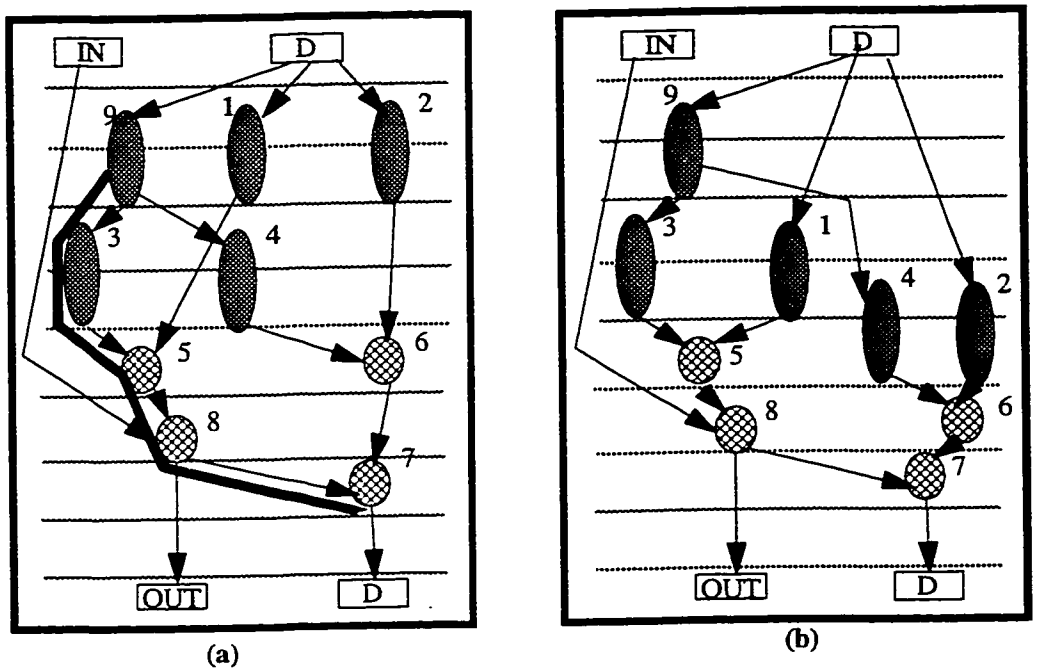


Fig. 2.4: The scheduling of the CFDG of Fig.2.3 (a) ASAP and (b) ALAP. The thick line shows the critical path.

The first ILP program will schedule the operations in specific cycle steps by giving the number of resources. With all the data given, minimum number of functional units, control steps, registers and storage are performed in this step. Fig.2.4 shows the schedul-ing and operations binding with two cases. In this example, the delay of the multiplier is two step-cycles and the that of the adder is one step-cycle. As shown in Fig.2.5a, one adder and one pipelined multiplier is used. The number of cycle-steps in that figure is one

cycle more than that in Fig.2.5b, where two multipliers and two adders are used. This shows the relationships between the cost of area and time: they are orthogonal to each other, i.e. the time is reduced by increasing the area and vice versa.

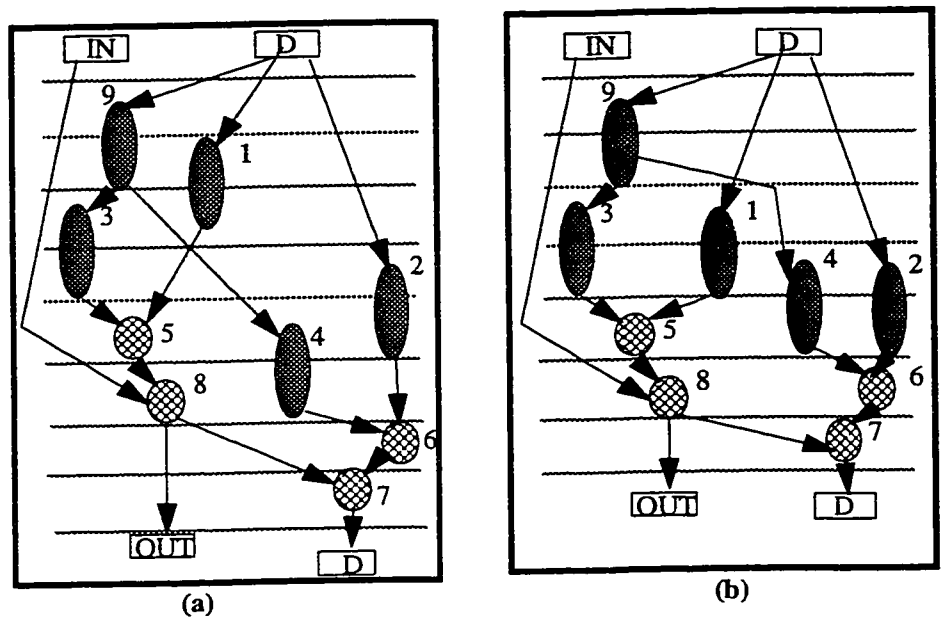
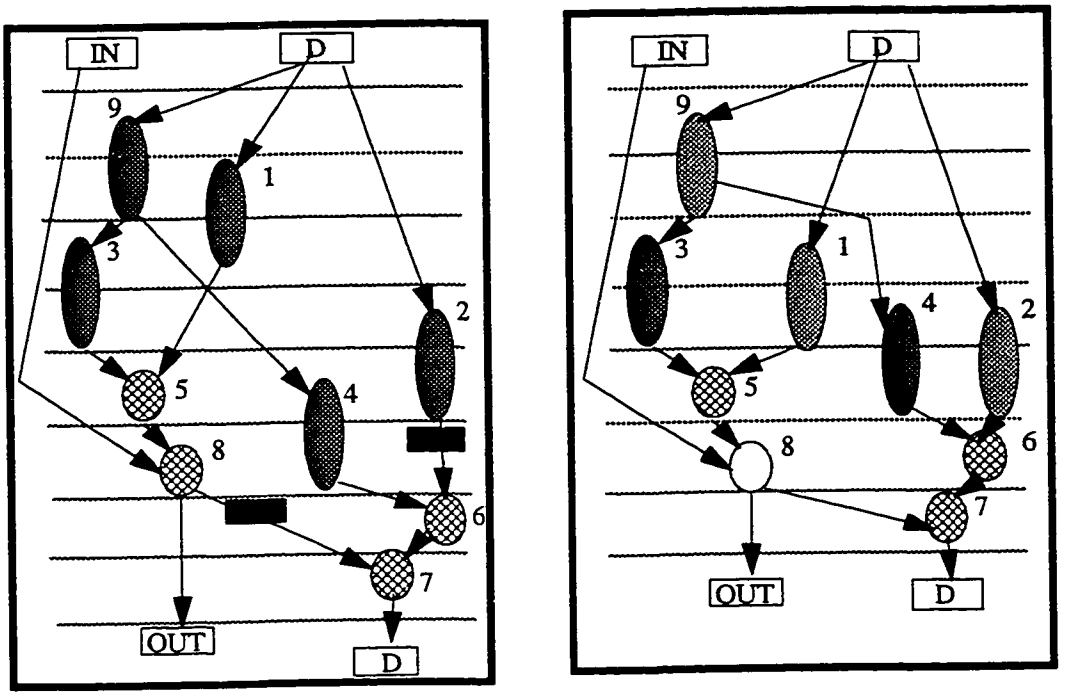


Fig. 2.5: Operation scheduling using (a) 1 adder and 1 multiplier, and (b) using two multipliers and two adders.

The objective in the second ILP formulation is to minimize the number of bus allocation, register file storage locations and bus loading[8]. In this stage, no register binding is made, only the scheduling of the bus transfers. In our example, a register is allocated between node 7 and node 8 as shown in (Fig.2.6a) shown in the next page.



(a) 2nd phase, two registers are added.

(b)2 multipliers and 2 adders are used and no register.

Fig. 2.6: The two graphs showing registers added to (a) and motifs are established in (b) between the two multipliers.

In the second case, in Fig.2.6b, where two adders and two multipliers are used, no registers are employed, and there are motifs used between the two multipliers and from one multiplier to an adder which leads to smaller interconnections. The data paths for these two cases are shown in Fig.2.7 and Fig.2.8 consecutively. The first case, Fig.2.7, where one adder and one multiplier are used, multiplexers are used to control the data transfer to and from the FU.

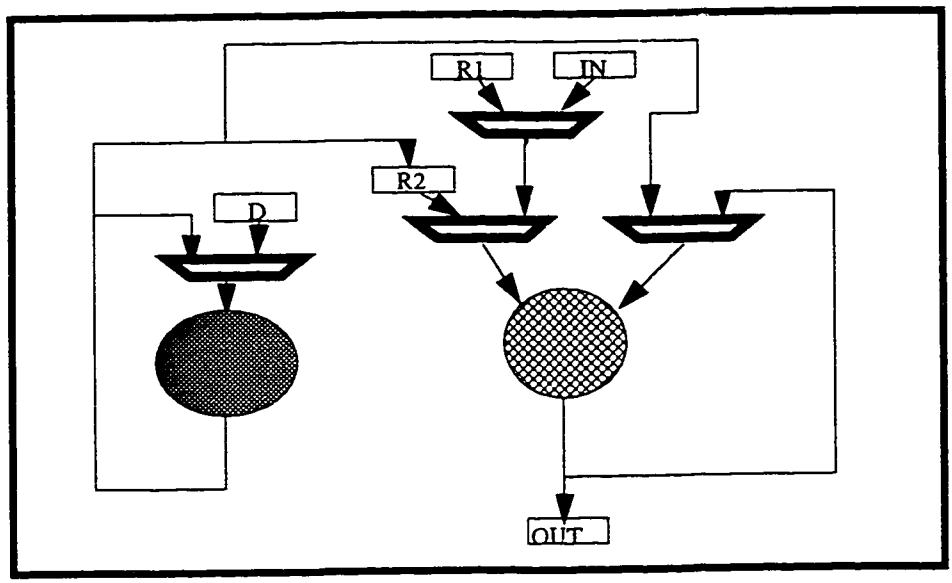


Fig. 2.7: The data path of the filter using one adder and one multiplier.

It is noticed in Fig.2.7 that two registers are also added to delay the same data for another operation. This is because the same data is needed for different operations in different step cycle.The second case, Fig.2.8, two adders and two multipliers are employed. The functional units are doubled but there are no registers used and the number of multiplexers are less.

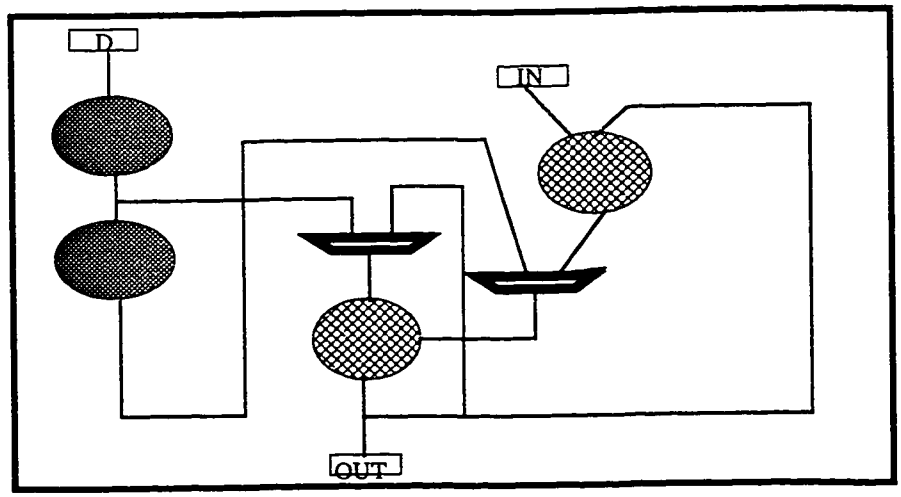


Fig. 2.8: The data path of the filter using 2 adders and 2 multipliers.

In summary, the whole tool is translating the circuit diagram drawn from the graphics user interface to database file that represent the CDFG. Once this information is ready, the code needed to run ILP formulations in GAMS (an ILP programming software) is generated and the ILP file is set to be running to produce the scheduling and binding of the operations. The solution to the ILP formulation is stored in a new file to be used for bus scheduling and register binding. The whole procedure is explained in the next figure below.

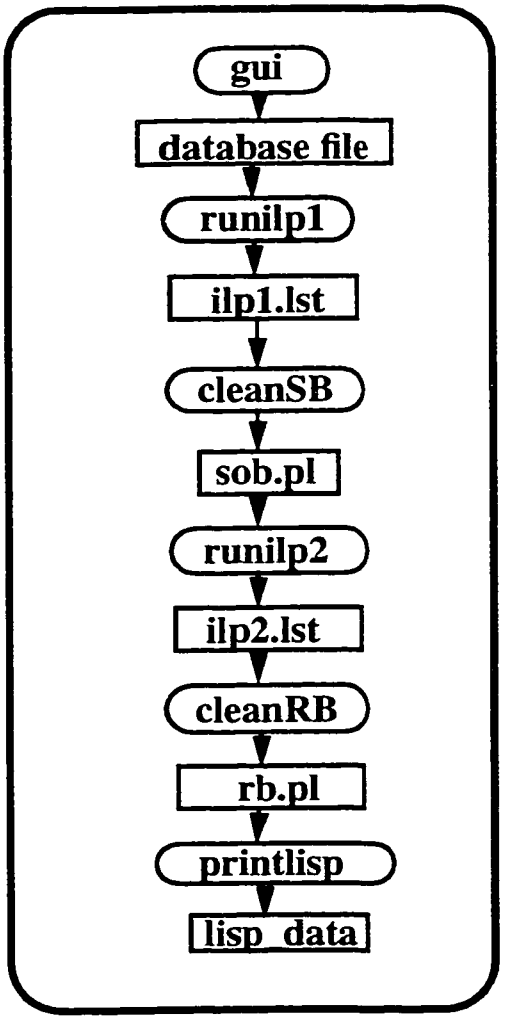


Fig. 2.9: The flowchart of the synthesis tool.

Fig.2.9 shows the sequence of commands needed to do the synthesis of any digital

circuit. The following is a brief detail of each command and the type of file produced by each command. The code written for the commands, might be in Prolog, GAMS and unix ex- editor commands. The elliptic shape in the figure is the command representation and the rectangle is a file representation.

gui: This command opens a graphical user interface that allows the user to draw the circuit diagram of any typical digital filter. The operations are mainly addition, multiplication and delay. The output of the command is the database file.

database file: This is the file produced from the gui command. It contains the control and data flow. The data in the file is mainly about the type of operations, and the control and data dependencies between these operations. The results are stored as variable elements in Prolog to be used as parameters to the next phase.

runilp1: This is the command that generates the ILP formulations given the data from the graph drawn. The number of functional units to be used for implementation, the delay of each functional unit in cycle-steps and the degree of optimization are given by the user interactively. Then the formulation is generated to run in GAMS. This is the scheduling and binding phase. It assigns each operation to a specific functional unit and determines the cycle-step the operation is taking place.

ILP1.lst: is the output file that results from running the formulation in GAMS. It contains the list of operation-functional unit binding and scheduling. The result is written in a GAMS format that needs an intermediate phase to interpret it as a prolog input.

cleanSB: This is an ex-editor file in the unix system. It scans the results from the listing and writes the data output in a Prolog format. The data output shows each operation binding and scheduling.

runilp2: This command shifts to the next stage, bus allocation and minimization. As mentioned before, the tool takes into consideration minimizing the interconnections as

well. Register allocation is a way of achieving that. The command runilp2 generates the ILP formulation needed to run GAMS and allows the user interactively to set the number of registers desired in the optimization.

ilp2.lst: It is the output file that results from running the GAMS for bus and register minimization.

cleanRB: It is like cleanSB, a unix ex-editor file that reads the data output from the listing of the second ILP formulation and writes the result in a prolog variable to be used for the final stage.

printlisp: This is a command that takes the results of both operation scheduling and binding with register minimization and produces the data path accordingly in the Lisp format.

lisp_file: This is the output file that contains the description of the data path in a lisp format. The lisp format is to do the floor planning for extended purposes.

As a result, the OSTA tool has an objective of optimizing the interconnections concurrently with the scheduling and binding of the operations. The number and the type of the functional units to be implemented are left to the choice of the user. The flexibility of these two ILP formulations allows to adjust the delay of the data-transfers between different partitions to some technology dependent value[8]. However, there are limitations to the tool: the running time to find the optimal solution may take hours if not days depending on the complexity of the design. The synthesis tool program is given in Appendix A.

Some of the synthesis techniques used in OSTA are applied in the implementation proposed in this thesis. As will be seen in the next chapters, the steps mentioned in the HLS methods, from the clustering technique to scheduling and binding of operations together with loop unfolding will prove to produce an effective and optimal implementation of the RS(n,k) decoder.

CHAPTER 3

Versatile Reed-Solomon Decoders

Reed-Solomon Codes are constructed from fields with a finite number of elements. A finite field with q elements is generally called a *Galois field* and denoted as Gift. Every field must have a **zero** element and **one** element. Hence the simplest field is GF(2). In general, when q is prime, we can construct the finite GF(q) consisting of the elements {0, 1,..., q - 1}. The addition and multiplication operations on the elements of GF(q) are defined modulo-q and denoted as (mod-q).

3.1 Galois Field Elements and Their Characteristics:

The finite field GF(q) can be constructed only if q is prime or a power of a prime. When q is prime, multiplication and addition are based on modulo-q arithmetic. If $q = p^m$ where p is a prime and m is any positive integer, it is possible to extend the field GF(p) to GF(p^m). This is called the *extension* field of GF(p). Multiplication and addition of the elements in the extension field are also based on modulo-p arithmetic. Because of implementation simplicity, normally RS codes are constructed in GF(2^m).

3.1.1 Properties of GF(2^m)

1. Elements of GF(2^m) can be represented in powers of the primitive element α as $GF(2^m) = \{0, 1, \alpha..., \alpha^i, ..., \alpha^{2^m-2}\}$, where α is a root of the irreducible generator polynomial $I(x)$.

2. The multiplication of two elements of GF(2^m) is defined as:-

$$0 \cdot \alpha^i = \alpha^i \cdot 0 = 0$$

$$1 \cdot \alpha^i = \alpha^i \cdot 1 = \alpha^i$$

$$\alpha^i \cdot \alpha^j = \alpha^{(i+j) \bmod -n} \quad \text{, where } n = 2^m - 1 \ .$$

3. α^i, any element in $GF(2^m)$ can be represented as a polynomial representation as $\alpha^i = \alpha_{i0} \cdot \alpha^0 + \ldots + \alpha_{i,m-2} \cdot \alpha^{m-2} + \alpha_{i,m-1} \cdot \alpha^{m-1}$ or in power representation as

$$\alpha^i = (\alpha_{i0}, \alpha_{i1}, \ldots, \alpha_{i,m-1}) \quad .$$

The power representation is convenient for multiplication while the polynomial representation is convenient for addition.

Let $m = 4$. The polynomial $p(x) = 1 + X + X^4$ is the primitive polynomial over GF(2)[1]. If set $p(\alpha) = 1 + \alpha + \alpha^4 = 0$, then

$$\alpha^4 = 1 + \alpha \quad . \tag{3.1}$$

The elements of GF(2^4) are given in the table below and using equation (3.1), we can form the polynomial representations as well. For example,

$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(1 + \alpha) = \alpha + \alpha^2$, and so on. To multiply two elements α^i and α^j, we simply add their exponents and use the fact that $\alpha^{15} = 1$. For example, $\alpha^7 \cdot \alpha^5 = \alpha^{12}$ and $\alpha^{12} \cdot \alpha^{10} = \alpha^{22} = \alpha^7$.

4. Addition in GF(2^m) is carried in using the polynomial representations in the table below. Thus, for adding α^i and α^j,

$$\alpha^i + \alpha^j = (\alpha_{i0} + \alpha_{jo}, \ldots, \alpha_{i,m-1} + \alpha_{j,m-1}) \quad [1], \text{ for example,}$$

$$\alpha^5 + \alpha^7 = (\alpha + \alpha^2) + (1 + \alpha + \alpha^3) = 1 + \alpha^2 + \alpha^3 = \alpha^{13} \ .$$

It is important to notice that adding the same element to itself yields a 0.

5. Squaring in GF(2^m) is linear

$\forall \alpha, \beta \in GF(2^m)$ $\quad (\alpha+\beta)^2 = \alpha^2+\beta^2$, because $2 \cdot \alpha\beta = \alpha\beta+\alpha\beta = 0$.

TABLE 3.1: Three representations for the elements of GF(2^4) generated by $p(x) = 1+X+X^4$

Power Representation	Polynomial Representation	4-Tuple Representation
0	0	(0000)
1	1	(1000)
α	α	(0100)
α^2	α^2	(0010)
α^3	α^3	(0001)
α^4	$1+\alpha$	(1100)
α^5	$\alpha+\alpha^2$	(0110)
α^6	$\alpha^2+\alpha^3$	(0011)
α^7	$1+\alpha+\alpha^3$	(1101)
α^8	$1+\alpha^2$	(1010)
α^9	$\alpha+\alpha^3$	(0101)
α^{10}	$1+\alpha+\alpha^2$	(1110)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0111)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1111)
α^{13}	$1+\alpha^2+\alpha^3$	(1011)
α^{14}	$1+\alpha^3$	(1001)

3.1.2 Implementation of Galois Field Arithmetic:-

Galois field arithmetic can be implemented more easily than ordinary arithmetic because there are no carries. To add two field elements, we simply add their vector representations. The resultant vector is then the vector representation of the sum of the two field elements. For example, we want to add α^7 and α^{13} of GF(2^4). From the table above, we find that their vector representations are (1101) and (1011) respectively. Their vector is (1101) + (1011) = (0110), which is the vector representation of α^5. Addition of the two

field elements can be accomplished with the circuit shown in the figure below.

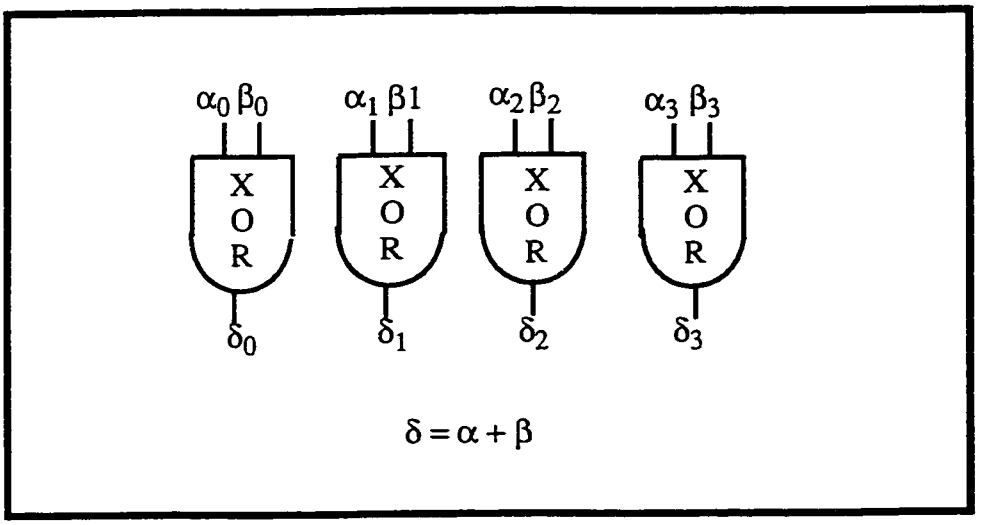


Fig. 3.1: The implementation of GF(2^4) adder.

The Galois Field multiplier has a different structure than the binary arithmetic multiplier. It is more complicated yet still pure combinational logic. It is implemented as parallel-in/parallel-out Galois Field multiplier [1].The choice of the multiplier depends on the irreducible polynomial that can generate $GF(2^m)$. The polynomial which introduces the least complex α-multiplier[5], should be chosen and is called the minimal polynomial. Suppose that we want to multiply a field element β in GF(2^4) by the primitive element α whose minimal polynomial is $p(x) = 1 + X + X^4$. The structure of α-multiplier is shown in Fig.3.2(a).The structure of the Galois Field GF(2^4) multiplier is shown in Fig.3.2(b).

After the properties of GF(2^m) have been stated, it has been seen that the multiplication and addition on these finite field elements are different than the ones performed on real numbers or integers. However, the implementation of the operational units of such elements is similar to that for reals or integers.For the case study, versatile time-domain Reed-Solomon decoder is considered since RS codes are defined in Galois Fields. It will

be shown that HLS techniques could be applied to implement such decoder and that this approach will yield interesting results.

RS codes have a very large burst error correcting capability when the symbols are transmitted bit by bit. These codes correct a large number of random errors. RS codes are used in many applications such as in satellite, spread spectrum and mobile communications as well as in magnetic and optical storage systems and this is the fact due to their error correcting capabilities and optimum structure.

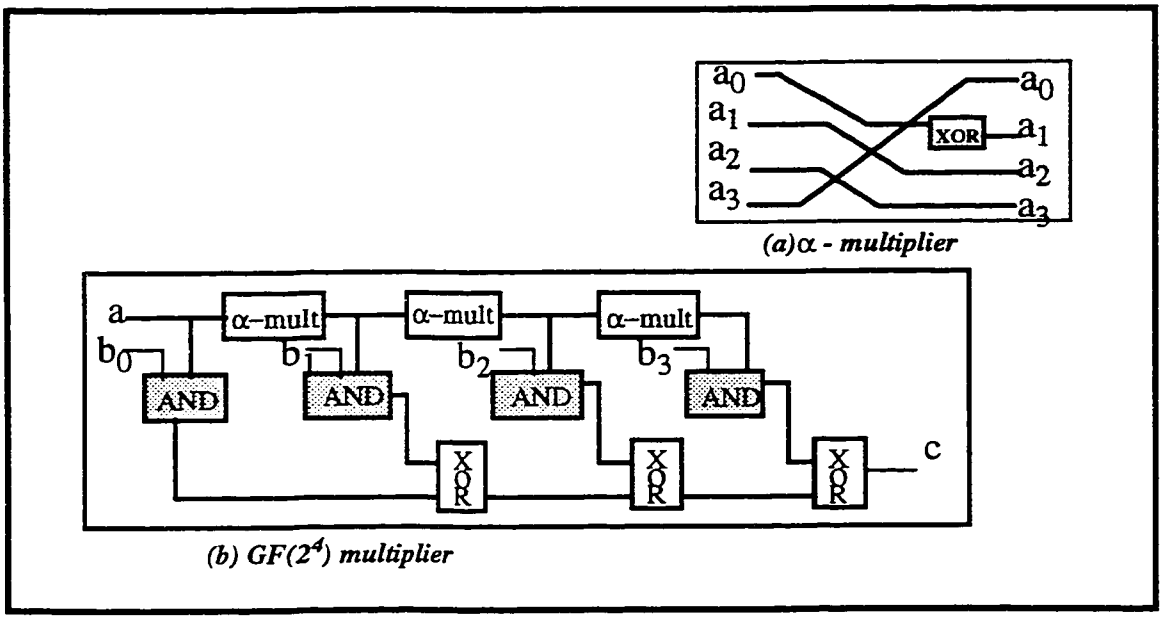


Fig. 3.2: The structures of α-multiplier[6] (a) and the $GF(2^4)$ multiplier employing α-multiplier[5] (b).

3.2 The Time-Domain Versatile RS(n,k) Decoders

An RS code is a block of symbols where each symbol is an element of $GF(2^m)$. An RS (n, k) code has the following parameters:-

m= number of bits per symbol.

$n = 2^m - 1$ = length of codewords in symbols.

t= maximum number of error symbols that can be corrected.

d= number of check symbols $=2t$.

$k = n - 2t$ = number of information symbols.

RS codes have a minimum distance of $2t + 1$ and $t -$ symbols error-correction capability. A received word with any combination of t or fewer symbols in error will be correctly decoded. Block codes such as RS codes, can use one bit of soft decision information per symbol, called erasure indicator, to inform that the received symbol is incorrect. In fact an erasure is an error with known location. An RS (n, k) code is able to correct up to $n - k$ erased symbols. Any pattern of ν symbol errors and ρ symbol erasures can be corrected provided that $2\nu + \rho \leq n - k$. To correct an error, the decoder must find both its location and its value. To correct an erasure, only its value needs to be found. A versatile RS (n, k) decoder has its error and erasure correction capability programmed, yet the code should be defined in $GF(2^m)$ with a fixed m[4]. The limitation is because the multiplier in Galois fields has different structures for different values of m[5].

For decoding, we have to solve a system of $2t$ simultaneous equations in the υ unknown error locations and υ unknown error values[28].

Let $c(x)$ be a code word polynomial of RS (n, k) and $e(x)$ be the error polynomial, the received vector then is

$$v(x) = c(x) + e(x) = v_o + v_1 x + \cdots + v_n x^{n-1}$$

where the polynomial coefficients are components of the received vector $\underline{v}$. Two main steps are involved in time-domain decoding.

3.2.1 The Time-Domain Algorithm[6]

The time-domain decoding algorithm, presented in Fig.3.3, the vector $\underline{\lambda}$ is the error locator vector, $\underline{s}$ is the errata value vector and $\underline{b}$ is the temporary storage of $\underline{\lambda}$. The value of L is initialized to zero, λ_i and b_i, $(i = 0, 1, \ldots, n-1)$, are initialized to $\alpha^0 = 1$, and the errata value vector $\underline{s} = \underline{y}$.

After initialization, there are n iterations, divided in three stages: according to the value of the parameters ρ, (the number of erasures) and k (the length of the message) in symbols.

In the *first stage*, we have ρ iterations, where the components of the error vector $\underline{\lambda}$ are evaluated according to the erasure locations j_r, $r = 0, 1, \ldots, \rho$, using following equations:

$$\Delta = \alpha^{j_r}$$

$$\delta = 0$$

$$\begin{bmatrix} \lambda_i \\ b_i \end{bmatrix} \leftarrow \begin{bmatrix} 1 & -\Delta\alpha^{-i} \\ \Delta^{-1} & (1-\delta)\,\alpha^{-i} \end{bmatrix} \begin{bmatrix} \lambda_i \\ b_i \end{bmatrix}$$

$$b_i = \lambda_i \tag{3.2}$$

In the iterations $r = \rho + 1, \rho + 2, \ldots n - k$, the *second stage*, the error vector $\underline{\lambda}$ is evaluated using the following set of recursive equations.

$$\Delta = \sum_{i=0}^{n-1} \lambda_i s_i$$

$$L \leftarrow \delta\,(r - L) + (1 - \delta)\,L$$

$$\begin{bmatrix} \lambda_i \\ b_i \end{bmatrix} \leftarrow \begin{bmatrix} 1 & -\Delta\alpha^{-i} \\ \Delta^{-1} & (1-\delta)\,\alpha^{-i} \end{bmatrix} \begin{bmatrix} \lambda_i \\ b_i \end{bmatrix} \tag{3.3}$$

where $\delta = 1$, if both $\Delta \neq 0$ and $2L - \rho \leq r - 1$, and $\delta = 0$, otherwise.

In the *third stage*, after the $(n-k)$ th iteration, the estimate of the errata value vector $\underline{s}$ is calculated according to,

$$\Delta = \sum_{i=0}^{n-1} \lambda_i s_i$$

$$s_i \leftarrow s_i - \Delta \tag{3.4}$$

and finally, if $\lambda_i s_i = 0$ for $i = 0, \ldots, n-1$, the received vector $\underline{y}$ is corrected to

$$\underline{c} = \underline{y} - \underline{s} \tag{3.5}$$

Hence, for an RS(n,k) code, we have a total of n iterations.

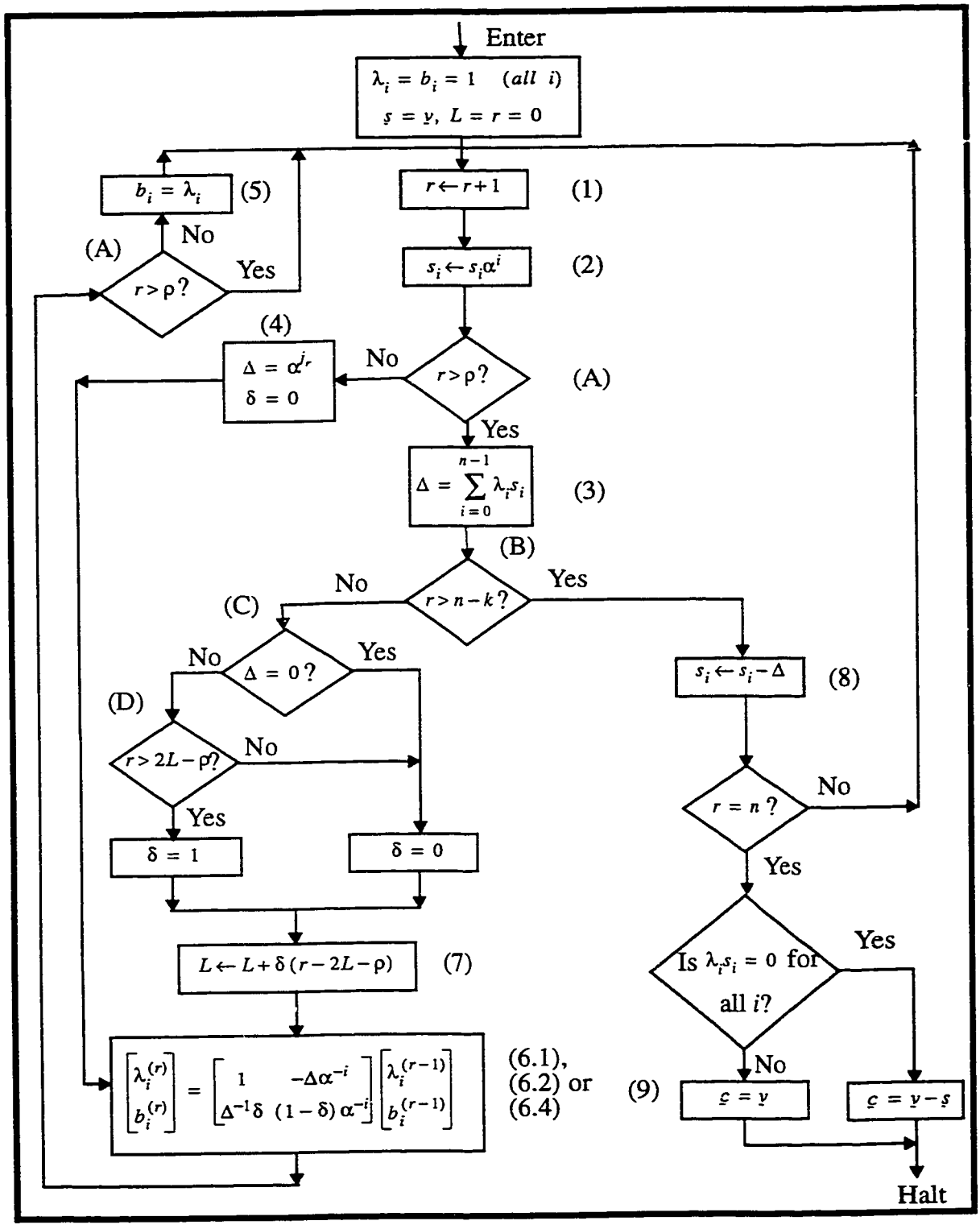


Fig. 3.3: The Versatile RS(n,k) Decoding Algorithm.[5]

3.3 High-level Synthesis Approach

Referring back to Fig.3.3, the calculations done in each iteration are not the same and depend on the values calculated previously, hence they involve decision making and loop branching which make the algorithm difficult to implement. As it is noticed, there are five conditions to be considered:

(1) A: $r > \rho$,

(2) B: $r > n - k$,

(3) C: $\Delta = 0$,

(4) D: $r > 2L - \rho$ and

(5) E: $r = n$

we see different paths according to the different possible conditions as in Fig.3.4.

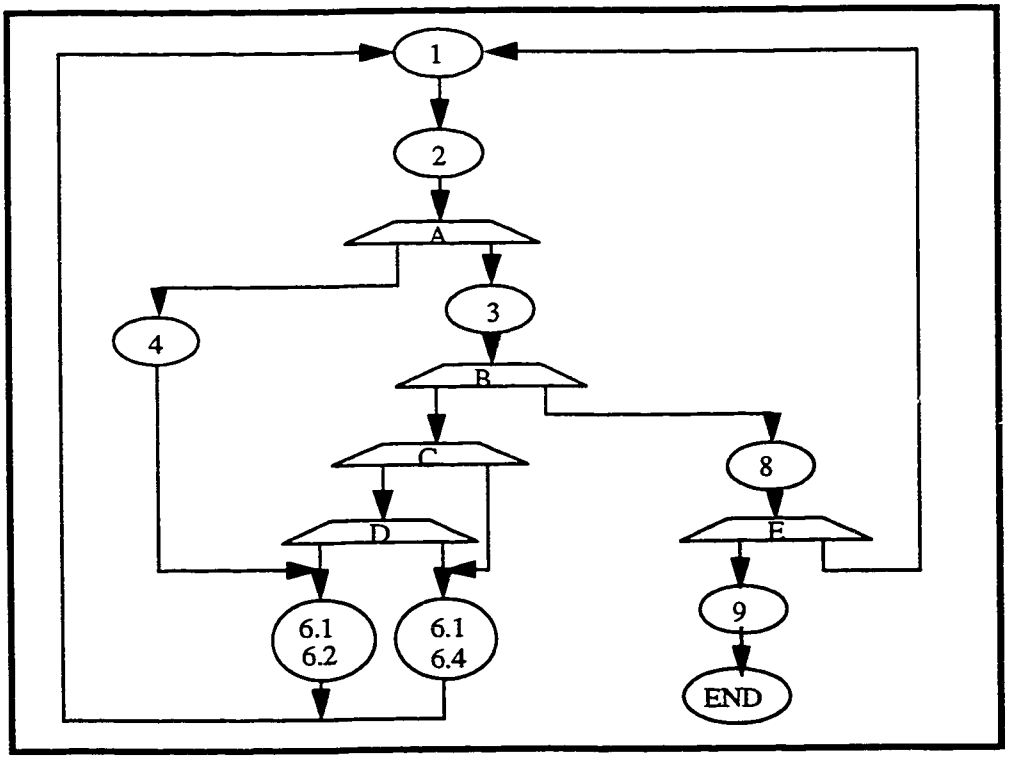


Fig. 3.4: The different paths of decoding algorithm.

Path π:

We have $(r \leq \rho)$ $\Delta = \alpha^{jr}, \delta = 0, L$ does not change, with these conditions, the sequence of operations is as shown in Fig.3.5:

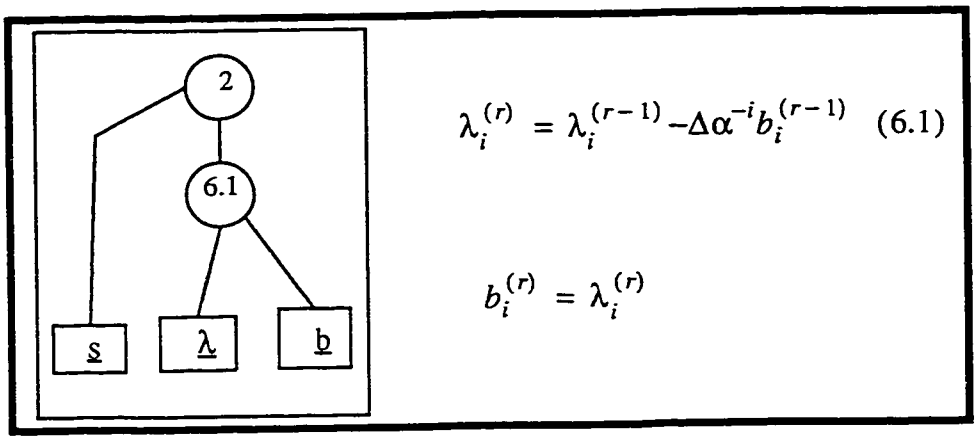


Fig. 3.5: Path π with the operations (2) and (6.1).

Path β:

With conditions $(r > \rho)$, $(r \leq n-k), \Delta \neq 0$, $r > 2L - \rho$, $\delta = 1$, we have the following operations as shown below in Fig.3.6

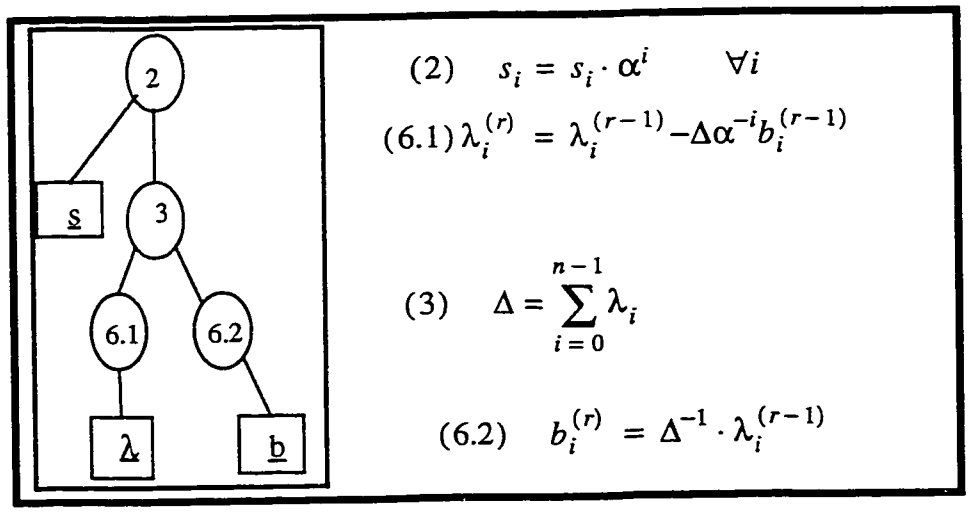
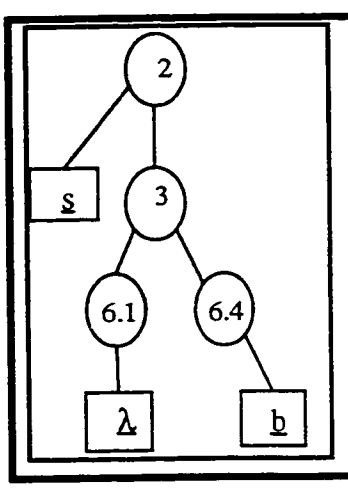


Fig. 3.6: Path β with the operations (2), (3), (6.1) and (6.2).

Path α:

With conditions $(r > \rho)$, $(r \leq n-k)$, $\Delta \neq 0$, $r \leq 2L-\rho$, $\delta = 0$ we have the following sequence of operations as shown in Fig.3.7



$$(2) \quad s_i = s_i \cdot \alpha^i \qquad \forall i$$

$$(3) \quad \Delta = \sum_{i=0}^{n-1} \lambda_i$$

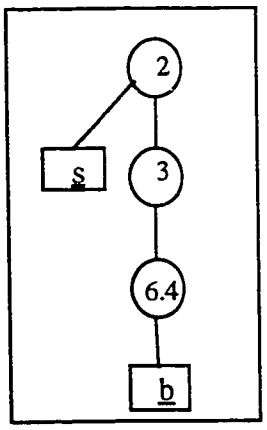
$$(6.1) \quad \lambda_i^{(r)} = \lambda_i^{(r-1)} - \Delta\alpha^{-i} b_i^{(r-1)}$$

$$(6.4) \quad b_i^{(r)} = \alpha^{-i} \cdot \lambda_i^{(r-1)}$$

Fig. 3.7: Path α with the operations (2), (3), (6.1) and (6.4).

Path Δ:

With conditions $((r > \rho), (r \leq n-k), \Delta = 0)$ the sequence of the operations becomes as shown below in Fig.3.8



$$(2) \quad s_i = s_i \cdot \alpha^i \qquad \forall i$$

$$\lambda_i^{(r)} = \lambda_i^{(r-1)}$$

$$(3) \quad \Delta = \sum_{i=0}^{n-1} \lambda_i \cdot s_i$$

$$(6.4) \quad b_i^{(r)} = \alpha^{-i} \cdot b_i^{(r-1)}$$

Fig. 3.8: Path Δ with the operations (2), (3) and (6.4).

Path σ:

When $(r = n)$, we have the following operation: if $\lambda_i s_i = 0$ for $i = 0, \ldots, n-1$, the received vector $\underline{y}$ is corrected to $\underline{c} = \underline{y} - \underline{s}$.

The above mentioned path, will be combined with path σ for simplicity, hence the operation (9) is inserted in Fig.3.9. With conditions $(r > \rho)$, $(r > n-k)$ the sequence becomes as shown in Fig.3.9

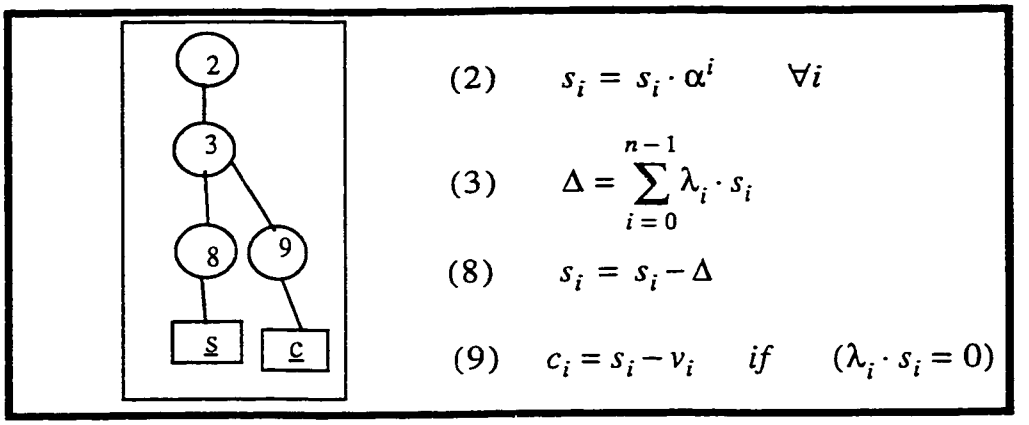


Fig. 3.9: Path σ with the operations (2), (3),(8) and (9).

As mentioned earlier, the decoding algorithm consists of a loop of n iterations with operations and decisions represented by circles and de-multiplexers. Implementing simple loops are easily done, but in our case the loop consists of conditional branches that depend on the data of the previous iterations which makes it more complex and difficult. From Fig.3.5 to Fig.3.9 we could see that there are five possible paths from which only one is taken at each iteration r , where $r = 0, \ldots, n-1$.

The conditions that control the choice are the values of ρ, k and Δ. The first two parameters are known in advance prior to any calculation, but the last, Δ, is calculated in the middle of each iteration in the loop. The fact that Δ is found later delays the decision made to choose the path. This is a major obstacle to make the design fast since the choice of the path, to be executed, must be determined before each iteration. It is noticed that

operation (2) that calculates the errata value vector $\underline{s}$, is performed in each path.

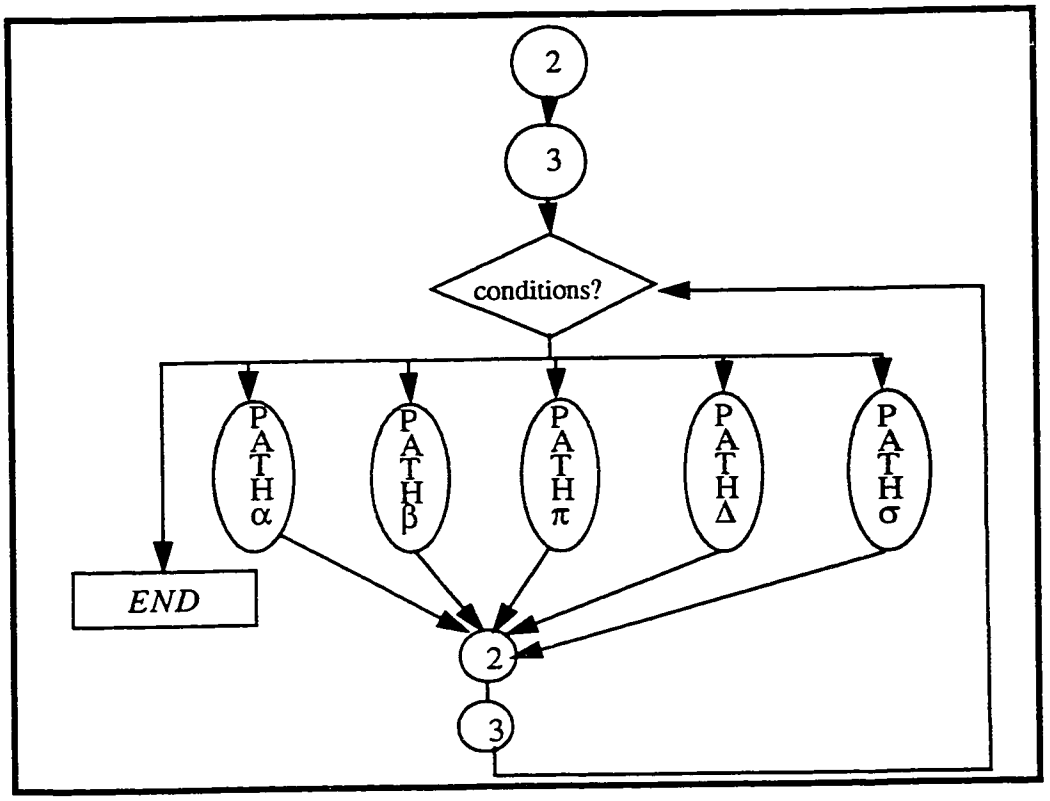


Fig. 3.10: The control flow graph of the decoder after retiming

To speed up the design, we can use one kind of transformation called retiming. Retiming has been used successfully in several areas of design synthesis and automation[2]. It has been used to reduce the critical path in the graph and to minimize the number of delays. Our approach is forwarding operations (2) and (3) in Fig.3.3, up to be done in the previous iteration, i.e. at iteration i, Δ_i and s_i are calculated in $(i-1)$ th iteration and the values of Δ_{i+1} and s_{i+1} are obtained in the ith iteration and so on.

By this transformation, the decision to chose the path can be determined at the beginning of each iteration, hence reducing the delay path and optimizing the resource utilization.The data flow graph, after retiming the operations mentioned previously shown in Fig.3.4, is transformed to the graph shown in Fig.3.10, above.

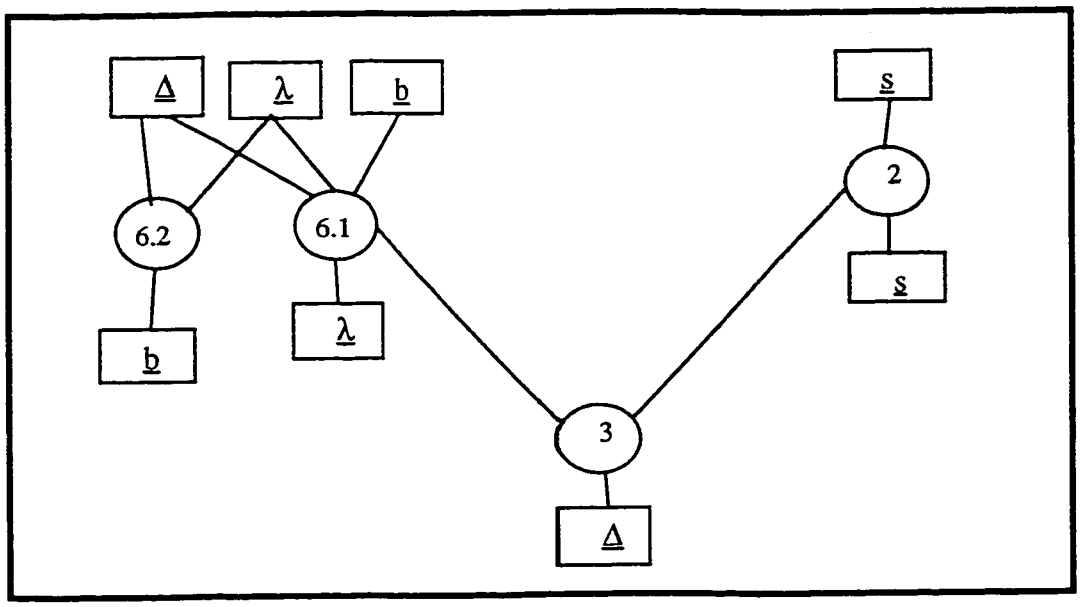


Fig. 3.11: Path β after retiming.

As the retiming is applied to the decoding algorithm, each of the paths mentioned previously is affected by this transformation. Fig.3.11 to Fig.3.15 represent the transformed paths shown in (Fig.3.5 - Fig.3.9).

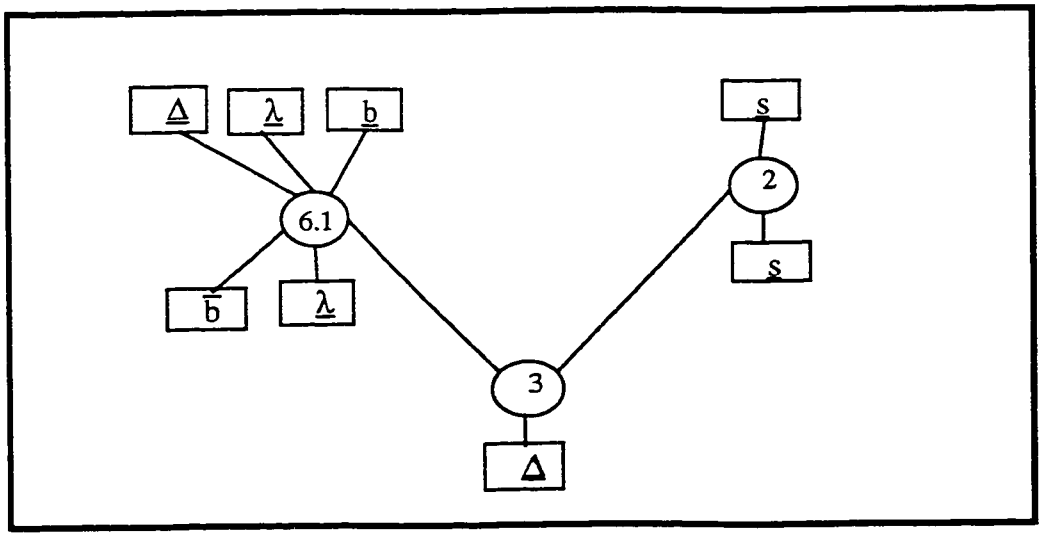


Fig. 3.12: Path π after retiming

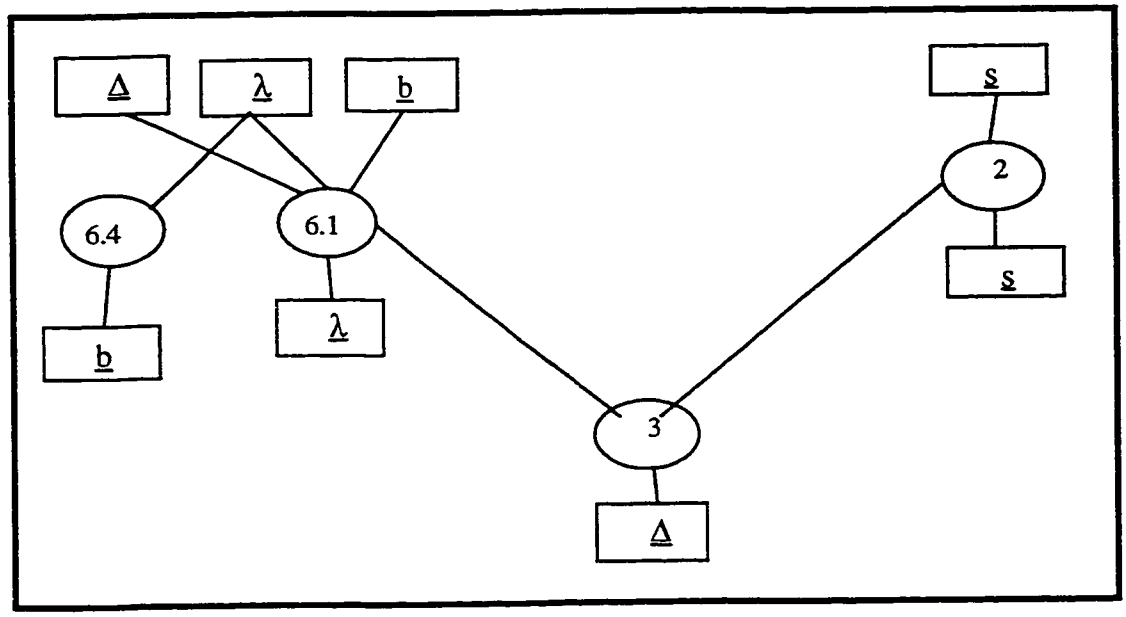


Fig. 3.13: Path α after retiming.

It is clearly shown that the time to do the operations in each path is reduced since the parameters that affect the decisions are evaluated at the beginning causing the design to be faster.

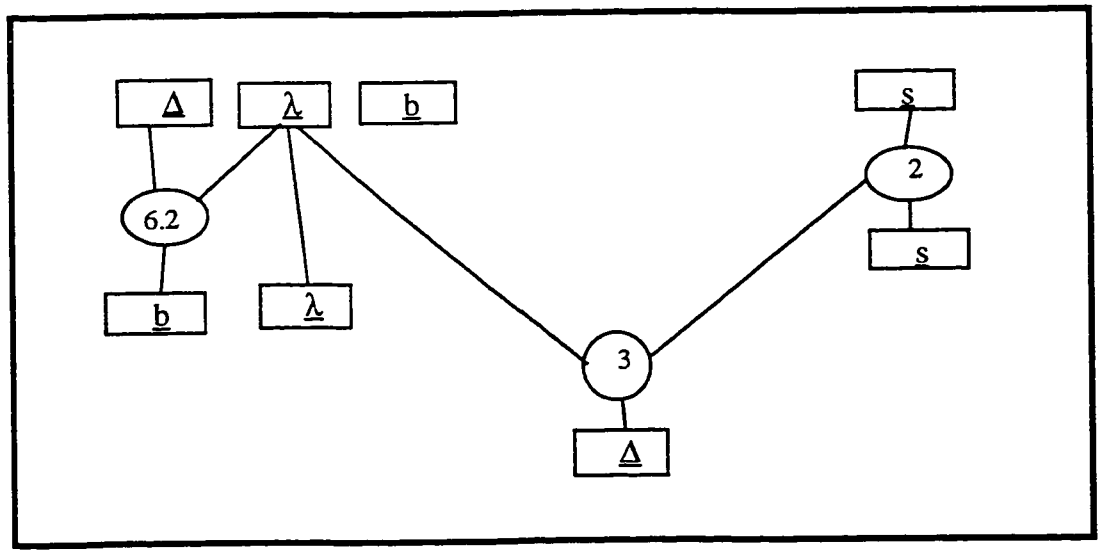


Fig. 3.14: Path Δ after retiming.

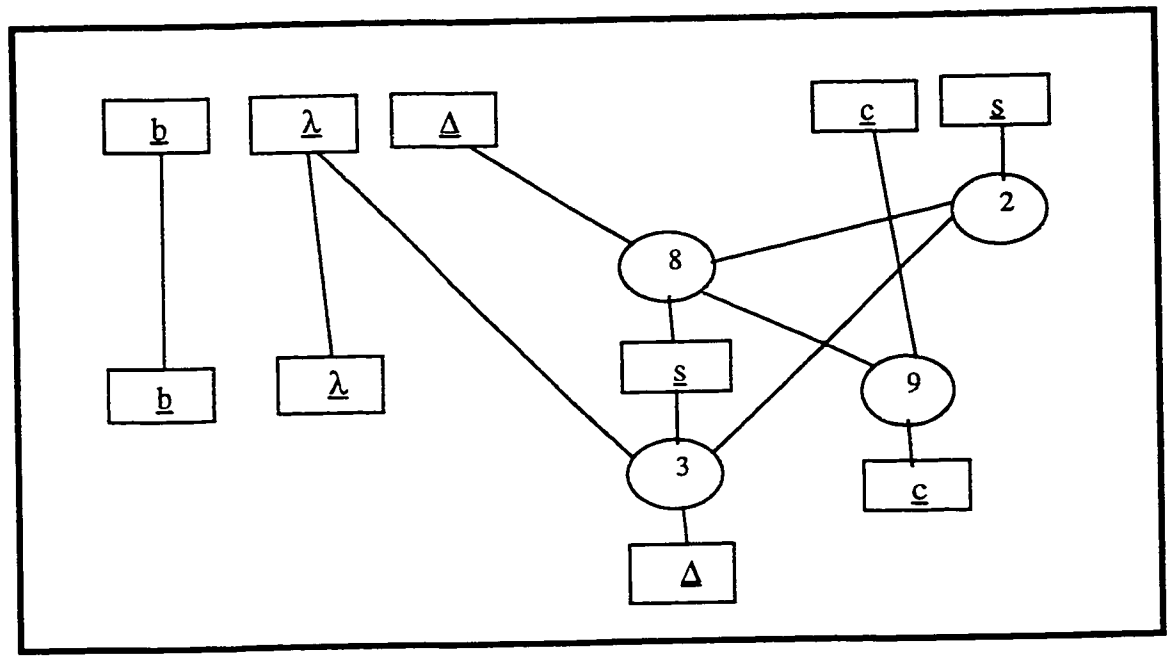


Fig. 3.15: Path σ after retiming.

The next chapter is devoted to implementing the hardware of the operations of the five possible paths. The implementation will be applied using the High-level Synthesis (HLS) approach in the OSTA tool. The techniques will be clustering functional units (FU) to form one unit that performs all the operations required on the data in one control-step. This control step will constitute the clock cycle of the decoder. The period of that clock will be dependent on the delay path of that cluster.

CHAPTER 4

HLS of Time-Domain RS(n,k) Decoders

The new micro- technology in VLSI design enhances any system in the hardware to result in three basic components: the data path, the control unit and the I/O unit. The data path contains the arithmetic and logic units where all the calculations are done by the operational units, the control unit, mainly represented by a finite state machine, controls the flow of data to the data path by defining the time and type of each operation together with the inputs and outputs. The I/O unit interacts by interfacing the input data to and from the data path.

4.1 The Structure of the RS(15,k) Decoder

Implementing the hardware of the RS*(n,k)* in the proposed design, requires that all the vectors being calculated to be $n \times m$ - bits wide. The input is m-bit symbols entering the decoder symbol by symbol and each nth symbol completes the $n \times m$ - bits symbols to constitute a codeword. To make the design faster and easier, all the codeword symbols are manipulated simultaneously which causes the connections in the data path to be $n \times m$ - bits wide. In our example, $m = 4$, and the code word length is then 15×4. Since the hardware does not support 60-bits width bus, the operational units are 32-bits wide built. Each of the operation presented in the previous graphs are constructed using two of the same type of functional unit. For example, the addition is built of two 32-bits wide adders and so is the multiplication. It is noticed that two of the adders constitute 64-bits operational units, and hence four most significant bits are added for symmetry purposes. The hardware cost might be increased slightly by adding these symmetry bits but the result is a more convenient design. This is due to the fact that having all the functional units to be of

the same width causes the design to be more consistent with higher throughput.

The structure of the RS(*n,k)* decoder will consist of the three units: In/Out unit, control unit and data path unit. The structure is shown in Fig.4.1..

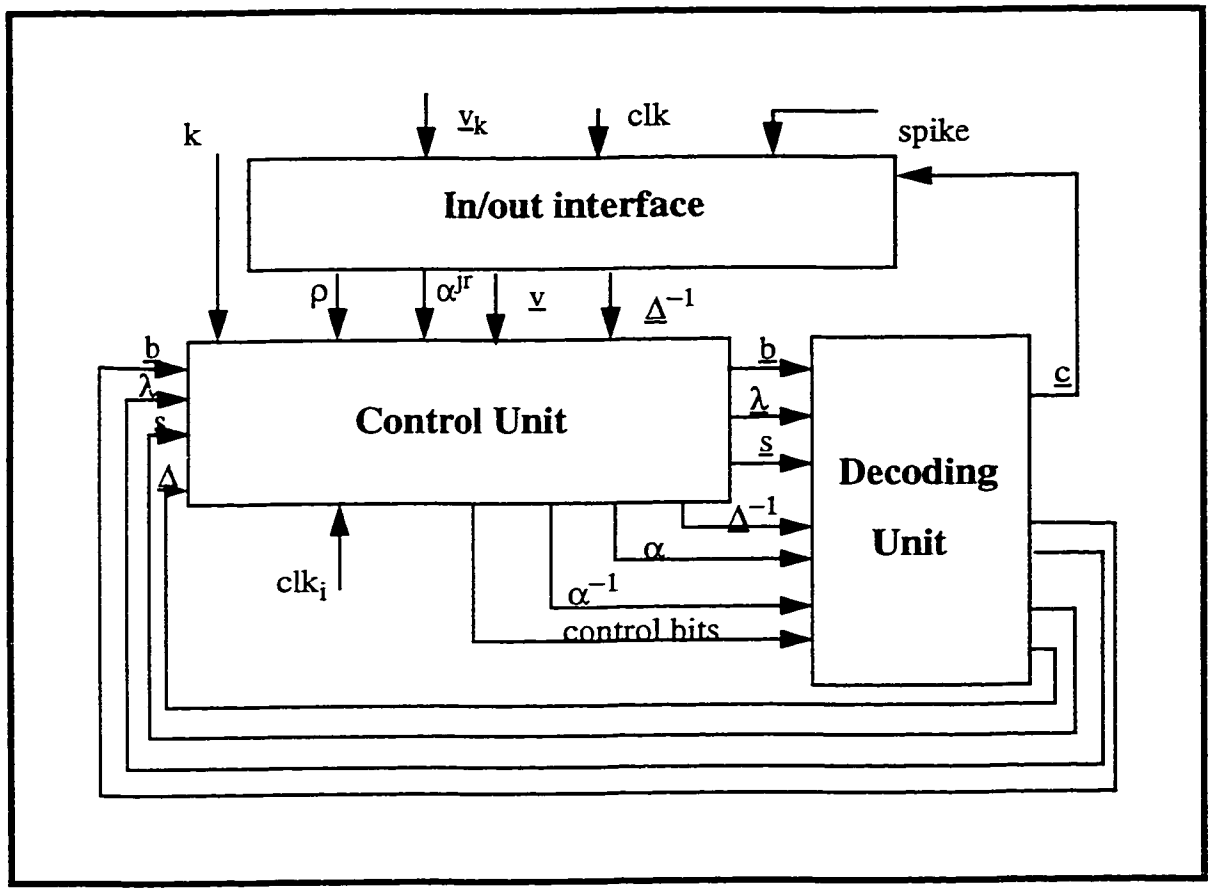


Fig. 4.1: The Block diagram of the RS (15,k) decoder

4.1.1 The Input/Output Unit

The In/Out unit has the following inputs: the received codeword vector $\underline{y}$ which comes as a series of symbols, the external clock, the length of the message *k* and the spike, the signal that shows an erasure. As the codeword is entered, if the spike is set to 1, an erasure is indicated to the relative symbol position. The position of each erasure is stored in

the vector d^{j_r}, which is to be used as initial values of the vector $\underline{\Delta}$ in the datapath or the decoding unit. The process of generating the input symbols, as one codeword to the decoding unit, is done using a finite state machine. If the reset signal is set to 1, the decoding process is initialized to start from the first symbol otherwise the next symbol is entered. Each time the symbol is entered, and an erasure is indicated, ρ will be incremented.In the last state, where the 15th symbol is entered, the process resets to the initial state where a new sequence of symbols is input. The state diagram of the input/output unit is shown in the figure below.

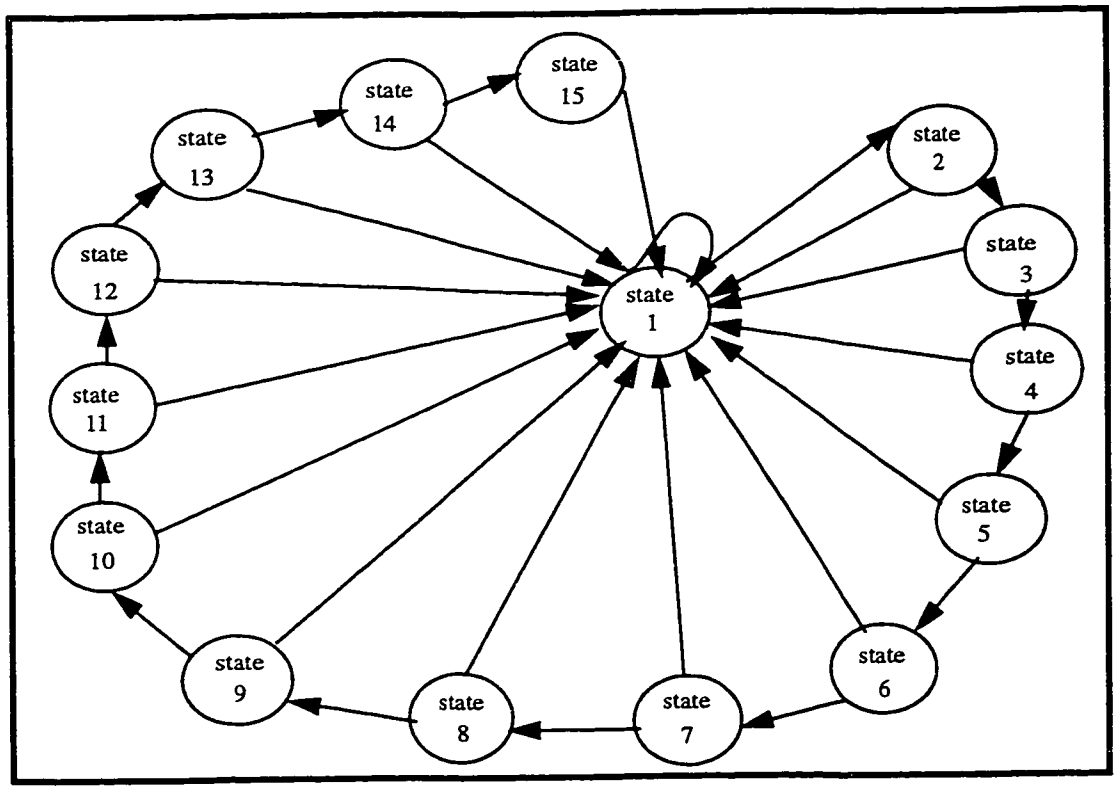


Fig. 4.2: The state diagram of the in/output unit.

The codeword is entered symbol by symbol at the rising edge of the clock. At the end of the *15th* cycle, the In/Out unit produces a 64-bits vector containing the whole codeword

that constitute the symbols entered with four redundant bits added. Meanwhile, the vectors $\underline{\lambda}$, $\underline{s}$, and $\underline{b}$ will be initialized so that the decoding process starts for this specific code word. The sequence of the operations is shown in Fig.4.3.

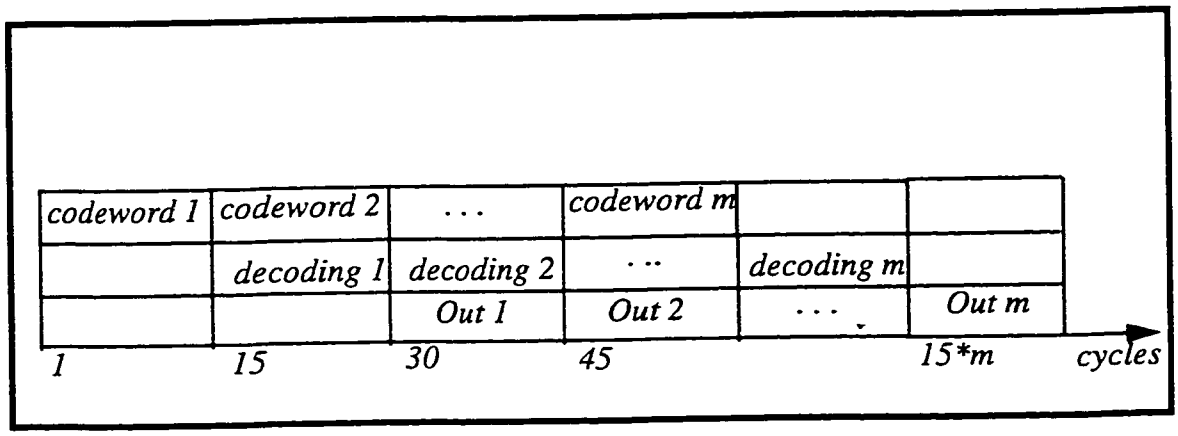


Fig. 4.3: Sequence of operations done in the decoder.

In conclusion, during the first n-cycles, the In/Out unit receives the codeword symbol by symbol, the second n-cycles the decoding unit decodes the first codeword while the In/Out receives the second codeword and so on. Therefore, while the in/out is receiving the *mth* codeword, the decoding unit is decoding the *(m-1)th* codeword and the *(m-2)th* codeword is ready for output.

4.1.2 The Decoding Unit

As mentioned in the previous chapter, the time domain decoding algorithm has five different possibilities of calculations in each iteration. Building the hardware for each possibility leads to an increase in the area, reduction in the throughput and hence the design is neither optimal nor efficient. The solution to this problem is to build a common hardware for all the possible operations with multiplexers added to the operational units. The multiplexers act as switches to control the data flow in and out of the common block. A state machine will be built to produce the control bits that supervise these multiplexers.

The calculations of the vectors are done for all the symbols simultaneously to improve the speed of the algorithm. The operations in the decoder, are: addition, multiplication, and compare-add ones. For example, the vector $\underline{s}$ is calculated as a codeword of 15 symbols simultaneously rather than calculating each symbol sequentially. This makes the calculation significantly faster with additional hardware that can be afforded.

For example, operation node (2) where

$$s_i = s_i \cdot \alpha^i \quad \forall \quad i = (0, \ldots n-1)$$

is evaluated by multiplying the whole vector $\underline{s}$ by the constant vector $\underline{\alpha}$ where $\underline{\alpha}$ is $[\alpha^0, \ldots \alpha^{14}]$ .This is the case for all the vectors.

The width of each vector is 60 bits (15×4 bits). Due to the limited capacity of implementing 64-bits bus, the vectors are divided into two 32-bit vectors where redundant 4-bit added are for symmetry purposes.

As mentioned in the previous section, the functional units are built to perform operations on 32-bit vectors. Hence each vector is divided into two 32-bit vectors. Galois field adders, are simply XOR gates with no carry bit to take care of. As shown in Fig.4.4, the basic adder, is composed of two 32-bit adders. Each 32-bit adder is composed of eight 4-bit adders, where the 4-bit adder is simply a 4-bit XOR gate as previously mentioned in chapter two.

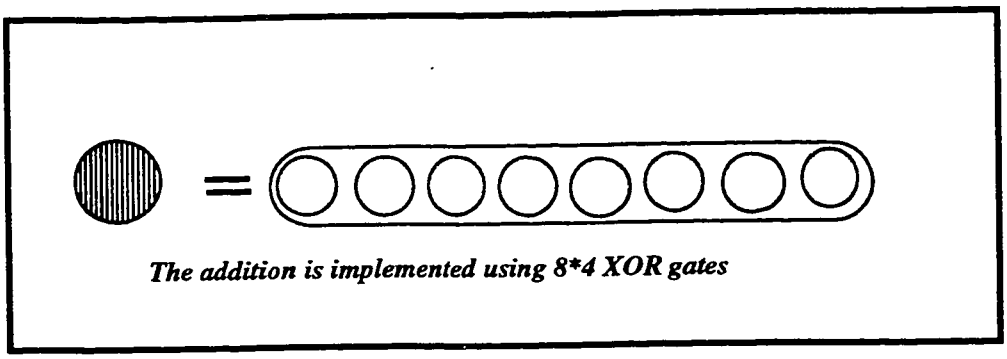


Fig. 4.4: Structure of 32-bits adder in the RS(15,k) decoder.

Similarly, the node for multiplication is composed of two 32-bit multipliers. Each multiplier is composed of eight 4-bit multipliers. The structure of the GF(2^4) multiplier was shown in Fig.3.2. The structure of the multiplication, or the multiplication node, in the RS(n,k) decoder algorithm is shown in Fig.4.5

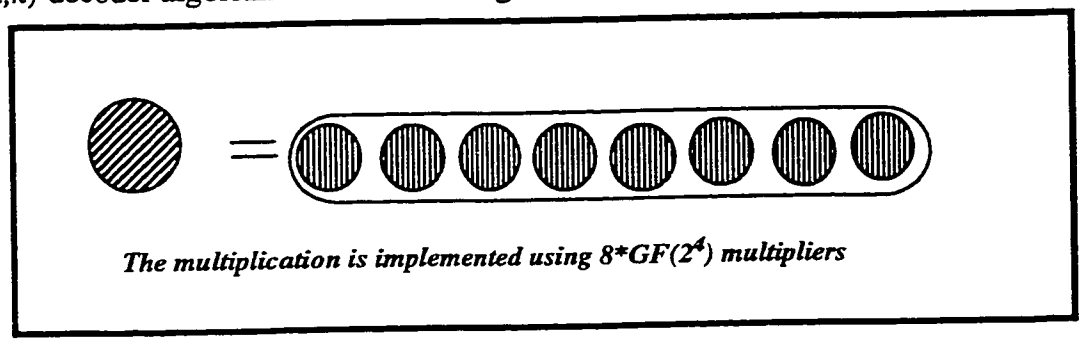


Fig. 4.5: Structure of the 32-bits multiplier in the RS(15,k) decoder.

The structure of the summer that sums up a multiplication calculating Δ, in equation (3.3), uses tree addition instead of the sequential addition to speed up the process from 15 addition delay to $\lceil \log_2 15 \rceil = 4$ additions. This transformation is applicable due to the associativity property of operations like the addition causing the delay of the whole addition to be smaller. The summer is shown in the figure below (Fig.4.6(b)). The summer node is shown in Fig.4.6..

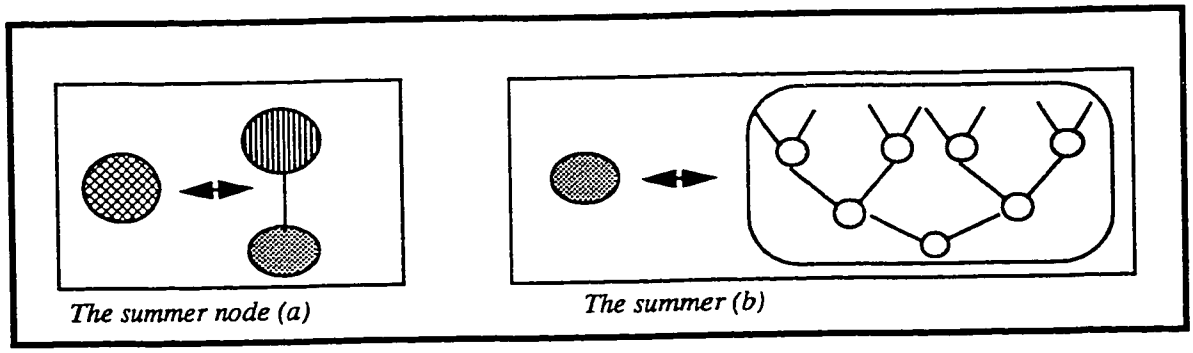


Fig. 4.6: Structure of the summer node.

As for the compare and add, a comparator is used to evaluate the product $\lambda_i s_i$ and

compared to zero. If zero, then do $c = v + s$, as shown in Fig.4.7.

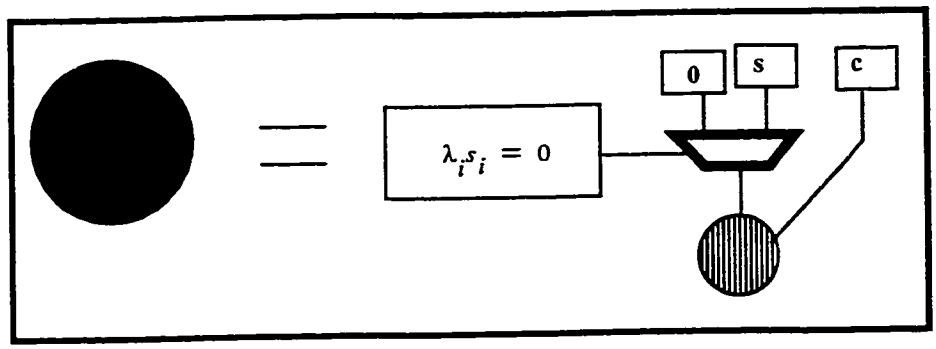


Fig. 4.7: Structure of compare and add in the RS(15,k) decoder.

In the proposed design, the functional units used are the adder, multiplier, summer and the compare-add elements presented above. Each of the operations in these different paths will be mapped to its related functional units.

There are six different paths, Fig.4.8 to Fig.4.13, which represent the functional units used as mentioned in the previous chapter. The total number of the operational units is: twenty-four multipliers, twelve adders and six summers.Building the hardware of each path results in larger area and hence more cost and less resource utilization. Also the throughput is very low because not all the paths are utilized in each iteration. Only one of them is busy while the other four are idle and hence we get 20% throughput leading to poor resource utilization. An important fact is that all the paths have some operations in common, which are two operations (2) and (3), (in Fig.3.3). These two operations constitute a major effect on the delay and area of the whole decoding process.

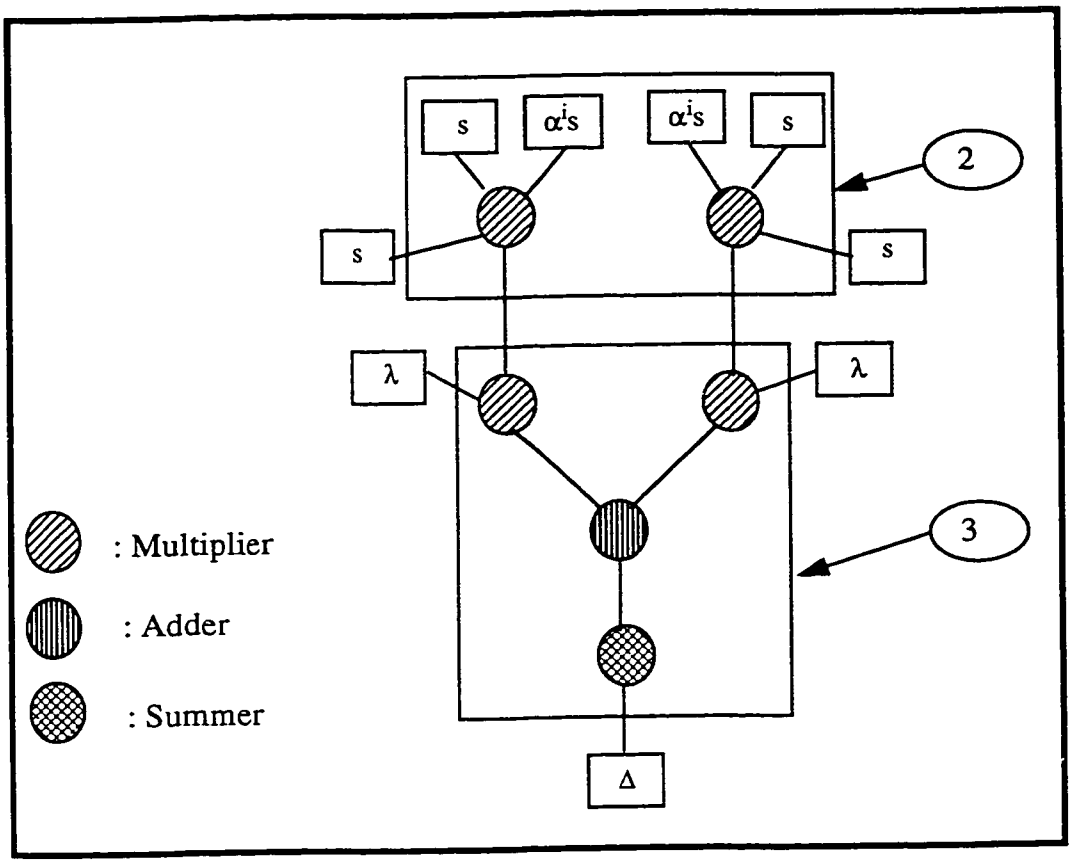


Fig. 4.8: The functional units used for state init.

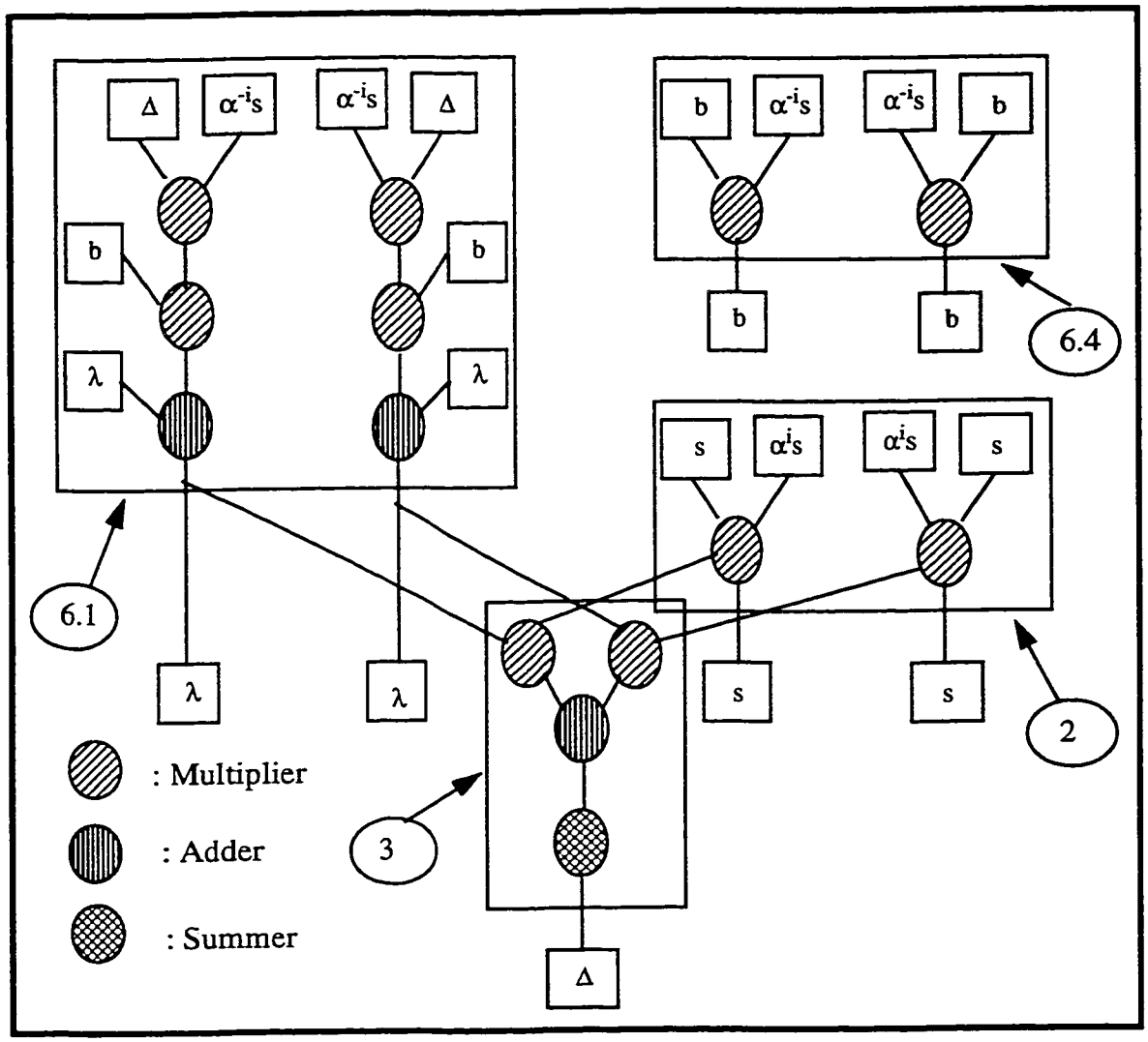


Fig. 4.9: Functional units used to perform operations in state α.

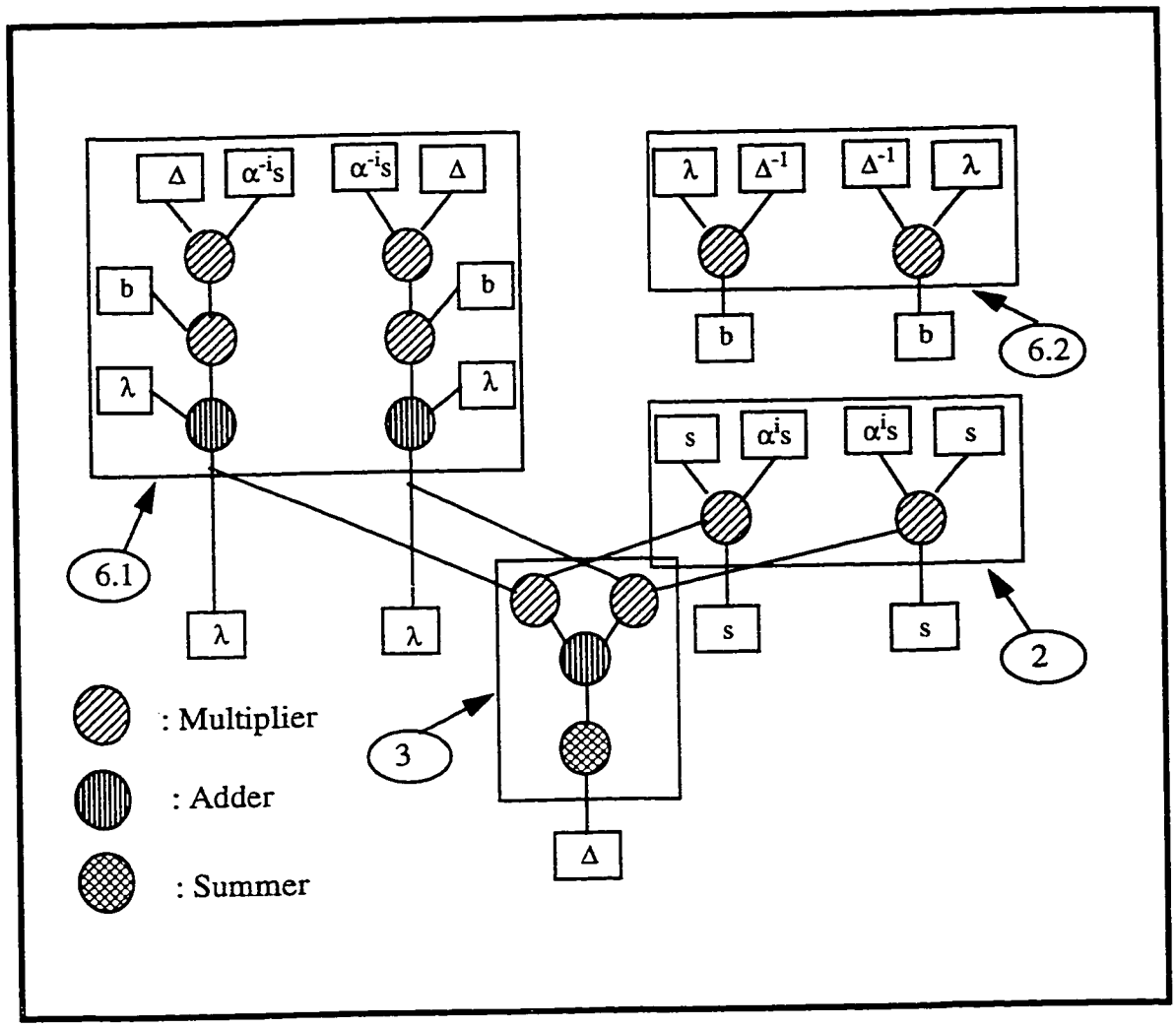


Fig. 4.10: Functional units used in state β

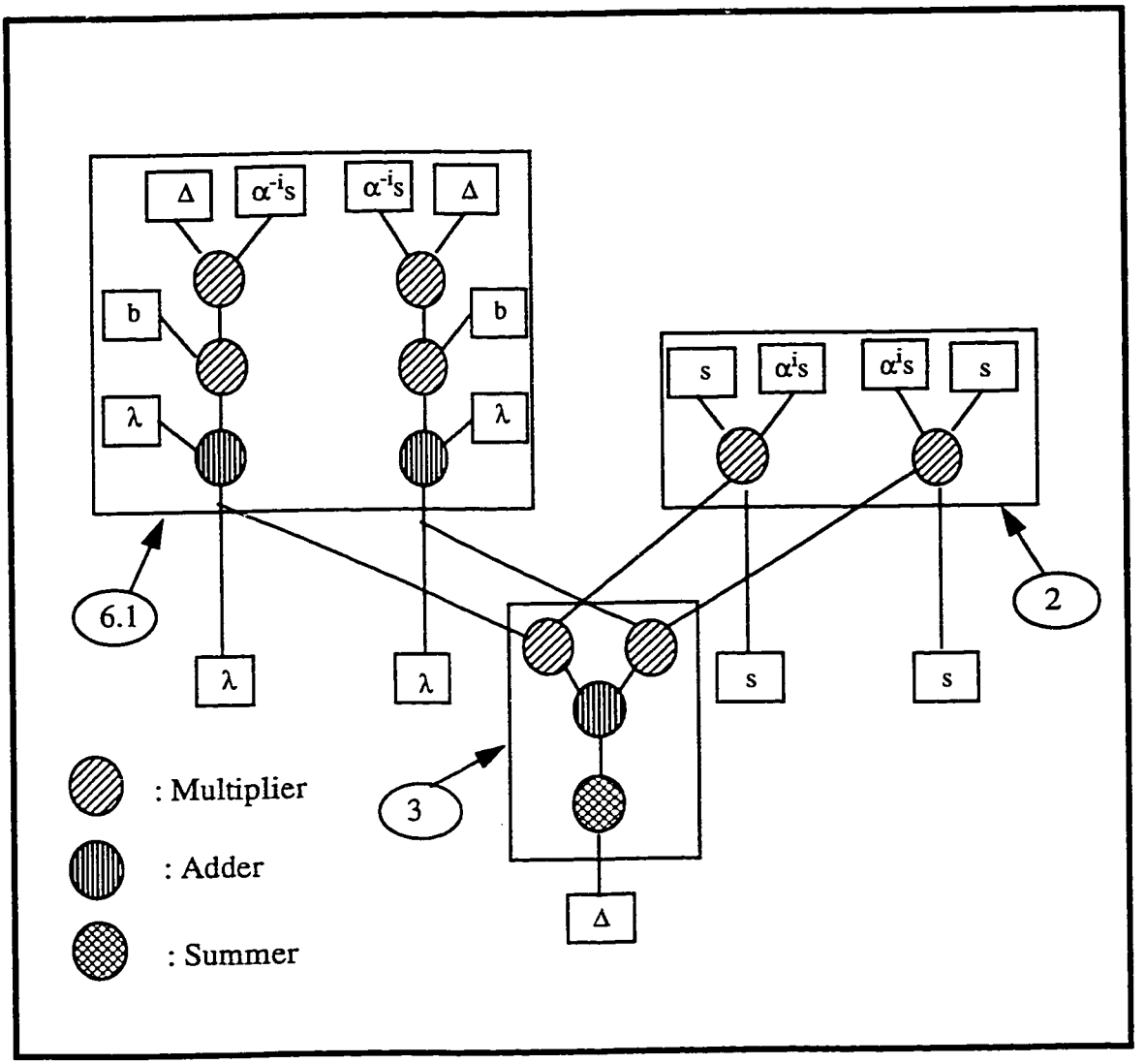


Fig. 4.11: Functional units used in state π

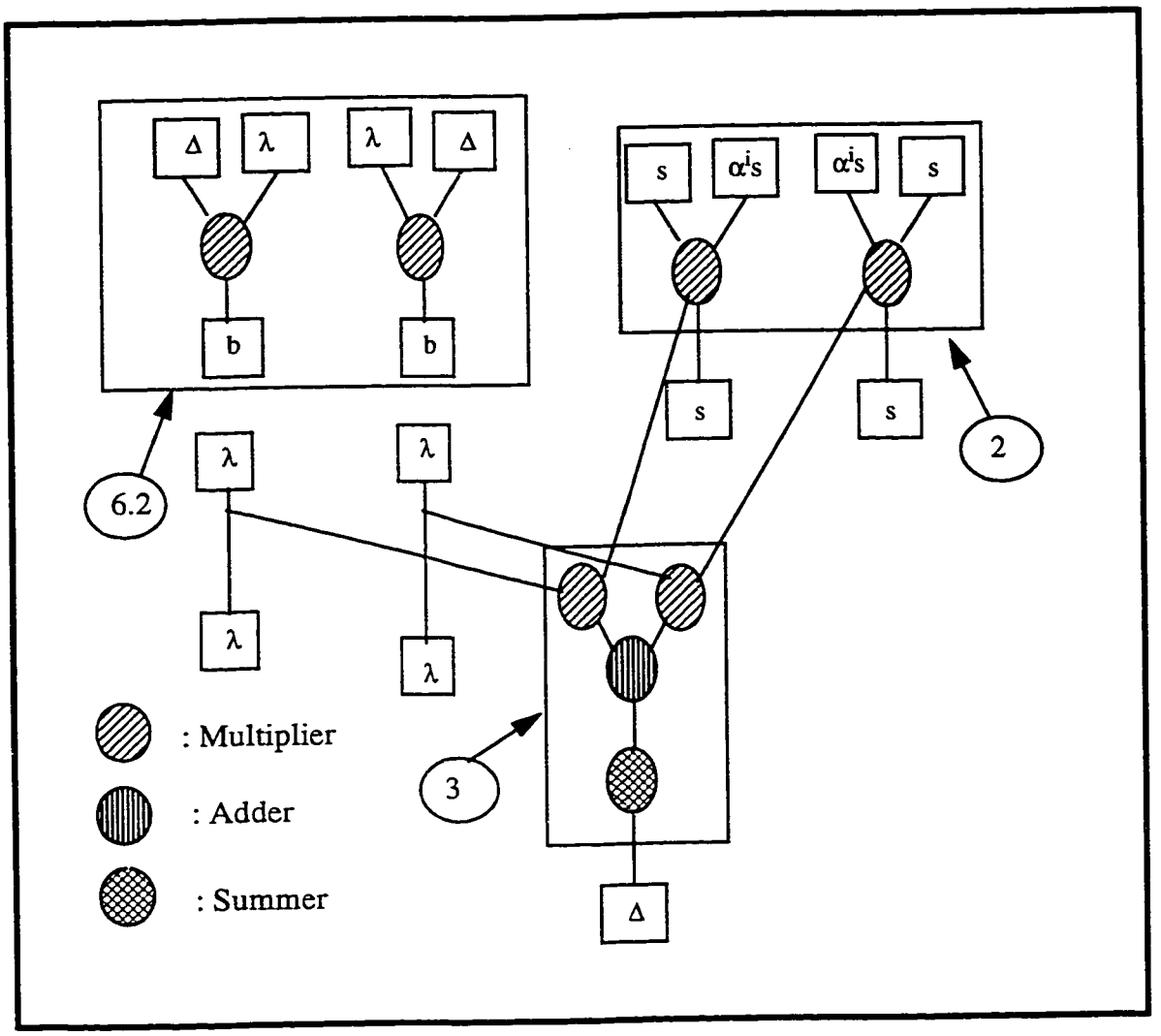


Fig. 4.12: Functional units used for operations in state Δ

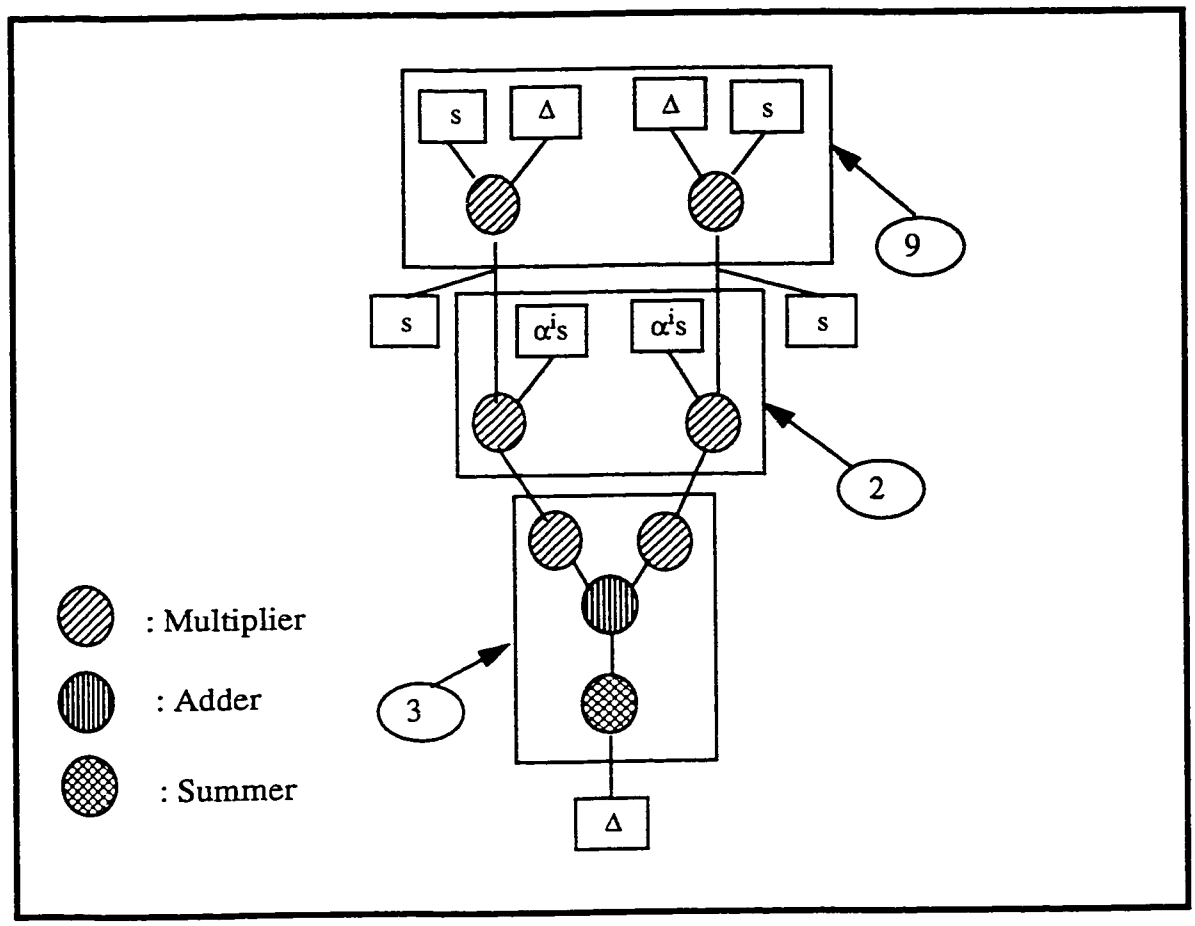


Fig. 4.13: Functional units in state σ.

The optimal solution for implementing an efficient design is to construct a common block that can perform the operations in all paths.This is applicable because the operations in each path are mutually exclusive.This reduces the area and increases the throughput of the resource. However, the price is adding multiplexers to control the flow of the data depending on the type of the path. In the common block multiplexers are employed and their cost are incomparable to implementing the five structures.

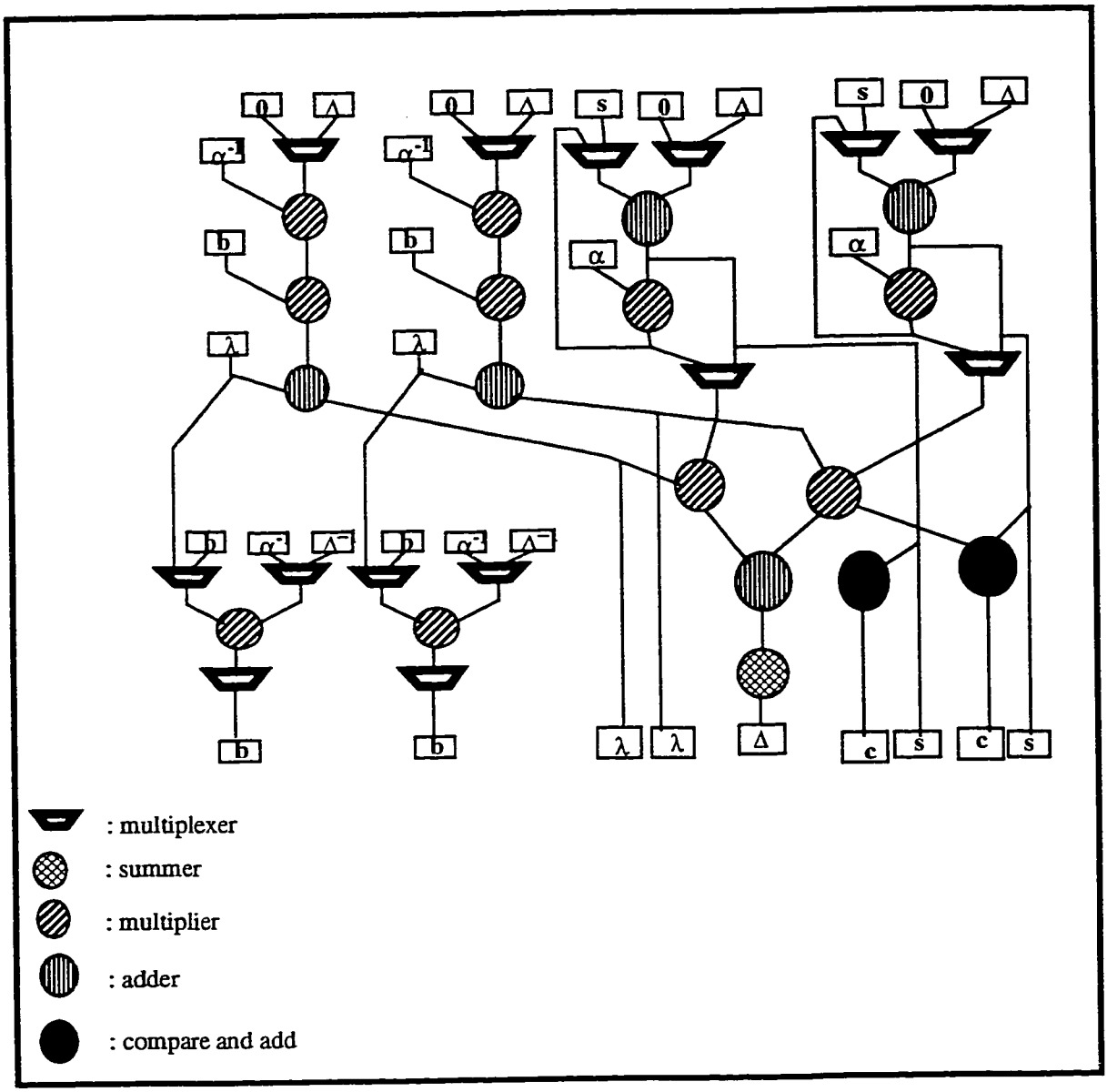


Fig. 4.14: The structure of the block common to all paths.

The selection of the inputs to those multiplexers is determined by the control unit, an additional cost, that makes the control unit more complicated. The common block is shown in the figure above (Fig.4.14) and contains all the functional units necessary to do the operations in all the paths. The data path is composed of 14 multiplexers, 2 compare

and adds, 10 multipliers, 5 adders and a summer. From Fig.4.14 it is clear that the resources are reduced to less than 50%, with a throughput of 100%.

4.1.3 The Control Unit

The control unit has the following as inputs: n, k, ρ, and Δ. These inputs are the parameters that determine the conditions. The output bits produced by the control unit are fed to the multiplexers in the decoding unit or the data path. These control unit monitors the flow of data. The following conditions are the ones that controls the choice of the path.

A: $r > \rho$, $\overline{A}$: $r \leq \rho$;

B: $r > n - k$, $\overline{B}$: $r \leq n - k$;

C: $\Delta = 0$, $\overline{C}$: $\Delta \neq 0$;

D: $r > 2L - \rho$, $\overline{D}$: $r \leq 2L - \rho$;

E: $r = n$

Fig.4.15 shows the decision being made in each iteration depending upon the conditions given above.

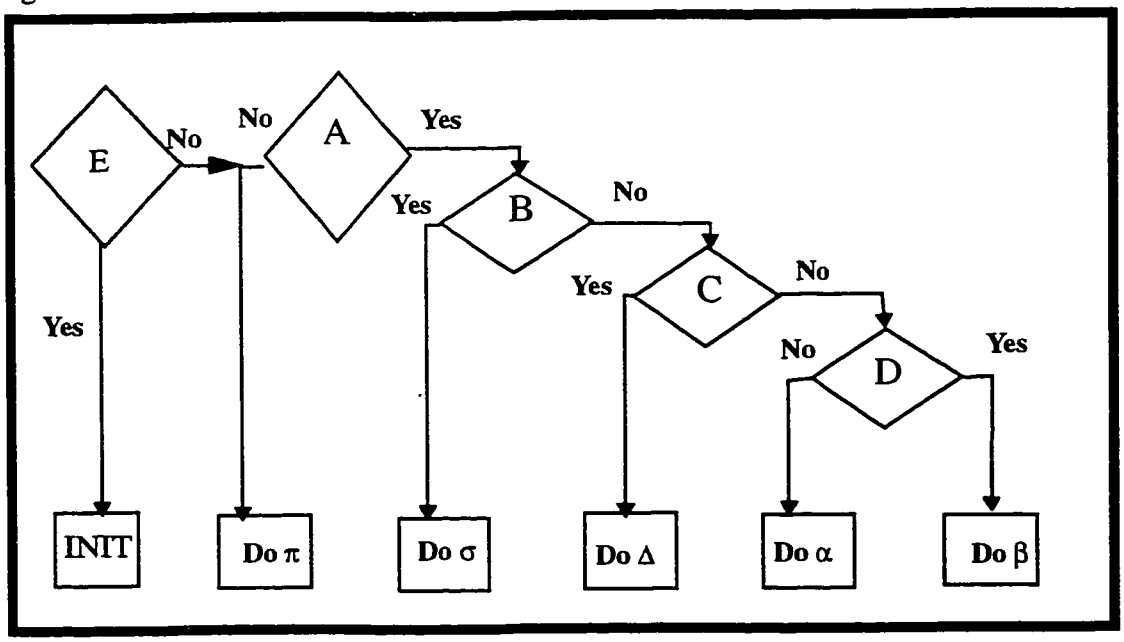


Fig. 4.15: The control flow graph of the decoding algorithm in each iteration.

In Fig.4.16, the control flow graph is mapped to a state diagram that shows the flow of control and output bits which controls the multiplexers in the data path.Each state in the state diagram, is named by the path it represents. The decoding algorithm has basically six states: state "INIT" is to initialize the vectors. The others represent the states in the other iterations.

The output bits in the state diagram are not shown for the sake of simplicity. These output bits are tabulated in Table 4.1. In each iteration r in the decoding algorithm, where $(r = 1, \ldots, n)$, the decoder is in one of the states and the control unit produces output bits to the multiplexers in the decoding unit.The control bits act as switches to the input data to the functional units in the decoding unit.

TABLE 4.1 The control bits selected by the control unit.

state	1	2	3	4	5	6	7	8	9	10	11	12	13	14
init	-	-	-	-	-	-	1	1	0	0	0	0	0	0
α	0	0	0	0	0	0	1	1	0	0	0	0	0	0
β	0	0	1	1	1	1	1	1	0	0	0	0	0	0
Δ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ	-	-	-	-	-	-	0	0	1	1	1	1	1	1
π	1	1	-	-	-	-	1	1	0	0	0	0	0	0

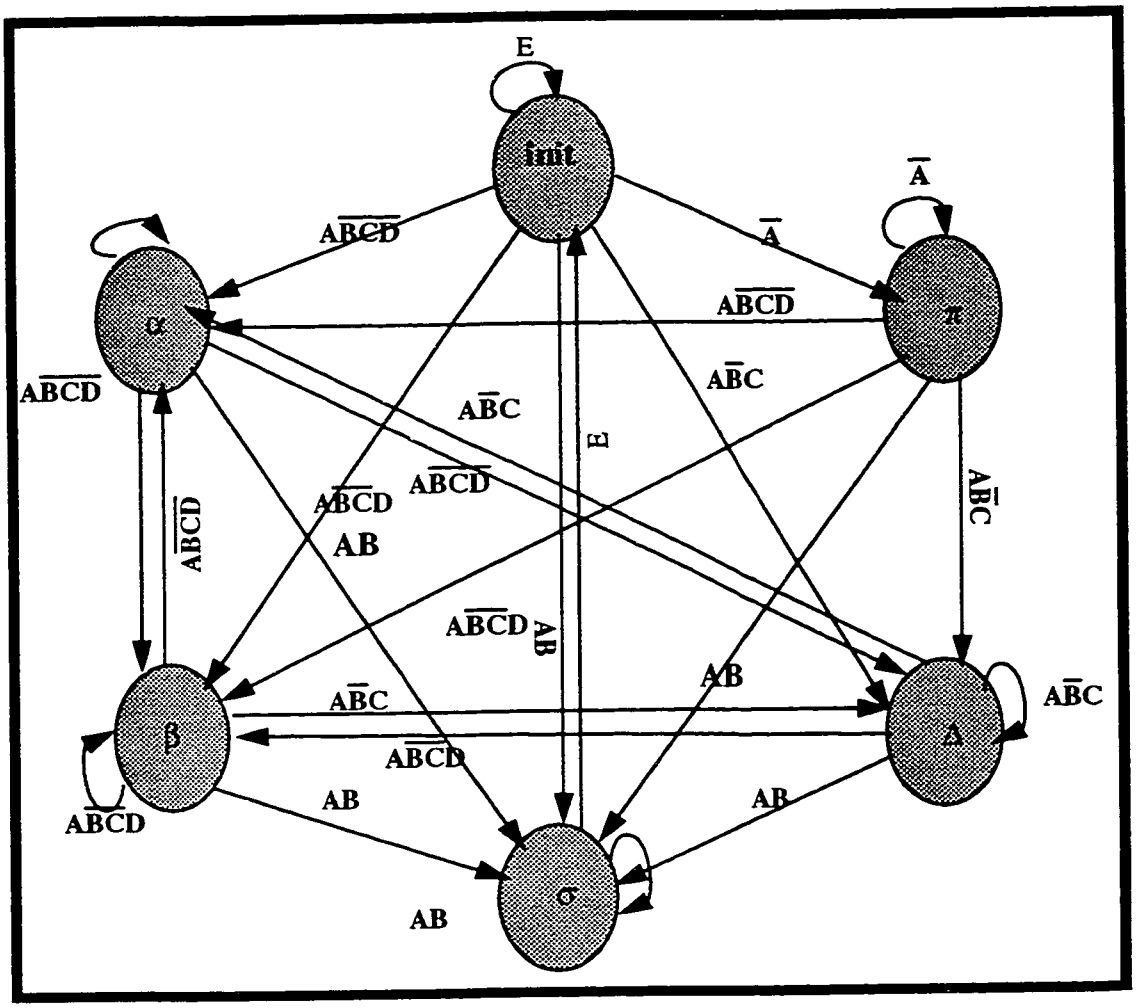


Fig. 4.16: State diagram of the decoding algorithm.

4.2 Analysis of the Implementation:

The proposed implementation has taken advantage of many high-level synthesis techniques. The approaches of High-level synthesis techniques mentioned before in chapter two, are applied here and are proved to produce an efficient and optimal design. These methods are: retiming, associativity, clustering and scheduling and binding.

- ***Retiming*** has been successfully used in several areas of design synthesis. It is

important to note that the richest source of delay in a program is the loop construct. Whenever a loop body uses information from a previous iteration, a delay is introduced. Therefore, retiming is applicable to instances which employ iteration or recursion as in this design. The application of retiming in this case is called software retiming in corresponding with the well known software pipelining transformation[10]. Retiming is also used here because the pipelining is ineffective namely in the optimization of recursive structures. Retiming has been exclusively used to reduce the critical path and is employed to the operations 2 and 8 that calculate Δ and $\underline{s}$ respectively. This shown in Fig.4.17.

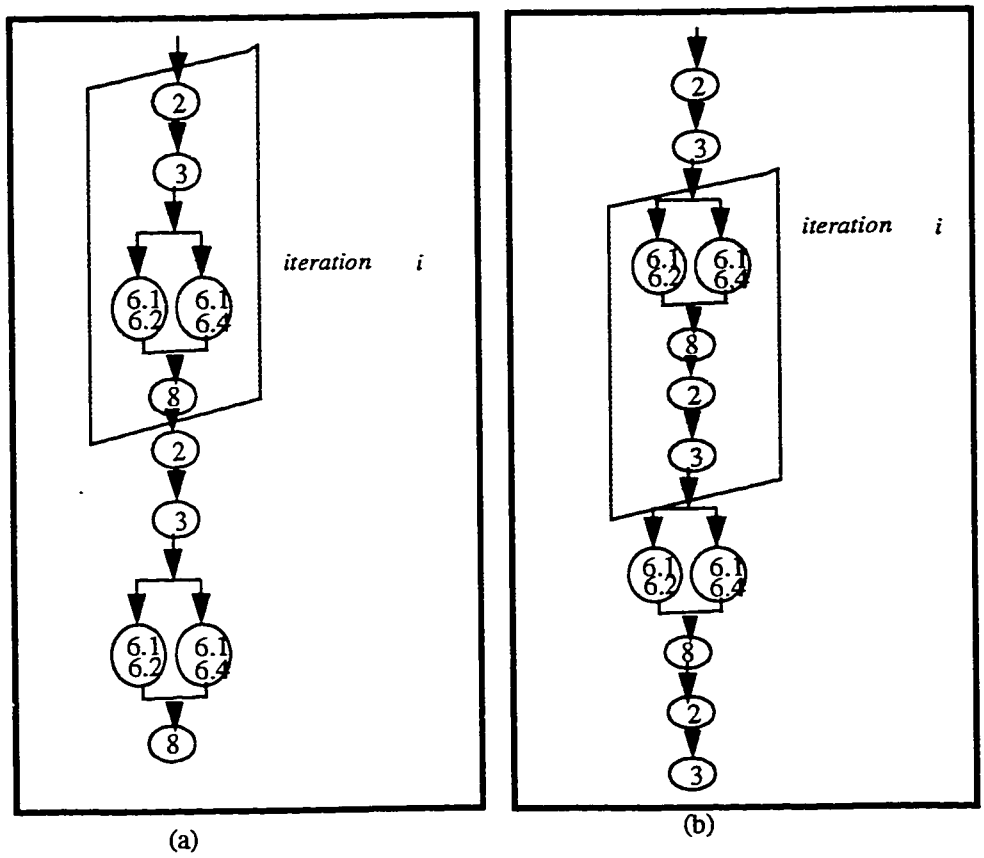


Fig. 4.17: Software Retiming: The loop in (a) is without retiming, the conditions are to be determined in the middle of the iteration. The loop in (b) is after retiming.

• ***Clustering*** is also used in the proposed implementation. Scheduling algorithms assign operations in a CDFG to control steps while preserving control and data dependencies between operations. Scheduling of operations into control steps can be viewed as the partitioning of operations into a set of clusters in which operations in the same cluster are executed in the same control step[15]. Hence in the proposed implementation, the common block is the cluster in our design. This is an optimal approach because partitioning the operations in smaller clusters proved to consume a huge amount of multiplexers that causes the design to have more delay and greater area.

• ***Associativity*** is employed in this implementation to calculate the vector Δ which is basically multiply and accumulate using the formula $\Delta = \sum_{i=0}^{14} \lambda_i \cdot s_i$. When the two vectors $\underline{\lambda}$ and $\underline{s}$ are multiplied, associativity namely "Tree Height Reduction" is used. This transformation causes the critical path of the operation to be reduced from the delay of fifteen adders to four adders delay. This is explained in the figure below.

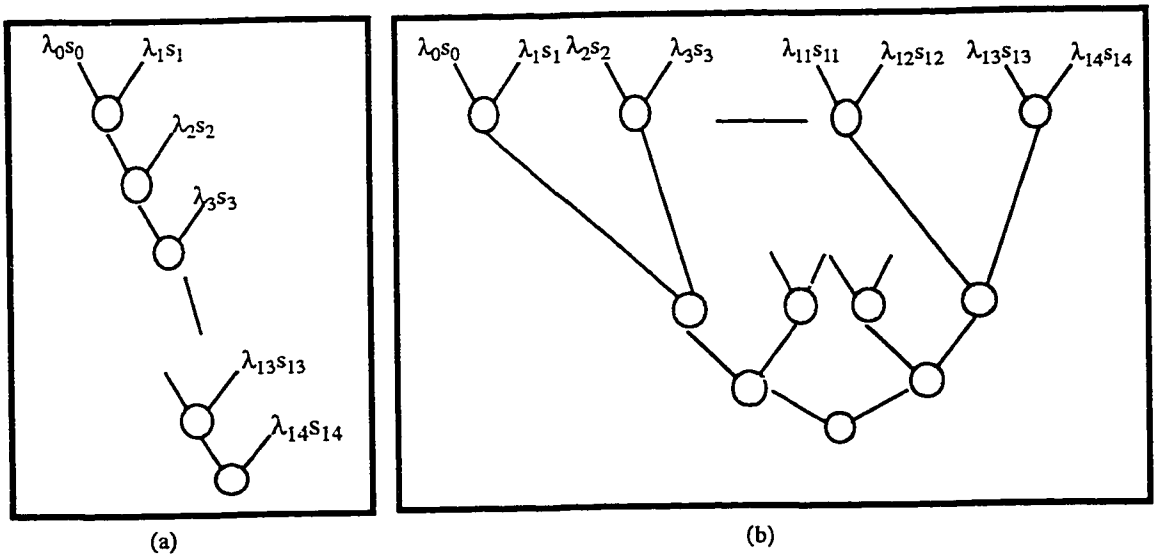


Fig. 4.18: Transformation from multiply and accumulate in (a) to Tree-height reduction in (b)

• ***Scheduling*** **and** ***Binding*** are performed when each operation in any node in the algorithm is bounded to the functional unit in the cluster formed.

CHAPTER 5

Implementation, Simulation and Synthesis: Procedure and Results

The versatile Reed-Solomon (n,k) decoder has the following features: The error and erasure correction capability can be programmed, i.e. the user can choose the length of the message, k. As cited before, the RS(n,k) decoder receives the codeword as a sequence of fifteen symbols. It is important to observe that the operations involved in the decoding algorithm are calculated symbol by symbol sequentially. Since the symbols have no data or control dependency between each other, they are updated parallel wise in all the vectors. This approach tends to result in a faster design with no internal clock as will be seen later.

In this chapter, for the case study, the RS(15,k) decoder where $m = 4$ will be implemented. The design of such decoder is implemented using VHDL. The process of synthesis and simulation will be described in the following sections.

5.1 The RS(n,k) Decoder Structure:

The decoder receives the data symbols continuously without any interruption. Each consecutive fifteen received symbols constitute one codeword. The decoder is designed to receive data symbols of the codeword and correct the previous codeword simultaneously. When the symbols of the decoder are entered and a spike is detected at a specific symbol, an erasure with its position is detected.

Fig.5.1, represents the schematic diagram of the decoder. At the rising edge of the clock, the symbols are entered one by one sequentially. The length of the message in the codeword is also entered and if a spike is detected, an erasure is indicated. Meanwhile, the

decoder is performing the iterations with every clock rise. When the whole codeword has been received, that is at the fifteenth clock cycle, the previous codeword would be corrected. The iterations of the decoding algorithm to correct the codeword is synchronized with the entrance of each symbol of the next codeword.

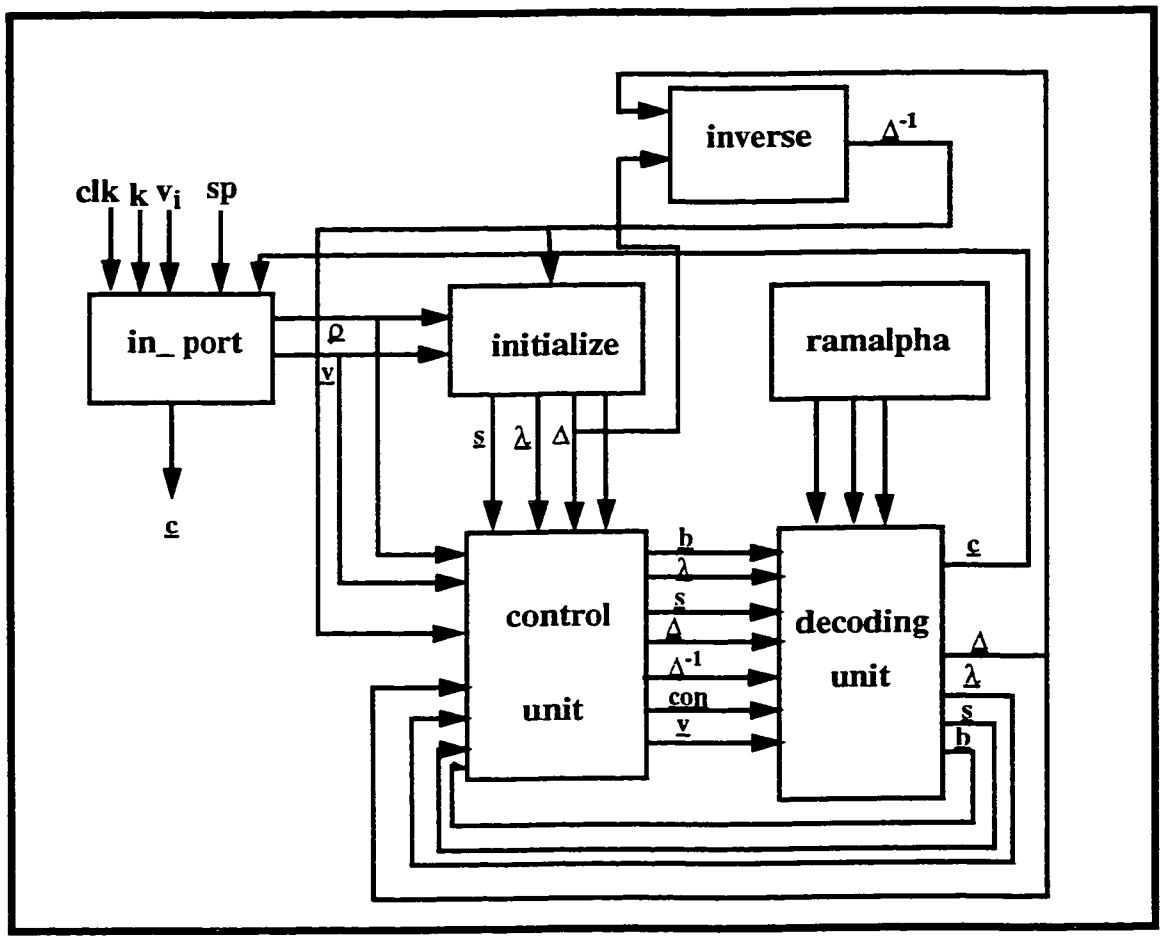


Fig. 5.1: The schematic diagram of the decoder.

At the arrival of the last symbol in the codeword, the vectors $\underline{s}$, $\underline{\lambda}$, Δ and $\underline{b}$ are initialized. Vectors $\underline{\lambda}$ and $\underline{b}$ are calculated for the first iteration while the calculation of Δ and $\underline{s}$ are done for the second iteration. This fact is due to the retiming procedure mentioned in

the previous chapter.

Two finite-state machines (FSM) are built in the proposed design. The first one is in the input port of the decoder and its function is to take care of the input data and produce a vector of 64-bits that contain one codeword symbol $\underline{v}$. The input port also detects the presence of erasures and stores them in a 60-bit vector, d^{j_r}. The second FSM is the control unit that monitors the flow of data into the muxes in the decoding unit.

The design of Fig.5.1 consists of five parts: in_port, ramalpha, inverse, initialize, control_unit and the decoding unit.

in_port: This is the interface of the decoder. It accepts the stream of symbols as input where each fifteen symbols constitute a codeword. The interface is a finite state machine with its clock taken as the clock of the decoder as a whole. With the beginning of each first cycle of the codeword, the codeword of the previous fifteen cycles are ready to be decoded. At the end of each fifteen cycle, the in_port would accept the next input codeword and have the previous one decoded and ready to be output simultaneously.

initialize: When the codeword is received, the vectors $\underline{\lambda}$, $\underline{b}$ are initialized. At the same time the vector $\underline{s}$ with the Δ of the second iteration would be calculated. As mentioned before, the procedure has all the conditions of the next iteration known where the path is chosen at this phase.

ramalpha: This is the storage of the constant pre-calculated values of α and α^{-1}.

inverse: This part gives the value of the inverse of α and α^{-1}. It is a pure combinational portion. It works as follows: given the value of α^i, its inverse is output.

decoding unit: This the part that does all the calculations in the decoder. It is the common block that contains all the paths in the algorithm. It contains the multiplexers that acts as data switches to the adders and multipliers. The multiplexers are controlled by the bits produced by the control unit.

control unit: It is the part of the design that controls the operations and the data flow into the functional units. It is a finite state machine that has its states representing the different paths of the decoding algorithm. According to the conditions reviewed in the algorithm of the decoder, the control unit sends the control bits to the multiplexers to monitor the operations and the flow of data.

As seen previously, the operations in the decoding algorithm are done symbol wise. For example, in the equation $s_i = s_i \cdot \alpha^i \quad \forall i = 0, \ldots, 14$, each symbol which is composed of 4-bits, is multiplied by α^i. Instead of implementing the multiplication of α, all the values of α^i are stored in a memory component. Also, the α^{-i} is stored in the same memory component.The values of α^i are defined according to function generator $g(x)$ of $GF(2^m)$. For the case of $m = 4$, the following table shows the values of α^i and α^{-i} in binary vector representation.

The memory component that contains these values is called ramalpa. It has those values stored. This method is efficient because an α-multiplier will be not needed to generate the powers of α.

TABLE 5.1 The values of α^i and α^{-i} in binary representation.

i	α^i	α^{-i}
0	0001	1001
1	0010	1101
2	0100	1111
3	1000	1110
4	0011	0111
5	0110	1010
6	1100	0101
7	1011	1011
8	0101	1100
9	1010	0110
10	0111	0011
11	1110	1000
12	1111	0100
13	1101	0010
14	1001	0001

5.2 Simulation and Synthesis:

The design process starts with the description of the circuit behavior in a high level design language such as VHDL in which the design of the decoder was coded. VHDL is a high level hardware design language that allows the user to specify the design's outputs in terms of the inputs over time using abstract data carriers and operators. It is similar to "C" or Pascal in which the programs are written using sequences of statements with expressions that assign values to variables. Yet, the semantics of a specific behavior in VHDL implies a hardware design to be built rather than to be executed on an existing machine.

Synopsys package was used as a CAD tool in the implementation phase. Synopsys is a synthesis and modelling tool that provides a high quality of results in IC area, performance - speed in particular - and power optimization. The Synopsys synthesis-based design methodology provides the technologies needed at the implementation levels appropriate for the design. Two tools of Synopsys were used: one for simulation and the other for synthesis.

VHDL System Simulator (VSS), used for simulation, provides performance throughout the design cycle through a unique, single simulation environment. It is offered by Synopsys and it is considered to be efficient when using VHDL, as a description language, for verifying the performance and functionality of an ASIC. It provides behavioral synthesis policy checking and powerful debugging environment.

Once the whole decoder was assembled, VSS was used to simulate the design either interactively or by writing the script code. The first method was adopted in checking the functionality of each of the small modules. If there was an error in the behavior, the design code was checked again and corrected to meet the specification. The second approach was a script written specifying the values of the input. It was used when the small modules were assembled to form the decoder.

To check the validity of the output, a C program was written to generate the test vectors. The same test vectors were applied in the simulation process of the hardware part. At the end, the errors were corrected and the simulation was complete. In the simulation, all the possible cases were examined to check wether the decoder can handle all the oddities possible. The cycle period could be specified at any value, since the delay of the critical path in each iteration has not yet physically been determined. Once the design was checked for errors and corrected, the next phase was the synthesis phase. Appendix B.1 shows the VHDL of the design and Appendix B.2 shows the simulation results.

Design Analyzer is a powerful analysis tool that gives the user synthesis control, design management and design analysis in a graphical environment. Design Analyzer enables the user to perform various design set-up and analysis functions, as well as viewing and interacting with the synthesized schematic. Synopsys provides an interactive synthesis procedure as well as indirectly using the script file to choose the criteria for the design. A script file was written trying again to optimize the design from smaller to larger modules.

Two approaches for optimization were selected: speed and area. Each sub-module was synthesized separately and the goal was to obtain a fast design. The emphasis in the synthesis process was to make a fast design, as an optimization criteria. This is because the present features of VLSI technologies available in the market nowadays allow to concern less about the number of gates required to build the design.

Reducing the area causes the design to be slower and vice versa. In order to obtain a design with an optimal area, one of the constraints for synthesis is to set the minimum limit of area to zero. Other constraint is to set the path delay to the minimum. This is possible if the design is purely combinational. If the design is sequential, i.e it has a finite state machine; the path delay, mainly the critical path, can be reduced through minimizing the clock period.

When choosing speed as an optimization criteria, several trial and error methods were tried. The combinational part of the design: the adders, the multiplexers and the multipliers were synthesized by setting the path delay from the input to the output to be as small as possible. When all the sub-modules were assembled to constitute the data path, an estimate of the delay of the path was reached to set the clock period for the decoder. As for the non combinational part mainly the control unit, which was the most complicated part, a value of the clock period was set so that to reduce the time as much as possible.

When assigning the clock period of the decoder, the delay of the data path was added to the delay of the control unit to obtain the delay of the decoder as a whole. The clock period of the decoder was set to a minimum when synthesizing the design of each unit. The delay of the output of each block was set to be within the clock period of the decoder.- Trials and errors were observed during the synthesis procedure and the slack of the timing in some output ports were violated.

5.3 Technology used for synthesis:

ASIC designers today are faced with a common dilemma. On the one hand, product specifications dictate their designs must consume minimum amounts of power to meet cost, reliability, and energy efficiency goals. On the other hand, functional specifications imply that the ASIC's must integrate more circuitry and run faster than ever to implement the desired functionality, which drives up overall power consumption.

At the present time, there are many different circuit families. A partial list would include the TTL (Transistor-Transistor Logic), MOS (Metal-Oxide-Semiconductor), and CMOS (Complementary MOS) families. The technology used for synthesis was restricted on CMOS and a combination of CMOS and BJT (Bipolar-Junction Transistor) families.

CMOS is currently the most popular digital circuit technology. CMOS logic circuits are available in as standard SSI and MSI packages for use in conventional digital system design. CMOS is also used in the design of general purpose VLSI circuits such as memory and microprocessors. CMOS is an inherently low power circuit technology with the capability of providing a lower power-delay product comparable in design rules to NMOS and PMOS technologies. Another advantage for CMOS is that there is no direct path between the voltage source and the ground for any combination of inputs. This is the basis for the low static power dissipation in CMOS[19].

Another VLSI circuit technology that is becoming increasingly popular is BiCMOS. As its name implies, BiCMOS technology combines Bipolar and CMOS circuit on the same IC chip. Therefore, circuitry retain the low-power, high input-impedance, and wide noise margins of CMOS and the high current-driving capability and high speed operation of Bipolar transistors. The result is a circuit technology that is capable of implementing very dense, high-speed, and low-power digital integrated circuit[19].

The two technologies mentioned above, CMOS and BiCMOS, were used in the libraries provided by the VLSI laboratory. Each library was set separately so that the synthesis was done for each of the two technologies. Two CMOS technologies were used in the synthesis: Nortel and Mitel technology. Table 5.2 shows the number of gates for specific parts of the decoder. The first row shows the number of gates in the decoder employing the three technologies. The next three row shows the gate number of the basic functional units used in the decoder. What is called 'decoding' is the data path of the decoder or the decoding unit. It is noticed that the BiCMOS technology required more number of gates than the others. The number of gates were calculated after obtaining the data results in terms of cell area. Each technology has its cell area linearly related to the number of gates per cell.,

TABLE 5.2 No. of gates in the design using the three technologies

Design	Bicmos	Mitel15	cmos4s
decoder	26k	15.8k	18.1k
data path	13k	8.7k	9.7k
I/O	7k	4.3k	4.8k
control	6k	2.8k	3.6k
summer	154	76	92
adder	346	85	128
multiplier	892	512	602

Table 5.3 displays the results of the speed of the decoder, in micro seconds, when employing the three libraries. It is obvious that the BiCMOS is faster than the CMOS and this is because of the reasons mentioned previously.

TABLE 5.3 Timing delay results of the design using the three technologies

Design	BiCMOS	Mitel15	CMOS4s
decoder	26.04 μs	39.33 μs	30.08 μs
data path	15.17 μs	25.94 μs	16.8 μs
I/O	7.8 μs	9.4 μs	8.2 μs
control	10.5 μs	15.6 μs	12.3 μs
summer	2.15 μs	3.78 μs	1.51 μs
adder	0.32 μs	1.13 μs	0.62 μs
multiplier	2.63 μs	4.17 μs	2.77 μs

In order to translate the numbers in the table above as the speed in MBits/s, the following formula is used:

$$throughput = \frac{n \times m}{the\ delay\ of\ the\ path \times number\ of\ iteration} \tag{5.1}$$

Hence from the equation above, (5.1), we tabulate the results obtained in the previous table and calculate the speed of the decoder in Mbits/sec:

As a result, we obtain new tabulated data that is shown below.

TABLE 5.4 The throughput of the decoder in Mbits/sec

technology employed	speed in Mbits/sec
Mitel15	101.7
BiCMOS	153.8
CMOS4S	133

The above results are displayed for $m = 4$ and $n = 2^4 - 1 = 15$. The same

approach can be applied to different values of m for the versatile RS(n,k) decoders where $m = 5, 6, 7, and\ 8$. Table 5.5 shows the estimated speed and area of the decoders of different values of m.

TABLE 5.5 Area of decoder with different values of m

m	adder	multiplier	summer	data path	decoder
4	128	602	92	9.7k	18.1k
5	400	1632	775	23.6k	48k
6	1152	4608	2268	64.8k	130k
7	3136	12096	6223	144.7k	288k
8	8192	30720	14400	420k	850k

It can been clearly deduced that the area increases significantly as m increases. This is due to the fact that the number of bits in each vector manipulated by the functional unit increases exponentially.

TABLE 5.6 Delay of decoder with different values of m in μs.

m	adder	multiplier	summer	data path	decoder
4	0.62	2.77	1.51	16.8	30.08
5	0.62	5	3.72	21.96	36.6
6	0.62	6.24	4.34	26.8	44.7
7	0.62	7.48	4.96	31.64	52.73
8	0.62	8.72	5.58	36.48	60.8

Using the equation (5.1), the results in the above table can be re-tabulated to show the throughput of the RS decoder for the various values of m.

TABLE 5.7 The throughput of the RS(n,k) decoder with different values of m.

m	throughput in Mbits/sec
4	133
5	136
6	134
7	132
8	131

It is deduced, from Table 5.7, that the throughput of the decoder is approximately the same for the various values of m. This is due to the fact that there is a linear relationship between the throughput and the number of bits per symbol, m. The formula in (5.1) can be changed to the following one, because the number of iterations in the decoding algorithm equals the number of symbols per code word n.

$$throughput = \frac{m}{delay\ path\ of\ the\ decoder} \tag{5.2}$$

CHAPTER 6

Summary and Conclusion

This thesis presents High-level Synthesis (HLS) techniques applied in the design of circuits dealing with finite field elements such as Galois Field elements. The HLS approach constitutes an important phase in the VLSI design of any digital circuit. High-level synthesis starts with an abstract behavioral specification of a digital system and finds a register-transfer level structure that realizes the given behavior. The HLS as implied by its name deals with the design in a very high level of abstraction. HLS has the goal of optimizing the design given its algorithm regardless of the type of data being processed. This allows the diversity in the types of designs being synthesized. Our case study is applying HLS methodologies on circuits that operates on Galois field elements specifically the versatile Reed-Solomon RS(n,k) decoders.

First, the following tasks of HLS were discussed including scheduling, binding, Control-Data-Flow-Graphs (CDFG), loop unfolding, transformation and clustering. It was shown how the synthesis tasks can be decomposed into a number of distinct but not independent subtasks. Then an automatization of Integer Linear Programming (ILP) formulation for scheduling the operations, as well as functional units and register binding were discussed. The ILP formulation was previously applied to the second order elliptic filter and now the tools could be used for performing certain subtasks of HLS for a general circuitry of any kind having basic multiplication, addition and delay as functional operations.

The properties of GF(2^m)and the structure of their adders and multipliers were studied. The algorithm of versatile RS(n,k) decoder, proposed in [7], was chosen as case study because of its structure and modularity. The algorithm of the decoder was analyzed in order to achieve an optimal design. To obtain the goal of an optimal design, several meth-

ods of High-level Synthesis were applied in the synthesis techniques. The decoding algorithm was modified accordingly. The behavioral synthesis of the proposed decoder algorithm was commonly achieved by dividing the task into a data path and a control path design.

The RS decoding algorithm contains conditional branching in each iteration. This made the design process complicated because the control unit could hardly be separated from the data-path unit as desired. These present conditions might have lead to larger hardware area due to the diversity of data operations in each iteration. However, several synthesis techniques applied to overcome the inconveniences mentioned above. A common block that could perform all the operations in any of the conditions was obtained and was considered to be the data path unit. A control unit was built to organize the flow of the data to and from the data path unit. This was made possible by applying some synthesis subtasks such as clustering, retiming and loop unfolding.

The design was coded using VHDL as a hardware description language and it was simulated using Synopsys CAD tool. For the design synthesis, Design Analyzer of Synopsys was used to optimize the speed of the decoder. The result of the design process revealed interesting figure numbered in terms of speed. The speed of the decoder using the adopted approach was significant with great amount of variations than the previous methods. As mentioned before, RS(n,k) decoders use GF(2^m) elements as their data. Studies were taken on different values of m, where $m = 5, 6, 7 \ and \ 8$. The area and speed of the decoder, under those values of m, were simulated and estimated using the same modified algorithm. The synthesis approach would be the same but as m increases, the area of the decoder increases significantly mainly, exponentially. However, larger area do not signify a problem in the advances in VLSI technology. It has been proved that results reveal a significantly fast decoder for all values of m.

REFERENCES

[1] Lin, Castello, "*Error Control Coding: Fundamentals and Applications*", Prentice Hall, New Jersey, Englewood Cliffs, 1983.

[2] C. Hwang, J.H. Lee, Y.C. Hsu, "A formal Approach to the scheduling Problem in High level Synthesis", *IEEE Trans on CAD*, Vol. 10, No. 4, pp 464-475, 1991.

[3] R. E. Blahut, *Theory and Practice of Error Control Codes"*, Reading, Mass. Addison-Wesley, MA, 1983.

[4] Y. Shayan, T. Le-Ngoc and V. Bhargava, "A Versatile Time-Domain Reed-Solomon Decoder", *IEEE JSAC*, Vol 8, No. 8, October 1990.

[5] S.A. Vastone,P.C. VanOorshot, *An Introduction to Error Correcting Codes, With Applications*. Kluwer Academic Publishers, Boston, 1989.

[6] E. R. Berlekamp, "Bit-Serial Reed-Solomon Encoders"*IEEE Trans On Information Theory*, Vol 28, No. 6, 1982. pp 869-874.

[7] Y. R. Shayan, *Versatile Reed-Solomon Decoders*, Ph.D. Thesis, Concordia University, Montreal, 1990.

[8] B. S. Haroun and M. Elmasry, "Synthesis of Multiple Bus Architectures for DSP Applications", Chapter in "*VLSI Design Methodologies for DSP Architectures & Applications"* Ed. M. Bayoumi, Kluwer, Boston, 1992.s

[9] M. McFarland, A. Parker and R. Composano, "The High-Level Synthesis of Digital Systems", *Proceedings of the IEEE*, Vol 78, No. 2, pp 301-317,February 1990.

[10] M. Potkojnak, J. Rabeay, "Maximally Fast and Arbitrarily Fast Implementation of Linear Computations", *IEEE Trans on CAD*, Vol 11, pp 304-308, 1992.

[11] M. Potkonjak, "Optimizing Resource Utilization Using Transformations", *IEEE Trans. on CAD of IC and Systems*, Vol. 13, No.3, pp 277-291 March 1994

[12] P. Paulin and J. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's", *IEEE Trans on CAD*, Vol. 8, No. 6, June 1989.

[13] B. Haroun and B. Sajjadi, "ILP Synthesis of Signal processing Architecture with minimum Structural Complexity", *CCIC*, May 1994, pp11.2.1-11.2.4.

[14] S. Malik, M. Sentovich, R.K. Bryton and A. Sangiovannivincentilli, "Retiming and Resynthesis: Optimizing sequential networks with combinational technique", *IEEE Trans. on CAD*, Vol 10 pp 74-84, 19941.

[15] B. Sajjadi, *Architectural Synthesis for FPGA Based Signal Processing Systems*, M. A. Sc thesis, Concordia University, Montreal, 1996.

[16] D. Gajski, N. Dutt, A. Wu and S. Lin, *High-level Synthesis, Introduction to Chip and System Design*, Kluwer, Boston, 1992.

[17] H. Mecha et al, "A Method for Area Estimation of Data-Path in High-Level Synthesis", *IEEE Trans. in CAD of Integrated Circuits and Systems*, Vol 15, No. 2, February 1996.

[18] M. Rim and R. Jain, "Valid Transformations: A New Class of Loop Transformations for High-level Synthesis and Pipelined Scheduling Applications", *IEEE Trans on Parallel and Distributed Systems*, Vol 7 No. 4 April 1996, pp 399-400.

[19] A. Sedra,K. C. Smith, *Microelectronic Circuits*, Saunders College Publishing, NY, 1990.

[20] J.Yang et al, "Scheduling and Control Generation with Environmental Constraints Based on Automata Representations", *IEEE Trans on CAD of Integrated Circuits and Systems*, Vol 15, No. 2, February 1996 pp 166-176.

[21] J.Sung and R. Redinbo, "Algorithm-Based Fault Tolerant Synthesis for Linear Operations", *IEEE Trans. on Computers*, Vol 45, No. 4, April 1996 pp 425-438.

[22] C. Gebotys and M. Elmasry, "Global Optimization Approach for Architectural

Synthesis", *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 12, No. 9, September 1993, pp 1266-1268.

[23] S. Wei and C. Wei, "High-Speed Decoder of Reed-Solomon Codes", *IEEE Trans on Communications*, Vol 41, No. 11, November 1993, pp 1588-1592.

[24] S. H. Jamali, T. Le-Ngoc, "*Coded Modulation Techniques for Fading Channels*", KAP, Montreal, 1994.

[25] S. Wicker, V. Bhargava, "*Reed-Solomon Codes and Their Applications*", pp 61-105, IEEE Press, NY, 1994.

[26] Glasser, Lance A, "*The Design and Analysis of VLSI Circuits*", Addison-Wesley, MA, 1985.

[27] E. Dillinger, "*VLSI Engineering*", Prentice Hall, NY, 1988.

[28] Y. Shayan, T. Le-Ngoc, "A Cellular Structure for a Versatile Reed-Solomon Decoder", *IEEE Trans. On Computers*, Vol 46, No. 1, Jan 199, pp 80-85.

APPENDIX A

Synthesis Tool Program in Prolog

```
%=============================================================================
% Package: ASAP
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: To find the ASAP of each operation.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================


runasap:-
initialise(List_nodes),
get_root(List_nodes,Roots),
delete_list(Roots,List_nodes,Remainings),
visit_path(Roots,Remainings),
setof(Asap,Node^asap(Node,Asap),Asaps),
last(Maximum,Asaps),assert(max_asap(Maximum)).


operations_in_cycle(Asap,Op1,Op2,Maximum):-
operations_in_cycle(Asap,Op1,Op2,Maximum,1).

%operations_in_cycle(Asap,Op1,Op2,Maximum,Maximum):-
%findall(Node,asap(Node,Maximum)

% this predicate allows to traverse the path starting from each visited node.
%-----------------------------------------------------------------------------

visit_path([],_).
visit_path([Start|Rest],Remainings):-
findall(Node,(Start^child_of_visited(Node,Start),
\+ visited(Node)),L_nodes),
remove_dups(L_nodes,List_nodes),
check_for_visited_parents(List_nodes,Remainings),
findall(Node,(member(Node,List_nodes),
Start^Edge^edge1(Edge,Start,Node),
visited(Node)),Right_child),
append(Right_child,Rest,New),
visit_path(New,Remainings).


child_of_visited(Child,Parent):-
edge1(Edge,Parent,Child),
type(Edge,direct),
visited(Parent).

check_for_visited_parents([],_).
check_for_visited_parents([Node|Others],Remaining):-
findall(Parent,(Edge^edge1(Edge,Parent,Node),
type(Edge,direct),
visited(Parent)),List),
findall(Parent,(Edge^edge1(Edge,Parent,Node),
type(Edge,direct),
\+ visited(Parent)),List1),
(List1 == [] ->
get_max_asap(List,Max,Delay),
Asap is Max + Delay,
assert(asap(Node,Asap)),
assert(visited(Node)),
delete(Remaining,Node,Rest)
;
Rest = Remaining
),
check_for_visited_parents(Others,Rest).

get_max_asap(List,Max,Delay):-
setof(Asap-Node,List^(member(Node,List),
asap(Node,Asap)),List_asaps),
```

```
last(Max-N,List_asaps),
operation_number(N,Type),
(Type == multiply ->
Type1 = mult
;
Type1 = Type),
delay(Type1,Delay).

% this predicate gets the list of nodes and the list of edges:
%-------------------------------------------------------------

initialise(List_nodes):-
      setof(Node,Y^operation_number(Node,Y),List_nodes).

% this predicate gets the parentless nodes and assign directly their
% asaps.
%-------------------------------------------------------------

get_root(List_nodes,Roots):-
setof(Root,parentless(Root,List_nodes),Roots),
assign_asap(Roots).

parentless(Node,List_nodes):-
member(Node,List_nodes),
((edge1(Edge,Node,_),type(Edge,direct))
;
      (edge1(Edge,_,Node),type(Edge,recursive))
),
\+ (edge1(Edge1,_,Node),type(Edge1,direct)).

assign_asap([]).
assign_asap([Root|Roots]):-
assert(asap(Root,1)),
assert(visited(Root)),
assign_asap(Roots).

delete_list(_,[],_).
delete_list([],R,R).
delete_list([Head|Rest],List,Rem):-
delete(List,Head,Res),
delete_list(Rest,Res,Rem).

%=============================================================================
% Package: ALAP
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: To find the ALAP of each operation.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================

runalap:-
retractall(visited(_)),
initialize(List_nodes),
get_roots(List_nodes,Roots),
del_list(Roots,List_nodes,Remainings),
visited_path(Roots,Remainings).

% this predicate allows to traverse the path starting from each visited node.
%-----------------------------------------------------------------------------

visited_path([],_).
visited_path([Start|Rest],Remainings):-
findall(Node,(Start^parent_of_visited(Node,Start),
\+ visited(Node)),Nodes),
check_visited_parents(Nodes,Remainings),
findall(Node,(member(Node,Nodes),
Start^Edge^edge1(Edge,Node,Start),
visited(Node)),Right_child),
append(Right_child,Rest,New),
visited_path(New,Remainings).
```

```
parent_of_visited(Parent,Child):-
edge1(Edge,Parent,Child),
type(Edge,direct),
visited(Child).

check_visited_parents([],_).
check_visited_parents([Node|Others],Remaining):-
findall(Parent,(Edge^edge1(Edge,Node,Parent),
type(Edge,direct),
visited(Parent)),List),
findall(Parent,(Edge^edge1(Edge,Node,Parent),
type(Edge,direct),
\+ visited(Parent)),List1),
(List1 == [] ->
get_min_alap(List,Min),
operation_number(Node,Type),
      (Type == multiply ->
            Type1 = mult
      ;
            Type1 = Type),
      delay(Type1,Delay),
Alap is Min - Delay ,
assert(alap(Node,Alap)),
assert(visited(Node)),
delete(Remaining,Node,Rest)
;
Rest = Remaining
),
check_visited_parents(Others,Rest).

get_min_alap(List,Min):-
setof(Alap,Node^List^(member(Node,List),
alap(Node,Alap)),List_alaps),
rev(List_alaps,Rev),
last(Min,Rev).


% this predicate gets the list of nodes and the list of edges:
%-------------------------------------------------------------

initialize(List_nodes):-
      setof(Node,Y^operation_number(Node,Y),List_nodes).


% this predicate gets the parentless nodes and assign directly their
% asaps.
%-------------------------------------------------------------

get_roots(List_nodes,Roots):-
setof(Root,childless(Root,List_nodes),Roots),
assign_alap(Roots).


childless(Node,List_nodes):-
member(Node,List_nodes),
edge1(Edge,_,Node),type(Edge,direct),
\+ (edge1(Edge1,Node,_),type(Edge1,direct)).

assign_alap([]).
assign_alap([Root|Roots]):-
max_asap(Max),operation_number(Root,Type),
(Type == multiply ->
            Type1 = mult
      ;
            Type1 = Type),
Alap is Max ,
assert(alap(Root,Alap)),
assert(visited(Root)),
assign_alap(Roots).

del_list(_,[],_).
del_list([],R,R).
del_list([Head|Rest],List,Rem):-
delete(List,Head,Res),
del_list(Rest,Res,Rem).

%=============================================================
% Package: Code
```

```
%% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: This program generates an ilp file to be run on Gams.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================

% the user input variables asked here affect the running of ilp since it
% influences the constraints...

% these variables are :-
%motisum.... if its coeff = 0 ->
%constraints 4,5 & 9, the variable motif,
%and addition3b are deleted from the ilp.

%mergsum... if its coeff = 0 ->
%constraints 6,10,11 & 12, the variable merg,
%and addition3a are deleted from ilp.

%tymaxov... if its coeff = 0 ->
%constraints 7 & 13, the variable maxovlap
%are deleted from the ilp.

%totovlap... if tis coeff = 0 ->
%constraint 8 is deleted from ilp.

%totincomp... if its coeff = 0 ->
%constraints 9 & 14, the variable incomp
%are deleted from ilp.

generate_code(Outfile):-

% asap.pl and alap.pl are programs that produces the asaps and alaps of each
% node or operation.
write('ENTER PLEASE THE MAXIMUM NUMBER OF CYCLES'),nl,

% here you should have previous knowledge of how the graph is being
% scheduled, so that at least you would know  the number of cycles the
% whole process needs.

write(' THE NUMBER SHOULD BE GREATER THAN CRITICAL PATH'),nl,
read(Cycles),nl,
write('DO YOU WANT THE MULTIPLIERS PIPELINED! '),nl,
write('IF SO TYPE YES '),nl,read(Ans),nl,
C is 1,
write(' HOW MANY ADDERS YOU REQUIRE ? '),nl,
read(NumAdder),asserta(numfu(add,NumAdder)),
(NumAdder \== 0 ->
C1 is C + 1 - 1,
assert(opntype(add,C1))
;
C1 is C
),
write(' HOW MANY MULTIPLIERS YOU REQUIRE'), nl,
read(NumMult),asserta(numfu(mult,NumMult)),
(NumMult \== 0 ->
C2 is C1 + 1,
assert(opntype(mult,C2))
;
C2 is C1
),
write(' HOW MANY SUBTRACTER YOU REQUIRE'), nl,
read(NumSub),asserta(numfu(sub,NumSub)),
(NumSub \== 0 ->
C3 is C2 + 1,
assert(opntype(sub,C3))
;
C3 is C2
),
write(' HOW MANY ALU YOU REQUIRE'),nl,
read(NumAlu),asserta(numfu(alu,NumAlu)),
(NumAlu \== 0 ->
        C4 is C3 + 1,
assert(opntype(alu,C4))
      ;
        C4 is C3
      ),
```

```
(NumAdder \== 0 ->
write(' LATENCY OF ADDERS YOU REQUIRE ? '),nl,
      read(LatAdder)
.
LatAdder is 0
),asserta(latency(add,LatAdder)),
(NumMult \== 0 ->
      write(' LATENCY OF MULTIPLIERS YOU REQUIRE'), nl,
      read(LatMult)
.
LatMult is 0
),asserta(latency(mult,LatMult)),
(NumSub \== 0 ->
      write(' LATENCY OF SUBTRACTER YOU REQUIRE'), nl,
      read(LatSub)
.
LatSub is 0
),asserta(latency(sub,LatSub)),
(NumAlu \== 0 ->
      write(' LATENCY OF ALU YOU REQUIRE'),nl,
      read(LatAlu)
.
LatAlu is 0
),asserta(latency(alu,LatAlu)),
write('ENTER PLEASE THE COST FUNCTION ....CSTEP'),nl,
read(Cstep),asserta(cost_function(cstep,Cstep)),
write('ENTER PLEASE THE INTERCONNECTION COST..... MOTIFSUM'),nl,
read(Motifsum),asserta(cost_inter(motifsum,Motifsum)),
write('ENTER THE MAX OVERLAP TYPE .... TYMAXOV'),nl,
read(Tymaxov),asserta(maxov(type_max_ov,Tymaxov)),
write('ENTER PLEASE THE TOTAL OVERLAP FACTOR .... TOTOVLAP'),nl,
read(Totovlap),asserta(cost_overlap(total_overlap,Totovlap)),
write('ENTER PLEASE THE MERSUM COEFF ... MERGSUM'), nl,
read(Mergesum), asserta(mergeable(mergesum,Mergesum)),
write('ENTER INCOMPATIBILITY FACTOR.....TOTINCOMP'),nl,
read(Totincomp), asserta(incomp(totincomp,Totincomp)),
get_max(Num1),
runasap,runalap,
tell(Outfile),
write('SETS'),nl,
write('S          /1*'),write(Cycles),
      write('/'),nl,
      write('P          /1*'),write(Cycles),
      write('/'),nl,
      nodes(operation,Num),
write('OP  /1*'),
write(Num),write('/'),nl,
write('OP2       /1*'),
      write(Num),write('/'),nl,
edges(direct,Num4),
write('PREC     /1*'),write(Num4),write('/'),nl,
write(' OPTYPE    /1*'),
      write(C4),write('/'),nl,
write(' OPTYPE2   /1*'),
      write(C4),write('/'),nl,
write(' ASSIGN    /1*'),
      write(Num1),write('/'),nl,
      write(' ASSIGN1   /1*'),
      write(Num1),write('/'),nl,
write(' ASSIGN2   /1*'),
      write(Num1),write('/'),nl,
total_edge(edge,Num2),
write('EDGE /1*'),
write(Num2),write('/'),nl,nl,
write(' EDGECON /1*'),
      write(Num4),write('/'),nl,
edges(recursive,Num3),
(Num3 \== 0 ->
   write('EDGEWRAP /1*'),
   write(Num3),write('/;'),nl
;
 true
),
write('SCALARS'), nl,
write('CRITICAL  /'),
find_max(Critical),
write(Critical),write(' /'),nl,
write('ADDCYCLE /'),
Diff is Cycles - Critical,
write(Diff),write('/'),nl,
```

```
write('PIPELINE        '),
(Ans == yes ->
write(' /1/ ;')
;
write('  /0/ ;')
),nl,
write('PARAMETERS'),nl,
write('ASAP(OP)        '),nl,
      write('/'),
      get_asap,write(' /'),nl,nl,
      write('ALAPORG(OP)     '),nl,
      write('/'),
      get_alap,write(' /'),nl,nl,
      write('ALAP(OP)        '),nl,
write('COUNTER(OP)     '),nl,nl,
write('TYPE(OP)        '),
      nl, write('/ '),
      type_of_ops,write('/'),nl,
write('TYPE2(OP2)      '),nl,
write_delay(C4),nl,
write_more,nl,
write('WOSUCC(OP)      '),nl,
      write('/'),
      get_wosucc,write(' /'),nl,nl,
write('SOURCE(PREC)'),nl,
write('/'),get_source_prec,write(' /'), nl,nl,
write('DEST(PREC)'),nl,
write('/'),get_dest_prec,write(' /'),nl,nl,
write('COMPREC(PREC,P) '),nl,
(Num3 \== 0 ->
write('COMPREC2(EDGEWRAP,P)'),nl,nl
;
      true
),
write('OPSTART(EDGE) '),nl,
      write('/'),
get_source,write('/'), nl ,nl,
write('OPEND(EDGE)   '),nl,
write('/'),
get_dest, write('/'), nl,nl,
write1_more,
write('CONSRC(EDGECON) '),nl,
      write('/'),
      direct_source,write('/'),nl,nl,
      write('CONDES(EDGECON) '),nl,
      write('/'),
      direct_dest,write('/'),nl,nl,
write('TYEDGECON(EDGECON,OPTYPE) '),nl,nl,nl,
(Num3 \== 0 ->
  write('WRAPSRC(EDGEWRAP)   '),nl,
        write('/'),
        source_wrap_around(Num4), write('/'),nl,nl,
  write('WRAPDES(EDGEWRAP)   '),nl,
  write('/'),
  dest_wrap_around(Num4), write('/'),nl,nl,
  write('TYEDGEWRAP(EDGEWRAP,OPTYPE) '),nl
;
  true
),
      write_parameters(Num3),nl,nl,
write_constraints(Num3),nl,
told.

% this predicate gives the max num of fu needed .
%-----------------------------------------

get_max(Num1):-
setof(Numfu,Op^numfu(Op,Numfu),List_of_numfu),
last(Num1,List_of_numfu).
          %-----------------------------------------


% this predicate gives the critical path of the graph.
%-------------------------------------------------

find_max(Critical):-
setof(Asaps,Node^asap(Node,Asaps),List_of_asap),
      last(Critical,List_of_asap).
          %-----------------------------------------
```

```
% this predicate prints the ASAP(OP) in the ILP1 File:-
%-------------------------------------------------

get_asap:-
setof([Node,Asapno],asap(Node,Asapno),
                        L_Asapnos),
      rev(L_Asapnos,List_of_Asap),
      length(List_of_Asap,Len),
      nth1(Len,L_Asapnos,Part,_Rest),
C is 0,
      write_asap(L_Asapnos,Part,C).

write_asap([],_,_).
write_asap([[Node,Asapno]|L_Asapnos],Part,9):-
nl,write(' '),
      ([Node,Asapno] \== Part ->
(write(Node),write('='),write(Asapno),write(','))
;
(write(Node),write('='),write(Asapno))),
      write_asap(L_Asapnos,Part,1).

write_asap([[Node,Asapno]|L_Asapnos],Part,C):-
C1 is C+1,
([Node,Asapno] \== Part ->
            (write(' '),write(Node),write('='),write(Asapno),write(','))
            ;
            (write(' '),write(Node),write('='),write(Asapno))),
      write_asap(L_Asapnos,Part,C1).
%-----------------------------------------
% this predicate prints the ALAP(OP) in the ILP1 File:-
%-------------------------------------------------


get_alap:-
setof([Node,Alapno],alap(Node,Alapno),
                        L_Alapnos),
      rev(L_Alapnos,List_of_Alap),
      length(List_of_Alap,Len),
      nth1(Len,L_Alapnos,Part,_Rest),
C is 0,
      write_asap(L_Alapnos,Part,C).

%-----------------------------------------------

% This predicate prints out the SOURCE(EDGE) in ilp1:-
%-------------------------------------------------
get_source:-
findall([Edge,Source],(source(Edge,Source),
\+type(Edge,output),
\+type(Edge,input)),L_sources),
rev(L_sources,List_of_sources),
      length(List_of_sources,Len1),
      nth1(Len1,List_of_sources,Parts,_Rests),
C is 0,
writing_source(List_of_sources,Parts,C).

get_source_prec:-
findall([Edge,Source],(source(Edge,Source),type(Edge,direct)),
                        L1_sources_prec),
      rev(L1_sources_prec,List1_of_sources_prec),
      length(List1_of_sources_prec,Len1),
      nth1(Len1,List1_of_sources_prec,Parts,_Rests),
C is 0,
      writing_source(List1_of_sources_prec,Parts,C).


writing_source([],_,_).

writing_source([[Edge,Source]|List_of_sources],Parts,9):-
nl,write(' '),
      ([Edge,Source] \== Parts ->
(write(Edge),write('='),write(Source),write(','))
;
(write(Edge),write('='),write(Source))),
      writing_source(List_of_sources,Parts,1).

writing_source([[Edge,Source]|List_of_sources],Parts,C):-
C1 is C+1,
([Edge,Source] \== Parts ->
            (write(' '),write(Edge),write('='),write(Source),write(','))
```

```
        ;
        (write(' '),write(Edge),write('='),write(Source))),
writing_source(List_of_sources,Parts,C1).

%-----------------------------------------------

% This predicate prints out the DEST(EDGE) in ilp1:-
%-----------------------------------------------

get_dest:-
    findall([Edge,Dest],(dest(Edge,Dest),
\+type(Edge,output),
\+type(Edge,input)),L_Dests),
rev(L_Dests,List_of_Dest),
length(List_of_Dest,Len),
nth1(Len,List_of_Dest,Part,_Rest),
C is 0,
    write_dest(List_of_Dest,Part,C).

get_dest_prec:-
    findall([Edge,Dest],(dest(Edge,Dest),type(Edge,direct)),
                    L1_Dests_prec),
    rev(L1_Dests_prec,List1_of_Dest_prec),
    length(List1_of_Dest_prec,Len1),
    nth1(Len1,List1_of_Dest_prec,Part,_Rest2),
C is 0,
    write_dest(List1_of_Dest_prec,Part,C).

write_dest([],_,_).

write_dest([[Edge,Dest]|List_of_Dests],Part,9):-
    nl,write(''),
    ([Edge,Dest] \== Part ->
(write(Edge),write('='),write(Dest),write(','))
;
(write(Edge),write('='),write(Dest))),
    write_dest(List_of_Dests,Part,1).

write_dest([[Edge,Dest]|List_of_Dests],Part,C):-
C1 is C+1,
([Edge,Dest] \== Part ->
        (write(' '),write(Edge),write('='),write(Dest),write(','))
        ;
        (write(' '),write(Edge),write('='),write(Dest))),
    write_dest(List_of_Dests,Part,C1).

%-----------------------------------------------

% This predicate prints out the WRAPSOURCE(EDGE) in ilp1:-
%-----------------------------------------------------

source_wrap_around(Num4):-
findall([Edge,Source],(source(Edge,Source),
                    type(Edge,recursive)),S_wraps),
length(S_wraps,Len4),
(Len4 == 0 ->
 write_source_wrap([],_,_,_)
;
     nth1(Len4,S_wraps,PartS,_RestS),
 C is 0,
 write_source_wrap(S_wraps,PartS,Num4,C)
).


write_source_wrap([],_,_,_).
write_source_wrap([[Edge,Source]|List_of_Dests],Part,Num4,9):-
    nl,([Edge,Source] \== Part ->
    (write(''),
 Edge1 is Edge - Num4,write(Edge1),write('='),
 write(Source),write(','))
;
    (write(''),Edge1 is Edge -Num4,
 write(Edge1),write('='),write(Source))
),
    write_source_wrap(List_of_Dests,Part,Num4,1).

write_source_wrap([[Edge,Source]|List_of_Dests],Part,Num4,C):-
    C1 is C+1,([Edge,Source] \== Part ->
    (write(' '),
     Edge1 is Edge - Num4,write(Edge1),write('='),
```

```
      write(Source),write(','))
     ;
     (write(' '),Edge1 is Edge -Num4,
      write(Edge1),write('='),write(Source))
     ),
     write_source_wrap(List_of_Dests,Part,Num4,C1).

%-----------------------------------------

% This predicate prints out the WRAPDEST(EDGE) in ilp1:-
%-----------------------------------------

dest_wrap_around(Num4):-
findall([Edge,Dest],(dest(Edge,Dest),
                     type(Edge,recursive)),D_wraps),
     length(D_wraps,Len3),
(Len3 == 0 ->
     write_dest_wrap([],_,_,_)
;
     nth1(Len3,D_wraps,PartD,_RestD),
     C is 0,
     write_dest_wrap(D_wraps,PartD,Num4,C)
).

write_dest_wrap([],_,_,_).

write_dest_wrap([[Edge,Dest]|List_of_Dests],Part,Num4,9):-
     nl,([Edge,Dest] \== Part ->
(write(' '),Edge1 is Edge - Num4,
      write(Edge1),write('='),write(Dest),write(','))
      ;
      (write(' '),Edge1 is Edge - Num4,
       write(Edge1),write('='),write(Dest))
      ),
      write_dest_wrap(List_of_Dests,Part,Num4,1).

write_dest_wrap([[Edge,Dest]|List_of_Dests],Part,Num4,C):-
C1 is C+1,
      ([Edge,Dest] \== Part ->
(write(' '),Edge1 is Edge - Num4,
 write(Edge1),write('='),write(Dest),write(','))
;
(write(' '),Edge1 is Edge - Num4,
 write(Edge1),write('='),write(Dest))
),
      write_dest_wrap(List_of_Dests,Part,Num4,C1).

%-----------------------------------------

% this predicate prints the source of all direct edges in the graph:-
%-----------------------------------------------------------------
direct_source:-
findall([Edge,Source],(source(Edge,Source),
                       type(Edge,direct)),L_direct_source),
length(L_direct_source,Len6),
nth1(Len6,L_direct_source,DirectLast,_RestDirect),
C is 0,
writing_source(L_direct_source,DirectLast,C).

%---------------------------------

% this predicate prints the destination of all direct edges in the graph:-
% ---------------------------------------------------------------------
direct_dest:-
findall([Edge,Source],(dest(Edge,Source),
                       type(Edge,direct)),L_direct_edge),
      length(L_direct_edge,Len5),
      nth1(Len5,L_direct_edge,DirectLastdest,_RestDirectEdge),
C is 0,
      write_dest(L_direct_edge,DirectLastdest,C).

%-----------------------------------

% this predicate prints the type of each operation (add,mult,sub,etc...):-
% ---------------------------------------------------------------

type_of_ops:-
```

```
findall([Num,Type],is_an_operation(Num,Type),List_of_Ops),
length(List_of_Ops,Len7),
nth1(Len7,List_of_Ops,LastType,_RestType),
C is 0,
write_type(List_of_Ops,LastType,C).

is_an_operation(Num,Type):-
operation_number(Num,Ops),
((Ops == multiply -> opntype(mult,Type));
(Ops == add -> opntype(add,Type));
(Ops == alu -> opntype(alu,Type));
(Ops == subtract -> opntype(sub,type))).

write_type([],_,_).

write_type([[Ops,Type]|List_of_Ops],LastType,9):-
      nl,write(''),([Ops,Type] \== LastType ->
            do_write(Ops,Type)
            ;
            write_end(Ops,Type)),
      write_type(List_of_Ops,LastType,1).

write_type([[Ops,Type]|List_of_Ops],LastType,C):-
C1 is C+1,
([Ops,Type] \== LastType ->
do_write(Ops,Type)
;
write_end(Ops,Type)),
write_type(List_of_Ops,LastType,C1).

do_write(Edge,Dest):-
      write(' '),write(Edge),write('='),write(Dest),write(',').

write_end(Edge,Dest):-
      write(' '),write(Edge),write('='),write(Dest).


%           ---------------------------------------------


% this predicate get all the without-successor nodes and prints them:-
%-------------------------------------------------------------------

get_wosucc:-
findall(Nodes,no_succ(Nodes),L_wosucc),
      remove_dups(L_wosucc,List_wosucc),
length(List_wosucc,L7),
nth1(L7,List_wosucc,Last,_Rest_list),
C is 0,
      write_succ(List_wosucc,Last,C).

write_succ([],_,_).

write_succ([Edge|List_wosucc],Last,9):-
      nl,write(''),(Edge \== Last ->
            write_still(Edge)
            ;
            write_end(Edge)),
      write_succ(List_wosucc,Last,1).

write_succ([Edge|List_wosucc],Last,C):-
C1 is C+1,
(Edge \== Last ->
write_still(Edge)
;
write_end(Edge)),
write_succ(List_wosucc,Last,C1).

write_still(Edge):-
write(Edge),write('='),write('1,').

write_end(Edge):-
write(' '),write(Edge),write('=1').


not_a_source(Node):-
\+ (edge1(Edge,Node,_Dest),type(Edge,direct)).

no_succ(Nodes):-
```

```
edge1(Edge,_Source,Nodes),
type(Edge,direct),
not_a_source(Nodes).

%           ----------------------------------------

write_parameters(Num3):-
write('POSTOP(EDGECON,P)'),nl,
write('POSCOM(EDGECON,P)'),nl,
write('POSCEN(EDGECON,P)    '),nl,
write('POSBOT(EDGECON,P)    '),nl,
write('LIFETIME(EDGECON,P)'),nl,nl,
(Num3 \== 0 ->
   write('POSBEG(EDGEWRAP,P) '),nl,
   write('POSBDEF(EDGEWRAP,P)  '),nl,
   write('POSFIN(EDGEWRAP,P)   '),nl,
   write('POSFDEF(EDGEWRAP,P)  '),nl
;
   true
),
write('SRCDELAY(EDGECON)    '),nl,
write('CHKMERG(EDGECON,P)   '),nl,
write('LIMIT10(EDGECON,ASSIGN1,ASSIGN2)'),nl,
(Num3 \== 0 ->
  write('MERGWRAP(EDGEWRAP)  '),nl,
  write('CHKMERG2(EDGEWRAP,P)  '),nl,
  write('LIMIT12(EDGEWRAP,ASSIGN1,ASSIGN2)'),nl
;
  true
),
write('MULTOUT(OP)    '),nl,nl,
write('EDGEMULT(EDGECON) '),nl,
(Num3 \== 0 ->
  write('WRAPMULT(EDGEWRAP)   ;'),nl,nl,nl
;
  write(';')
).


write_constraints(Num3):-
write('ALAP(OP) = ALAPORG(OP) + ADDCYCLE;'),nl,
write('TYPE2(OP2) = SUM(OP$(ORD(OP) EQ ORD(OP2)), TYPE(OP));'),
nl, write('COUNTER(OP) = 1;'), nl,nl,
write('DELAYOP(OP) = SUM(OPTYPE$(TYPE(OP) EQ ORD(OPTYPE)), '),
write('DELAYFU(OPTYPE));'),
nl,nl,
write('LATENCYOP(OP) = SUM(OPTYPE$(TYPE(OP) EQ ORD(OPTYPE)), '),
write('LATENCYFU(OPTYPE));'),nl,
write('LATEOPM1(OP) $ (LATENCYOP(OP) GT 0) = '),
write('LATENCYOP(OP) - 1 ;'),nl,
write('LATEOPP1(OP) $ (LATENCYOP(OP) GT 0) = '),
      write('LATENCYOP(OP) + 1 ;'),nl,nl,
write('DELAYOPM1(OP) = SUM(OPTYPE$(TYPE(OP) EQ ORD(OPTYPE)),'),
write('DELAYFU(OPTYPE)) - 1;'), nl,
write('*INPUT(EDGEIN) = 1;'),nl,
write('*OUTPUT(EDGEOUT) = 1;'),nl,nl,
write('*TOPPART(EDGEIN,P) = 1$(SUM(OP$((INPUT(EDGEIN) EQ '),
write('ORD(OP)) AND '),nl,
write('*(ORD(P) GE ASAP(OP)) AND (ORD(P) LT ALAP(OP)) )'),
write(' COUNTER(OP)) GT 0) ;'),nl,nl,
write('*BOTPART(EDGEOUT,P) = 1$(SUM(OP$((OUTPUT(EDGEOUT) EQ '),
write('ORD(OP)) AND '), nl,
write('*(ORD(P) GE (ASAP(OP)+DELAYOPM1(OP))) ), '),
write('COUNTER(OP)) GT 0);'), nl,nl,
write('POSTOP(EDGECON,P) = 1$((SUM(OP$((CONSRC(EDGECON) '),
write('EQ ORD(OP)) AND '),nl,
write('(ORD(P) GE (ASAP(OP) + DELAYOPM1(OP)))'),nl,
write('AND (ORD(OP) LE (ALAP(OP) + DELAYOPM1(OP))) ), '), nl,
write('COUNTER(OP)) GT 0) '), nl,
write('AND (SUM(OP$((CONDES(EDGECON) EQ ORD(OP)) AND '),
write('(ORD(OP) LT ASAP(OP)) ), '),nl,
write('COUNTER(OP)) GT 0) );'), nl,nl,
write('POSBOT(EDGECON,P) = 1$((SUM(OP$((CONSRC(EDGECON) '),
write('EQ ORD(OP)) AND '),nl,
write('(ORD(P) GT ALAP(OP))'), COUNTER(OP)) GT 0)'),nl,
write('AND (SUM(OP$((CONDES(EDGECON) EQ ORD(OP)) '),
write('AND (ORD(P) GE ASAP(OP))'),nl,
write('AND (ORD(P) LT ALAP(OP)) ), COUNTER(OP)) GT 0) );'),nl,
```

```
      write('POSCOM(EDGECON,P) = 1$((SUM(OP$((CONSRC(EDGECON) '),
      write('EQ ORD(OP)) AND '), nl,
      write('(ORD(P) LE ALAP(OP)) ), COUNTER(OP)) GT 0) AND'),nl,
      write('(SUM(OP$((CONDES(EDGECON) EQ ORD(OP)) AND '),
      write('(ORD(P) GE ASAP(OP)) ), '),nl,
      write('COUNTER(OP)) GT 0) );'),nl,nl,
      write('POSCEN(EDGECON,P) = 1$((SUM(OP$((CONSRC(EDGECON) '),
      write('EQ ORD(OP)) AND '), nl,
      write('(ORD(P) GT (ALAP(OP)+ DELAYOPM1(OP))) ), '),
      write('COUNTER(OP)) GT 0) AND'),nl,
      write('(SUM(OP$((CONDES(EDGECON) EQ ORD(OP)) AND (ORD(P) '),
      write(' LT ASAP(OP)) ), '), nl,
      write('COUNTER(OP)) GT 0) ); '), nl,
      write('LIFETIME(EDGECON,P)=1 $((POSCEN(EDGECON,P) GT 0) OR'),nl,
      write('(POSCOM(EDGECON,P) GT 0) OR (POSTOP(EDGECON,P) GT 0) OR '),nl,
      write('(POSBOT(EDGECON,P) GT 0));'),nl,nl,
      (Num3 \== 0 ->
                  write('POSBEG(EDGEWRAP,P)=1   $((SUM(OP$((WRAPSRC(EDGEWRAP)   EQ
ORD(OP))'),nl,
        write('AND (ORD(P) GE (ASAP(OP)+DELAYOPM1(OP))) AND '),nl,
        write('(ORD(P) LE (ALAP(OP) + DELAYOP(OP) -2))), COUNTER(OP)) GT 0));'),nl,nl,
                  write('POSBDEF(EDGEWRAP,P)=1   $((SUM(OP$((WRAPSRC(EDGEWRAP)   EQ
ORD(OP))'),nl,
        write('AND (ORD(P) GT (ALAP(OP) + DELAYOP(OP) -2))), COUNTER(OP)) GT 0));'),nl,nl,
        write('POSFIN(EDGEWRAP,P)=1 $((SUM(OP$((WRAPDES(EDGEWRAP) EQ ORD(OP))'),nl,
        write('AND (ORD(P) GE ASAP(OP)) AND '),nl,
        write('(ORD(P) LT ALAP(OP)) ), COUNTER(OP)) GT 0));'),nl,nl,
                  write('POSFDEF(EDGEWRAP,P)=1   $((SUM(OP$((WRAPDES(EDGEWRAP)   EQ
ORD(OP))'),nl,
        write('AND (ORD(P) LT ASAP(OP)) ), COUNTER(OP)) GT 0));'),nl,nl
      ;
        true
      ),
      write('NUMFU2(OPTYPE2)=  SUM(OPTYPE$(ORD(OPTYPE)  EQ  ORD(OPTYPE2)),  NUM-
FU(OPTYPE));') ,nl,
      write('NUMOP(OP)=      SUM(OPTYPE$(TYPE(OP)      EQ      ORD(OPTYPE)),NUMFU(OP-
TYPE));'),nl,nl,
      write('RANGEX(OP,S)= 1$((ORD(S) GE ASAP(OP)) AND (ORD(S) LE ALAP(OP)));'),nl,nl,
      write('RANGEXP(OP,S,P)$((PIPELINE EQ 0) AND (ORD(P) GE ASAP(OP)) AND '),nl,
      write('(ORD(P) LE (ALAP(OP) + DELAYOP(OP) -1)))= 1$((ORD(S) LE ORD(P)) AND'),nl,
      write('(ORD(S) GE (ORD(P) - DELAYOP(OP) + 1)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
      write('(ORD(S) LE ALAP(OP)));'),nl,
      write('RANGEXP(OP,S,P)$((PIPELINE EQ 1) AND (ORD(P) GE ASAP(OP)) AND '),nl,
      write('(ORD(P) LE (ALAP(OP) + LATEOPM1(OP))))= 1$((ORD(S) LE ORD(P)) AND'),nl,
      write('(ORD(S) GE (ORD(P) - LATEOPM1(OP))) AND (ORD(S) GE ASAP(OP)) AND'),nl,
      write('(ORD(S) LE ALAP(OP)));'),nl,nl,
      write('COMPREC(PREC,P)= 1$('),nl,
      write('(SUM((OP,S)$((SOURCE(PREC) EQ ORD(OP)) AND '),nl,
      write('(ORD(S) LE (ALAP(OP) + DELAYOP(OP) - 1)) AND'),nl,
      write('(ORD(S) EQ ORD(P)) ), COUNTER(OP))  GT 0) AND'),nl,
      write('(SUM((OP,S)$((DEST(PREC) EQ ORD(OP)) AND'),nl,
      write('(ORD(S) GE ASAP(OP)) AND'),nl,
      write('(ORD(S) EQ ORD(P)) ), COUNTER(OP)) GT 0) );'),nl,nl,
      (Num3 \== 0 ->
        write('COMPREC2(EDGEWRAP,P)$('),nl,
        write('WRAPSRC(EDGEWRAP) NE WRAPDES(EDGEWRAP)) =1$('),nl,
        write('(SUM((OP,S)$((WRAPDES(EDGEWRAP) EQ ORD(OP)) AND '),nl,
            write('(ORD(S) LE (ALAP(OP) + DELAYOP(OP) - 1)) AND'),nl,
            write('(ORD(S) EQ ORD(P)) ), COUNTER(OP))  GT 0) AND'),nl,
            write('(SUM((OP,S)$((WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND'),nl,
            write('(ORD(S) GE ASAP(OP)) AND'),nl,
            write('(ORD(S) EQ ORD(P)) ), COUNTER(OP)) GT 0) );'),nl,nl
      ;
        true
      ),
      write('POSMOTIF(EDGE,OPTYPE,OPTYPE2) = 1$'),nl,
      write('(SUM(OP$(ORD(OP) EQ OPSTART(EDGE)),TYPE(OP)) EQ ORD(OPTYPE) ) AND '),nl,
      write('(SUM(OP$(ORD(OP) EQ OPEND(EDGE)), TYPE(OP)) EQ ORD(OPTYPE2) );'),nl,nl,
      write('SETMOTIF(OPTYPE,OPTYPE2) =1$'),nl,
      write('(SUM(EDGE$POSMOTIF(EDGE,OPTYPE,OPTYPE2),'),nl,
      write('POSMOTIF(EDGE,OPTYPE,OPTYPE2)) GT 0);'),nl,nl,
      write('MINMOTIF=SUM((OPTYPE,OPTYPE2),SETMOTIF(OPTYPE,OPTYPE2));'),nl,nl,
      write('NUMVAR(OP,OPTYPE)$(TYPE(OP) EQ ORD(OPTYPE)) = '),nl,
      write('(ALAP(OP) - ASAP(OP)) * NUMOP(OP);'),nl,nl,
      write('MAXVAR(OPTYPE) = SMAX(OP,NUMVAR(OP,OPTYPE));'),nl,nl,
      write('SELECT(OP,OPTYPE)$(TYPE(OP) EQ ORD(OPTYPE)) = 1$'),nl,
      write('(NUMVAR(OP,OPTYPE) EQ MAXVAR(OPTYPE));'),nl,nl,
      write('SELECT(OP,OPTYPE) = ORD(OP)*SELECT(OP,OPTYPE);'),nl,nl,
      write('CHOSE1(OPTYPE) = SMAX(OP$SELECT(OP,OPTYPE),SELECT(OP,OPTYPE));'),nl,nl,
      write('SRCDELAY(EDGECON)=SUM(OP$(ORD(OP)    EQ    CONSRC(EDGECON)),    DELAY-
```

```
OP(OP));'),nl,nl,
    write('CHKMERG(EDGECON,P)= 1 $'),nl,
    write('((ORD(P) GE SUM(OP$(ORD(OP) EQ CONSRC(EDGECON)), ASAP(OP))) AND '),nl,
    write(' (ORD(P) LE SUM(OP$(ORD(OP) EQ CONSRC(EDGECON)), ALAP(OP))) AND'),nl,
    write(' ((ORD(P) + SRCDELAY(EDGECON)) GE '),nl,
    write('SUM(OP$(ORD(OP) EQ CONDES(EDGECON)), ASAP(OP))));'),nl,nl,
    write('LIMIT10(EDGECON,ASSIGN1,ASSIGN2)=1$'),nl,
    write('((SUM(OP$(CONSRC(EDGECON) EQ ORD(OP)), NUMOP(OP)) GE ORD(ASSIGN1))
AND'),nl,
    write('  (SUM(OP$(CONDES(EDGECON) EQ ORD(OP)), NUMOP(OP)) GE ORD(AS-
SIGN2)));'),nl,nl,
    (Num3 \== 0 ->
      write('MERGWRAP(EDGEWRAP)=1$ ('),nl,
      write('(SUM(OP$(WRAPDES(EDGEWRAP) EQ ORD(OP)), ASAP(OP)) EQ 1) AND'),nl,
        write('(SUM(OP$(WRAPSRC(EDGEWRAP) EQ ORD(OP)), ALAP(OP) + DELAYOP-
M1(OP))'),nl,nl,
      write('EQ (CARD(S) + ADDCYCLE)) );'),nl,nl,
      write('CHKMERG2(EDGEWRAP,P)$MERGWRAP(EDGEWRAP) =1 $'),nl,
      write('((ORD(P) GE SUM(OP$(ORD(OP) EQ WRAPSRC(EDGEWRAP)),'),nl,
      write('CRITICAL - DELAYOPM1(OP))) AND '),nl,
      write('(ORD(P) LE SUM(OP$(ORD(OP) EQ WRAPSRC(EDGEWRAP)), ALAP(OP))) );'),nl,nl,
      write('LIMIT12(EDGEWRAP,ASSIGN1,ASSIGN2)$MERGWRAP(EDGEWRAP) =1$'),nl,
        write('((SUM(OP$(WRAPSRC(EDGEWRAP) EQ ORD(OP)), NUMOP(OP)) GE ORD(AS-
SIGN1)) AND'),nl,
        write('  (SUM(OP$(WRAPDES(EDGEWRAP) EQ ORD(OP)), NUMOP(OP)) GE ORD(AS-
SIGN2)) );'),nl,nl
    ;
      true
    ),
    write('TYEDGECON(EDGECON,OPTYPE)      =      1$(SUM(OP$(CONDES(EDGECON)      EQ
ORD(OP)), TYPE(OP))'),nl,
    write('EQ ORD(OPTYPE));'),nl,nl,
    (Num3 \== 0 ->
      write('TYEDGEWRAP(EDGEWRAP,OPTYPE) =1$(SUM(OP$(WRAPDES(EDGEWRAP) EQ
ORD(OP)), TYPE(OP))'),nl,
      write('EQ ORD(OPTYPE));'),nl,nl
    ;
      true
    ),
    write('MULTOUT(OP) =1$(SUM(EDGES(OPSTART(EDGE) EQ ORD(OP)),'),nl,
    write('(OPSTART(EDGE)/OPSTART(EDGE))) GT 1);'),nl,nl,
    write('EDGEMULT(EDGECON) =1$(SUM(OP$((MULTOUT(OP) EQ 1) AND (CONSRC(EDGE-
CON) EQ'),nl,
    write('ORD(OP))), COUNTER(OP)) GT 0);'),nl,nl,
    (Num3 \== 0 ->
              write('WRAPMULT(EDGEWRAP)  =1$(SUM(OP$((MULTOUT(OP)  EQ  1)  AND
(WRAPSRC(EDGEWRAP) EQ'),nl,
      write('ORD(OP))), COUNTER(OP)) GT 0);'),nl,nl
    ;
      true
    ),
    write('DISPLAY COUNTER, DELAYOP, NUMOP, RANGEX, RANGEXP, COMPREC,'),nl,
    write('      POSMOTIF, NUMVAR,  MAXVAR, SELECT, CHOSE1, SETMOTIF, MINMOTIF,'),nl,
    write('      POSTOP, POSBOT, POSCEN, POSCOM, LIFETIME,'),nl,
    (Num3 \== 0 ->
      write('      POSBEG, POSFIN, POSBDEF, POSFDEF,'),nl
    ;
      true
    ),
    write('      LATENCYOP, LATEOPM1, LATEOPP1,'),nl,
    (Num3 \== 0 ->
      write('      MERGWRAP, ')
    ;
      true
    ),write('CHKMERG, LIMIT10,'),nl,
    write('      TYEDGECON '),
    (Num3 \== 0 ->
    write(',TYEDGEWRAP'),nl
    ;
       nl
    ),
    write('      MULTOUT, EDGEMULT '),
    (Num3 \== 0 ->
    write(', WRAPMULT'),nl,nl,nl,nl
    ;
      nl,nl,nl,nl
    ),
    write('VARIABLES'),nl,write('X(OP,ASSIGN,S)'),nl,
    write('CSTEP'),nl,nl,
    cost_inter(motifsum,Motifsum),
```

```
(Motifsum \== 0 ->
write('MOTIF(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)'),nl,
write(' MOTIFSUM'),nl,nl
;
true
),
cost_overlap(total_overlap,Totovlap),
(Totovlap \== 0 ->
write('TOTOVLAP'),nl,nl
;
true),
incomp(totincomp,Totincomp),
(Totincomp \== 0 ->
write('INCOMP(OPTYPE2)'),nl,nl
;
true
),
mergeable(mergesum,Mergesum),
( Mergesum \== 0 ->
write('MERG(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)'),nl,nl,
write('MERGSUM'),nl,nl
;
true
),
maxov(type_max_ov,Tymaxov),
(Tymaxov \== 0 ->
write('MAXOVLAP(OPTYPE) '),nl,nl,
write('TYMAXOV'),nl,nl
;
true),
(Totincomp \== 0 ->
write('TOTINCOMP'),nl,nl
;
true
),
write('OBJ;'),nl,nl,nl,nl,
write('BINARY VARIABLESX'),
(Motifsum \== 0 ->

write(', MOTIF')
;
true
),
(Mergesum \== 0 ->
write(', MERG ')
;
true
),
write(';'), nl,nl,nl,nl,
write('EQUATIONS'),nl,
write('ADDITION1'),nl,write('ADDITION2'),nl,
(Motifsum \== 0 ->
write('ADDITION3A'),nl
;
true
),
(Mergesum \== 0 ->
write('ADDITION3B'),nl
;
true
),write('ASSIGNONE(OPTYPE)'),nl,nl,
write('CONS1(OP)'),nl,write('CONS2(ASSIGN,OPTYPE,P)'),nl,
write('CONS3A(PREC,P)'),nl,
(Num3 \== 0 ->
    write('CONS3B(EDGEWRAP,P)'),nl
;
    true
),
write('CONS4(OP)'),nl,
(Motifsum \== 0 ->
write('CONS5(EDGE,OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)'),nl,
write('CONS6'),nl
;
true
),
(Tymaxov \== 0 ->
write('CONS7(P,OPTYPE)'),nl
;
true
),
```

```
( Totovlap \== 0 ->
write('CONS8'),nl
;
true
),
(Totincomp \== 0 ->
write('CONS9(OPTYPE2,ASSIGN2)'),nl
;
true
),
(Mergesum \== 0 ->
write('CONS10(EDGECON,P,ASSIGN1,ASSIGN2)'),nl,
write('CONS11'),nl
;
true
),
((Num3 \== 0 ), (Mergesum \== 0 ))->
write('CONS12(EDGEWRAP,P,ASSIGN1,ASSIGN2)'),nl
;
true
),
(Tymaxov \== 0 ->
write('CONS13'),nl
;
true
),
(Totincomp \== 0 ->
write('CONS14'),nl
;
true
),write('TOTAL;'),nl,nl,
write('ADDITION1.. CSTEP =G= CRITICAL;'),nl,
write('ADDITION2.. CSTEP =L= CARD(S);'),nl,
(Motifsum \== 0 ->
write('ADDITION3A.. MOTIFSUM =G= 0;'),nl
;
true),
(Mergesum \== 0 ->
write('ADDITION3B.. MERGSUM =G= 0;'),nl
;
true
),write('ASSIGNONE(OPTYPE)'),
write('$((CHOSE1(OPTYPE) LE CARD(OP))'),nl,
write('AND (CHOSE1(OPTYPE) GE 1)..'),nl,
write('SUM((OP,S)$((RANGEX(OP,S) EQ 1) AND (CHOSE1(OPTYPE) EQ ORD(OP))),'),nl,
write('X(OP,"1",S)) =E= 1;'),nl,nl,
write_cons1,
write_cons2,
write_cons3(Num3),
write_cons4,
(Motifsum \== 0 ->
write_cons5,
write_cons6
;
true
),
(Tymaxov \== 0 ->
write_cons7(Num3)
;
true
),
(Totovlap \== 0 ->
write_cons8(Num3)
;
true
),
(Totincomp \== 0 ->
write_cons9
;
true
),
(Mergesum \== 0 ->
write_cons10,
write_cons11,
write_cons12(Num3)
;
true
),
(Tymaxov \== 0 ->
write_cons13
```

```
        ;
        true
        ),
        (Totincomp \== 0 ->
        write_cons14
        ;
        true
        ),
        write_rest.
        write_cons1:-
        write('CONS1(OP)..'),nl,write('SUM(S$RANGEX(OP,S),'),nl,
        write('SUM(ASSIGN$(ORD(ASSIGN) LE NUMOP(OP)), X(OP,ASSIGN,S))) =E= 1;'),
        nl,nl.

        write_cons2:-
        write('CONS2(ASSIGN,OPTYPE,P)$(ORD(ASSIGN) LE NUMFU(OPTYPE))..'),nl,
        write('SUM(OP$(TYPE(OP) EQ ORD(OPTYPE)),'),nl,
        write('SUM(S$RANGEXP(OP,S,P), X(OP,ASSIGN,S)) ) =L= 1;'),nl,nl.

        write_cons3(Num3):-
        write('CONS3A(PREC,P)$COMPREC(PREC,P)..'),nl,
        write('SUM((OP,ASSIGN,S)$((SOURCE(PREC) EQ ORD(OP)) AND'),nl,
        write('(RANGEX(OP,S) EQ 1) AND (ORD(S) GE (ORD(P)-DELAYOP(OP)+1)) AND'),nl,
        write('(ORD(ASSIGN) LE NUMOP(OP)) ),X(OP,ASSIGN,S)) +'),nl,
        write('SUM((OP,ASSIGN,S)$((DEST(PREC) EQ ORD(OP)) AND'),nl,
        write('(ORD(S) LE ORD(P)) AND (RANGEX(OP,S) EQ 1) AND'),nl,
        write('(ORD(ASSIGN) LE NUMOP(OP)) ), X(OP,ASSIGN,S)) =L= 1;'),nl,nl,
        (Num3 \== 0 ->
        write('CONS3B(EDGEWRAP,P)$COMPREC2(EDGEWRAP,P)..'),nl,
              write('SUM((OP,ASSIGN,S)$((WRAPDES(EDGEWRAP) EQ ORD(OP)) AND'),nl,
              write('(RANGEX(OP,S) EQ 1) AND (ORD(S) GE (ORD(P)-DELAYOP(OP)+1)) AND'),
        nl,
             write('(ORD(ASSIGN) LE NUMOP(OP)) ),X(OP,ASSIGN,S)) +'),nl,
              write('SUM((OP,ASSIGN,S)$((WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND'),nl,
              write('(ORD(S) LT ORD(P)) AND (RANGEX(OP,S) EQ 1) AND'),nl,
              write('(ORD(ASSIGN) LE NUMOP(OP)) ), X(OP,ASSIGN,S)) =L= 1;'),nl,nl
        ;
        true
        ).

        write_cons4:-
        write('CONS4(OP)$WOSUCC(OP)..'),nl,
        write('SUM(S$RANGEX(OP,S), ORD(S)* SUM(ASSIGN$(ORD(ASSIGN) LE NUMOP(OP)),'),nl,
        write('X(OP,ASSIGN,S))) -CSTEP =L= -DELAYOP(OP) +1;'),nl,nl.

        write_cons5:-
        write('CONS5(EDGE,OPTYPE,ASSIGN1,OP-
TYPE2,ASSIGN2)$((POSMOTIF(EDGE,OPTYPE,OPTYPE2) EQ 1)'),nl,
        write('AND (ORD(ASSIGN1) LE NUMFU(OPTYPE)) AND (ORD(ASSIGN2) LE NUMFU2(OP-
TYPE2)) )..'),nl,
        write('SUM(OP$(ORD(OP) EQ OPSTART(EDGE)), SUM(S$RANGEX(OP,S), SUM(ASSIGN$'),nl,
        write('(ORD(ASSIGN) EQ ORD(ASSIGN1)), X(OP,ASSIGN,S) ) ) ) +'),nl,
        write('SUM(OP$(ORD(OP) EQ OPEND(EDGE)), SUM(S$RANGEX(OP,S),SUM(ASSIGN$'),nl,
        write('(ORD(ASSIGN) EQ ORD(ASSIGN2)), X(OP,ASSIGN,S) ) ) ) -'),nl,
        write('MOTIF(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2) =L= 1;'),nl,nl.

        write_cons6;-
        write('CONS6 ..SUM((OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)$'),nl,
        write('((SETMOTIF(OPTYPE,OPTYPE2) EQ 1)'),nl,
        write('AND (ORD(ASSIGN1) LE NUMFU (OPTYPE)) AND (ORD(ASSIGN2) LE NUMFU2
(OPTYPE2))),'),nl,
        write('MOTIF(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)) -MOTIFSUM =E=0;'),nl,nl.

        write_cons7(Num3):-
        write('CONS7(P,OPTYPE)..'),nl,
        write('0.9 * ('),nl,
        write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
        write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) EQ 1) AND'),nl,
        write('(POSTOP(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
        write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
        write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) +'),nl,
        write('SUM(EDGECON$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
        write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) EQ 1) AND'),nl,
        write('(POSCEN(EDGECON,P) EQ 1)), LIFETIME(EDGECON,P)) +'),nl,
        write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
        write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) EQ 1) AND '),nl,
        write('(POSCOM(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
        write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
        write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) -'),nl,
        write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
```

```
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) EQ 1) AND'),nl,
write('(POSCOM(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE ORD(P))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) EQ 1) AND'),nl,
write('(POSBOT(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GT ORD(P)) AND'),nl,
write('(ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S))) +'),nl,nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) NE 1) AND'),nl,
write('(POSTOP(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) +'),nl,
write('SUM(EDGECON$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) NE 1) AND'),nl,
write('(POSCEN(EDGECON,P) EQ 1)), LIFETIME(EDGECON,P)) +'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) aND (EDGEMULT(EDGECON) NE 1) AND'),nl,
write('(POSCOM(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) -'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) NE 1) AND'),nl,
write('(POSCOM(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND '),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE ORD(P))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((TYEDGECON(EDGECON,OPTYPE) EQ 1)'),nl,
write('AND (LIFETIME(EDGECON,P) EQ 1) AND (EDGEMULT(EDGECON) NE 1) AND'),nl,
write('(POSBOT(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GT ORD(P)) AND'),nl,
write('(ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S))  '),
(Num3 \== 0 ->
write('+  '),nl,write('0.9 * ('),nl,
        write('SUM((EDGEWRAP,OP,ASSIGN,S)$((TYEDGEWRAP(EDGEWRAP,OPTYPE)  EQ
1)'),nl,
write('AND (POSBEG(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) EQ 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) + 1))), X(OP,ASSIGN,S))+'),nl,
write('SUM((EDGEWRAP,OP)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (WRAPMULT(EDGEWRAP) EQ 1) AND'),nl,
write('(POSBDEF(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) EQ 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) ), COUNTER(OP)) +'),nl,
write('SUM((EDGEWRAP,OP,ASSIGN,S)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (POSFIN(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) EQ 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GT ORD(P)) AND (ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGEWRAP,OP)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (POSFDEF(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) EQ 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP))), COUNTER(OP)) ) +'),nl,nl,
write('SUM((EDGEWRAP,OP,ASSIGN,S)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (POSBEG(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) NE 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) + 1))), X(OP,ASSIGN,S))+'),nl,
write('SUM((EDGEWRAP,OP)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (WRAPMULT(EDGEWRAP) NE 1) AND'),nl,
write('(POSBDEF(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) NE 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) ), COUNTER(OP)) +'),nl,
write('SUM((EDGEWRAP,OP,ASSIGN,S)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (POSFIN(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) NE 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GT ORD(P)) AND (ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGEWRAP,OP)$((TYEDGEWRAP(EDGEWRAP,OPTYPE) EQ 1)'),nl,
write('AND (POSFDEF(EDGEWRAP,P) EQ 1) AND (WRAPMULT(EDGEWRAP) NE 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP))), COUNTER(OP))  )
;
true
),
write(' - MAXOVLAP(OPTYPE) =L= 0;'),nl,nl.

write_cons8(Num3):-
write('CONS8.. SUM(P,'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((LIFETIME(EDGECON,P) EQ 1) AND'),nl,
write('(POSTOP(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) +'),nl,
write('SUM(EDGECON$((LIFETIME(EDGECON,P) EQ 1) AND'),nl,
write('(POSCEN(EDGECON,P) EQ 1)), LIFETIME(EDGECON,P)) +'),nl,
```

```
write('SUM((EDGECON,OP,ASSIGN,S)$((LIFETIME(EDGECON,P) EQ 1) AND'),nl,
write('(POSCOM(EDGECON,P) EQ 1) AND (CONSRC(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) +1))), X(OP,ASSIGN,S)) -'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((LIFETIME(EDGECON,P) EQ 1) AND'),nl,
write('(POSCOM(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE ORD(P))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGECON,OP,ASSIGN,S)$((LIFETIME(EDGECON,P) EQ 1) AND'),nl,
write('(POSBOT(EDGECON,P) EQ 1) AND (CONDES(EDGECON) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) LE NUMOP(OP)) AND (ORD(S) GT ORD(P)) AND'),nl,
write('(ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S))  '),nl,
(Num3 \== 0 ->
nl,write('+  SUM((EDGEWRAP,OP,ASSIGN,S)$((POSBEG(EDGEWRAP,P) EQ 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GE ASAP(OP)) AND'),nl,
write('(ORD(S) LE (ORD(P) - DELAYOP(OP) + 1))), X(OP,ASSIGN,S))+'),nl,
write('SUM((EDGEWRAP,OP)$((POSBDEF(EDGEWRAP,P) EQ 1) AND'),nl,
write('(WRAPSRC(EDGEWRAP) EQ ORD(OP)) ), COUNTER(OP)) +'),nl,
write('SUM((EDGEWRAP,OP,ASSIGN,S)$((POSFIN(EDGEWRAP,P) EQ 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP)) AND (ORD(ASSIGN) LE NUMOP(OP))'),nl,
write('AND (ORD(S) GT ORD(P)) AND (ORD(S) LE ALAP(OP))), X(OP,ASSIGN,S)) +'),nl,
write('SUM((EDGEWRAP,OP)$((POSFDEF(EDGEWRAP,P) EQ 1) AND'),nl,
write('(WRAPDES(EDGEWRAP) EQ ORD(OP))), COUNTER(OP))')
;
true
),
write(' ) - TOTOVLAP =L= 0;'),nl,nl.

write_cons9:-
write('CONS9(OPTYPE2,ASSIGN2)$(ORD(ASSIGN2) LE NUMFU2(OPTYPE2)) ..'),nl,
write('SUM((OPTYPE,ASSIGN1)$'),nl,
write('((SETMOTIF(OPTYPE,OPTYPE2) EQ 1)'),nl,
write('AND (ORD(ASSIGN1) LE NUMFU(OPTYPE))),'),nl,
write('MOTIF(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)) -INCOMP(OPTYPE2) =L= 0;'),
nl,nl.


write_cons10:-
write('CONS10(EDGECON,P,ASSIGN1,ASSIGN2)$'),nl,
write('((CHKMERG(EDGECON,P) EQ 1) AND'),nl,
write('(LIMIT10(EDGECON,ASSIGN1,ASSIGN2) EQ 1)) ..'),nl,
write('SUM((OP,S,ASSIGN)$((CONSRC(EDGECON) EQ ORD(OP)) AND '),nl,
write('(ORD(P) EQ ORD(S)) AND (ORD(ASSIGN) EQ ORD(ASSIGN1))), '),nl,
write('X(OP,ASSIGN,S)) +'),nl,
write('SUM((OP,S,ASSIGN)$((CONDES(EDGECON) EQ ORD(OP)) AND '),nl,
write('((ORD(P)+SRCDELAY(EDGECON)) EQ ORD(S)) AND'),nl,
write('(ORD(ASSIGN) EQ ORD(ASSIGN2))), X(OP,ASSIGN,S)) -'),nl,
write('SUM((OP,OP2,OPTYPE,OPTYPE2)$((CONSRC(EDGECON) EQ ORD(OP))'),nl,
write('AND (TYPE(OP) EQ ORD(OPTYPE)) AND (CONDES(EDGECON) EQ ORD(OP2))'),nl,
write('AND (TYPE2(OP2) EQ ORD(OPTYPE2))),'),nl,
write('MERG(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)) =L= 1;'),nl,nl.

write_cons11:-
write('CONS11 ..SUM((OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)$'),nl,
write('((SETMOTIF(OPTYPE,OPTYPE2) EQ 1)'),nl,
write('AND (ORD(ASSIGN1) LE NUMFU(OPTYPE)) AND (ORD(ASSIGN2) LE NUMFU2(OP-
TYPE2))),'),nl,
write('MERG(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)) -MERGSUM =E=0;'),nl,nl.

write_cons12(Num3):-
(Num3 \== 0 ->
write('CONS12(EDGEWRAP,P,ASSIGN1,ASSIGN2)$'),nl,
write('((CHKMERG2(EDGEWRAP,P) EQ 1) AND'),nl,
write('(LIMIT12(EDGEWRAP,ASSIGN1,ASSIGN2) EQ 1)) ..'),nl,
write('SUM((OP,ASSIGN)$((WRAPDES(EDGEWRAP) EQ ORD(OP)) AND'),nl,
write('(ORD(ASSIGN) EQ ORD(ASSIGN2))),'),nl,
write('X(OP,ASSIGN,"1")) +'),nl,
write('SUM((OP,S,ASSIGN)$((WRAPSRC(EDGEWRAP) EQ ORD(OP)) AND'),nl,
write('(ORD(P) EQ ORD(S)) AND'),nl,
write('(ORD(ASSIGN) EQ ORD(ASSIGN1))), X(OP,ASSIGN,S)) -'),nl,
write('SUM((OP,OP2,OPTYPE,OPTYPE2)$((WRAPSRC(EDGEWRAP) EQ ORD(OP))'),nl,
write('AND (TYPE(OP) EQ ORD(OPTYPE)) AND (WRAPDES(EDGEWRAP) EQ
ORD(OP2))'),nl,
write('AND (TYPE2(OP2) EQ ORD(OPTYPE2))),'),nl,
write('MERG(OPTYPE,ASSIGN1,OPTYPE2,ASSIGN2)) =L= 1;'),nl,nl
;
true
).
```

```
write_cons13:-
write('CONS13.. SUM(OPTYPE, MAXOVLAP(OPTYPE)) - TYMAXOV =E=0;'),nl,nl.

write_cons14:-
write('CONS14.. SUM(OPTYPE2, INCOMP(OPTYPE2)) - TOTINCOMP =E=0;'),nl,nl.

write_rest:-
write('TOTAL.. '),cost_function(cstep,Cstep),write(Cstep),
write('*CSTEP '),cost_inter(motifsum,Motifsum),
(Motifsum \== 0 ->
write(' + '),
write(Motifsum),
write('*MOTIFSUM ')
;
true
),maxov(type_max_ov,Tymaxov),
(Tymaxov \== 0 ->
write(' + '),
write(Tymaxov),
write('*TYMAXOV ')
;
true
),cost_overlap(total_overlap,Totovlap),
(Totovlap \== 0 ->
write(' + '),
write(Totovlap),
write('*TOTOVLAP ')
;
true
),mergeable(mergesum,Mergesum),
(Mergesum \== 0 ->
write(' + '),
write(Mergesum),
write('*MERGSUM ')
;
true
),nl,incomp(totincomp,Totincomp),
(Totincomp \== 0 ->
write(' + '),
write(Totincomp),
write('*TOTINCOMP ')
;
true
),
write(' =E= OBJ;'),nl,nl,
write('MODEL SCHEDULE /ALL/ ;'),nl,
write('SCHEDULE.optfile =1;'),nl,
write('OPTIONS ITERLIM = 2000000, RESLIM=1000000,'),nl,
write('       WORK=500000, LIMROW=300, OPTCR=0.001;'),nl,
write('SOLVE SCHEDULE USING MIP MINIMIZING OBJ ;'),nl.

write_more:-
write('NUMFU2(OPTYPE2)   '),nl,
write('NUMOP(OP)         '),nl,
write('RANGEX(OP,S)'),nl,
write('RANGEXP(OP,S,P)'),nl.

write1_more:-
write('POSMOTIF(EDGE,OPTYPE,OPTYPE2)   '),nl,
write('SETMOTIF(OPTYPE,OPTYPE2)    '),nl,
write('MINMOTIF'),nl,nl,
write('NUMVAR(OP,OPTYPE)     '),nl,
write('MAXVAR(OPTYPE)        '),nl,
write('SELECT(OP,OPTYPE)     '),nl,
write('CHOSE1(OPTYPE)        '),nl,nl,nl.


write_delay(C4):-
write('DELAYFU(OPTYPE)    '),nl,
write_delays(C4),nl,
write('DELAYOP(OP)'),nl,
write('DELAYOPM1(OP)'),nl,
write('LATENCYFU(OPTYPE)'),nl,
write('/'),
write_latency(C4),nl,
write('LATENCYOP(OP)'),nl,
write('LATEOPM1(OP)'),nl,
write('NUMFU(OPTYPE)'),
write('/'),write_numfu(C4),
nl,write('LATEOPP1(OP)'),nl.
```

```
write_numfu(C):-
write_numfu(C,1).

write_numfu(C,C):-
write(C), write('='),
opntype(Optype,C),
numfu(Optype,Numfu),
write(Numfu),write('    /'),nl.

write_numfu(C,C1):-
write(C1),write('='),
opntype(Optype,C1),
numfu(Optype,Numfu),
write(Numfu),write(' , '),
C2 is C1 + 1,
write_numfu(C,C2).

write_latency(C):-
write_latency(C,1).

write_latency(C,C):-
write(C), write('='),
opntype(Optype,C),
latency(Optype,Latency),
write(Latency),write('    /').

write_latency(C,C1):-
write(C1),write('='),
opntype(Optype,C1),
latency(Optype,Latency),
write(Latency), write(' , '),
C2 is C1 + 1,
write_latency(C,C2).


write_delays(C):-
write('/'),write_delays(C,1).

write_delays(C,C):-
write(C),write('='),
opntype(Optype,C),
delay(Optype,Delay),
write(Delay),write('    /').

write_delays(C,C1):-
write(C1),write('='),
opntype(Optype,C1),
delay(Optype,Delay),
write(Delay), write(' , '),
C2 is C1 + 1,
write_delays(C,C2).


% this is the main call to the program that generates ilp2....
% it reads from output of ilp1, and generates the codes for ilp2.
:- dynamic destination_edge/3.

rdb:-
 read_and_build_db.


read_and_build_db:-
      compile('~/SCHD_BIND/sob.pl'),
% this file gives you the output of ilp1 formulation in the form:
%     nodenum_funcid_cycleno(Node,FU,Cycle).  .....

compile('~/SCHD_BIND/del.pl'),
% this file gives the delay of each operator (adder,multiplier,etc...)

compile('~/SCHD_BIND/optype.pl'),
% this is to assert the database of Nodenumber-OperationType from file optype.pl

      assert_opn_type,
% this is to assert the delay of each operation related to the type of function
% operating.....

assert_delay_type,
```

```
% this is to assert database of Edgenumber-SourceNode.
% it takes data from the list in edge_srce.pl that gives the source of each
% edge.....

      compile('~/SCHD_BIND/edge_srce.pl'),
      assert_edge_srce,

% this is to assert database of Edgenumber-DestinationNode.
      compile('~/SCHD_BIND/edge_dest.pl'),
      assert_edge_dest.


% this predicate allows to define the operation type of each node
% and puts the data in a file called op.pl.....
%-------------------------------------------------------------

assert_opn_type:-
      tell('~/SCHD_BIND/op.pl'),
      nodnum_optyp(Operation_Fu),
      assert_opn_type(Operation_Fu),
told.

assert_opn_type([]).
assert_opn_type([NodeNum-OpType|Operation_Fu]):-
      assert(nodenum_optype(NodeNum,OpType)),
      write(nodenum_optype(NodeNum,OpType)),write('.'),nl,
      assert_opn_type(Operation_Fu).

%-------------------------------------------


% this predicate asserts the dealy of each node related to the type of
% operation done....
%-----------------------------------------------------------------------

assert_delay_type:-
tell('~/SCHD_BIND/delay.pl'),
optype_delay(Delays),
assert_delay(Delays),
findall(Nodenum-Delay,(Op^nodenum_optype(Nodenum,Op),
  operation_delay(Op,Delay)),List_delay),
assert_operation_delay(List_delay),
told.

assert_delay([]).
assert_delay([OpType-Delay|Rest]):-
assert(operation_delay(OpType,Delay)),
assert_delay(Rest).

assert_operation_delay([]).
assert_operation_delay([Op-Delay|Rest]):-
assert(op_delay(Op,Delay)),
write(op_delay(Op,Delay)),write('.'),nl,
assert_operation_delay(Rest).

%-----------------------------------------------

% this predicate asserts the data that defines the source of each edge.
% the data is written in a file calledsource.pl1
assert_edge_srce:-
      tell('~/SCHD_BIND/source.pl'),
      edge_srce(Edge_Source),
      assert_edge_srce(Edge_Source),
      told.

assert_edge_srce([]).
assert_edge_srce([Edge-Source|Edge_Source]):-
assert(source_edge(Source,Edge,1)),
      write(source_edge(Source,Edge,1)),write('.'),nl,
      assert_edge_srce(Edge_Source).


assert_edge_dest:-
      tell('~/SCHD_BIND/dest.pl'),
      edge_dest(Edge_Destination),
      assert_edge_dest(Edge_Destination),told.

assert_edge_dest([]).
assert_edge_dest([Edge-Destination|Edge_Destination]):-
set_of_all(E,Destination^St^destination_edge(Destination,E,St),Edges),
```

```
length(Edges,Len),
St_time is Len + 1,
      assert(destination_edge(Destination,Edge,St_time)),
      write(destination_edge(Destination,Edge,St_time)),write('.'),nl,
      assert_edge_dest(Edge_Destination).

%=============================================================================
% Package: ILP2.
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: To generate an ilp2 file tp be run on gams.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================



 :- dynamic cycle/2.


% this is the main call to the program that generates ilp2.....
% it reads from output of ilp1, and generates the codes for ilp2.


generate(Filename):-
compile('~/SCHD_BIND/in_put'),
% the in_put file is the file produced from data_ilp2.pl and gives the database
% for the input of ilp2.

compile('~/SCHD_BIND/sob.pl'),
% the sob.pl file gives the information about the operation cycle and functional
% id for the node.

findall(Edge,Cycle^starting(Edge,Cycle),Edges),
% the above findall is to get the starting time of edges that needs bus
% bus allocation.....

findall(Node,New^node_new(Node,New),Nodes),

% the second findall gets the original number of the node.

findall(Op-Edge,mult(Op,_,Edge),Multiple),
% the third findall is to get the multiple output nodes and their edges.

write('Enter the number of buses needed'),nl,read(Number_buses),
tell(Filename),
format_input(Edges,Nodes,Multiple,Number_buses),
told.

format_input(Edges,Nodes,Multiple,Number_buses):-
length(Edges,Length_Edges), % gets the no. of edges that need allocation
length(Nodes,Length_Nodes), % gets the no. of ops that mult_output.
write('SETS'),nl,
num_of_cycles(Max),
create_list_cycle(Max),
findall(Ops-Delay-Cycle,(op_delay(Ops,Delay),
            FU^nodenum_funcid_cycleno(Ops,FU,Cycle)
  ),L_Operations),
remove_dups(L_Operations,Operations),
% the above findall gets the delay and the starting time for each operation.

visit_each_node(Operations,Max),
findall(Edge-Source-Phase,
edge_two(Edge,Source,Phase),
Two_cycle_edges),

% the above findall gets the list of edges that two cycles long it gives
% the starting time, and the source of that edge.

findall(Edge-Source-Time,Max^wrap_two_cycle(Edge,Source,Time,Max),
 Wrap_two_edge),

% the above findall gets the list of edges that are wrap around edges, of
% length 2 and starts at the maximum cycle, it gets the source,and the starting
% time of that edge.

append(Wrap_two_edge,Two_cycle_edges,List1),
```

```
Before is Max - 1,
findall(Edge-Source-Time,
Before^two_cycle_edge(Edge,Source,Time,Before),
Two_cycles),

% the above findall gets the edges that are two cycles long, wrap_around edges,
% and starts one cycle before the last cycle. It gets the source and the
% starting time of these edges.

append(Two_cycles,List1,L_Final),
remove_dups(L_Final,Final),
inc_cycles(Final),
findall(Cycle-Reg,(cycle(Cycle,Reg),Reg \== 0),List_regs),

% the above findall gets the cycles that have needs registers.

findall(Node-Edge-Cycle,
    (member(Edge-Node-Cycle,Final),
New^node_new(Node,New),
multiple_op_node(Node)),
List_Reg_Cycle),
% the above findall gets the list of edges that are two cycles long and comes
% from a multiple output node.

write('Eset of the edges / 1*'),
write(Length_Edges),write('/'),nl,
write('Snumber of cycles /1*'),write(Max),write('/'),nl,
write('Pnumber of cycles /1*'),write(Max),write('/'),nl,
write('Nnumber of buses  /1*'),write(Number_buses),
write('/'),nl,
write('OPno. of multiple output ops  /1*'),
write(Length_Nodes),write('/   ;'),nl,nl,nl,
write('PARAMETERS'),nl,
write('ASAPS(E)asap cycle for bus tr. of each edge'),nl,
write('/'),
write_asaps_edges(Edges),write(' /'),nl,nl,
write('ALAPT(E)alap cycle for bus tr. of each edge'),nl,
write('/'),
write_alaps_edges(Edges),write(' /'),nl,nl,
write('MULT(OP,E)'),nl,
write('/'),
write_mult_ops(Multiple),write(' /'),nl,nl,
write('REG(P)'),nl,
write('/'),write_registers(List_regs),write(' /'),nl,nl,
format1_ilp2,
format2_ilp2,
format3_ilp2,
format4_ilp2,
format5_ilp2,
format6_ilp2,
(List_Reg_Cycle \== [] ->
assert_additions(List_Reg_Cycle)
;
true
),
format7_ilp2,
(List_Reg_Cycle \== [] ->
write_additions(List_Reg_Cycle)
;
true
),
format8_ilp2,
format9_ilp2,
format10_ilp2,nl,nl.

% this predicate updates the no. of registers needed in each cycle as a
% final result.
% ------------------------------------------------------------------

inc_cycles([]).
inc_cycles([_-_-CyclelFinal]):-
cycle(Cycle,C),
C1 is C + 1,
retract(cycle(Cycle,C)),assert(cycle(Cycle,C1)),
inc_cycles(Final).

%----------------------------------------

edge_two(Edge,Source,Phase):-
source_edge(Source,Edge,_),
```

```
op_delay(Source,Delay),
destination_edge(Dest,Edge,_),
nodenum_funcid_cycleno(Source,_,Start),
nodenum_funcid_cycleno(Dest,_,End),
      Length is End - Start - Delay + 1,
Length = 2, Phase is Start + 1.

% this predicate initializes  the no. of buses needed in each cycle to zero.
%-------------------------------------------------------------------

create_list_cycle(Max):-
create_list_cycle(Max,1).

create_list_cycle(Max,Max):-
assert(cycle(Max,0)).

create_list_cycle(Max,C):-
assert(cycle(C,0)),
C1 is C + 1,
create_list_cycle(Max,C1).

%-----------------------------------------


two_cycle_edge(Edge,Source,Time,Before):-
source_edge(Source,Edge,_),
op_delay(Source,Delay),
      destination_edge(Dest,Edge,_),
Max is Before + 1,
      nodenum_funcid_cycleno(Source,_,Before),
nodenum_funcid_cycleno(Dest,_,1),
Length is 1 - Before + Max - Delay + 1 ,
Length = 2 , Time is Max.

wrap_two_cycle(Edge,Source,Time,Max):-
source_edge(Source,Edge,_),
      destination_edge(Dest,Edge,_),
nodenum_funcid_cycleno(Source,_,Max),
nodenum_funcid_cycleno(Dest,_,2),
Time is 1.


% this predicate updates the no. of buses needed in each cycle.
% ------------------------------------------------------------
visit_each_node([],_).
visit_each_node([Op-Delay-Cycle|Operation],Max):-
op_delay(Op,Delay),
 (Delay == 1 -> New_Cycle is Cycle,
cycle(New_Cycle,C),
C1 is C + 1
 ;
       New_Cycle is Cycle + Delay - 1,
      cycle(New_Cycle,C),
C1 is C + 1
      ),
retract(cycle(New_Cycle,C)),assert(cycle(New_Cycle,C1)),
visit_each_node(Operation,Max).

%---------------------------------------

% this predicate writes the alap of each edge formatted in ilp2 file:-
% ------------------------------------------------------------------

write_alaps_edges(Edges):-
length(Edges,L),
      nth1(L,Edges,Last_edge,_Rest),
      write_alaps_edges(Edges,Last_edge,0).


write_alaps_edges([],_,_).

write_alaps_edges([Edge|Edges],Last_edge,9):-
      nl,write('      '),
      ending(Edge,Cycle),
      (Edge \== Last_edge ->
            write(' '),write(Edge),write('='),write(Cycle),write(',')
      ;
            write(' '),write(Edge),write('='),write(Cycle)
      ),
```

```
    write_alaps_edges(Edges,Last_edge,1).

write_alaps_edges([Edge|Edges],Last_edge,C):-
    C1 is C + 1,
    ending(Edge,Cycle),
    (Edge \== Last_edge ->
        write(' '),write(Edge),write('='),write(Cycle),write(',')
    ;
        write(' '),write(Edge),write('='),write(Cycle)
    ),
    write_alaps_edges(Edges,Last_edge,C1).

%---------------------------------------

% this predicate writes the number of registers needed in each cycle:-
% ------------------------------------------------------------------

write_registers(List_regs):-
length(List_regs,Max),
nth1(Max,List_regs,Last,_),
write_registers(List_regs,Last,0).

write_registers([],_,_).

write_registers([Cycle-Reg|List],Last,9):-
nl,write(' '),
cycle(Cycle,Reg),
(Cycle-Reg \== Last ->
write(' '),write(Cycle),write('='),write(Reg),write(',')
;
write(' '),write(Cycle),write('='),write(Reg)
),
write_registers(List,Last,0).

write_registers([Cycle-Reg|List],Last,C):-
C1 is C + 1,
    cycle(Cycle,Reg),
    (Cycle-Reg \== Last ->
        write(' '),write(Cycle),write('='),write(Reg),write(',')
    ;
        write(' '),write(Cycle),write('='),write(Reg)
    ),
    write_registers(List,Last,C1).

%-----------------------------------------

% This predicate writes the multiple output nodes formatted in  file for ilp2:-
% ------------------------------------------------------------------------------

write_mult_ops(Multiple):-
length(Multiple,L),
nth1(L,Multiple,Last_node,_),
write_mult_ops(Multiple,Last_node,0).


write_mult_ops([],_,_).

write_mult_ops([Node-Edge|Nodes],Last_node,7):-
    nl,write('        '),
node_new(Node,New_Node),
    (Node-Edge \== Last_node ->
        write(' '),write(New_Node),write('.'),
      write(Edge),write('=1'),write(',')
    ;
      write(' '),write(New_Node),write('.'),
        write(Edge),write('=1')
    ),
    write_mult_ops(Nodes,Last_node,1).

write_mult_ops([Node-Edge|Nodes],Last_node,C):-
C1 is C + 1,
    node_new(Node,New_Node),
    (Node-Edge \== Last_node ->
        write(' '),write(New_Node),write('.'),
        write(Edge),write('=1'),write(',')
    ;
        write(' '),write(New_Node),write('.'),
        write(Edge),write('=1')
    ),
```

```
    write_mult_ops(Nodes,Last_node,C1).

%-----------------------------------------

% this predicate writes the asaps of each formatted in file for ilp2 :-
% ---------------------------------------------------------------------

write_asaps_edges(Edges):-
length(Edges,L),
nth1(L,Edges,Last_edge,_Rest),
write_asaps_edges(Edges,Last_edge,0).

write_asaps_edges([],_,_).

write_asaps_edges([Edge|Edges],Last_edge,9):-
    nl,write('    '),
starting(Edge,Cycle),
    (Edge \== Last_edge ->
        write(' '),write(Edge),write('='),write(Cycle),write(',')
    ;
        write(' '),write(Edge),write('='),write(Cycle)
    ),
    write_asaps_edges(Edges,Last_edge,1).


write_asaps_edges([Edge|Edges],Last_edge,C):-
C1 is C + 1,
starting(Edge,Cycle),
(Edge \== Last_edge ->
    write(' '),write(Edge),write('='),write(Cycle),write(',')
;
    write(' '),write(Edge),write('='),write(Cycle)
),
write_asaps_edges(Edges,Last_edge,C1).

%-----------------------------------------

multiple_op_node(Node):-
findall(Edge,source_edge(Node,Edge,_),List),
length(List,L),
L > 1.


% this predicate writes the additional equations for multiple output nodes
% and getting the starting time of their edges.
% ---------------------------------------------------------------------

assert_additions(List_Reg_Cycle):-
length(List_Reg_Cycle,L),
write_equation(L,1).

write_equation(L,L):-
write('ADDITION'),write(L).

write_equation(L,C):-
write('ADDITION'),write(C),nl,
C1 is C + 1,
write_equation(L,C1).

write_additions(List_Reg_Cycle):-
nl,nl,nl,
length(List_Reg_Cycle,Length),
%nth1(L,List_Reg_Cycle,Last,_),
write_additions(List_Reg_Cycle,Length,1),nl,nl.

write_additions([Node-_Edge-Cycle|_List],L,L):-
write('ADDITION'),write(L),write('..   R('),
    node_new(Node,New),write('"'),
    write(New),write('"'),
    write(Cycle),write('") =E= 1;'),nl.

write_additions([Node-_Edge-Cycle|List],Length,C):-
write('ADDITION'),write(C),write('..   R('),
    node_new(Node,New),write('"'),
    write(New),write('"'),
    write(Cycle),write('") =E= 1;'),nl,
C1 is C + 1,
write_additions(List,Length,C1).
```

```
%----------------------------------------

format1_ilp2:-
format('ALAPS(E)
        ASAPT(E)
        RANGES(E,S)
        RANGESP(E,P)
        RANGET(E,S)
        RANGETP(E,P)
        COMSTE(E,P)
STFACET(E,P,S)
        TEFACET(E,P,s)
',[]).

format2_ilp2:-
        format('RANGEE(E,P)
        STRCOUNT(E,P,S)
        TERCOUNT(E,P,S)
        SINGLE(E)
        RANGEOP(OP,P)
        RANGEMST(OP,S)
        RANGEMTE(OP,S) ;
',[]).

format3_ilp2:-

format('
ASAPT(E)= 1;
ASAPT(E)$(ASAPS(E) LT CARD(S))= ASAPS(E) + 1;

ALAPS(E)= CARD(S);
ALAPS(E)$(ALAPT(E) GT 1)= ALAPT(E) - 1;

RANGES(E,S)$(ASAPS(E) LE ALAPS(E))= 1$
((ORD(S) GE ASAPS(E)) AND (ORD(S) LE ALAPS(E)));

RANGES(E,S)$(ASAPS(E) GT ALAPS(E))= 1$
((ORD(S) GE ASAPS(E)) OR (ORD(S) LE ALAPS(E)));

RANGET(E,S)$(ASAPT(E) LE ALAPT(E))= 1$
((ORD(S) GE ASAPT(E)) AND (ORD(S) LE ALAPT(E)));

RANGET(E,S)$(ASAPT(E) GT ALAPT(E))= 1$
((ORD(S) GE ASAPT(E)) OR (ORD(S) LE ALAPT(E)));

RANGESP(E,P)= SUM(S$(ORD(S) EQ ORD(P)), RANGES(E,S));

RANGETP(E,P)= SUM(S$(ORD(S) EQ ORD(P)), RANGET(E,S));

COMSTE(E,P)$((RANGESP(E,P) EQ 1) AND (RANGETP(E,P) EQ 1))= 1 ;',[]).

format4_ilp2:-
format('

STFACET(E,P,S)$
((COMSTE(E,P) EQ 1) AND (ORD(P) LE ALAPS(E)))= 1$
((ORD(S) GE ORD(P)) AND (ORD(S) LE ALAPS(E)));

STFACET(E,P,S)$
((COMSTE(E,P) EQ 1) AND (ORD(P) GT ALAPS(E)))= 1$
((ORD(S) GE ORD(P)) OR (ORD(S) LE ALAPS(E)));

TEFACET(E,P,S)$
((COMSTE(E,P) EQ 1) AND (ORD(P) GE ASAPT(E)))= 1$
((ORD(S) LE ORD(P)) AND (ORD(S) GE ASAPT(E)));
```

```
TEFACET(E,P,S)$
((COMSTE(E,P) EQ 1) AND (ORD(P) LT ASAPT(E)))= 1$
((ORD(S) LE ORD(P)) OR (ORD(S) GE ASAPT(E)));

RANGEE(E,P)= 1$((RANGESP(E,P) EQ 1) OR (RANGETP(E,P) EQ 1));

STRCOUNT(E,P,S)$
((RANGEE(E,P) EQ 1) AND (ORD(P) GE ASAPS(E))) =1$
((ORD(S) LE ORD(P)) AND
(ORD(S) GE ASAPS(E)) AND (RANGES(E,S) EQ 1));',[]).

format5_ilp2:-
format('

STRCOUNT(E,P,S)$
((RANGEE(E,P) EQ 1) AND (ORD(P) LT ASAPS(E))) =1$
( ((ORD(S) LE ORD(P)) AND (RANGES(E,S) EQ 1))
OR (ORD(S) GE ASAPS(E)));

TERCOUNT(E,P,S)$
((RANGETP(E,P) EQ 1) AND (ORD(P) GE ASAPT(E))) =1$
((ORD(S) LE ORD(P)) AND
(ORD(S) GE ASAPT(E)) AND (RANGET(E,S) EQ 1));

TERCOUNT(E,P,S)$
((RANGETP(E,P) EQ 1) AND (ORD(P) LT ASAPT(E))) =1$
( ((ORD(S) LE ORD(P)) AND (RANGET(E,S) EQ 1))
OR (ORD(S) GE ASAPT(E)));

SINGLE(E)=1 $(SUM(OP, MULT(OP,E)) EQ 0);

RANGEOP(OP,P)= 1$
(SUM(E$MULT(OP,E), (RANGESP(E,P)+RANGETP(E,P))) GT 0);

RANGEMST(OP,S)= 1$
(SUM(E$MULT(OP,E), RANGES(E,S)) GT 0);

RANGEMTE(OP,S)= 1$
(SUM(E$MULT(OP,E), RANGET(E,S)) GT 0);',[]).

format6_ilp2:-
format('

DISPLAY ASAPS, ALAPS, ASAPT, ALAPT, RANGES, RANGET,
     COMSTE, STFACET, TEFACET, SINGLE,
     RANGEE, RANGEOP ,RANGEMST, RANGEMTE,
     STRCOUNT, TERCOUNT ;


VARIABLES
     ST(E,S,N)      start var. for the bus transfer
     TE(E,S,N)      termination var. for the bus transfer
     MST(OP,S,N)    start var. for a fictious edge
*                   representing the multiple output op.
     MTE(OP,S,N)    termination var. for a fictious edge
*                   representing the multiple output op.
     R(OP,P)        representing a reg. per cycle for
                    multiple output op.
     MINREG         min number of reg. at each cycle

     TOTREG         total cycles for reg. usage

     OBJ ;

BINARY VARIABLES  ST, TE, MST, MTE, R ;

EQUATIONS
',[]).


format7_ilp2:-
format('
CONS1(E)
CONS2(E)
CONS3(E,N)
CONS4(E,P)
CONS5(OP,E,S,N)
```

```
CONS6(OP,E,S,N)
CONS7(S,N)
CONS8(S,N)
CONS9(OP,E,P)
CONS10(OP,E,P)
CONS11(P)
CONS12
TOTAL;

',[]).

format8_ilp2:-
format('
CONS1(E)..     SUM(N,SUM(S$RANGES(E,S), ST(E,S,N))) =L= 1;

CONS2(E)..     SUM(N,SUM(S$RANGET(E,S), TE(E,S,N))) =L= 1;

CONS3(E,N)..   SUM(S$RANGES(E,S), ST(E,S,N)) -
          SUM(S$RANGET(E,S), TE(E,S,N)) =E= 0;

CONS4(E,P)$COMSTE(E,P)..
          SUM(N,SUM(S$STFACET(E,P,S), ST(E,S,N))) +
          SUM(N,SUM(S$TEFACET(E,P,S), TE(E,S,N))) =L= 1;

CONS5(OP,E,S,N)$((MULT(OP,E) EQ 1) AND
          (RANGES(E,S) EQ 1)).. ST(E,S,N) - MST(OP,S,N) =L= 0;

CONS6(OP,E,S,N)$((MULT(OP,E) EQ 1) AND
          (RANGET(E,S) EQ 1)).. TE(E,S,N) - MTE(OP,S,N) =L= 0;

CONS7(S,N)..   SUM(E$((SINGLE(E) EQ 1) AND
          (RANGES(E,S) EQ 1)), ST(E,S,N)) +
          SUM(OP$RANGEMST(OP,S), MST(OP,S,N)) =L= 1;
',[]).

format9_ilp2:-
format('CONS8(S,N)..   SUM(E$((SINGLE(E) EQ 1) AND
          (RANGET(E,S) EQ 1)), TE(E,S,N)) +
          SUM(OP$RANGEMTE(OP,S), MTE(OP,S,N)) =L= 1;

CONS9(OP,E,P)$((MULT(OP,E) EQ 1) AND (RANGESP(E,P) EQ 1))..
          1 - SUM(N, SUM(S$RANGES(E,S), ST(E,S,N))) -
          R(OP,P) =L= 0;

CONS10(OP,E,P)$((MULT(OP,E) EQ 1) AND (RANGEOP(OP,P) EQ 1))..
          (1 -
          SUM(N,SUM(S$STRCOUNT(E,P,S), ST(E,S,N))) +
          SUM(N,SUM(S$TERCOUNT(E,P,S), TE(E,S,N))) )
          - R(OP,P) =L= 0;

CONS11(P)..    SUM(E$((SINGLE(E) EQ 1) AND (RANGEE(E,P))),
          (1 -
          SUM(N,SUM(S$STRCOUNT(E,P,S), ST(E,S,N))) +
          SUM(N,SUM(S$TERCOUNT(E,P,S), TE(E,S,N))) ) ) +
          REG(P) + SUM(OP$RANGEOP(OP,P), R(OP,P))
          - MINREG =L= 0;',[]).

format10_ilp2:-
format('
CONS12..       SUM(P, SUM(E$RANGEE(E,P),
          (1 -
          SUM(N,SUM(S$STRCOUNT(E,P,S), ST(E,S,N))) +
          SUM(N,SUM(S$TERCOUNT(E,P,S), TE(E,S,N))) ) ) )
          - TOTREG =E= 0;

TOTAL..        20*MINREG + 10*TOTREG +
          SUM((OP,S,N)$RANGEMST(OP,S), MST(OP,S,N)) +
          SUM((OP,S,N)$RANGEMTE(OP,S), MTE(OP,S,N)) +
          SUM((OP,P)$RANGEOP(OP,P), R(OP,P)) - OBJ =E= 0;


MODEL SCHEDULE /ALL/ ;
*SCHEDULE.optfile =1;
OPTIONS  ITERLIM = 2000000, RESLIM=1000000,
     WORK=500000, LIMROW=300, OPTCR=0.01;
SOLVE SCHEDULE USING MIP MINIMIZING OBJ ;

',[]).
```

```
%=============================================================================
% Package: Code_ilp2.
%% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: creates a database file to be read to generate an ilp2 file.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================



% this program generates the database to be ready as an input to the ilp2
% formulation the database is stored in a file named 'in_put',
% if there is no need to run ilp2 for bus allocation,hence no file is produced,
% and start fails.


start:-
tell('~/SCHD_BIND/in_put'),
max_node_id,
cycles(Max),
write(num_of_cycles(Max)),write('.'),nl,

% this statement gets the list of edges that are longer than two cycles, hence
% they need data bus transfer.
% the first findall is for direct edges,
% .....

findall(Edge-Start-End,(Length^edge_length(Edge,Start,End,Length),
   Length > 2),List1_edges),
% the next findall is for the wrap around edges,
%.....

findall(Edge-Start-End,
      Max^wrap_edge_length(Edge,Start,End,Max),
            List2_edges),
append(List1_edges,List2_edges,List_edges),
remove_dups(List_edges,Edges),
assert_edge_num(Edges),
filter_edge(Edges),

% if there are no edges of length greater than two cycles, then there is no
% need for running ilp2 and hence no data bus transfer.
% hence start fails and gives a message to skip running ilp2

( Edges == [] ->
told,
fail
;
setof(Nodes,Old^New^mult(Nodes,Old,New),Nodes),
remove_dups(Nodes,New_Nodes),
number_nodes(New_Nodes),
told
).

% this file is to get the no of operations the whole graph contains.
% ------------------------------------------------------------

max_node_id:-
setof(N,FU^C^nodenum_funcid_cycleno(N,FU,C),Nodes),
last(Max,Nodes),
assert(max_id(Max)).

%-------------------------------------


% this file is to get the no of cycles the whole operation takes.
% ------------------------------------------------------------

cycles(Max):-
setof(C,FU^N^nodenum_funcid_cycleno(N,FU,C),Cycles),
last(Max,Cycles),
assert(num_of_cycles(Max)).

%-------------------------------------
```

```
number_nodes(Nodes):-
number_nodes(Nodes,1).

number_nodes([],_).
number_nodes([Node|Nodes],C):-
assert(node_new(Node,C)),write(node_new(Node,C)),write('.'),nl,
C1 is C + 1,
number_nodes(Nodes,C1).

%---------------------------------------


% this predicate allows to filter the long edges and only take the edges
% that come from multiple output node, and hence deleting the edge that does not
% come from a multiple output node since it does not need bus to transfer data.
%-------------------------------------------------------------------------

filter_edge(List_edges):-
      filter_edges(List_edges,_List_edges,1).

filter_edges([],_,_).
filter_edges([Edge-_-_|List_edges],Lst,C):-
source_edge(Source,Edge,_),
findall(Source-Edges,mult_out_node(Source,Edges),List),
length(List,L),
(L =< 1 ->
   delete_element(Edge-_-_,Lst,NewList),
   C1 is C
;
   assert_mult(Source,Edge),
   C1 is C + 1,
   NewList = Lst) ,
filter_edges(List_edges,NewList,C1).
% this predicate gets the multiple output nodes:
% ----------------------------------------

mult_out_node(Source,Edges):-
num_of_cycles(Max),
source_edge(Source,Edges,_),
(edge_length(Edges,_,_,Length) , Length >=1
;
wrap_edge_length(Edges,_,_,Max)).


assert_mult(Source,Edge):-
old_new(Edge,New),
assert(mult(Source,Edge,New)),
write(mult(Source,Edge,New)),write('.'),nl.
%---------------------------------------


%---------------------------------------

% this predicate renumbers the long edges so that they are fit for the ilp2.
% ----------------------------------------------------------------------

assert_edge_num(List_edges):-
assert_edge_num0(List_edges,1).

assert_edge_num0([],_).
assert_edge_num0([Edge-S-E|List_edges],C):-
edge_st_end(Edge,S,E),
assert(starting(C,S)),write(starting(C,S)),write('.'),nl,
assert(old_new(Edge,C)),write(old_new(Edge,C)),write('.'),nl,
assert(ending(C,E)),write(ending(C,E)),write('.'),nl,
C1 is C + 1,
assert_edge_num0(List_edges,C1).

%---------------------------------------


% this predicate calculates the length of the wrap around edges:
% ----------------------------------------------------

wrap_edge_length(Edge,Start,End,Max):-
source_edge(Source_Node,Edge,_),
      destination_edge(Dest_Node,Edge,_),
      nodenum_funcid_cycleno(Source_Node,_,Start_Cycle),
      nodenum_funcid_cycleno(Dest_Node,_,Term_Cycle),
op_delay(Source_Node,Delay),
```

```
Diff is Term_Cycle - Start_Cycle - Delay,
Diff < 0 ,
Len is Diff + Max + Delay,
Len > 2.
(Term_Cycle == 1 -> End is Max
;
End is Term_Cycle - 1),
(Start_Cycle == Max -> Start is Delay
;
Start is Start_Cycle + Delay),
      assert(edge_st_end(Edge,Start,End)).

%-------------------------------------

% this predicate calculates the length of the direct edge:-
% ----------------------------------------------------------

edge_length(Edge,Start,End,Length):-
source_edge(Source_Node,Edge,_),
destination_edge(Dest_Node,Edge,_),
nodenum_funcid_cycleno(Source_Node,_,Start_Cycle),
nodenum_funcid_cycleno(Dest_Node,_,Term_Cycle),
op_delay(Source_Node,Delay),
Start is Start_Cycle + Delay - 1, End is Term_Cycle ,
Length is Term_Cycle - Start ,
assert(edge_st_end(Edge,Start,End)).

%-------------------------------------

delete_element(_,[],[]).

delete_element(X,[X|L],M):-
    !,
    delete_element(X,L,M).

delete_element(X,[Y|L1],[Y|L2]):-
    delete_element(X,L1,L2).

%=============================================================================
% Package: SBIF.
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: To generate an SBIF file for Safir's tool.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================


:- ensure_loaded(library(basics)).
:- ensure_loaded(library(lists)).
:- ensure_loaded(library(setof)).
:- ensure_loaded(wr_tk).

:- dynamic
      cost_function/2, max_asap/1,
      cost_inter/2,visited/1,alap/2,asap/2,
      maxov/2,numfu/2,opntype/2,
      latency/2,
      cost_overlap/2,
      mergeable/2,
      enable_bus/0, function_index/2,
      max_node_id/1 ,
      incomp/2 ,  optype_futype/2.


% this program produces a file that prints all the information about
% each node and edge and their scheduling and binding, and the information
% about bus allocation, etc...


printlisp(Filename):-
assert(function_index(add,1)),
assert(function_index(mult,2)),
assert(function_index(sub,3)),
assert(function_index(alu,4)),
assert(optype_futype(add,adder)),
assert(optype_futype(mult,multiplier)),
assert(optype_futype(bus,bus_transfer)),
```

```
assert(optype_futype(sub,subtractor)),
maximum_node_id,
findall(Node-Type,nodenum_optype(Node,Type),Nodes),
assert_op_type(Nodes),
(enable_bus ->
compile('~/SCHD_BIND/in_put'),
compile('~/SCHD_BIND/bus.pl')
;
true
),
tell(Filename),
all_opn_bindings(All),
print_member_at_a_time(All),
told.

assert_op_type([]).
assert_op_type([Node-Type|Nodes]):-
nodenum_optype(Node,Type),
function_index(Op,Type),
optype_futype(Op,OpType),
assert(node_operation(Node,OpType)),
assert_op_type(Nodes).

all_opn_bindings(All):-
      \+ enable_bus,
      set_of_all(Out,operation_binding(Out),All).

all_opn_bindings(All):-
      enable_bus,
      set_of_all(Out,operation_binding(Out),All).
      %all_bus_binding(All2),
%append(All1,All2,All).

print_member_at_a_time([]).
print_member_at_a_time([Opn|Rest]):-
      write_tokens:test_write(Opn),
      nl,
      print_member_at_a_time(Rest).

% this predicate gets the information about the node, its type FU used,
% edges leaving it, and edges entering it.
% ----------------------------------------------------------------------

operation_binding([Opn_id,Opn_type,FU,nil,FU_id,EdgesTo,EdgesFrom]):-
      nodenum_funcid_cycleno(Opn_id ,FU_id,_Cycle),
      nodenum_optype(Opn_id,Opn_type),
      function_index(FU,Opn_type),
      edges_to_opn(EdgesTo,Opn_id),
      edges_from_opn(EdgesFrom,Opn_id).


%-----------------------------------------

% This predicate gets the information of the edge entering the node,with
% its source, strting and ending time.
% ----------------------------------------------------------------------


edges_to_opn(EdgesTo,Opn_id):-
  set_of_all([Idst,[Edge_id,Write,Read,Src,SrcFU,Reg_id]],
      Idst^Reg_id^Write^Read^Src^
      edge_to_opn(Opn_id,Read,Write,Src,SrcFU,Idst,Reg_id,Edge_id),
%sorted by Idst
      EdgesTo).
edge_to_opn(Opn_id,Read,Write,Srcx,SrcFU,Idst,Reg_id,Edge_id):-
      destination_edge(Opn_id,Edge_id,Idst),
      source_edge(Src,Edge_id,_Isrc),
      nodenum_funcid_cycleno(Opn_id ,_Fu_id, Read ),
      Reg_id = nil,%until we get that binding
( uses_sb_edge_bus(Edge_id,_Start,Termination,Bus_id,Bus_Opn_id)->
            (
            Srcx = Bus_Opn_id,
            SrcFU = [bus,Bus_id],
            Write  is Termination
            );
            (
            Srcx = Src,
            nodenum_funcid_cycleno(Src,Fu_idsrc, StartSrc),
            nodenum_optype(Src,FU_type_id),
```

```
function_index(Futype,FU_type_id),
optype_futype(Futype,FU),
            SrcFU = [FU,Fu_idsrc],
            op_delay(Src,DelaySrc),
            Write  is StartSrc + DelaySrc - 1
            )
      ).

%-----------------------------------------

maximum_node_id:-
      setof(N,FU^C^nodenum_funcid_cycleno(N , FU,C),Nodes),
      last(Max,Nodes),
      assert(max_node_id(Max)).

% this predicate gives information about bus allocation if there was:-
% ------------------------------------------------------------------
uses_sb_edge_bus(Sb_edge,Start,Termination,Bus_id, Bus_Opn_id):-
      enable_bus,
      old_new(Sb_edge,Bus_edge),
      source_edge(Src,Sb_edge,_Isrc),
      max_node_id(Max),
      Bus_Opn_id is Max + Src,
      edge_start_bus(Bus_edge ,Start ,Bus_id  ),
      edge_end_bus(Bus_edge,Termination,Bus_id  ).

%-----------------------------------------


% this predicate gives information about the edges leaving the node
% with the dest, and starting and ending time.
% ------------------------------------------------------------

edges_from_opn(EdgesFrom,Opn_id):-
  set_of_all([Isrc,EdgesFromIsrc],
      Isrc^setof([Edge_id,Write,Read,Dst,DstFU,Reg_id],
      Reg_id^Write^Read^Dst^DstFU^
edge_from_opn(Opn_id,Read,Write,Dst,DstFU,Isrc,Reg_id,Edge_id),
      EdgesFromIsrc),
      EdgesFrom).

edge_from_opn(Opn_id,Read,Write,Dstx,DstFU,Isrc,Reg_id,Edge_id):-
      source_edge(Opn_id,Edge_id,Isrc),
      destination_edge(Dst,Edge_id,_IDst),
      nodenum_funcid_cycleno(Opn_id , _Fu_id, StartSrc),
            nodenum_optype(Opn_id,FU_type_id),
            op_delay(Opn_id,DelaySrc),
            Write  is StartSrc + DelaySrc - 1,
      Reg_id = nil,%until we get that binding
( uses_sb_edge_bus(Edge_id,Startbus,_Termination,Bus_id,Bus_Opn_id)->
            (
            Dstx =Bus_Opn_id,
            DstFU = [bus,Bus_id],
            Read is Startbus
            );
            (
Dstx = Dst,
            nodenum_funcid_cycleno(Dst,Fu_iddst, StartDst),
node_operation(Dst,DstFUname),
            DstFU = [DstFUname,Fu_iddst],
            Read is StartDst
            )
      ).
%-----------------------------------------


%=============================================================================
% Package: RUNILP
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: this programs interfaces the different steps from the SFG
%   to get an Sbif file.

% copyright (c) 1994, Concordia Univ... All rights reserved.
%=============================================================================


:- ensure_loaded(library(basics)).
:- ensure_loaded(library(lists)).
```

```
:- ensure_loaded(library(setof)).
:- ensure_loaded(wr_tk).
:- ensure_loaded(library(strings)).
:- ensure_loaded(library(read_sent)).

 uesr:runtime(start):-
runilp.
:- dynamic

cost_function/2, max_asap/1,
      cost_inter/2,visited/1,alap/2,asap/2,
      maxov/2,numfu/2,opntype/2,
      latency/2,
      cost_overlap/2,
      mergeable/2,
      enable_bus/0,
      max_node_id/1 ,
      incomp/2 .


runilp:-
runilp1(Dr),
runilp2(Dr).

%=============================================================================
% Package: RUN2ILP
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: this programs interfaces the different steps from the SFG
%        to get an Sbif file.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================
:- ensure_loaded(library(read_sent)).

runilp1(Dr2):-
% this file generates the code for ilp1 and ilp1 is the integer linear
% programming formulation for scheduling and binding the operations together.

      (unix(system('mkdir $HOME/SCHD_BIND')) ->
            true
      ;
            true
),
append("~/SCHD_BIND/","",Home),
      write('Do you want to generate the file for ilp1 ? y/n'),nl,
      read(Ans),
      (Ans == y ->
write(' Enter please the file name of the database '),nl,
            read(Filename),atom_chars(Filename,F1),append(Home,F1,Fil1),
            atom_chars(Fi1,Fil1),
            nl, write(' ENTER PLEASE THE NAME OF THE GENERATED FILE...'),nl,
            read(Outfile),atom_chars(Outfile,F2),append(Home,F2,Fil2),
atom_chars(Fi2,Fil2),
            consult(Fi1),
            write(' The ILP1 file is generated now .....'),nl,
            nl,nl,nl,
            generate_code(Fi2)
      ;
            write(' ENTER PLEASE THE NAME OF THE GENERATED FILE...'),nl,
            read(Outfile),atom_chars(Outfile,Fil2),
            append(Home,Fil2,F2),
            atom_chars(Fi2,F2)
      ),
      write('Do you want to run gams for ILP1 (Scheduling & binding? [y/n]'),nl,
      read(Answ),
      %atom_chars(Outfile,Ofile),
      unix(system('pwd > this_dir')),
      see(this_dir),
      read_line(Dir),close(this_dir),unix(system('rm this_dir')),
      append(Dr1,[10],Dir),append(Dr1,"/",Dr2),
      (Answ == y ->
            write('The program now is running gams to File '),
            write(Outfile),
            write('.....'),nl,nl,nl,nl,
            write('The output of the file will be in the '),
            write(Outfile),
            write('.lst.....'),nl,nl,nl,nl,nl,nl,
```

```
        write('The program generates the scheduling and binding to the '
),
        write('operations.....'),nl,nl,nl,nl,

        write(' ILP for scheduling and Binding is running '),nl,
        write(' ----------Waiting for results----------'),nl
,
        write(' If your graph is very intricate, then running gams takes
 time'),
        nl,nl,nl,

% logging remotely to chloe to be able to run gams on the file that was
% produced by the command generate_code.

        write(' Enter please your username in the ece account....'),nl,
        read(Name),
        atom_chars(Name,Nom),
        append("rlogin -l ", Nom, Tcmp),
        append(Tcmp," chloe ",NewCom),atom_chars(Lcom,NewCom),
        unix(system(Lcom))
    ;
        true
    ),
        atom_chars(Outfile,Ofile),append("$HOME/SCHD_BIND/",Ofile,Fr),
        append(Fr,".lst",File),
        append(Dr2,"cleanSB ",TempCommand),
        append(TempCommand,File,Com),
        atom_chars(Command,Com),
        unix(system(Command)).



%=============================================================================
% Package: RUNILP
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose: this programs interfaces the different steps from the SFG
%       to get an Sbif file.

% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================
:-ensure_loaded(library(environ)).


runilp2(Dr2):-
rdb,
% Now data_ilp2.pl is to arrange the  data for the input for ILP2 or for the
% sbif file.
compile('~/SCHD_BIND/op.pl'),
compile('~/SCHD_BIND/delay.pl'),
compile('~/SCHD_BIND/dest.pl'),
compile('~/SCHD_BIND/source.pl'),
    (start ->
        write('Do you to generate a file to run ILP2 ? [y/n]'),
        nl,
        read(Anse),
        write('Enter the name of the file to be run for ilp2'),nl,
        read(Filme),atom_chars(Filme,Film1),
        environ('HOME',HOMt),atom_chars(HOMt,HOME),
        append(HOME,"/SCHD_BIND/",SCHD_BIND),
        append(SCHD_BIND,Film1,Film2),
        atom_chars(Filw2,Film2),
        (Anse == y ->
            generate(Filw2),
            write('enter please your username again in the ece accou
nt....'),
            nl,nl,nl, read(Lcom),
            atom_chars(Lcom,User),
            append("rlogin -l ",User,Commd),
            append(Commd," chloe ",Tcom),
            atom_chars(Fcom,Tcom),
            unix(system(Fcom))
        ;
            true
        ),
        assert(enable_bus),
        append(Dr2,"cleanBUS ",TmpC),
        append(Film2,".lst",Lst_com),
```

```
        append(TmpC,Lst_com,Com2),
        atom_chars(Com,Com2),
        unix(system(Com))
    ;
        nl,nl,write('.....NO ILP2.....'),nl,
        write(' No need to run ilp2 --> no bus transfer....'),nl,
        nl,nl,nl,nl,nl
    ),
    write('Enter the name of the sbif file '),nl,
    read(Nam),atom_chars(Nam,NamL),append("~/SCHD_BIND/",NamL,PrintL),
atom_chars(Name,PrintL),
    printlisp(Name).


%  Package: writetokens.pl
%  Author : Richard A. O'Keefe
%  Updated: 8/29/89
%  Purpose: Convert a term to a list of tokens.

%  Adapted from shared code written by the same author; all changes
%  Copyright (C) 1989, Quintus Computer Systems, Inc.  All rights reserved.

:- module(write_tokens, [
term_to_tokens/2,
term_to_tokens/4
  ]).

sccs_id('"@(#)89/08/29 writetokens.pl33.1"').

/* This is essentially the same as the public-domain write.pl, except
   the instead of writing characters to the current ouput stream, it
   returns a list of tokens.  There are three kinds of tokens:
   (1) punctuation marks, which are represented by atoms.
'('
'['
'{'
   (2) constants, wrapped around to say what they are
atom(Atom)
integer(Integer)
float(Float)
var(Var)
   (3) atoms used other ways have other wrappers
functor(Functor)% THE ( IS IMPLICIT IN THIS TOKEN!!!
prefix(Operator)
infix(Operator)
postfix(Operator)
   Note that ',' and ';', when used as operators, will be reported
   as infix(_) tokens.

   There is nothing to indicate spacing; the point of this package is
   to let the caller do such formatting.
*/

%  term_to_tokens(+Term, ?Tokens)
%  unifies Tokens with a list of tokens which represent the Term.
%  Extra parentheses are added just as write/1 would add them.
%  This predicate is steadfast in Tokens.

term_to_tokens(Term, Tokens) :-
term_to_tokens(Term, 1200, Tokens, []).

term_to_tokens(Term, Priority) -->
(   {var(Term)} ->[var(Term)]
;   {integer(Term)} ->[integer(Term)]
;   {float(Term)} ->[float(Term)]
;   {atom(Term)} ->
    ({prefix(Term, P, _), P > Priority} ->
['(',atom(Term),')']
    ;/* not prefix operator, or in priority range */
[atom(Term)]
    )
;   {functor(Term, F, N)},
    ({N =:= 1} ->
        unary_to_tokens(F, Term, Priority)
    ;{N =:= 2} ->
binary_to_tokens(F, Term, Priority)
    ;  /* there is no special syntax for terms with >2 arguments */
[functor(F)],% THE ( IS IMPLICIT IN THIS TOKEN!!!
```

```
arguments_to_tokens(1, N, Term)
    )
).


%. arguments_to_tokens(+I, +N, +Term)(S0,S)
%  converts the Ith to Nth arguments of Term to tokens, adding
%  appropriate punctuation.

arguments_to_tokens(I, N, Term) --> !,
{arg(I, Term, Arg)},
term_to_tokens(Arg, 999),
(  {I < N} ->[','],
   {J is I+1},
   arguments_to_tokens(J, N, Term)
;  /* finished */ [')']
).


%. unary_to_tokens(+Functor, +Term, +Priority)(S0,S)
%  converts a unary term to a sequence of tokens.  In addition to
%  handling user-defined operators and the {_} form, it watches out
%  for a nasty little special case:  we do NOT want -(4) to be written
%  out as -4; that's quite a different term!

unary_to_tokens(-, -(N), _) --> {number(N)}, !,
[functor(-)], term_to_tokens(N, 999), [')'].
unary_to_tokens({}, {Term}, _) --> !,
['{'], term_to_tokens(Term, 1200), ['}'].
unary_to_tokens(F, Term, Priority) -->
{arg(1, Term, Arg)},
(  {prefix(F, O, Q)} ->
   (  {O > Priority} ->
['(',prefix(F)], term_to_tokens(Arg, Q), [')']
   ;     [prefix(F)], term_to_tokens(Arg, Q)
   )
;  {postfix(F, P, O)} ->
   (  {O > Priority} ->
['('], term_to_tokens(Arg, P), [postfix(F),')']
   ;     term_to_tokens(Arg, P), [postfix(F)]
   )
;  [functor(F)], term_to_tokens(Arg, 999), [')']
).


%. binary_to_tokens(+Functor, +Term, +Priority)(S0,S)
%  converts a binary term to a sequence of tokens.  In addition to
%  handling user-defined operators, it handles the bracket notation
%  for lists.  The public-domain version of write/1 also has a case
%  here to catch (A,B) so that the comma here is written as a plain
%  ",", not as a quoted atom "','" as using infix(',') might do.
%  I chose to omit that case here.  [Code available on request.]
%  Sorry about the layout in the second clause, but I wanted to make
%  the relation between the with-parens and the without-parens cases clear.

binary_to_tokens('.', [Head|Tail], _) --> !,
['['],
term_to_tokens(Head, 999),
tail_to_tokens(Tail).
binary_to_tokens(F, Term, Priority) -->
{infix(F, P, O, Q)},
!,
{arg(1, Term, A)},
{arg(2, Term, B)},
(  {O > Priority} ->
   ['('],term_to_tokens(A, P),[infix(F)],term_to_tokens(B, Q),[')']
;     term_to_tokens(A, P),[infix(F)],term_to_tokens(B, Q)
).
binary_to_tokens(F, Term, _) -->
[functor(F)],
arguments_to_tokens(1, 2, Term).


%. tail_to_tokens(Tail)(S0,S)
%  converts the tail of a list to a sequence of tokens.  This Tail
%  was preceded by a Head, which has been converted.

tail_to_tokens(Var) --> {var(Var)}, !,
['|',var(Var),']'].
```

```
tail_to_tokens([]) --> !,
[']'].
tail_to_tokens([Head|Tail]) --> !,
[','],
term_to_tokens(Head, 999),
tail_to_tokens(Tail).
tail_to_tokens(Term) -->
['|'],
term_to_tokens(Term, 999).


%. The original public-domain code was written to go with a similarly
%  public-domain version of op/3 and current_op/3 where the following
%  three tables were the primary reality. Whether they are or aren't,
%  only current_op/3 is (currently) directly available to customers.

prefix(F, O, Q) :-
(  current_op(O, fx, F) -> Q is O-1
;  current_op(O, fy, F) -> Q is O
).

postfix(F, P, O) :-
(  current_op(O, xf, F) -> P is O-1
;  current_op(O, yf, F) -> P is O
).

infix(F, P, O, Q) :-
(  current_op(O, xfy, F) -> P is O-1, Q is O
;  current_op(O, xfx, F) -> P is O-1, Q is P
;  current_op(O, yfx, F) -> Q is O-1, P is O
).


%  test_write/1, write_tokens/1, and write_token/1 are a test harness to
%  ensure that this file is working properly.
%  This provides a very rough and inefficient first approximation to an
%  output routine which takes line length into account. The things it
%  doesn't do include reducing unnecessary spaces between operators and
%  operands when no confusion would result, getting the length right in
%  the presence of quoting, atoms which are too big to fit no matter
%  what you do (the answer is quote, use escapes, and insert \<newline>).
%  Basically, what we want is a "write_token" primitive down at the C
%  level somewhere.

test_write(Term) :-
term_to_tokens(Term, Tokens),
current_output(Stream),
line_position(Stream, Column),
write_tokens(Tokens, Column).


write_tokens([], _).
write_tokens([Token|Tokens], C0) :-
write_token(Token, C0, C1),
write_tokens(Tokens, C1).

write_token(Token, C0, C1) :-
token_length(Token, Length),
(  C0+Length > 75 ->
   C1 is Length,
   nl,
   write_token(Token)
;  C1 is C0+Length,
   write_token(Token)
).

token_length('(', 1).
token_length(',', 1).
token_length(')', 1).
token_length('[', 1).
token_length('|', 1).
token_length(']', 1).
token_length('{', 1).
token_length('}', 1).
token_length(var(_), 8).% sometimes an over-estimate
token_length(integer(X), L):- constant_length(X, 0, L).
token_length(float(X), L):- constant_length(X, 0, L).
token_length(atom(X), L):- constant_length(X, 0, L).
token_length(prefix(X), L):- constant_length(X, 1, L).
```

```
token_length(infix(X), L):- constant_length(X, 2, L).
token_length(postfix(X), L):- constant_length(X, 1, L).
token_length(functor(X), L):- constant_length(X, 1, L).

constant_length(Constant, L0, L1) :-
name(Constant, Chars),
length(Chars, L),
L1 is L0+L.

write_token('('):- put("(").
write_token(','):- put(",").
write_token(')'):- put(")").
write_token('['):- put("(").
write_token('|'):- put("|").
write_token(']'):- put(")").
write_token('{'):- put("{").
write_token('}'):- put("}").
write_token(var(X)):- write(X).
write_token(integer(X)):- write(X).
write_token(float(X)):- write(X).
write_token(atom([])):- !, put("("),put(")").
write_token(atom(X)):- write(X).
write_token(prefix(X)):- write(X), put(" ").
write_token(infix(X)):- put(" "), write(X), put(" ").
write_token(postfix(X)):- put(" "), write(X).
write_token(functor(X)):- write(X), put("(").

end_of_file.
```

```
%=============================================================================
% Package : GUI
% Authors: S. Hijazie and B. Haroun.
% Updated: 12/16/94
% Purpose : provides a graphical interface for dsp circuits, and produces
% an output file the discribes the SFG.
% copyright (c) 1994, Concordia Univ... All rights reserved.

%=============================================================================
  %creating the main
%-----------------------------

:- module(gui,[
        gui/0
    ]).
:- ensure_loaded(library(proxt)).
:- ensure_loaded(library(proxl)).
:- ensure_loaded(library(xif)).
:- ensure_loaded(library(basics)).
:- ensure_loaded(library(strings)).

:- dynamic
        selected_region/4,
        last_element/1,
        node1/1,erased_group/2,
        node2/1,erased_lines/2,
        delay_element/2,erased_connection/2,
        path/4,looping/1,group_ports/2,
        connection/4,
        exit_loop/1,relate/3,
        draw_mode/2,lines_connected/7,
        ending_point/2,
        drawn_figure/3,
        list_data/5,erased/5,
        data/6,group/2,group_connection/2,
        element/2,port/3,group_line/2,
        delay_data/5,group/2,fu_data/5,
        starting_point/2,
 connection_list/1,
 in1/3,in2/3,out/3,line/4,
 data_figure/5,line/4,
        drawing_window/9.

  user:runtime_entry(start) :-
        gui.
 gui :-
      initialize_state,
      window_data(title,Label),
      window_data(vertical_space,VerticalSpace),
      window_data(horizontal_space,HorizontalSpace),
```

```
        window_data(height,DrawHeight),
        window_data(width,DrawWidth),
        window_data(boundary_height,BoundaryHeight),
        window_data(menu_option_size,MenuBoxSize),
        find_font(titlefont,TitleFont),
        find_font(menu_font,MenuFont),
        find_font(box_font,BoxFont),
        get_font_attributes(TitleFont,[height(FontHeight)]),
        get_font_attributes(BoxFont,[height(BoxFontHeight)]),
        create_rubberband_gc(RubberBandGC,MenuFont),
        text_extents(TitleFont,Label,_,_,LabelWidth,Ascent,_),
        BoxWidth is LabelWidth // 6,
        MinTitleWidth is LabelWidth + 200,
        MenuWidth is 4 * HorizontalSpace + 2*MenuBoxSize,
        MinWidth is DrawWidth + MenuWidth,
        max(MinTitleWidth,MinWidth,Width),
        WindowWidth is Width + HorizontalSpace // 2,
        BoxHeight is 2 * VerticalSpace + BoxFontHeight,
        TitleHeight is 2 * VerticalSpace + FontHeight,
        RootHeight is DrawHeight + BoundaryHeight + TitleHeight,
        create_drawing_gcs(DrawingGC,InvertingGC,FgPixel,BgPixel,
                           MenuFont),
%xif_initialize(test,test,Shell),
%xtCreateWidget(shell,widgetClass,Shell,[],RootWidget),
%widget_window(RootWidget,RootWin),
        create_root_window(RootWin,WindowWidth,RootHeight,RootCursor),
        create_title_window(TitleWin,WindowWidth,TitleHeight,LabelWidth,
                            Ascent,RootWin,TitleFont),
create_drawing_window(DrawWin,DrawWidth,DrawHeight,TitleHeight,
                      RootWin,RubberBandGC,DrawingGC),
        create_exit_box(ByeBox,TitleWin,WindowWidth,BoxWidth,
                        BoxHeight,Ascent,
                        TitleHeight,
                        BoxFont,RootWin),
        create_menu_window(MenuWin,MenuWidth,DrawHeight,
                           DrawWidth,TitleHeight,RootWin,
                           RootCursor,DrawWin,DrawingGC,
                           InvertingGC,FgPixel,BgPixel),
        put_window_attributes(RootWin,[mapped(true)]),
        assert(drawing_window(RootWin,WindowWidth,RootHeight,DrawWin,
                              DrawWidth,DrawHeight,
                              MenuWin,TitleWin,ByeBox)),
write('Is it a new file ?'),nl,
read(Answer),
(Answer == no ->
    open_and_draw(DrawWin,DrawingGC)
  ;
    true),
%xif_main_loop(looping(yes)).
handle_events(looping(yes)).



%creating the windows:
%_____________________

% This is the rootwindow and it contains all the windows created in this
% software.
%          ____________________________________________________

  create_root_window(RootWin,RootWidth,RootHeight,RootCursor):-
      window_data(root_cursor,RootCursorName),
      create_cursor(RootCursorName,RootCursor),
      create_window(RootWin,
                    [mapped(true),
                     size(RootWidth,RootHeight),
                     position(100,100),
                     border_width(2),
                     property('WM_NAME','GUI'),
                     cursor(RootCursor),
                     callback(configure_notify,[size(W,H)],
                          new_root_size(RootWin,W,H))]).

%          ____________________________________________________

% creating the title window:
%___________________________
% This is the title window that displays the title of the graphics user
```

```
% interface.
%          ------------------------------------------------------

   create_title_window(TitleWin,TitleWidth,TitleHeight,LabelWidth,
                       Ascent,RootWin,Font):-
      create_window(TitleWin,[position(0,0),size(TitleWidth,TitleHeight),
                              border_width(1),parent(RootWin),
                              mapped(true),
                              callback(expose,[count(0)],
                                    title_head(TitleWin,
                                          LabelWidth,Ascent))],
                              [font(Font)]).
%          ------------------------------------------------------

% creating the exit box:
% ----------------------:
% This is the box that allows the user to exit from the gui.
%          ------------------------------------------------------

   create_exit_box(ByeBox,TitleWin,TitleWidth,BoxWidth,BoxHeight,Ascent,
                   TitleHeight,BoxFont,RootWin):-
      window_data(box_size,BoxSize),
      BoxX is TitleWidth - 2*BoxSize ,
      BoxN is BoxSize//2,
      BoxY is (TitleHeight -  BoxN)//3,
      create_window(ByeBox,[position(BoxX,BoxY),
                            size(BoxWidth,BoxHeight),
                            border_width(1),
                            parent(TitleWin),
                            win_gravity(north_east),
                            mapped(true),
                            callback(expose,[count(0)],
                                  title_box(ByeBox,BoxN,
                                        Ascent)),
                            callback(button_press,[],
                                  looping(yes),
                                  goodbye(RootWin))],
                            [font(BoxFont)]).
%          ------------------------------------------------------

% creating the drawing window:
% ----------------------------

% This is the place where all the drawing takes place:
% You can draw an adder, multiplier,a delay element and connect the nodes.
%          ------------------------------------------------------------------

   create_drawing_window(DrawWin,WindowWidth,WindowHeight,TitleHeight,
                         RootWin,RubberBandGC,DrawingGC) :-
      window_data(boundary_height,BoundaryHeight),
      window_data(draw_cursor,CursorName),
      DrawWinY is TitleHeight + BoundaryHeight,
      create_cursor(CursorName,DrawCursor),
 asserta(selected_region(0,0,0,0)),
      create_window(DrawWin,[size(WindowWidth,WindowHeight),
                             position(0,DrawWinY),border_width(0),
                             parent(RootWin),
                             cursor(DrawCursor),
                             gc(DrawingGC),
                             mapped(true),
                             callback(expose,[count(0)],
                                   refresh(DrawWin,DrawingGC)),
                             callback(button_press([2]),[position(X,Y)],
                                   draw_button(DrawWin,X,Y)),
                             callback(button_press([1]),
                                   [position(X,Y)],
                                   start_point(X,Y)),
                             callback(motion_notify([1]),
                                   [position(X,Y)],
                                   new_point(DrawWin,
                                         RubberBandGC,X,Y)),
                             callback(button_press([3]),
                                   [position(X,Y)],
                                   label_at(DrawWin,X,Y)),
                             callback(button_release([1]),
                                   [position(X,Y)],
                                   end_point(DrawWin,RubberBandGC,
                                         DrawingGC,X,Y)),
```

```
            callback(configure_notify,[],
                  new_draw_size(DrawWin))]).
%        ____________________________________________-

%creating the menu window:
%_________________________
%This window allows to display the choice of event the user picks.
% it could be either add or delete or move one of the proeviously mentioned
% elements.
%    ______________________________________________________

  create_menu_window(MenuWin,MenuWidth,MenuHeight,X,Y,
            RootWin,RootCursor,DrawWin,
            DrawingGC,InvertingGC,FgPixel,BgPixel):-
    window_data(boundary_height,BoundaryHeight),
    window_data(horizontal_space,HorizontalSpace),
    window_data(vertical_space,VerticalSpace),
    window_data(menu_option_size,MenuOptionSize),
    MenuY is Y + BoundaryHeight,
    create_window(MenuWin, [size(MenuWidth,MenuHeight),
                position(X,MenuY),
                parent(RootWin),
                cursor(RootCursor),
                border_width(1),
                background(BgPixel),
                mapped(true)]),
    FirstColumnX is HorizontalSpace,
    SecondColumnX is FirstColumnX + MenuOptionSize + 2* HorizontalSpace,
    FirstRowY is 2* VerticalSpace,
    RowInc is 2*VerticalSpace + MenuOptionSize,
    create_menu_options([[adder,multiplier],
              [delay1,alu],
  [subtract,fu],
              [connect,delete],
              [undo,c],
              [m,g],
  [s]],
             MenuWin,
             [FirstColumnX,SecondColumnX],FirstRowY,RowInc,
             MenuOptionSize,DrawWin,DrawingGC,InvertingGC,
             FgPixel,BgPixel).

%    ____________________________________________

% creating menu options:
%_________________________

  create_menu_options([],_,_,_,_,_,_,_,_,_,_).
  create_menu_options([RowOptions|Options],MenuWin,Columns,Row,RowInc,
            Size,DrawWin,DrawingGC,InvertingGC,FgPixel,BgPixel):-
     create_menu_row(RowOptions,MenuWin,Columns,Row,Size,DrawWin,
            DrawingGC,InvertingGC,FgPixel,BgPixel),
     NextRow is Row + RowInc,
     create_menu_options(Options,MenuWin,Columns,NextRow,RowInc,Size,
            DrawWin,DrawingGC,InvertingGC,FgPixel,BgPixel).

%      ______________________________________________

% creating menu row:
%_____________________

  create_menu_row([],_,_,_,_,_,_,_,_,_).
  create_menu_row([Option|RowOptions],MenuWin,[Column|Columns],Row,
          Size,DrawWin,DrawingGC,InvertingGC,FgPixel,BgPixel):-
     Highlight = highlight(Option,Window,Size,Size,InvertingGC,FgPixel),
     Select   = select(Option,Window,Size,Size,X,Y,DrawWin,
             DrawingGC,BgPixel),
     Redraw   = redraw(Option,Window,Size,Size),
     Ignore = unhighlight(Option,Window,Size,Size,DrawingGC,BgPixel),
     Button1 = buttons(down,_,_,_,_),
     create_window(Window,[size(Size,Size),
                position(Column,Row),
                parent(MenuWin),
                border_width(1),
                gc(DrawingGC),
                background(BgPixel),
                callback(button_press([1]),[],Highlight),
```

```
                    callback(button_release([1]),[position(X,Y)],
                        Select),
                    callback(expose,[count(0)],Redraw),
                    callback(leave_notify,[state(Button1,_)],Ignore),
                    mapped(true)]),
    default_setup(Option,Window,InvertingGC,FgPixel),
    create_menu_row(RowOptions,MenuWin,Columns,Row,Size,DrawWin,
            DrawingGC,InvertingGC,FgPixel,BgPixel).

%       ------------------------------------------------------

% The implementing of main callbacks( to handle event):
% ------------------------------------------------------

% 1. new_root_size:
%-----------------
  new_root_size(Root,NewWidth,NewHeight):-
    drawing_window(Root,OldWidth,OldHeight,DrawWin,OldDrawWidth,
            OldDrawHeight,MenuWin,TitleWin,ByeBox),
    ( NewWidth =:= OldWidth, NewHeight =:=OldHeight -> true
    ; put_window_attributes(TitleWin,[width(NewWidth)]),
      window_data(box_size,BoxSize),
      BoxX is NewWidth -2*BoxSize,
      put_window_attributes(ByeBox,[x(BoxX)]),
      DeltaW is NewWidth - OldWidth,
      DeltaH is NewHeight - OldHeight,
      DrawWidth is OldDrawWidth + DeltaW,
      DrawHeight is OldDrawHeight + DeltaH,
      put_window_attributes(DrawWin,[size(DrawWidth,DrawHeight)]),
      put_window_attributes(MenuWin,[x(DrawWidth),height(DrawHeight)])
    ).
%       ---------------------------------------------


% 2.1 title for the Heading:
%--------------------------

  title_head(Windows,LabelWidth,Ascent):-
      window_data(vertical_space,VerticalSpace),
      window_data(title,Label),
      get_window_attributes(Windows,[width(TitleWidth)]),
      TitleX is (TitleWidth - LabelWidth)//2,
      TitleY is VerticalSpace + Ascent,
      draw_string(Windows,TitleX,TitleY,Label).
%         ---------------------------------------------

% creating interactive box:
%--------------------------


% 2.2 title for the exit box:
%--------------------------

  title_box(Windows,NameWidth,Ascent):-
      window_data(vertical_space,VerticalSpace),
      window_data(box_name,Name),
      get_window_attributes(Windows,[width(BoxWidth)]),
      TitleX is (( BoxWidth - NameWidth ) // 2 - 10),
      TitleY is (VerticalSpace + Ascent) //2,
      draw_string(Windows,TitleX,TitleY,Name).
%         ---------------------------------------------
% 3. REfresh:
%------------

   refresh(DrawWin,DrawingGC):-

      clear_window(DrawWin),
findall(Figure-Label-X1-Y1-X2^Y2^(drawn_figure(element(Figure,Label),
            info(X1,Y1,X2,Y2),_),Figure \== rectangle),
    List_drawns),
draw_them(List_drawns,DrawWin,DrawingGC).


draw_them([],_,_).
draw_them([Figure-Label-X1-Y1|List_drawns],DrawWin,DrawingGC):-
      drawn_figure(element(Figure,Label),info(X1,Y1,X2,Y2),_),
(( Figure == adder ; Figure == subtract ; Figure == multiplier ;
  Figure == delay1 ; Figure == alu_adder ; Figure == alu_sub ; Figure == fu ) ->
          drawing_figure(element(Figure,Label),
```

```
                DrawWin,DrawingGC,X1,Y1,X2,Y2))
  ;
( Figure == connect ->
     draw_figure(connect,DrawWin,DrawingGC,X1,Y1,X2,Y2))
  ;
( Figure == in1 ->
     draw_port1(DrawWin,DrawingGC,Label,X1,Y1))
  ;
( Figure == in2 ->
     draw_port2(DrawWin,DrawingGC,Label,X1,Y1))
  ;
( Figure == out ->
     draw_port3(DrawWin,DrawingGC,Label,X1,Y1))),
     draw_them(List_drawns,DrawWin,DrawingGC).


  draw_port1(DrawWin,GC,Label,Xs,Ys):-
Xi is Xs - 60, Xm is Xi + 40,
     Yi is Ys - 10, Xin is Xi + 5, Yin is Yi +20,
%write(in1(Label,Xi,Yi)),write('.'),nl,
     draw_rectangle(DrawWin,GC,Xi,Yi,40,20),
     draw_line(DrawWin,GC,Xm,Ys,Xs,Ys),
     draw_string(DrawWin,GC,Xin,Yin,Label).


  draw_port2(DrawWin,GC,Label,Xs,Ys):-
Xi is Xs - 90, Xm is Xi + 40,
     Yi is Ys - 10, Xin is Xi + 5, Yin is Yi +20,
%write(in2(Label,Xs,Ys)),write('.'),nl,
     draw_rectangle(DrawWin,GC,Xi,Yi,40,20),
     draw_line(DrawWin,GC,Xm,Ys,Xs,Ys),
     draw_string(DrawWin,GC,Xin,Yin,Label).

  draw_port3(DrawWin,GC,Label,Xs,Ys):-
Xi is Xs + 20, Yi is Ys - 10,Xin is Xi + 5,Yin is Yi + 20,
%write(out(Label,Xi,Yi)),write('.'),nl,
draw_rectangle(DrawWin,GC,Xi,Yi,40,20),
draw_line(DrawWin,GC,Xi,Ys,Xs,Ys),
draw_string(DrawWin,GC,Xin,Yin,Label).


%       ---------------------------------------------
% label the input:
%---------------

  label_at(DrawWin,X,Y):-
   (  near_end_point([Xs,Ys],X,Y)->
 get_window_attributes(DrawWin,[gc(GC)]),
     (( data(element(Mode,Name),_,_,ending_point(Xs,Ys),
                ending_point(_,_),
                ending_point(_,_)) ->
        Node is 1,
    gensym(i,Label),
    asserta(in1(Label,Xs,Ys)),
    write(in1(Label,Xs,Ys)),write('.'),nl,
    assertz(drawn_figure(element(in1,Label),info(Xs,Ys,_,_),_)),
    draw_port1(DrawWin,GC,Label,Xs,Ys),
    asserta(list_data(elemenet(in1,Label),Xs,Ys,_,_))
      ; data(element(Mode,Name),_,_,ending_point(_,_),
                ending_point(Xs,Ys),
                ending_point(_,_)) ->
        Node is 2,
    gensym(i,Label),
    asserta(in2(Label,Xs,Ys)),
        write(in2(Label,Xs,Ys)),write('.'),nl,
    assertz(drawn_figure(element(in2,Label),info(Xs,Ys,_,_),_)),
    draw_port2(DrawWin,GC,Label,Xs,Ys),
    asserta(list_data(element(in2,Label),Xs,Ys,_,_))
      ;  data(element(Mode,Name),_,_,ending_point(_,_),
                ending_point(_,_),
                ending_point(Xs,Ys)) ->
        Node is 3,gensym(o,Label),
    asserta(out(Label,Xs,Ys)),
        write(out(Label,Xs,Ys)), write('.'),nl,
    assertz(drawn_figure(element(out,Label),info(Xs,Ys,_,_),_)),
    draw_port3(DrawWin,GC,Label,Xs,Ys),
    asserta(list_data(element(out,Label),Xs,Ys,_,_))
      )
```

```
      ;
       (delay_data(element(Mode,Name),_,_,ending_point(Xs,Ys),
                   ending_point(_,_)) ->
         Node is 1,gensym(i,Label),
    asserta(in1(Label,Xs,Ys)),
    write(in1(Label,Xs,Ys)),write('.'),nl,
    assertz(drawn_figure(element(in1,Label),info(Xs,Ys,_,_),_)),
    draw_port1(DrawWin,GC,Label,Xs,Ys),
    asserta(list_data(element(in1,Label),Xs,Ys,_,_))
       ;
        delay_data(element(Mode,Name),_,_,ending_point(_,_),
                   ending_point(Xs,Ys)) ->
         Node is 2,gensym(o,Label),
    asserta(out(Label,Xs,Ys)),
    write(out(Label,Xs,Ys)), write('.'),nl,
    assertz(drawn_figure(element(out,Label),info(Xs,Ys,_,_),_)),
    draw_port3(DrawWin,GC,Label,Xs,Ys),
    asserta(list_data(element(out,Label),Xs,Ys,_,_))
       )),
       asserta(port(Name,Node,Label)),
       write(port(Name,Node,Label)),write('.'),nl
    ;
true
    ).
% 4. start_point:
%----------------
    start_point(X,Y):-
      asserta(selected_region(X,Y,X,Y)),
      draw_mode(ModeX,_),
      (ModeX == connect ->
       ( near_end_point([Xs,Ys],X,Y)->
        (( data(element(Mode,Name),_,_,ending_point(Xs,Ys),
                   ending_point(_,_),
                   ending_point(_,_)) ->
          Node is 1
         ; data(element(Mode,Name),_,_,ending_point(_,_),
                   ending_point(Xs,Ys),
                   ending_point(_,_)) ->
          Node is 2
         ;  data(element(Mode,Name),_,_,ending_point(_,_),
                   ending_point(_,_),
                   ending_point(Xs,Ys)) ->
          Node is 3
        )
        ;
         ( delay_data(element(Mode,Name),_,_,ending_point(Xs,Ys),
                   ending_point(_,_)) ->
          Node is 1
         ;
          delay_data(element(Mode,Name),_,_,ending_point(_,_),
                   ending_point(Xs,Ys)) ->
          Node is 2
        )
        ),
        retract(selected_region(X,Y,X,Y)),
        asserta(node1(Node)),
        asserta(selected_region(Xs,Ys,Xs,Ys)),
  asserta(lines_connected(Mode,Name,Node,[Xs-Ys],Mode,Name,Node)),
        asserta(path(element(Mode,Name),node1(Node),_,_))

        ;
  (in_line([Xi,Yi,Xj,Yj],X,Y) ->
  (line(Xi,Yi,Xj,Yj),
  lines_connected(ModeI,NameI,Node1,List,ModeJ,NameJ,NodeJ),
  a_member(Xi-Yi,List),a_member(Xj-Yj,List)),
  copy_list(Xj-Yj,List,NList),
  check_for_connections(ModeI,NameI,Node1,ModeJ,NameJ,NodeJ,NList))
        ;
         true
        )
        ;
       ModeX == m ->
  (in_region([Xi,Yi],X,Y) ->
       ( retract(selected_region(X,Y,X,Y)),
   asserta(selected_region(Xi,Yi,Xi,Yi)))
         ;
          true
  )
        ;
       ModeX == g -> true
```

```
    ).

%     ------------------------------------------------

check_for_connections(ModeI,NameI,NodeI,ModeJ,NameJ,NodeJ,Last):-
connection(element(ModeI,NameI),NodeI,element(ModeJ,NameJ),NodeJ),
(((NodeI == 1);(NodeI == 2, ModeI \== delay1)) ->
   asserta(path(element(ModeJ,NameJ),node1(NodeJ),_,_)),
   asserta(lines_connected(ModeJ,NameJ,NodeJ,Last,_,_,_))
;
   asserta(path(element(ModeI,NameI),node1(NodeI),_,_)),
   asserta(lines_connected(ModeI,NameI,NodeI,Last,_,_,_))
).

copy_list(Xj-Yj,[Xj-Yj|_],_).
copy_list(Xj-Yj,[Xs-Ys|List],NList):-
insert(Xs-Ys,NList,NewList),
copy_list(Xj-Yj,List,NewList).


% 5. end_point:
%--------------

   end_point(DrawWin,RubberBandGC,DrawingGC,X,Y):-
      selected_region(SX,SY,LX,LY),
      draw_mode(Mode,_),
      (Mode == connect ->
         (near_end_point([Xl,Yl],X,Y) ->
            ( (data(element(Mode1,Name),_,_,ending_point(Xl,Yl),
                        ending_point(_,_),
                        ending_point(_,_)) ->
               Node is 1
             ; data(element(Mode1,Name),_,_,ending_point(_,_),
                        ending_point(Xl,Yl),
                        ending_point(_,_)) ->
               Node is 2
             ; data(element(Mode1,Name),_,_,ending_point(_,_),
                        ending_point(_,_),
                        ending_point(Xl,Yl)) ->
               Node is 3
            )
            ;
             (delay_data(element(Mode1,Name),_,_,
                        ending_point(Xl,Yl),
                        ending_point(_,_)) ->
               Node is 1
             ;
              delay_data(element(Mode1,Name),_,_,
                        ending_point(_,_),
                        ending_point(Xl,Yl)) ->
               Node is 2
   )
            ;
               true
 ),
lines_connected(ModeI,NameI,Node1,List,ModeJ,NameJ,NodeJ),
insert(Xl-Yl,List,NewList),
asserta(lines_connected(ModeI,NameI,Node1,NewList,
                  Mode1,Name,Node)),
retract(lines_connected(ModeI,NameI,Node1,List,
                  ModeJ,NameJ,NodeJ)),
      draw_rubberband(connect,DrawWin,RubberBandGC,SX,SY,Xl,Yl),
            draw_figure(connect,DrawWin,DrawingGC,SX,SY,Xl,Yl),
            asserta(line(SX,SY,Xl,Yl)),
asserta(list_data(element(connect,_),SX,SY,Xl,Yl)),
write(line(SX,SY,Xl,Yl)),write('.'),nl,
assertz(drawn_figure(element(connect,_),info(SX,SY,Xl,Yl),_)),
            asserta(node2(Node)),
(path(element(ModeI,NameI),node1(Node1),_,_) ->
               asserta(connection(element(ModeI,NameI),
Node1,element(Mode1,Name),Node)),
               write('connection('),
               write(NameI),write(','),write(Node1),
               write(','),write(Name),write(','),
  write(Node),write(').'),nl,
               retract(path(element(ModeI,_),node1(Node1),
element(Mode1,_),node2(Node))),
  write(NewList),nl,
  write(lines_connected(ModeI,NameI,Node1,NewList,
```

```
                Mode1,Name,Node)),nl
            ;
    true)
;
in_line([Xi,Yi,Xj,Yj],X,Y) ->
  path(element(ModeX,NameX),node1(NodeX),_,_),
       line(Xi,Yi,Xj,Yj),
       lines_connected(ModeI,NameI,Node1,List,ModeJ,NameJ,NodeJ),
       a_member(Xi-Yi,List),a_member(Xj-Yj,List),
       %copy_list(Xj-Yj,List,NList),
       check_for_connections(ModeI,NameI,Node1,ModeJ,NameJ,NodeJ,List),
  (path(element(ModeI,NameI),node1(Node1),_,_) ->
    asserta(connection(element(ModeX,NameX),NodeX,
                element(ModeI,NameI),Node1)),
    write(connection(NameX,NodeX,NameI,Node1)),nl
  ;
    path(element(ModeJ,NameJ),node1(NodeJ),_,_)->
    asserta(connection(element(ModeX,NameX),NodeX,
                element(ModeJ,NameJ),NodeJ)),
    write(connection(NameX,NodeX,NameJ,NodeJ)),nl
    ),
  asserta(line(SX,SY,X,Y)),write(line(SX,SY,X,Y)),
       write('.'),nl,
       draw_rubberband(connect,DrawWin,RubberBandGC,SX,SY,X,Y),
       draw_figure(connect,DrawWin,DrawingGC,SX,SY,X,Y),
       asserta(list_data(element(connect,_),SX,SY,X,Y)),
       assertz(drawn_figure(element(connect,_),info(SX,SY,X,Y),_))
;
  lines_connected(ModeI,NameI,Node1,List,ModeJ,NameJ,NodeJ),
       insert(X-Y,List,NewList),
  asserta(lines_connected(ModeI,NameI,Node1,NewList,
                ModeJ,NameJ,NodeJ)),
  retract(lines_connected(ModeI,NameI,Node1,List,
                ModeJ,NameJ,NodeJ)),
       asserta(line(SX,SY,X,Y)),write(line(SX,SY,X,Y)),
  write('.'),nl,
       draw_rubberband(connect,DrawWin,RubberBandGC,SX,SY,X,Y),
       draw_figure(connect,DrawWin,DrawingGC,SX,SY,X,Y),
  asserta(list_data(element(connect,_),SX,SY,X,Y)),
  assertz(drawn_figure(element(connect,_),info(SX,SY,X,Y),_)),
  write(NewList),nl,
  write(lines_connected(ModeI,NameI,Node1,NewList,
                Mode1,Name,Node)),nl
       )
    ;
       Mode == m ->
  ((data(element(Mode1,Name),SX,SY,ending_point(X1,Y1),
    ending_point(X2,Y2),
ending_point(X3,Y3));
    delay_data(element(Mode1,Name),SX,SY,ending_point(X1,Y1),
ending_point(X2,Y2))) ->
    Xf is LX + 30, Yf is LY + 30,
    drawing_figure(element(Mode1,Name),DrawWin,LX,LY,Xf,Yf),
    assertz(drawn_figure(element(Mode1,Name),info(LX,LY,Xf,Yf),_)),
    (data(element(Mode1,Name),LX,LY,ending_point(Xn1,Yn1),
                ending_point(Xn2,Yn2),
                ending_point(Xn3,Yn3));
     delay_data(element(ModeI,Name),LX,LY,ending_point(Xn1,Yn1),
ending_point(Xn2,Yn2))
    ),
        (line(X1,Y1,Xj,Yj) ->
retract(line(X1,Y1,Xj,Yj)),
retract(drawn_figure(element(connect,_),
       info(X1,Y1,Xj,Yj),_)),
    assertz(drawn_figure(element(connect,_),
  info(Xn1,Yn1,Xj,Yj),_)),
asserta(line(Xn1,Yn1,Xj,Yj))
;
 true),
(line(Xj,Yj,X1,Y1) ->
    retract(drawn_figure(element(connect,_),
       info(Xj,Yj,X1,Y1),_)),
    assertz(drawn_figure(element(connect,_),
       info(Xj,Yj,Xn1,Yn1),_)),
retract(line(Xj,Yj,X1,Y1)),
asserta(line(Xj,Yj,Xn1,Yn1))
;
 true),
(line(X2,Y2,Xj,Yj) ->
retract(line(X2,Y2,Xj,Yj)),
```

```
retract(drawn_figure(element(connect,_),
        info(X2,Y2,Xj,Yj),_)),
      assertz(drawn_figure(element(connect,_),
        info(Xn2,Yn2,Xj,Yj),_)),
asserta(line(Xn2,Yn2,Xj,Yj))
;
true),
(line(Xj,Yj,X2,Y2) ->
retract(drawn_figure(element(connect,_),
        info(Xj,Yj,X2,Y2),_)),
      assertz(drawn_figure(element(connect,_),
   info(Xj,Yj,Xn2,Yn2),_)),
retract(line(Xj,Yj,X2,Y2)),
asserta(line(Xj,Yj,Xn2,Yn2))
;
true),
(line(X3,Y3,Xj,Yj) ->
retract(drawn_figure(element(connect,_),
        info(X3,Y3,Xj,Yj),_)),
assertz(drawn_figure(element(connect,_),
        info(Xn3,Yn3,Xj,Yj),_)),
retract(line(X3,Y3,Xj,Yj)),
asserta(line(Xn3,Yn3,Xj,Yj))
    ;
          true),
(line(Xj,Yj,X3,Y3) ->
      retract(drawn_figure(element(connect,_),
        info(Xj,Yj,X3,Y3),_)),
      asserta(drawn_figure(element(connect,_),
        info(Xj,Yj,Xn3,Yn3),_)),
retract(line(Xj,Yj,X3,Y3)),
asserta(line(Xj,Yj,Xn3,Yn3))
;
true),
( in1(Label,X1,Y1) ->
          retract(drawn_figure(element(in1,Label),
       info(X1,Y1,_,_),_)),
      asserta(in1(Label,Xn1,Yn1)),
retract(in1(Label,X1,Y1)),
assertz(drawn_figure(element(in1,Label),
       info(Xn1,Yn1,_,_),_))
            ;
              true),
     ( in2(Label,X2,Y2) ->
                retract(drawn_figure(element(in2,Label),
     info(X2,Y2,_,_),_)),
retract(in2(Label,X2,Y2)),
asserta(in2(Label,Xn2,Yn2)),
assertz(drawn_figure(element(in2,Label),
       info(Xn2,Yn2,_,_),_))
     ;
  true),
          ( out(Label,X3,Y3) ->
      retract(drawn_figure(element(out,Label),
          info(X3,Y3,_,_),_)),
asserta(out(Label,Xn3,Yn3)),
retract(out(Label,X3,Y3)),
      assertz(drawn_figure(element(out,Label),
                info(Xn3,Yn3,_,_),_))
            ;
              true
           ),
((connection(element(Mode2,Name2),Node2,
  element(Mode1,Name),Node1);
 connection(element(Mode1,Name),Node1,
        element(Mode2,Name2),Node2)
) ->
          (data(element(Mode2,Name2),NX,NY,_,_,_);
           delay_data(element(Mode2,Name2),NX,NY,_,_)),
      findall(Xi1-Yi1,
line_element1(Mode1,Name,X1,Y1,Xi1,Yi1),Lines1),
          mvlines(Lines1,DrawWin,Xn1,Yn1),
          findall(Xi2-Yi2,
line_element2(Mode1,Name,X2,Y2,Xi2,Yi2),Lines2),
          mvlines(Lines2,DrawWin,Xn2,Yn2),
    (((Mode1 \== delay1) ) ->
          findall(Xi3-Yi3,
line_element3(Mode1,Name,X3,Y3,
Xi3,Yi3),Lines3),
          mvlines(Lines3,DrawWin,Xn3,Yn3)
```

```
    ; true)

    ;
     true
          ),
     (retract(data(element(Mode1,Name),SX,SY,_,_,_));
retract(delay_data(element(Mode1,Name),SX,SY,_,_)),
retract(drawn_figure(element(Mode1,Name),info(SX,SY,_,_),_))
    ;
(drawn_figure(element(rectangle,Group),info(SX,SY,_,_),_),
 group(Group,List_inside),
      group_line(Group,Lines_inside),
 findall(Na-Xi-Yi,(Fig^drawn_figure(element(Fig,Na),
           info(Xi,Yi,_,_),_),
 Fig^Dx^Dy^a_member(Fig-Na-Dx-Dy,List_inside)),
 Old_inside),
  move_block(LX,LY,DrawWin,DrawingGC,List_inside),
  reproduce_line(LX,LY,DrawWin,DrawingGC,Lines_inside),
  remove_list(Old_inside),
  delete_lines(Lines_inside),
  retract(drawn_figure(element(rectangle,Group),
              info(SX,SY,_,_),_))
          )),
refresh(DrawWin,DrawingGC)
    ;
Mode == g ->
  draw_rubberband(rectangle,DrawWin,DrawingGC,SX,SY,X,Y),
  draw_figure(rectangle,DrawWin,DrawingGC,SX,SY,X,Y),
  gensym(g,Group),
  asserta(drawn_figure(element(rectangle,Group),info(SX,SY,X,Y),_)),
  write(drawn_figure(element(rectangle,Group),info(SX,SY,X,Y),_)),nl,
  findall(Figure-Name-Dx-Dy,(Figure \== connect, Figure \== rectangle,
          under_group(SX,SY,X,Y,Figure,Name,Dx,Dy)), List_inside),
  asserta(group(Group,List_inside)),
  write(group(Group,List_inside)),nl,
  findall(Xi-Yi-Xj-Yj-Dx-Dy-Dx2-Dy2,(line(Xi,Yi,Xj,Yj),
                  inside(Xi,Yi,SX,SY,X,Y),Dx2 is Xj - SX,
    inside(Xj,Yj,SX,SY,X,Y),Dy2 is Yj - SY,
    Dx is Xi - SX,Dy is Yi - SY),Line_inside),
  asserta(group_line(Group,Line_inside)),
  findall(Name1-Node1-Name2-Node2,
          (Fig1^Dx1^Dy1^member(Fig1-Name1-Dx1-Dy1,List_inside),
     Fig2^Dx2^Dy2^member(Fig2-Name2-Dx2-Dy2,List_inside),
          is_connection(Name1,Node1,Name2,Node2)),

List_connections),

  write(List_connections),nl,
  asserta(group_connection(Group,List_connections)),
  write(group_line(Group,Line_inside)),nl,
  findall(Label-Element,(Node^port(Element,Node,Label),
          Port^Dx^Dy^a_member(Port-Label-Dx-Dy,List_inside)),
                List_ports),
  asserta(group_ports(Group,List_ports)),
  asserta(list_data(element(rectangle,Group),SX,SY,X,Y))
      ),
      retractall(selected_region(_,_,_,_)).

%       ------------------------------------------------

move_block(_,_,_,[]).
move_block(LX,LY,DrawWin,GC,[Figure-Name-DX-DY|List_inside]):-
drawn_figure(element(Figure,Name),_,_),
Xn1 is DX + LX,Yn1 is DY + LY,
Xn2 is Xn1 + 30,Yn2 is Yn1 + 30,
(((Figure == adder ; Figure == multiplier ; Figure == subtract;
 Figure == delay1 ; Figure == alu_adder ; Figure == alu_sub ; Figure == fu ) ->
   drawing_figure(element(Figure,Name),DrawWin,GC,Xn1,Yn1,Xn2,Yn2),
   assertz(drawn_figure(element(Figure,Name),info(Xn1,Yn1,Xn2,Yn2),_)),
   asserta(element(Figure,Name)),
   asserta(list_data(element(Figure,Name),Xn1,Yn1,_,_)))
    ;
(Figure == in1 ->
   draw_port1(DrawWin,GC,Name,Xn1,Yn1),
   assertz(drawn_figure(element(in1,Name),info(Xn1,Yn1,_,_),_)),
   asserta(in1(Name,Xn1,Yn1)))
    ;
(Figure == in2 ->
   draw_port2(DrawWin,GC,Name,Xn1,Yn1),
   assertz(drawn_figure(element(in2,Name),info(Xn1,Yn1,_,_),_)),
```

```
   asserta(in2(Name,Xn1,Yn1)))
;
(Figure == out ->
   draw_port3(DrawWin,GC,Name,Xn1,Yn1),
   assertz(drawn_figure(element(out,Name),info(Xn1,Yn1,_,_),_)),
   asserta(out(Name,Xn1,Yn1)))
),
move_block(LX,LY,DrawWin,GC,List_inside).


line_element1(Mode,Name,Xi,Yi,Xj,Yj):-
(line(Xj,Yj,Xi,Yi);line(Xi,Yi,Xj,Yj)),
(data(element(Mode,Name),_,_,ending_point(Xi,Yi),_,_);
 delay_data(element(Mode,Name),_,_,ending_point(Xi,Yi),_)).



line_element2(Mode,Name,Xi,Yi,Xj,Yj):-
      (line(Xj,Yj,Xi,Yi);line(Xi,Yi,Xj,Yj)),
      (data(element(Mode,Name),_,_,_,ending_point(Xi,Yi),_);
 delay_data(element(Mode,Name),_,_,_,ending_point(Xi,Yi))).



line_element3(Mode,Name,Xi,Yi,Xj,Yj):-
      (line(Xj,Yj,Xi,Yi);line(Xi,Yi,Xj,Yj)),
      data(element(Mode,Name),_,_,_,_,ending_point(Xi,Yi)).


% mvlines:- moves the lines that are connected to the element moved.

mvlines([],_,_,_).
mvlines([Xi-Yi|List_lines],DrawWin,Xa,Ya):-
      (line(Xi,Yi,Xj,Yj) ->
                  asserta(line(Xa,Ya,Xi,Yi)),
                  assertz(drawn_figure(element(connect,_),
               info(Xa,Ya,Xi,Yi),_)),
                  retract(line(Xi,Yi,Xj,Yj)),
                  retract(drawn_figure(element(connect,_),
               info(Xi,Yi,Xj,Yj),_)));
      (line(Xj,Yj,Xi,Yi) ->
                  asserta(line(Xa,Ya,Xi,Yi)),
                  assertz(drawn_figure(element(connect,_),
               info(Xa,Ya,Xi,Yi),_)),
                  retract(line(Xj,Yj,Xi,Yi)),
                  retract(drawn_figure(element(connect,_),
               info(Xj,Yj,Xi,Yi),_))),
      mvlines(List_lines,DrawWin,Xa,Ya).


% 6. new_point:
%--------------

    new_point(DrawWin,RubberBandGC,X,Y):-
       clause(selected_region(SX,SY,LX,LY),_,Ref),
       draw_mode(Mode,_),
       (Mode == connect ->
          draw_rubberband(connect,DrawWin,RubberBandGC,SX,SY,LX,LY),
          draw_rubberband(connect,DrawWin,RubberBandGC,SX,SY,X,Y)
  ;
  Mode == g ->
     draw_rubberband(rectangle,DrawWin,RubberBandGC,SX,SY,LX,LY),
     draw_rubberband(rectangle,DrawWin,RubberBandGC,SX,SY,X,Y)

       ; true
       ),
  asserta(selected_region(SX,SY,X,Y)),
       flush,
       erase(Ref).

%      ------------------------------------------------

% new_draw_size:
%---------------

    new_draw_size(DrawWin):-
       drawing_window(_,_,_,DrawWin,_,_,_,_,_).
%      ------------------------------------------------
```

```
%goodbye(RootWidget):-
      %xif_initialize(test,'Demo',QuitWidget),
      %xmCreateMessageDialog(QuitWidget,'quit',
            %[xmNwidth(400),xmNheight(200)],Button),
      %xtManageChild(Button),
      %xmMessageBoxGetChild(Button,xmDIALOG_HELP_BUTTON,Help),
      %xtUnmanageChild(Help),
      %xmMessageBoxGetChild(Button,xmDIALOG_CANCEL_BUTTON,Cancel),
      %xtManageChild(Cancel),
      %xmMessageBoxGetChild(Button,xmDIALOG_OK_BUTTON,Ok),
      %xtManageChild(Ok),
      %xtAddCallback(Button,xmNokCallback,ok_callback(Button,RootWidget)
            %,exit_loop(yes)),
      %xtAddCallback(Button,xmNcancelCallback,cancel(Button),looping(yes)),
      %xtRealizeWidget(Button),
      %xif_main_loop(exit_loop(yes)).


ok_callback(Button,RootWidget) :-
xtDestroyWidget(RootWidget),
xtDestroy(Button).

   cancel(Button):-
      asserta(exit_loop(yes)),
      xtDestroyWidget(Button).


% 8. goodbye:
%-------------

    goodbye(Window):-
write('Do you want to save the data?yes/no'),nl,
read(Ans),
(Ans == yes ->
( write('Enter the filename of the netlist....'),nl,
 read(Filename),
  tell(Filename),
 findall(Element-Label,element(Element,Label),List_elements),
 write_elements(List_elements),
 findall(E1-N1-E2-N2,is_connection(E1,N1,E2,N2),List_connections),
 write_connections(List_connections),
 findall(E1-N1-Label,port(E1,N1,Label),List_ports),
 write_ports(List_ports),
      told)
;
true),
      destroy_window(Window).

%       ------------------------------------------------


is_connection(E1,N1,E2,N2):-
connection(element(_,E1),N1,element(_,E2),N2).


write_elements([]).
write_elements([Element-Label|List_elements]):-
element(Element,Label),
write(element(Element,Label)),write('.'),nl,
write_elements(List_elements).

write_connections([]).
write_connections([E1-N1-E2-N2|List_connections]):-
is_connection(E1,N1,E2,N2),
write(connection(E1,N1,E2,N2)),write('.'),nl,
write_connections(List_connections).

write_ports([]).
write_ports([E1-N1-Label|List_ports]):-
port(E1,N1,Label),
write(port(E1,N1,Label)),write('.'),nl,
write_ports(List_ports).


%   The Menu Callbacks:
%----------------------
```

```
% 1.highlight:
%--------------
    highlight(Option,Window,Width,Height,InvertingGC,FgPixel):-
        not_selected(Option,_,_,_,_),
        put_window_attributes(Window,[background(FgPixel),gc(InvertingGC)]),
        clear_window(Window),
        redraw(Option,Window,Width,Height).

%          ------------------------------------------------

% 2. select:
%-----------
    select(c,Window,Width,Height,X,Y,DrawWin,DrawingGC,BgPixel):-
        !,
        unhighlight(c,Window,Width,Height,DrawingGC,BgPixel),
        (
          X =< Width, Y =<Height ->
            clear_window(DrawWin),
            retractall(drawn_figure(_,_,_))
        ;
          true
        ).
    select(s,Window,Width,Height,X,Y,DrawWin,DrawingGC,BgPixel):-
        !,
        unhighlight(s,Window,Width,Height,DrawingGC,BgPixel),
        (
          X =< Width, Y =<Height ->
          %refresh(DrawWin,DrawingGC),
save_data_base(DrawWin)
        ;
          true
        ).

    select(Option,Window,Width,Height,X,Y,_,DrawingGC,BgPixel):-
        (
        X =< Width, Y=< Height ->
            not_selected(Option,Mode,ModeWindow,MSize,Ref),
            unhighlight(Mode,ModeWindow,MSize,MSize,DrawingGC,BgPixel),
            erase(Ref),
            asserta(draw_mode(Option,Window))
        ;
            unhighlight(Option,Window,Width,Height,DrawingGC,BgPixel)
        ).

open_and_draw(DrawWin,DrawingGC):-
      write('ENTER PLEASE THE NAME OF THE FILE....'),nl,
      read(Filename),
      consult(Filename),
      clear_window(DrawWin),
      findall(Figure-Name,
              (figure(element(Figure,Name),_,_),Figure\== connect,
      Figure \== rectangle),
                  List_data_base),
findall(X1-Y1,X2^Y2^lin(X1,Y1,X2,Y2),List_lines),
draw_the_file(List_data_base,DrawWin,DrawingGC),
draw_the_list(List_lines,DrawWin,DrawingGC),
findall(Name1-Node1-Name2-Node2,Mode1^Mode2^conn(Mode1,Name1,Node1,
                                    Mode2, Name2,Node2),
                        List_connect),
findall(Na-No-Label,port_label(Na,No,Label),List_ports),
assert_ports(List_ports),
assert_connection(List_connect).

assert_ports([]).
assert_ports([Na-No-Label|List_ports]):-
asserta(port(Na,No,Label)),write(port(Na,No,Label)),nl,
assert_ports(List_ports).

assert_connection([]).
assert_connection([Name1-Node1-Name2-Node2|List_connect]):-
conn(Mode1,Name1,Node1,Mode2,Name2,Node2),
asserta(connection(element(Mode1,Name1),Node1,
          element(Mode2,Name2),Node2)),
write(connection(Name1,Node1,Name2,Node2)),nl,
assert_connection(List_connect).

draw_the_list([],_,_).
draw_the_list([Xs-Ys|List_lines],DrawWin,DrawingGC):-
lin(Xs,Ys,Xe,Ye),
assertz(drawn_figure(element(connect,_),info(Xs,Ys,Xe,Ye),_)),
```

```
draw_figure(connect,DrawWin,DrawingGC,Xs,Ys,Xe,Ye),
asserta(line(Xs,Ys,Xe,Ye)),
      draw_the_list(List_lines,DrawWin,DrawingGC).


draw_the_file([],_,_).
draw_the_file([Head-Name|List_data_base],DrawWin,DrawingGC):-
figure(element(Head,Name),info(X1,Y1,X2,Y2),_),
(((Head == adder ; Head == multiplier ; Head == subtract ;
 Head == delay1 ; Head == alu_adder ; Head == alu_sub ; Head == fu) ->
     drawing_figure(element(Head,Name),DrawWin,DrawingGC,
              X1,Y1,X2,Y2),
     assertz(drawn_figure(element(Head,Name),info(X1,Y1,X2,Y2),_)),
     asserta(element(Head,Name)),
     write(element(Head,Name)),write('.'),nl,
     asserta(list_data(element(Head,Name),X1,Y1,_,_)))
;
 (Head == in1  ->
    asserta(in1(Name,X1,Y1)),
    draw_port1(DrawWin,DrawingGC,Name,X1,Y1),
    assertz(drawn_figure(element(in1,Name),info(X1,Y1,_,_),_)))
;
 (Head == in2 ->
    asserta(in2(Name,X1,Y1)),
    draw_port2(DrawWin,DrawingGC,Name,X1,Y1),
    assertz(drawn_figure(element(in2,Name),info(X1,Y1,_,_),_)))
;
 (Head == out ->
    asserta(out(Name,X1,Y1)),
    draw_port3(DrawWin,DrawingGC,Name,X1,Y1),
    assertz(drawn_figure(element(out,Name),info(X1,Y1,_,_),_)))
),
      draw_the_file(List_data_base,DrawWin,DrawingGC).


save_data_base(DrawWin):-
write('ENTER PLEASE THE NAME OF THE FILE....'),nl,
read(Filename),
tell(Filename),
findall(Figure-Name,
(drawn_figure(element(Figure,Name),_,_),
      Figure \== connect,Figure \== rectangle),
List_data_base),
write_to_file(List_data_base,DrawWin),
findall(X1-Y1,X2^Y2^line(X1,Y1,X2,Y2),List_of_lines),
write_lines(List_of_lines,DrawWin),
findall(Name1-Node1-Name2-Node2,Mode1^Mode2^connection(
            element(Mode1,Name1),Node1,
  element(Mode2,Name2),Node2),List_connections),
wr_connections(List_connections),
told.
save_data_base(_).


wr_connections([]).
wr_connections([Name1-Node1-Name2-Node2|List_connections]):-
connection(element(Mode1,Name1),Node1,element(Mode2,Name2),Node2),
write(conn(Mode1,Name1,Node1,Mode2,Name2,Node2)),write('.'),nl,
wr_connections(List_connections).

write_lines([],_).
write_lines([Sx-Sy|List_of_lines],DrawWin):-
line(Sx,Sy,Ex,Ey),
write(lin(Sx,Sy,Ex,Ey)),write('.'),nl,
      write(figure(element(connect,_),info(Sx,Sy,Ex,Ey),_)),
write('.'),nl,
write_lines(List_of_lines,DrawWin).


write_to_file([],_).
write_to_file([Head-Name|List_data_base],DrawWin):-
drawn_figure(element(Head,Name),info(X1,Y1,X2,Y2),_),
( Head == in1 ->
   port(Mode,Node,Name),
   write(port_label(Mode,Node,Name)),write('.'),nl,
   write(figure(element(in1,Name),info(X1,Y1,_,_),_)),write('.'),nl
;
 Head == in2 ->
```

```
    port(Mode,Node,Name),
    write(port_label(Mode,Node,Name)),write('.'),nl,
    write(figure(element(in2,Name),info(X1,Y1,_,_),_)),write('.'),nl
;
  Head == out ->
    port(Mode,Node,Name),
    write(port_label(Mode,Node,Name)),write('.'),nl,
    write(figure(element(out,Name),info(X1,Y1,_,_),_)),write('.'),nl
;
  write(figure(element(Head,Name),info(X1,Y1,X2,Y2),_)),
  write('.'),nl
),
write_to_file(List_data_base,DrawWin).


    unhighlight(Option,Window,Width,Height,DrawingGC,BgPixel):-
        put_window_attributes(Window,[background(BgPixel), gc(DrawingGC)]),
        clear_window(Window),
        redraw(Option,Window,Width,Height).
%       ---------------------------------------------------


look_at_list([]).
look_at_list([[Mode1-Name1-Node1-Mode2-Name2-Node2]|List]):-
(connection(element(Mode1,Name1),Node1,
                element(Mode2,Name2),Node2);
 connection(element(Mode2,Name2),Node2,
                element(Mode1,Name1),Node1)),
(Node1 == 1 ->
(data(element(Mode1,Name1),_,_,ending_point(Xa,Ya),_,_);
 delay_data(element(Mode1,Name1),_,_,ending_point(Xa,Ya),_))
;
 Node1 == 2 ->
(data(element(Mode1,Name1),_,_,_,ending_point(Xa,Ya),_);
 delay_data(element(Mode1,Name1),_,_,_,ending_point(Xa,Ya)))
;
 Node1 == 3 ->
 data(element(Mode1,Name1),_,_,_,_,ending_point(Xa,Ya))
),
(Node2 == 1 ->
 (data(element(Mode2,Name2),_,_,ending_point(Xb,Yb),_,_);
 delay_data(element(Mode2,Name2),_,_,ending_point(Xb,Yb),_))
;
 Node2 == 2 ->
 (data(element(Mode2,Name2),_,_,_,ending_point(Xb,Yb),_);
  delay_data(element(Mode2,Name2),_,_,_,ending_point(Xb,Yb)))
;
 Node2 == 3 ->
 data(element(Mode2,Name2),_,_,_,_,ending_point(Xb,Yb))
),
(connection(element(Mode1,Name1),Node1,
                element(Mode2,Name2),Node2) ->
        show_path(Xa,Ya,Xb,Yb)
;
 connection(element(Mode2,Name2),Node2,
                element(Mode1,Name1),Node1) ->
        show_path(Xb,Yb,Xa,Ya)
),
(retract(connection(element(Mode1,Name1),Node1,
                element(Mode2,Name2),Node2));
 retract(connection(element(Mode2,Name2),Node2,
                element(Mode1,Name1),Node1))),
look_at_list(List).


show_path(Xa,Ya,Xb,Yb):-
line(Xa,Ya,Xi,Yi),
retract(line(Xa,Ya,Xi,Yi)),
retract(drawn_figure(element(connect,_),
        info(Xa,Ya,Xi,Yi),_)),
path1(Xi,Yi,Xb,Yb).

path1(Xb,Yb,Xb,Yb).

path1(Xi,Yi,Xb,Yb):-
      line(Xi,Yi,Xj,Yj),
retract(line(Xi,Yi,Xj,Yj)),
retract(drawn_figure(element(connect,_),
        info(Xi,Yi,Xj,Yj),_)),
```

```
path1(Xj,Yj,Xb,Yb).

del_list([]).
del_list([Label|List2]):-
port(Name,Node,Label),
(Node == 1 -> (in1(Label,_,_),retract(in1(Label,_,_)),
       retract(drawn_figure(element(in1,Label),_,_)))
     ;
 Node == 2 -> ((in2(Label,_,_),retract(in2(Label,_,_)),
       retract(drawn_figure(element(in2,Label),_,_)));
          (out(Label,_,_),retract(out(Label,_,_))),
 retract(drawn_figure(element(out,Label),_,_)))
     ;
 Node == 3 -> (out(Label,_,_),retract(out(Label,_,_)),
       retract(drawn_figure(element(out,Label),_,_)))
    ),
retract(port(Name,Node,Label)),
del_list(List2).

% draw according to selection:
%----------------------------

   draw_button(DrawWin,X,Y):-
     draw_mode(Mode,_),
get_window_attributes(DrawWin,[gc(GC)]),
     asserta(selected_region(X,Y,X,Y)),
X2 is X + 30,
     Y2 is Y + 30,
     get_window_attributes(DrawWin,[size(DrawWidth,DrawHeight)]),
     (((Mode == adder)  -> gensym(a,Label)
       ; (Mode == multiplier) -> gensym(m,Label)
       ; (Mode == delay1) -> gensym(d,Label)
       ; (Mode == subtract) -> gensym(s,Label)
       ; (Mode == fu) -> gensym(f,Label)
      ),
        asserta(element(Mode,Label)),
        write(element(Mode,Label)),write('.'),nl,
        drawing_figure(element(Mode,Label),DrawWin,X,Y,X2,Y2),
        RSX is X/DrawWidth, RX is X2/DrawWidth,
        RSY is Y/DrawHeight,RY is Y2/DrawHeight,
        assertz(drawn_figure(element(Mode,Label),info(X,Y,X2,Y2),
        info(RSX,RSY,RX,RY))),
        asserta(list_data(element(Mode,Label),X,Y,_,_))
     ;
 Mode == alu ->
   write('it it an adder or subtracter?'),nl,
   write('enter please a or s  ----- '),nl,
   read(Ans),
   (Ans == a ->
ModeT = alu_adder,
gensym(ala,Label)
   ;
     ModeT = alu_sub,
gensym(als,Label)
   ),
   X2 is X + 30,
        Y2 is Y + 30,
        get_window_attributes(DrawWin,[size(DrawWidth,DrawHeight)]),
        asserta(element(ModeT,Label)),
        write(element(ModeT,Label)),write('.'),nl,
        drawing_figure(element(ModeT,Label),DrawWin,X,Y,X2,Y2),
        RSX is X/DrawWidth, RX is X2/DrawWidth,
        RSY is Y/DrawHeight,RY is Y2/DrawHeight,
        assertz(drawn_figure(element(ModeT,Label),info(X,Y,X2,Y2),
        info(RSX,RSY,RX,RY))),
        asserta(list_data(element(ModeT,Label),X,Y,_,_))
;
      Mode == delete ->
        (in_region([Xi,Yi],X,Y) ->
   (( data(element(M1,N1),Xi,Yi,_,_),
     findall([M1-N1-No1-M2-N2-No2],
   (connection(element(M1,N1),No1,
            element(M2,N2),No2);
    connection(element(M2,N2),No2,
            element(M1,N1),No1)),List),
     look_at_list(List),
findall(Label,N1^Node^port(N1,Node,Label),List2),
del_list(List2),
            retract(data(element(M1,N1),Xi,Yi,ending_point(_,_),
                         ending_point(_,_),
```

```
                        ending_point(_,_))),
retract(drawn_figure(element(M1,N1),info(Xi,Yi,_,_),_))                    ;
      delay_data(element(M1,N1),Xi,Yi,_,_),
      findall([M1,N1,No1,M2,N2,No2],
     (connection(element(M1,N1),No1,
            element(M2,N2),No2);
      connection(element(M2,N2),No2,
            element(M1,N1),No1)),List),
      look_at_list(List),
findall(Label,N1^Node^port(N1,Node,Label),List2),
del_list(List2),
            retract(delay_data(_,Xi,Yi,ending_point(_,_),
                        ending_point(_,_))),
retract(drawn_figure(_,info(Xi,Yi,_,_),_))
    ;
     out(Label,Xi,Yi),retract(out(Label,Xi,Yi)),
retract(port(N,P,Label)),
            retract(drawn_figure(element(out,Label),info(Xi,Yi,_,_),_))
    ;
     in1(Label,Xi,Yi),retract(in1(Label,Xi,Yi)),
retract(port(N,P,Label)),
retract(drawn_figure(element(in1,Label),info(Xi,Yi,_,_),_))
    ;
     in2(Label,Xi,Yi),retract(in2(Label,Xi,Yi)),
retract(port(N,P,Label)),
retract(drawn_figure(element(in2,Label),info(Xi,Yi,_,_),_))
    ;
     (line(Xi,Yi,Xl,Yl) ->
  retract(drawn_figure(element(connect,_),
            info(Xi,Yi,Xl,Yl),_)),
retract(line(Xi,Yi,Xl,Yl))
;
      line(Xl,Yl,Xi,Yi) ->
retract(drawn_figure(element(connect,_),
            info(Xl,Yl,Xi,Yi),_)),
retract(line(Xl,Yl,Xi,Yi))
)
    ;
     drawn_figure(element(rectangle,Group),info(Xi,Yi,_,_),_),
     group(Group,List_inside),
     delete_list(List_inside),
     group_line(Group,List_lines),
     delete_lines(List_lines),
     group_connection(Group,List_connections),
     delete_connections(List_connections),
     retract(group(Group,List_inside)),
     retract(group_line(Group,List_lines)),
     retract(group_connection(Group,List_connections)),
     retract(drawn_figure(element(rectangle,Group),_,_))
   ))
     ),
 refresh(DrawWin,GC)
      ;
      Mode == undo ->
  list_data(element(Mode1,Label),X1,Y1,Xl,Yl),
       (Mode1 \== undo ->
     drawn_figure(element(Mode1,Label),info(X1,Y1,Xl,Yl),_),
          retract(list_data(element(Mode1,Label),X1,Y1,Xl,Yl)),
asserta(list_data(element(Mode,_),X1,Y1,_,_)),
     asserta(erased(element(Mode1,Label),X1,Y1,Xl,Yl)),

          ((Mode1 == adder ; Mode1 == multiplier ;
            Mode1 == subtract ; Mode1 == delay1 ;
Mode1 == alu_adder ; Mode1 == alu_sub ; Mode1 == fu) ->
         retract(element(Mode1,Label)),
              (retract(data(element(Mode1,_),X1,Y1,_,_,_))
              ;
              retract(delay_data(element(Mode1,_),X1,Y1,_,_))
              ),
   retract(drawn_figure(element(Mode1,Label),
                        info(X1,Y1,_,_),_))
            ;
  retract(drawn_figure(element(rectangle,Label),
          info(X1,Y1,Xl,Yl),_)),
  group(Label,List_inside),
       group_line(Label,List_lines),
  asserta(erased_group(Label,List_inside)),
  asserta(erased_lines(Label,List_lines)),
  %delete_list(List_inside),
       %delete_lines(List_lines),
```

```
        group_connection(Label,List_connections),
   asserta(erased_connection(Label,List_connections)),
        %delete_connections(List_connections),
        retract(group(Label,List_inside)),
        retract(group_line(Label,List_lines)),
        retract(group_connection(Label,List_connections))
;
              retract(drawn_figure(element(connect,_),
info(X1,Y1,Xl,Yl),_)),
              retract(line(X1,Y1,Xl,Yl))
 ;
  retract(drawn_figure(element(out,Label),info(Xi,Yi,_,_),_)),
  retract(out(Label,Xi,Yi)),
  retract(port(_,_,Label))
 ;
  retract(drawn_figure(element(in1,Label),info(Xi,Yi,_,_),_)),
  retract(in1(Label,Xi,Yi)),
  retract(port(_,_,Label))
 ;
  retract(drawn_figure(element(in2,Label),info(Xi,Yi,_,_),_)),
  retract(in2(Label,Xi,Yi)),
        retract(port(_,_,Label))
         )
            ;
  erased(element(Element,Label),X1,Y1,Xl,Yl),
            (( Element == adder; Element == multiplier;
    Element == subtract; Element == delay1;
              Element == alu_adder ; Element == alu_sub ; Element == fu) ->
               X2 is X1+ 30,
               Y2 is Y1 + 30,
               drawing_figure(element(Element,Label),
DrawWin,X1,Y1,X2,Y2),
 assertz(drawn_figure(element(Element,Label),
info(X1,Y1,X2,Y2),_)),
 asserta(element(Element,Label))
            ;
  (Element == rectangle ->
  erased_group(Label,New_list),
  assertz(drawn_figure(element(rectangle,Label),
        info(X1,Y1,Xl,Yl),_)),
       asserta(group(Label,New_list)),
  erased_lines(Label,Lines_inside),
  asserta(group_line(Label,Lines_inside)),
  erased_connection(Label,Connections),
  asserta(group_connection(Label,Connections)),
  draw_figure(rectangle,DrawWin,GC,X1,Y1,Xl,Yl)
  %build(DrawWin,GC,Lines_inside),
  %draw_again(DrawWin,GC,New_list),
  %build_connection(Connections)
  );
  (Element == connect ->
  assertz(drawn_figure(element(connect,_),
info(X1,Y1,Xl,Yl),_)),
  asserta(line(X1,Y1,Xl,Yl)),
  asserta(list_data(element(connect,_),X1,Y1,Xl,Yl)),
  asserta(element(connect,_))
            );
            (Element == in1 ->
  draw_port1(DrawWin,GC,Label,X1,Y1),
  assertz(drawn_figure(element(in1,Label),
info(X1,Y1,_,_),_)),
  asserta(in1(Label,X1,Y1)),
  asserta(list_data(element(in1,Label),X1,Y1,_,_)),
  asserta(port(Mode,Node,Label)),
  asserta(element(in1,Label))
  );
  (Element == in2 ->
  draw_port2(DrawWin,GC,Label,X1,Y1),
  assertz(drawn_figure(element(in2,Label),
info(X1,Y1,_,_),_)),
  asserta(in2(Label,X1,Y1)),
  asserta(list_data(element(in2,Label),X1,Y1,_,_)),
  asserta(port(Mode,Node,Label)),
  asserta(element(in2,Label))
  );
  (Element == out ->
  draw_port3(DrawWin,GC,Label,X1,Y1),
  assertz(drawn_figure(element(out,Label),
info(X1,Y1,_,_),_)),
  asserta(out(Label,X1,Y1)),
```

```
  asserta(list_data(element(out,Label),X1,Y1,_,_)),
  asserta(port(Mode,Node,Label)),
  asserta(element(out,Label))
  )),
        retract(erased(element(Element,Label),X1,Y1,Xl,Yl)),
  asserta(list_data(element(Element,Label),X1,Y1,Xl,Yl))
        ),
        refresh(DrawWin,GC)
;
Mode == g ->
  drawn_figure(element(rectangle,Group),info(X1,Y1,X2,Y2),_),
  group(Group,List_inside),
  write(group(Group,List_inside)),nl,
  group_line(Group,Line_inside),
  group_connection(Group,List_connections),
  write(group_line(Group,Line_inside)),nl,
  group_ports(Group,List_ports),
  reproduce(X,Y,DrawWin,GC,List_inside),
  reproduce_port(List_ports),
  reproduce_connection(List_connections),
  reproduce_line(X,Y,DrawWin,GC,Line_inside),
  asserta(list_data(element(rectangle,Group),X1,Y1,X2,Y2)),
        refresh(DrawWin,GC)
      ;
       true
      ).
%  ------------------------------------------------------

delete_list([]).
delete_list([Fig-Name-_-_|List_inside]):-
drawn_figure(element(Fig,Name),info(X1,Y1,_,_),_),
((Fig == adder ; Fig == subtract ; Fig == multiplier ;
   Fig == delay1 ; Fig == alu_adder ; Fig == alu_sub ; Fig == fu) ->
  retract(drawn_figure(element(Fig,Name),info(X1,Y1,_,_),_)),
  retract(element(Fig,Name)),
  retract(list_data(element(Fig,Name),X1,Y1,_,_)),
        (retract(data(element(Fig,Name),X1,Y1,_,_,_));
      retract(delay_data(element(Fig,Name),X1,Y1,_,_)))
;
  retract(drawn_figure(element(out,Name),info(X1,Y1,_,_),_)),
  retract(out(Name,X1,Y1)), retract(port(N,P,Name))
      ;
  retract(drawn_figure(element(in1,Name),info(X1,Y1,_,_),_)),
  retract(in1(Name,X1,Y1)), retract(port(N,P,Name))
      ;
  retract(drawn_figure(element(in2,Name),info(X1,Y1,_,_),_)),
  retract(in2(Name,X1,Y1)), retract(port(N,P,Name))
      ),
delete_list(List_inside).

remove_list([]).
remove_list([Name-Xi-Yi|Old_list]):-
drawn_figure(element(Fig,Name),info(Xi,Yi,_,_),_),
((Fig == adder ; Fig == subtract ; Fig == multiplier ;
   Fig == delay1 ; Fig == alu_adder ; Fig == alu_sub ;Fig == fu) ->
  retract(drawn_figure(element(Fig,Name),info(Xi,Yi,_,_),_)),
  retract(element(Fig,Name)),
  retract(list_data(element(Fig,Name),Xi,Yi,_,_)),
        (retract(data(element(Fig,Name),Xi,Yi,_,_,_));
      retract(delay_data(element(Fig,Name),Xi,Yi,_,_)))
;
  retract(drawn_figure(element(out,Name),info(Xi,Yi,_,_),_)),
  retract(out(Name,Xi,Yi)), retract(port(_,_,Name))
      ;
  retract(drawn_figure(element(in1,Name),info(Xi,Yi,_,_),_)),
  retract(in1(Name,Xi,Yi)), retract(port(_,_,Name))
      ;
  retract(drawn_figure(element(in2,Name),info(Xi,Yi,_,_),_)),
  retract(in2(Name,Xi,Yi)), retract(port(_,_,Name))
      ),
remove_list(Old_list).

draw_again(_,_,[]).
draw_again(DrawWin,GC,[Fig-Name-Xi-Yi|Old_list]):-
((Fig == adder ; Fig == subtract ; Fig == multiplier ;
   Fig == delay1 ;Fig == alu_adder ; Fig == alu_sub ; Fig == fu) ->
   X2 is Xi + 30, Y2 is Yi + 30,
   drawing_figure(element(Fig,Name),DrawWin,GC,Xi,Yi,X2,Y2),
   asserta(element(Fig,Name)),
   assertz(drawn_figure(element(Fig,Name),info(Xi,Yi,X2,Y2),_)),
```

```
   asserta(list_data(element(Fig,Name),Xi,Yi,X2,Y2))
;
(Fig == out ->
   assertz(drawn_figure(element(out,Name),info(Xi,Yi,_,_),_)),
   draw_port3(DrawWin,GC,Name,Xi,Yi),
   asserta(out(Name,Xi,Yi)),
   asserta(list_data(element(out,Name),Xi,Yi,_,_)))
;
(Fig == in1 ->
   assertz(drawn_figure(element(in1,Name),info(Xi,Yi,_,_),_)),
   draw_port1(DrawWin,GC,Name,Xi,Yi),
   asserta(in1(Name,Xi,Yi)),
   asserta(list_data(element(in1,Name),Xi,Yi,_,_)))
;
(Fig == in2 ->
   assertz(drawn_figure(element(in2,Name),info(Xi,Yi,_,_),_)),
   draw_port2(DrawWin,GC,Name,Xi,Yi),
   asserta(in2(Name,Xi,Yi)),
   asserta(list_data(element(in2,Name),Xi,Yi,_,_)))
),
draw_again(DrawWin,GC,Old_list).

build(_,_,[]).
build(DrawWin,GC,[Xi-Yi-Xj-Yj-_-_-_-_|Lines]):-
asserta(line(Xi,Yi,Xj,Yj)),
assertz(drawn_figure(element(connect,_),info(Xi,Yi,Xj,Yj),_)),
asserta(list_data(element(connect,_),Xi,Yi,Xj,Yj)),
build(DrawWin,GC,Lines).


build_connection([]).
build_connection([Mode1-Node1-Mode2-Node2|Connections]):-
element(Fig1,Mode1),element(Fig2,Mode2),
asserta(connection(element(Fig1,Mode1),Node1,
          element(Fig2,Mode2),Node2)),
build_connection(Connections).



delete_lines([]).
delete_lines([Xi-Yi-Xj-Yj-_-_-_-_|List_lines]):-
line(Xi,Yi,Xj,Yj),retract(line(Xi,Yi,Xj,Yj)),
retract(drawn_figure(element(connect,_),info(Xi,Yi,Xj,Yj),_)),
delete_lines(List_lines).

delete_connections([]).
delete_connections([Name1-Node1-Name2-Node2|List_connections]):-
connection(element(Mode1,Name1),Node1,element(Mode2,Name2),Node2),
retract(connection(element(Mode1,Name1),Node1,element(Mode2,Name2),Node2)),
delete_connections(List_connections).

a_member(A,L):-
\+ nonmember(A,L).


under_group(X1,Y1,X2,Y2,Figure,Name,Dx,Dy):-
drawn_figure(element(Figure,Name), info(Xi,Yi,_,_),_),
inside(Xi,Yi,X1,Y1,X2,Y2),
Figure \== connect,Figure \== rectangle,
Dx is Xi - X1, Dy is Yi - Y1.

inside(Xi,Yi,X1,Y1,X2,Y2):-
Xi =< X2, Xi > X1 , Yi =< Y2 , Yi > Y1 .

reproduce_port([]).
reproduce_port([Label-Mode|List_ports]):-
port(Mode,_,Label),
relate(_,Label,Name),relate(_,Mode,New),
asserta(port(New,Node,Name)),write(port(New,Node,Name)),nl,
reproduce_port(List_ports).

reproduce_connection([]).
reproduce_connection([Na1-No1-Na2-No2|List_connections]):-
connection(element(Figure1,Na1),No1,element(Figure2,Na2),No2),
relate(Figure1,Na1,New1),relate(Figure2,Na2,New2),
asserta(connection(element(Figure1,New1),No1,element(Figure2,New2),No2)),
write(connection(element(Figure1,New1),No1,element(Figure2,New2),No2)),
nl,reproduce_connection(List_connections).
```

```
reproduce(_,_,_,_,[]).
reproduce(X,Y,DrawWin,GC,[Figure-Name-DX-DY|List_inside]):-
drawn_figure(element(Figure,Name),_,_),
Xn1 is DX + X,Yn1 is DY + Y,
Xn2 is Xn1 + 30,Yn2 is Yn1 + 30,
(Figure == adder -> gensym(a,New)
;
Figure == subtract -> gensym(s,New)
;
Figure == multiplier -> gensym(m,New)
;
Figure == delay1 -> gensym(d,New)
;
Figure == alu_adder -> gensym(ala,New)
;
Figure == alu_sub -> gensym(als,New)
;
Figure == fu -> gensym(fu,New)
),
(((Figure == adder ; Figure == multiplier ; Figure == subtract;
  Figure == delay1 ; Figure == alu_adder; Figure == alu_sub ; Figure == fu) ->
    drawing_figure(element(Figure,New),DrawWin,GC,Xn1,Yn1,Xn2,Yn2),
    assertz(drawn_figure(element(Figure,New),info(Xn1,Yn1,Xn2,Yn2),_)),
    asserta(element(Figure,New)),write(element(Figure,New)),nl,
    asserta(relate(Figure,Name,New)),
    asserta(list_data(element(Figure,New),Xn1,Yn1,_,_)))
;
(Figure == in1 ->
    draw_port1(DrawWin,GC,New,Xn1,Yn1),
    assertz(drawn_figure(element(in1,New),info(Xn1,Yn1,_,_),_)),
    asserta(in1(New,Xn1,Yn1)))
;
(Figure == in2 ->
    draw_port2(DrawWin,GC,New,Xn1,Yn1),
    assertz(drawn_figure(element(in2,New),info(Xn1,Yn1,_,_),_)),
    asserta(in2(New,Xn1,Yn1)))
;
(Figure == out ->
    draw_port3(DrawWin,GC,New,Xn1,Yn1),
    assertz(drawn_figure(element(out,New),info(Xn1,Yn1,_,_),_)),
    asserta(out(New,Xn1,Yn1)))
),
%insert(Figure-New-Xn1-Yn1,New_list,List),
asserta(relate(Figure,Name,New)),
reproduce(X,Y,DrawWin,GC,List_inside).


reproduce_line(_,_,_,_,[]).
reproduce_line(X,Y,DrawWin,GC,[Xi-Yi-Xj-Yj-DX-DY-Dx2-Dy2|Line_inside]):-
 line(Xi,Yi,Xj,Yj),
 Xn1 is DX + X,Yn1 is Y + DY,Xn2 is X + Dx2, Yn2 is Y + Dy2,
 draw_figure(connect,DrawWin,GC,Xn1,Yn1,Xn2,Yn2),
 asserta(line(Xn1,Yn1,Xn2,Yn2)),
 assertz(drawn_figure(element(connect,_),info(Xn1,Yn1,Xn2,Yn2),_)),
 reproduce_line(X,Y,DrawWin,GC,Line_inside).

% connect the elements:
%---------------------

% 3. redraw:
%-----------

    redraw(adder,Window,Width,Height):-
        LRX is Width - 15,
        LRY is Height - 15,
        draw_figure(adder,Window,15,15,LRX,LRY).


    redraw(multiplier,Window,Width,Height):-
        LRX is Width - 15,
        LRY is Height - 15,
        draw_figure(multiplier,Window,15,15,LRX,LRY).

    redraw(subtract,Window,Width,Height):-
        LRX is Width - 15,
        LRY is Height - 15,
        draw_figure(subtract,Window,15,15,LRX,LRY).
```

```
  redraw(delay1,Window,Width,Height):-
      LRX is Width - 15,
      LRY is Height - 15,
      draw_figure(delay1,Window,15,15,LRX,LRY).

  redraw(alu,Window,Width,Height):-
      LRX is Width - 30,
      LRY is Height - 30,
      draw_figure(alu,Window,0,0,LRX,LRY).

  redraw(fu,Window,Width,Height):-
LRX is Width - 30,
      LRY is Height - 30,
draw_figure(fu,Window,0,0,LRX,LRY).

  redraw(undo,Window,Width,Height):-
      text_extents(Window,u,LBearing,_,TextWidth,Ascent,Descent),
      VS is (Height - TextWidth)//2,
      TextHeight is Ascent + Descent,
      HS is (Width - TextHeight)//2,
      X is HS - LBearing,
      Y is VS + 2*Ascent,
      draw_string(Window,X,Y,'U').

  redraw(connect,Window,Width,Height):-
      URX is Width - 5,
      LLY is Height - 5,
      draw_figure(connect,Window,5,LLY,URX,5).

  redraw(delete,DrawWin,Width,Height):-
      URX is Width - 5,
      URY is Height - 5,
      get_window_attributes(DrawWin,[gc(GC)]),
      draw_line(DrawWin,GC,5,URY,URX,5),
      draw_line(DrawWin,GC,5,5,URX,URY).

  redraw(m,Window,Width,Height):-
      text_extents(Window,m,LBearing,_,TextWidth,Ascent,Descent),
      VS is (Height - TextWidth)//2,
      TextHeight is Ascent + Descent,
      HS is (Width - TextHeight)//2,
      X is HS - LBearing,
      Y is VS + 2*Ascent,
      draw_string(Window,X,Y,'M').

  redraw(c,Window,Width,Height):-
      text_extents(Window,c,LBearing,_,TextWidth,Ascent,Descent),
      VS is (Height - TextWidth)//2,
      TextHeight is Ascent + Descent,
      HS is (Width - TextHeight)//2,
      X is HS - LBearing,
      Y is VS + 2*Ascent,
      draw_string(Window,X,Y,'C').

  redraw(s,Window,Width,Height):-
      text_extents(Window,s,LBearing,_,TextWidth,Ascent,Descent),
      VS is (Height - TextWidth)//2,
      TextHeight is Ascent + Descent,
      HS is (Width - TextHeight)//2,
      X is HS - LBearing,
      Y is VS + Ascent,
      draw_string(Window,X,Y,'S').

  redraw(g,Window,Width,Height):-
      text_extents(Window,p,LBearing,_,TextWidth,Ascent,Descent),
      VS is (Height - TextWidth)//2,
      TextHeight is Ascent + Descent,
      HS is (Width - TextHeight)//2,
      X is HS - LBearing,
      Y is VS + Ascent,
      draw_string(Window,X,Y,'G').

%     ---------------------------------------------

% drawing the figures:
%---------------------
  draw_figure(Figure,Window,X1,Y1,X2,Y2):-
      get_window_attributes(Window,[gc(GC)]),
      draw_figure(Figure,Window,GC,X1,Y1,X2,Y2).
```

```
draw_figure(rectangle,DrawWin,GC,X1,Y1,X2,Y2):-
Width is X2 - X1,
Height is Y2 - Y1,
draw_rectangle(DrawWin,GC,X1,Y1,Width,Height).

draw_figure(adder,DrawWin,GC,X1,Y1,X2,Y2):-
    Size is 30,
    draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
    X11 is X1 + 15,
    X12 is X1 + 5,
    X21 is X2 - 5,
    Y11 is Y1 + 15,
    Y12 is Y1 + 5,
    Y21 is Y2 - 5,
    draw_line(DrawWin,GC,X11,Y12,X11,Y21),
    draw_line(DrawWin,GC,X12,Y11,X21,Y11),
    X13 is X1 - 15,
    Y22 is Y2 + 15,
    X22 is X2 + 15,
    draw_line(DrawWin,GC,X13,Y11,X1,Y11),
    draw_line(DrawWin,GC,X11,Y2,X11,Y22),
    draw_line(DrawWin,GC,X2,Y11,X22,Y11),
    X14 is X1 - 10,
    Y13 is Y1 + 10,
    fill_ellipse(DrawWin,GC,X14,Y13,10,10),
    X15 is X1 + 10,
    fill_ellipse(DrawWin,GC,X15,Y2,10,10),
    draw_ellipse(DrawWin,GC,X2,Y13,10,10).


draw_figure(fu,DrawWin,GC,X1,Y1,X2,Y2):-
    draw_string(DrawWin,GC,X1,Y1,Label),
    Size is 30,Size1 is 60,
Xtemp is X1 + 15, X2temp is X2 + 15,
    draw_rectangle(DrawWin,GC,Xtemp,Y1,Size,Size1),
    X0 is Xtemp - 10, X3 is X2temp + 10,
    Y11 is Y1 + 10, Y12 is Y11 + 8, Y13 is Y11 + 16,
    Y14 is Y11 + 24 , Y15 is Y11 + 32 , Y16 is Y11 + 40,
    draw_line(DrawWin,GC,X0,Y11,Xtemp,Y11),
    draw_line(DrawWin,GC,X0,Y12,Xtemp,Y12),
    draw_line(DrawWin,GC,X0,Y13,Xtemp,Y13),
    draw_line(DrawWin,GC,X0,Y14,Xtemp,Y14),
    draw_line(DrawWin,GC,X0,Y15,Xtemp,Y15),
    draw_line(DrawWin,GC,X0,Y16,Xtemp,Y16),
    draw_line(DrawWin,GC,X2temp,Y11,X3,Y11),
    draw_line(DrawWin,GC,X2temp,Y12,X3,Y12),
    draw_line(DrawWin,GC,X2temp,Y13,X3,Y13),
    draw_line(DrawWin,GC,X2temp,Y14,X3,Y14),
    draw_line(DrawWin,GC,X2temp,Y15,X3,Y15),
    draw_line(DrawWin,GC,X2temp,Y16,X3,Y16),
Xc1 is Xtemp - 4 , Yc1 is Y11 - 2,
    Yc2 is Y12 - 2, Yc3 is Y13 - 2, Yc4 is Y14 - 2,
    Yc5 is Y15 - 2, Yc6 is Y16 - 2,
    draw_ellipse(DrawWin,GC,Xc1,Yc1,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc2,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc3,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc3,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc4,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc5,4,4),
    draw_ellipse(DrawWin,GC,Xc1,Yc6,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc1,4,4),
    draw_ellipse(DrawWin,GC,X2tempYc2,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc3,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc3,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc4,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc5,4,4),
    draw_ellipse(DrawWin,GC,X2temp,Yc6,4,4).


draw_figure(subtract,DrawWin,GC,X1,Y1,X2,Y2):-
    Size is 30,
    draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
    X11 is X1 + 15,
    X12 is X1 + 5,
    X21 is X2 - 5,
    Y11 is Y1 + 15,
    draw_line(DrawWin,GC,X12,Y11,X21,Y11),
    X13 is X1 - 15,
    Y22 is Y2 + 15,
```

```
    X22 is X2 + 15,
    draw_line(DrawWin,GC,X13,Y11,X1,Y11),
    draw_line(DrawWin,GC,X11,Y2,X11,Y22),
    draw_line(DrawWin,GC,X2,Y11,X22,Y11),
    X14 is X1 - 10,
    Y13 is Y1 + 10,
    fill_ellipse(DrawWin,GC,X14,Y13,10,10),
    X15 is X1 + 10,
    fill_ellipse(DrawWin,GC,X15,Y2,10,10),
    draw_ellipse(DrawWin,GC,X2,Y13,10,10).

draw_figure(multiplier,DrawWin,GC,X1,Y1,X2,Y2):-
    Size is 30,
    draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
    X11 is X1 + 9,
    Y21 is Y2 - 7,
    X21 is X2 -9,
    Y11 is Y1 + 7,
    draw_line(DrawWin,GC,X11,Y21,X21,Y11),
    draw_line(DrawWin,GC,X11,Y11,X21,Y21),
    X22 is X2 + 15,
    Y22 is Y2 + 15,
    Y12 is Y1 + 15,
    X13 is X1 - 15,
    draw_line(DrawWin,GC,X13,Y12,X1,Y12),
    draw_line(DrawWin,GC,X2,Y12,X22,Y12),
    X12 is X1 + 15,
    draw_line(DrawWin,GC,X12,Y2,X12,Y22),
    X16 is X1 - 10,
    Y15 is Y1 + 10,
    fill_ellipse(DrawWin,GC,X16,Y15,10,10),
    X17 is X1 + 10,
    fill_ellipse(DrawWin,GC,X17,Y2,10,10),
    draw_ellipse(DrawWin,GC,X2,Y15,10,10).

 draw_figure(delay1,DrawWin,GC,X1,Y1,X2,Y2):-
    Size is 30,
    draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
    Y11 is Y2 - 15,
    X13 is X1 - 15,
    X22 is X2 + 15,
    draw_line(DrawWin,GC,X13,Y11,X1,Y11),
    draw_line(DrawWin,GC,X2,Y11,X22,Y11),
    X16 is X1 - 10,
    Y15 is Y1 + 10,
    fill_ellipse(DrawWin,GC,X16,Y15,10,10),
    X is X1 + 10,
    Y is Y1 + 25,
    draw_string(DrawWin,X,Y,'1'),
    draw_ellipse(DrawWin,GC,X2,Y15,10,10).


draw_figure(alu,DrawWin,GC,X,Y,Xi,Yi):-
X1 is X + 20, Y1 is Y + 10, X3 is Xi + 10, Y2 is Y + 50,
Y4 is Y + 40, Y3 is Y + 20, X9 is X + 60,
draw_line(DrawWin,GC,X1,Y1,X1,Y2),
draw_line(DrawWin,GC,X3,Y3,X3,Y4),
draw_line(DrawWin,GC,X,Y3,X1,Y3),
draw_line(DrawWin,GC,X,Y4,X1,Y4),
draw_line(DrawWin,GC,X1,Y1,X3,Y3),
draw_line(DrawWin,GC,X1,Y2,X3,Y4),
draw_line(DrawWin,GC,X3,Yi,X9,Yi),
      Xc is X1 - 8,
      Yc is Y3 - 4, Yc2 is Y4 - 4,
      fill_ellipse(DrawWin,GC,Xc,Yc,8,8),
fill_ellipse(DrawWin,GC,Xc,Yc2,8,8),
      Yc3 is Yi - 4,
      draw_ellipse(DrawWin,GC,X3,Yc3,8,8).

draw_figure(connect,DrawWin,GC,X1,Y1,X2,Y2):-
      draw_line(DrawWin,GC,X1,Y1,X1,Y2),
      draw_line(DrawWin,GC,X1,Y2,X2,Y2).


%drawing_figure is a predicate to do so&so
 drawing_figure(element(Figure,Name),Window,X1,Y1,X2,Y2):-
    get_window_attributes(Window,[gc(GC)]),
    drawing_figure(element(Figure,Name),Window,GC,X1,Y1,X2,Y2).

 drawing_figure(element(adder,Label),DrawWin,GC,X1,Y1,X2,Y2):-
```

```
draw_string(DrawWin,GC,X1,Y1,Label),
      Size is 30,
      draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
        X11 is X1 + 15,
        X12 is X1 + 5,
        X21 is X2 - 5,
        Y11 is Y1 + 15,
        Y12 is Y1 + 5,
        Y21 is Y2 - 5,
        draw_line(DrawWin,GC,X11,Y12,X11,Y21),
        draw_line(DrawWin,GC,X12,Y11,X21,Y11),
        X13 is X1 - 15,
        Y22 is Y2 + 15,
        X22 is X2 + 15,
        draw_line(DrawWin,GC,X13,Y11,X1,Y11),
        draw_line(DrawWin,GC,X11,Y2,X11,Y22),
        draw_line(DrawWin,GC,X2,Y11,X22,Y11),
        asserta(ending_point(X13,Y11)),
%         write(ending_point(X13,Y11)), nl,
        asserta(ending_point(X11,Y22)),
%         write(ending_point(X11,Y22)), nl,
        asserta(ending_point(X22,Y11)),
%         write(ending_point(X22,Y11)), nl,
        asserta(data(element(adder,Label),X1,Y1,
                          ending_point(X13,Y11),
                          ending_point(X11,Y22),
                          ending_point(X22,Y11))),
%         write(data(element(adder,Label),X1,Y1,
                          %ending_point(X13,Y11),
%                          ending_point(X11,Y22),
%                          ending_point(X22,Y11))),nl,
        X14 is X1 - 10,
        Y13 is Y1 + 10,
        fill_ellipse(DrawWin,GC,X14,Y13,10,10),
        X15 is X1 + 10,
        fill_ellipse(DrawWin,GC,X15,Y2,10,10),
        draw_ellipse(DrawWin,GC,X2,Y13,10,10).

  drawing_figure(element(subtract,Label),DrawWin,GC,X1,Y1,X2,Y2):-
draw_string(DrawWin,GC,X1,Y1,Label),
      Size is 30,
      draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
        X11 is X1 + 15,
        X12 is X1 + 5,
        X21 is X2 - 5,
        Y11 is Y1 + 15,
        draw_line(DrawWin,GC,X12,Y11,X21,Y11),
        X13 is X1 - 15,
        Y22 is Y2 + 15,
        X22 is X2 + 15,
        draw_line(DrawWin,GC,X13,Y11,X1,Y11),
        draw_line(DrawWin,GC,X11,Y2,X11,Y22),
        draw_line(DrawWin,GC,X2,Y11,X22,Y11),
        asserta(ending_point(X13,Y11)),
%         write(ending_point(X13,Y11)), nl,
        asserta(ending_point(X11,Y22)),
%         write(ending_point(X11,Y22)), nl,
        asserta(ending_point(X22,Y11)),
%         write(ending_point(X22,Y11)), nl,
        asserta(data(element(subtract,Label),X1,Y1,
                          ending_point(X13,Y11),
                          ending_point(X11,Y22),
                          ending_point(X22,Y11))),
%         write(data(element(subtract,Label),X1,Y1,
                          %ending_point(X13,Y11),
%                          ending_point(X11,Y22),
%                          ending_point(X22,Y11))),nl,
        X14 is X1 - 10,
        Y13 is Y1 + 10,
        fill_ellipse(DrawWin,GC,X14,Y13,10,10),
        X15 is X1 + 10,
        fill_ellipse(DrawWin,GC,X15,Y2,10,10),
        draw_ellipse(DrawWin,GC,X2,Y13,10,10).

  drawing_figure(element(fu,Label),DrawWin,GC,X1,Y1,X2,Y2):-
      draw_string(DrawWin,GC,X1,Y1,Label),
Size is 30,Size1 is 60,
draw_rectangle(DrawWin,GC,X1,Y1,Size,Size1),
X0 is X1 - 10, X3 is X2 + 10,
Y11 is Y1 + 10, Y12 is Y11 + 8, Y13 is Y11 + 16,
```

```
Y14 is Y11 + 24 , Y15 is Y11 + 32 , Y16 is Y11 + 40,
draw_line(DrawWin,GC,X0,Y11,X1,Y11),
draw_line(DrawWin,GC,X0,Y12,X1,Y12),
draw_line(DrawWin,GC,X0,Y13,X1,Y13),
draw_line(DrawWin,GC,X0,Y14,X1,Y14),
draw_line(DrawWin,GC,X0,Y15,X1,Y15),
draw_line(DrawWin,GC,X0,Y16,X1,Y16),
draw_line(DrawWin,GC,X2,Y11,X3,Y11),
      draw_line(DrawWin,GC,X2,Y12,X3,Y12),
      draw_line(DrawWin,GC,X2,Y13,X3,Y13),
      draw_line(DrawWin,GC,X2,Y14,X3,Y14),
      draw_line(DrawWin,GC,X2,Y15,X3,Y15),
      draw_line(DrawWin,GC,X2,Y16,X3,Y16),
Xc1 is X1 - 4 ,  Yc1 is Y11 - 2,
Yc2 is Y12 - 2, Yc3 is Y13 - 2, Yc4 is Y14 - 2,
Yc5 is Y15 - 2, Yc6 is Y16 - 2,
draw_ellipse(DrawWin,GC,Xc1,Yc1,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc2,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc3,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc3,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc4,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc5,4,4),
draw_ellipse(DrawWin,GC,Xc1,Yc6,4,4),
draw_ellipse(DrawWin,GC,X2,Yc1,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc2,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc3,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc3,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc4,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc5,4,4),
      draw_ellipse(DrawWin,GC,X2,Yc6,4,4),
assert(ending_point(X0,Y11)),assert(ending_point(X0,Y12)),
assert(ending_point(X0,Y13)),assert(ending_point(X0,Y14)),
assert(ending_point(X0,Y15)),assert(ending_point(X0,Y16)),
assert(ending_point(X3,Y11)),assert(ending_point(X3,Y12)),
assert(ending_point(X3,Y13)),assert(ending_point(X3,Y14)),
assert(ending_point(X3,Y15)),assert(ending_point(X3,Y16)),
assert(data_fu(element(fu,Label),X1,X2,[(X0,Y11),(X0,Y12),(X0,Y13),
      (X0,Y14),(X0,Y13),(X0,Y14),(X0,Y15),(X0,Y16)],
      [(X3,Y11),(X3,Y12),(X3,Y13),(X3,Y14),(X3,Y15),
 (X3,Y16)])).

  drawing_figure(element(multiplier,Label),DrawWin,GC,X1,Y1,X2,Y2):-
draw_string(DrawWin,GC,X1,Y1,Label),
      Size is 30,
      draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
      X11 is X1 + 9,
      Y21 is Y2 - 7,
      X21 is X2 -9,
      Y11 is Y1 + 7,
      draw_line(DrawWin,GC,X11,Y21,X21,Y11),
      draw_line(DrawWin,GC,X11,Y11,X21,Y21),
      X22 is X2 + 15,
      Y22 is Y2 + 15,
      Y12 is Y1 + 15,
      X13 is X1 - 15,
      draw_line(DrawWin,GC,X13,Y12,X1,Y12),
      draw_line(DrawWin,GC,X2,Y12,X22,Y12),
      X12 is X1 + 15,
      draw_line(DrawWin,GC,X12,Y2,X12,Y22),
      asserta(ending_point(X13,Y12)),
      %write(ending_point(X13,Y12)), nl,
      asserta(ending_point(X12,Y22)),
      %write(ending_point(X12,Y22)), nl,
      asserta(ending_point(X22,Y12)),
      %write(ending_point(X22,Y12)), nl,
      asserta(data(element(multiplier,Label),X1,Y1,
                        ending_point(X13,Y12),
                        ending_point(X12,Y22),
                        ending_point(X22,Y12))),
      %write(data(element(multiplier,Label),X1,Y1,
                        %ending_point(X13,Y12),
      %                  ending_point(X12,Y22),
      %                  ending_point(X22,Y12))),nl,
      X16 is X1 - 10,
      Y15 is Y1 + 10,
      fill_ellipse(DrawWin,GC,X16,Y15,10,10),
      X17 is X1 + 10,
      fill_ellipse(DrawWin,GC,X17,Y2,10,10),
      draw_ellipse(DrawWin,GC,X2,Y15,10,10).
```

```
  drawing_figure(element(delay1,Label),DrawWin,GC,X1,Y1,X2,Y2):-
draw_string(DrawWin,GC,X1,Y1,Label),
      Size is 30,
      draw_ellipse(DrawWin,GC,X1,Y1,Size,Size),
      Y11 is Y2 - 15,
      X13 is X1 - 15,
      X22 is X2 + 15,
      draw_line(DrawWin,GC,X13,Y11,X1,Y11),
      draw_line(DrawWin,GC,X2,Y11,X22,Y11),
      asserta(ending_point(X13,Y11)),
      asserta(ending_point(X22,Y11)),
      asserta(delay_data(element(delay1,Label),X1,Y1,
                         ending_point(X13,Y11),
                         ending_point(X22,Y11))),
      X16 is X1 - 10,
      Y15 is Y1 + 10,
      X is X1 + 10,
      Y is Y1 + 25,
      draw_string(DrawWin,X,Y,'1'),
      fill_ellipse(DrawWin,GC,X16,Y15,10,10),
      draw_ellipse(DrawWin,GC,X2,Y15,10,10).

drawing_figure(element(alu_adder,Label),DrawWin,GC,X,Y,Xi,Yi):-
draw_string(DrawWin,GC,X,Y,Label),
X1 is X + 20, Y1 is Y + 10, X3 is X + 40, Y2 is Y + 50,
Y4 is Y + 40, Y3 is Y + 20, Y5 is Y + 35, Y6 is Y + 25,
X9 is X + 60, X6 is X + 25, X5 is X + 35,
draw_line(DrawWin,GC,X1,Y1,X1,Y2),
draw_line(DrawWin,GC,X3,Y3,X3,Y4),
draw_line(DrawWin,GC,X,Y3,X1,Y3),
draw_line(DrawWin,GC,X,Y4,X1,Y4),
draw_line(DrawWin,GC,X1,Y1,X3,Y3),
draw_line(DrawWin,GC,X1,Y2,X3,Y4),
draw_line(DrawWin,GC,X3,Yi,X9,Yi),
draw_line(DrawWin,GC,X6,Yi,X5,Yi),
draw_line(DrawWin,GC,Xi,Y6,Xi,Y5),
      Xc is X1 - 8,
      Yc is Y3 - 4, Yc2 is Y4 - 4,
      fill_ellipse(DrawWin,GC,Xc,Yc,8,8),
fill_ellipse(DrawWin,GC,Xc,Yc2,8,8),
      Yc3 is Yi - 4,
      draw_ellipse(DrawWin,GC,X3,Yc3,8,8),
asserta(ending_point(X,Y3)),
asserta(ending_point(X,Y4)),
asserta(ending_point(X9,Yi)),
draw_line(DrawWin,GC,Xi,Y5,Xi,Y6),
      asserta(data(element(alu_adder,Label),X,Y,
                         ending_point(X,Y3),
                         ending_point(X,Y4),
ending_point(X9,Yi))).

drawing_figure(element(alu_sub,Label),DrawWin,GC,X,Y,Xi,Yi):-
draw_string(DrawWin,GC,X,Y,Label),
X1 is X + 20, Y1 is Y + 10, X3 is X + 40, Y2 is Y + 50,
Y4 is Y + 40, Y3 is Y + 20, Y5 is Y + 35, Y6 is Y + 25,
X9 is X + 60,
draw_line(DrawWin,GC,X1,Y1,X1,Y2),
draw_line(DrawWin,GC,X3,Y3,X3,Y4),
draw_line(DrawWin,GC,X,Y3,X1,Y3),
draw_line(DrawWin,GC,X,Y4,X1,Y4),
draw_line(DrawWin,GC,X1,Y1,X3,Y3),
draw_line(DrawWin,GC,X1,Y2,X3,Y4),
draw_line(DrawWin,GC,X3,Yi,X9,Yi),
draw_line(DrawWin,GC,Xi,Y6,Xi,Y5),
      Xc is X1 - 8,
      Yc is Y3 - 4, Yc2 is Y4 - 4,
      fill_ellipse(DrawWin,GC,Xc,Yc,8,8),
fill_ellipse(DrawWin,GC,Xc,Yc2,8,8),
      Yc3 is Yi - 4,
      draw_ellipse(DrawWin,GC,X3,Yc3,8,8),
asserta(ending_point(X,Y3)),
asserta(ending_point(X,Y4)),
asserta(ending_point(X9,Yi)),
draw_line(DrawWin,GC,Xi,Y5,Xi,Y6),
      asserta(data(element(alu_sub,Label),X,Y,
                         ending_point(X,Y3),
                         ending_point(X,Y4),
ending_point(X9,Yi))).
```

```
%       ----------------------------------------

% RubberBanding the figures:
%-------------------------
  draw_rubberband(rectangle,DrawWin,GC,X1,Y1,X2,Y2):-
Width is X2 - X1,
Height is Y2 - Y1,
draw_rectangle(DrawWin,GC,X1,Y1,Width,Height).

  draw_rubberband(rect_m,DrawWin,GC,X1,Y1,X2,Y2):-
draw_line(DrawWin,GC,X1,Y1,X1,Y2),
draw_line(DrawWin,GC,X1,Y1,X2,Y1).


  draw_rubberband(connect,DrawWin,GC,X1,Y1,X2,Y2):-
      draw_line(DrawWin,GC,X1,Y1,X1,Y2),
      draw_line(DrawWin,GC,X1,Y2,X2,Y2).


% Drawing Graphics Context:
%-------------------------

   create_drawing_gcs(NormalGC, InvertingGC,FgPixel,BgPixel,Font):-
      get_screen_attributes([black_pixel(FgPixel),
                             white_pixel(BgPixel)]),
      create_gc(NormalGC,[foreground(FgPixel),
                          background(BgPixel),
                          function(copy),
                          font(Font),
                          line_width(2)]),

      create_gc(InvertingGC,[foreground(BgPixel),
                             background(FgPixel),
                             function(copy),
                             font(Font),
                             line_width(2)]).
%       ----------------------------------------------------

% Drawing Rubberband font:
%-----------------------
    create_rubberband_gc(GC,Font):-
       get_screen_attributes([black_pixel(FgPixel),
                              white_pixel(BgPixel)]),
       xor(FgPixel,BgPixel,XorPixel),
       create_gc(GC, [foreground(XorPixel),
                      background(BgPixel),
                      function(xor),
                      font(Font),
                      line_width(0)]).

%       ----------------------------------------------------

% to maintain the state of the window:
%-----------------------------------

% 1.
    initialize_state:-
  predicate_property(gui:P,(dynamic)),
       retractall(P),fail.
    initialize_state.
       %retractall(draw_state(_,_,_)),
       %retractall(ending_point(_,_)),
       %retractall(list_data(_,_,_,_)),
       %retractall(drawn_figure(_,_,_)).

% 2.
    default_setup(Option,Window,InvertingGC,FgPixel):-
       window_data(default_mode,Option),
       !,
       put_window_attributes(Window, [background(FgPixel),gc(InvertingGC)]),
       asserta(draw_mode(Option,Window)).

    default_setup(_,_,_,_).

%       ----------------------------------------------------
```

```
% Some User Built Functions:
%-------------------------

   find_font(FontData,Font):-
      (
       window_data(FontData,FontSpec),
       current_font(FontSpec,FontName) ->
            load_font(FontName,Font);
       otherwise ->
            format('[~p: Cannot Find Requested Fonts ]~n',gui),
            fail
      ).

   xor(A,B,C):-
      C is (\(A) /\ B) \/ ( A /\ \(B)).


   not_selected(Option,Mode,Window,Size,Ref) :-
      clause(draw_mode(Mode,Window),_,Ref),
      window_data(menu_option_size,Size),
      Mode \== Option.

   max(A,B,Max):-
      (A >= B -> Max = A ; Max = B).

   min(A,B,Min):-
      ( A < B -> Min = A ; Min = B).

   square_constraint(Apparent,Start,Size,Real):-
      (Apparent < Start -> Real is Start - Size ; Real is Start).

   position(A,B,Min,Delta):-
       ( A >= B -> Min is B, Delta is A - B ; Min is A,Delta is B - A).


   difference(A1,B1,A2,B2,D):-
       ((A1 - A2) > 0 ->
         Dx is A1 - A2
        ;
         Dx is A2 - A1
        )
        ,
        ((B1 - B2) > 0 ->
         Dy is B1 - B2
        ;
         Dy is B2 - B1
        ),
max(Dx,Dy,D).

   near(X,Y,X1,Y1,Delt):-
       difference(X,Y,X1,Y1,Delt),
       Delt < 5.


   near_end_point([Xs,Ys],X,Y):-
   setof(T,near_end_point1(T,X,Y),[[_,Xs,Ys]|_]).

   near_end_point1([Diff,Xs,Ys],X,Y):-
       ending_point(Xs,Ys),
       near(Xs,Ys,X,Y,Diff).


   in_region([Xi,Yi],X,Y):-
  (( data(Label,Xi,Yi,_,_,_);delay_data(Label,Xi,Yi,_,_)),
     Xf is Xi + 45,
          Yf is Yi + 45,
     X >= Xi , X < Xf , Y >= Yi , Y < Yf)
        ;
    (( out(Label,Xi,Yi) ,
          Xf is Xi+ 60, Yf1 is Yi + 10,Yf2 is Yi - 10,
     X >= Xi , X =< Xf , Y >= Yf2 , Y =< Yf1
    ;
          in1(Label,Xi,Yi),
     Xf is Xi - 50, Yf1 is Yi + 10,Yf2 is Yi - 10,
     X =< Xi, X >= Xf, Y >= Yf2 , Y =< Yf1
    ;
     in2(Label,Xi,Yi),
     Xf is Xi - 80, Yf1 is Yi + 10,Yf2 is Yi - 10,
```

```
    X =< Xi, X >= Xf, Y >= Yf2 , Y =< Yf1 )
   )
     ;
   ( drawn_figure(element(rectangle,Label),info(Xi,Yi,Xl,Yl),_),
     near(X,Y,Xi,Yi,Diff)
   )
  ;
   ((line(Xi,Yi,Xl,Yl);line(Xl,Yl,Xi,Yi)),
((Xi =< Xl,Yi =< Yl,
       X =< Xl, X >= Xi, Y =< Yl, Y >= Yi)
     ;
      (Xi >= Xl, Yi >= Yl,
       X =< Xi, X >= Xl, Y =< Yi, Y >= Yl)
;
(Xi =< Xl, Yi >= Yl,
  X =< Xl, X >= Xi, Y =< Yi, Y >= Yl)
;
(Xi >= Xl, Yi =< Yl,
  X =< Xi, X >= Xl, Y =< Yl, Y >= Yi)
)
   )
      ).


  in_line([Xi,Yi,Xl,Yl],X,Y):-
(line(Xi,Yi,Xl,Yl);line(Xl,Yl,Xi,Yi)),
((Xi =< Xl,Yi =< Yl,
       X =< Xl, X >= Xi, Y =< Yl, Y >= Yi)
     ;
      (Xi >= Xl, Yi >= Yl,
       X =< Xi, X >= Xl, Y =< Yi, Y >= Yl)
;
(Xi =< Xl, Yi >= Yl,
  X =< Xl, X >= Xi, Y =< Yi, Y >= Yl)
;
(Xi >= Xl, Yi =< Yl,
  X =< Xi, X >= Xl, Y =< Yl, Y >= Yi)
).


   delete(_,[],[]).

   delete(X,[X|L],M):-
     !,
     delete(X,L,M).

   delete(X,[Y|L1],[Y|L2]):-
     delete(X,L1,L2).

   insert([],L,L).
   insert(X,List,Big):-
member(X,List),
insert([],List,Big) , !.

   insert(X,List,Big):-
nonmember(X,List),
append(List,[X],Big), !.

   insert(X,List,Big):-
append(List,[X],Big),  !.



%     ------------------------------------------------------------


window_data(width,500).
window_data(height,500).
%window_data(titlefont,'*-charter-medium-r-*-240-*').
window_data(titlefont,'*-times-bold-i-*-240-*').
window_data(box_font,'*-lucidabright-demibold-*-normal-*-12-*').
window_data(title,'GUI FOR DIG.FILTERS').
window_data(menu_wide,100).
window_data(menu_width,350).
window_data(menu_height,350).
window_data(boundary_height,3).
window_data(horizontal_space,5).
```

```
window_data(vertical_space,5).
window_data(menu_option_size,60).
window_data(menu_font,'*-times-bold-i-*-240-*').
window_data(default_mode,adder).
window_data(default_line,2).
window_data(box_name,'QUIT').
window_data(root_cursor,left_ptr).
window_data(draw_cursor,crosshair).
window_data(box_size,40).
```

APPENDIX B

Implementation in C and Simulation Results

B.1 Program in C

```
#include<stdio.h>
#include <string.h>

main()

  {
  int enc[15],message[15],rec[15];
  int k,i,jjj;
  k = 7;
  message[0] = 1; message[1] = 6; message[2] = 1; message[3] = 2;
  message[4] = 5; message[5] = 0; message[6] = 3;

  printf(" the message is as follows:\n");
  for ( i=k; i<15;i++)
message[i] = -1;
  for ( i=0;i<15;i++)
     printf(" message %d is %d \n",i,message[i]);
  printf("\n");
  encode(message,enc,k);
  channel(enc,rec);
  decode(k,rec);
  for (i=0;i<=14;i++)
    printf("  %d -- message  is %d  , encoded is %d ,  corrected is %d \n",i
        ,message[i],enc[i],rec[i]);

 }


#include<stdio.h>
#include <string.h>
int get_inverse(x)
   {
    if ( x == -1)
      return(-1);
    else
      {
      if (x != 0)
        return(15 - x);
      else
       return(0);
       }

   }

void  decode(k,rec)

int    k,rec[];
  {
  /* trying to build the data structure... we have three tables:
     addition table: it gives the the value of the GF numbers.

     inverse table: it gives the inverse of the GF element.

     transformation table: it transforms the binary input to the
     power representation of alpha.   */

  int  lem[15],erasure[15],
keppa[15], b[15],pos[15],
k_temp[15],e[15],v[15];

  int  i,j,temp,l,r,row,
limit,tem,
del,delta;
```

```
/*  initializing the error vector */
   for (i=0;i<=14;i++)
v[i] = rec[i];

/*  calculating the syndrome elements;
------------------------------- */
    printf(" enter please the number of erasures : row\n ");
    printf("remember that the relation between the errors and erasures: \n");
    printf(" row + 2*no. of errors <= n - k \n");
    scanf("%d",&row);
    if(row>15-k)
    {
     printf(" Error in entering row violating rule row + 2*t <= 15-k\n");
     scanf("%d",&row);
    }
    limit = 15-k+row;
  /*calculating the erasure polynomial :
-------------------------------------*/
    i=1;
    while(i<=row)
      {
       printf("enter please the position :\n");
       scanf("%d",&pos[i]);
       if (pos[i] >15)
         printf(" error message  wrong position reenter again \n");
       else
       {
         printf("The position of the erasure is %d \n",pos[i]);
         i++;
       }
      }
/* initializing lem[i] i=0,...,n-1  */
   for (i=0;i<15;i++)
lem[i] = 0;
   r = 1;
   while(r<=row)
    {
     for(i=0;i<15;i++)
{
tem = get_sum(0,(get_inverse(i)+pos[r])%15);
if ((tem == -1) || (lem[i] == -1))
  lem[i] = -1;
else
  lem[i] = (lem[i] + tem)%15;
printf(" lem[%d] is now %d\n ",i,lem[i]);
}
     r++;
    }
 printf("\n");

/*  step 2. calcultating erata locator vector using Berlekamp-Massey algo.
-------------------------------------------------------------------------*/
    /* initialization of keppa and b ---------*/

  for (i=0; i<15; i++)
   {
keppa[i] = lem[i];
b[i] = lem[i];
   }
  l = 0;
  while(r<=limit)
   {

/* calculating delta-r ----------*/

    delta = -1;
    for (i=0; i<15; i++)
     {
      if ((keppa[i] == -1) || (v[i] == -1))
        temp = -1;
      else
        temp = (i*r + keppa[i] + v[i])%15;
      delta = get_sum(delta,temp);
printf(" delta is now  %d ",delta);
     }
printf("\n");
/* determining the value of del----------*/
```

```
   if((delta != -1) & (2*l-row <= r-1))
     {
del = 1;
l = r-l - row;
     }
   else
del = 0;
   printf(" l is now %d , r is %d, and del is %d., \n" ,l,r,del);
   for (i=0; i<15; i++)
k_temp[i] = keppa[i];
/* calculating the eqns 2.52 from Shayan's thesis
-----------------------------------------------------*/
    if (del == 1)
     {
if (delta == -1)
{
for (i=0; i<15; i++)
  b[i] = k_temp[i];
     }
      else
{
for ( i=0; i<15; i++)
 {
 if (b[i] == -1)
        temp = -1;
 else
   temp = ( delta + get_inverse(i) + b[i])%15;
 if ( k_temp[i] == -1)
 {
   keppa[i] = temp;
   b[i] = -1;
     }
 else
 {
   keppa [i] = get_sum(k_temp[i],temp);
    b[i] = (get_inverse(delta) + k_temp[i])%15;
 }
      }
     }
    }
   else   /* del = 0   */
    {
    if ( delta != -1)
    {
     for (i=0; i<15; i++)
       {
        if (b[i] == -1)
        {
          temp = -1;
          b[i] = -1;
        }
        else
          temp = ( delta + get_inverse(i) + b[i])%15;
        if ( k_temp[i] == -1)
          keppa[i] = temp;
        else
          keppa [i] = get_sum(k_temp[i],temp);
        b[i] = (get_inverse(i) + b[i])%15;
 printf(" keppa[%d] is %d ,b[%d] is %d ",i,keppa[i],i,b[i]);
 printf("\n");
       }
    }
    }
for (i=0; i<15; i++)
{
printf(" keppa[%d] is %d ,b[%d] is %d ",i,keppa[i],i,b[i]);
      printf("\n");
}
   r++;
    }
/* doing step 3:-
 To find the errata values e[i] from eqns 2.54 and 2.55 in
  shayan's thesis.....
-----------------------------------------------------------*/
/* initialization for e[i]   .....*/
   for (i=0; i<15; i++)
e[i] =v[i];
   for ( r=limit+1; r<15; r++)
    {
    delta = -1;
```

```
    for (i=0; i<15; i++)
      {
       if ((keppa[i] == -1) || (e[i] == -1))
          temp = -1;
       else
          temp = (i*r + keppa[i] + e[i])%15;
       delta = get_sum(delta,temp);
      }
    for ( i=0; i<15; i++)
      {
       if (delta != -1)
       {
        e[i] = get_sum(e[i],(delta + get_inverse(r)*i)%15);
       }
printf(" e[%d] is %d  , ",i, e[i]);
    }
printf("\n");
    }
   /* doing now step four eqn 2.71
   ----------------------------*/

   for ( i=0; i<15; i++)
     {
       if ( keppa[i] != -1)
 e[i] = -1;
     }
   for ( i=0; i<15; i++)
     {
       rec[i] = get_sum(v[i],e[i]);
     }
  }

#include <stdio.h>
#include <string.h>

void    encode(inf,enc,k)

int  inf[],enc[],k;
   {
   int  i,b[15][15],
        init_b[15],j,q,count,
        n,m,limit;

   int  divider[15],init_d[15],
        a[15][15],rem[15],
        upper,r[15],temp;

   /* this is to initialise the multiplier -
   ========================================== */
   init_b[0] = 1;
   init_b[1] = 0;
   for (i=2;i<=14;i++)
        init_b[i] = -1;
   /* this is to initialize the divider which equals information bits
      shifted by 15-k to the right -
   ================================================================================*/
   j=0;
   for (i=0; i<=14; i++)
        init_d[i] = -1;
   for (i=14-k+1;i<=14;i++)
     {
       init_d[i] = inf[j];
       j++;
     }
   limit = 15-k;
  /* initializing the generator polyinomial---
  ------------------------------------------*/
   b[0][0] = 0;
   b[1][0] = 1;
   b[1][1] = 0;
   for (i=2; i< limit; i++)
   {
     b[i][0] = (b[i-1][0] + i)%15;      /*bnext = bprev*alpha_i */
     for (j=1; j<i;j++)
       {
         if ( b[i-1][j] == -1)
            temp = -1;
         else
            temp = (i + b[i-1][j])%15;
```

```
        b[i][j] = get_sum(b[i-1][j-1],temp);
      }
    b[i][i] = 0;
  }
     /* now calculating the division-
     ==========================================*/
    upper = 14;
    i = 1;
    while(i <= limit)
      {
      for (q=0; q<=upper;q++)
      divider[upper-1] = init_d[upper];
      for (j=2; j<upper+1; j++)
        {
        if (divider[upper-j+1] == -1)
          temp = -1;
        else
          temp = (i+ divider[upper-j+1])%15;
        divider[upper-j] = get_sum(init_d[upper-j+1],temp);
        if (upper-j == 0)
          {
          if(divider[0] == -1)
            temp = -1;
          else
            temp = (i + divider[0])%15;
          r[i-1] = get_sum(init_d[0],temp);
          }
        }
      i++;
      upper--;
      for ( q=0;q<=14; q++)
        init_d[q] = divider[q];
      }
     /*  for(j=14-i;j>=0;j++)
          init_d[j] = divider[j];*/
/* Now calculating the check parity symbols--
-------------------------------------------*/

    count = 0;
    while(count < limit)
    {
      for (j=0; j<=count;j++)
      {
        if (r[count] == -1)
          a[count][j] = -1;
        else
        {
          if (b[count][j] == -1)
            a[count][j] = -1;
          else
            a[count][j] = (r[count] + b[count][j])%15;
        }
      }
      count++;
    }
/* initializing the remainder--
-----------------------------*/

  for (i=0;i<limit;i++)
    rem[i] = -1;
  j = 0;
  while (j<limit)
    {
    for (i=0;i<=j;i++)
    {
      rem[i] = get_sum(a[j][i],rem[i]);
    }
    j++;
    }
    for(j=0;j<limit;j++)
       enc[j] = rem[j];
    for (i=limit;i<15;i++)
      enc[i] = inf[i-limit];
  }

#include<stdio.h>
#include <string.h>
```

```
void   channel(enc,rec)

int    enc[],rec[];

/* this simulates the noisy channel by allowing the user to enter the
change in the recieved vector.     */


    {
    int   i,error,pos;

    for (i=0; i<=14; i++)
rec[i] = enc[i];
    printf(" How manny errors you would like to introduce?\n");
    printf(" REMMEMBER : if the number of errors exceeds (15 - k)/2 ==>\n");
    printf(" then the program will not be able to correct the errors\n");
    scanf("%d",&error);
    i = 1;
    while (i <= error)
{
printf(" enter please the position of the error %d\n  ",i);
scanf("%d",&pos);
if ( pos > 14)
printf(" error in position reenter again \n");
else
{
printf(" enter the value of the error \n");
scanf("%d",&rec[pos]);
if ( rec[pos] > 14 || rec[pos] < -1)
printf(" error in value reenter the value again\n");
else
{
printf(" the error value of %d is %d\n",pos, rec[pos]);
i++;
}
}
}
    }



#include <stdio.h>
#include<string.h>

int get_sum(x,y)
  {int table[15][15],
      i,j,nw;
  if(x>y)
    { nw=y;
      y=x;
      x=nw;
    }
  table[0][0]=-1; table[0][1]=4; table[0][2]=8; table[0][3]=14; table[0][4]=1;
  table[0][5]=10; table[0][6]=13; table[0][7]=9; table[0][8]=2; table[0][9]=7;
  table[0][10]=5; table[0][11]=12; table[0][12]=11;table[0][13]=6;
  table[0][14]=3;table[1][2]=5; table[1][3]=9; table[1][4]=0; table[1][5]=2;
  table[1][6]=11; table[1][7]=14; table[1][8]=10; table[1][9]=3; table[1][10]=8;
  table[1][11]=6; table[1][12]=13; table[1][13]=12; table[1][14]=7; table[2][3]=6;
  table[2][4]=10; table[2][5]=1; table[2][6]=3; table[2][7]=12; table[2][8]=0;
  table[2][9]=11; table[2][10]=4; table[2][11]=9; table[2][12]=7; table[2][13]=14;
  table[2][14]=13; table[3][4]=7; table[3][5]=11; table[3][6]=2; table[3][7]=4;
  table[3][8]=13; table[3][9]=1; table[3][10]=12; table[3][11]=5; table[3][12]=10;
  table[3][13]=8; table[3][14]=0; table[4][5]=8; table[4][6]=12; table[4][7]=3;
  table[4][8]=5; table[4][9]=14; table[4][10]=2; table[4][11]=13; table[4][12]=6;
  table[4][13]=11; table[4][14]=9; table[5][6]=9; table[5][7]=13; table[5][8]=4;
  table[5][9]=6; table[5][10]=0; table[5][11]=3; table[5][12]=14; table[5][13]=7;
  table[5][14]=12;  table[6][7]=10; table[6][8]=14; table[6][9]=5; table[6][10]=7;
  table[6][11]=1; table[6][12]=4; table[6][13]=0; table[6][14]=8; table[7][8]=11;
  table[7][9]=0; table[7][10]=6; table[7][11]=8; table[7][12]=2; table[7][13]=5;
  table[7][14]=1; table[8][9]=12; table[8][10]=1; table[8][11]=7; table[8][12]=9;
  table[8][13]=3; table[8][14]=6;  table[9][10]=13; table[9][11]=2; table[9][12]=8;
  table[9][13]=10; table[9][14]=4; table[10][11]=14; table[10][12]=3;
  table[10][13]=9; table[10][14]=11; table[11][12]=0;table[11][13]=4; table[11][14]=10;
  table[12][13]=1; table[12][14]=5; table[13][14]=2;


  if (x == y)
    return (-1);
  if (x == -1)
```

```
    return(y);
  if (y == -1)
    return(x);
  else
    return(table[x][y]);

}
```

B.2 Simulation Results

```
enter please the length of the message :
7
Enter all the the terms of the message in power of alpha...

enter the message 0 1
enter the message 1 6
enter the message 2 1
enter the message 3 2
enter the message 4 5
enter the message 5 0
enter the message 6 3
the message is as follows:
message 0 is 1
message 1 is 6
message 2 is 1
message 3 is 2
message 4 is 5
message 5 is 0
message 6 is 3
message 7 is -1
message 8 is -1
message 9 is -1
message 10 is -1
message 11 is -1
message 12 is -1
message 13 is -1
message 14 is -1

limit is 8
the multiplication polynomial is being calculated
Now calculating the division :

the parity symbols are the following:
encoded vector 0  is : 1
encoded vector 1  is : 4
encoded vector 2  is : 1
encoded vector 3  is : 12
encoded vector 4  is : 5
encoded vector 5  is : 1
encoded vector 6  is : 4
encoded vector 7  is : 7
encoded vector 8  is : 1
encoded vector 9  is : 6
encoded vector 10  is : 1
encoded vector 11  is : 2
encoded vector 12  is : 5
encoded vector 13  is : 0
encoded vector 14  is : 3
How manny errors you would like to introduce?
REMMEMBER : if the number of errors exceeds (15 - k)/2 ==>
then the program will not be able to correct the errors
2
enter please the position of the error 1
 1
enter the value of the error
6
the error value of 1 is 6
enter please the position of the error 2
 6
enter the value of the error
2
the error value of 6 is 2
the recieved message is as following.....
the rec [0] is 1
the rec [1] is 6
the rec [2] is 1
the rec [3] is 12
```

```
the rec [4] is 5
the rec [5] is 1
the rec [6] is 2
the rec [7] is 7
the rec [8] is 1
the rec [9] is 6
the rec [10] is 1
the rec [11] is 2
the rec [12] is 5
the rec [13] is 0
the rec [14] is 3
the vector 0 is : 1
the vector 1 is : 6
the vector 2 is : 1
the vector 3 is : 12
the vector 4 is : 5
the vector 5 is : 1
the vector 6 is : 2
the vector 7 is : 7
the vector 8 is : 1
the vector 9 is : 6
the vector 10 is : 1
the vector 11 is : 2
the vector 12 is : 5
the vector 13 is : 0
the vector 14 is : 3
enter please the number of erasures : row
remember that the relation between the errors and erasures:
row + 2*no. of errors <= n - k
1
enter please the position :
3
The position of the erasure is 3
lem is now 14 ... lem is now 8 ... lem is now 4 ... lem is now -1 ...
lem is now 3 ... lem is now 6 ... lem is now 11 ... lem is now 12 ..
. lem is now 5 ... lem is now 7 ... lem is now 2 ... lem is now 9 ...
lem is now 13 ... lem is now 10 ... lem is now 1 ...
lemda[0] is 14 ..
lemda[1] is 8 ..
lemda[2] is 4 ..
lemda[3] is -1 ..
lemda[4] is 3 ..
lemda[5] is 6 ..
lemda[6] is 11 ..
lemda[7] is 12 ..
lemda[8] is 5 ..
lemda[9] is 7 ..
lemda[10] is 2 ..
lemda[11] is 9 ..
lemda[12] is 13 ..
lemda[13] is 10 ..
lemda[14] is 1 ..
r is now 2 ...
delta is now 0... and temp is 0delta is now 4... and temp is 1delta is
 now 14... and temp is 9delta is now 14... and temp is -1delta is now 7
... and temp is 1delta is now 12... and temp is 2delta is now 3...
and temp is 10delta is now -1... and temp is 3delta is now 7...
and temp is 7delta is now 14... and temp is 1delta is now 6...
and temp is 8delta is now 2... and temp is 3delta is now 7...
and temp is 12delta is now 10... and temp is 6delta is now 4...
and temp is 2
delta is 4  ...
 del is now 1  ....
 keppa[0] is 0 ..., b[0] is 10...
 keppa[1] is 7 ..., b[1] is 4...
 keppa[2] is 12 ..., b[2] is 0...
 keppa[3] is -1 ..., b[3] is -1...
 keppa[4] is -1 ..., b[4] is 14...
 keppa[5] is 9 ..., b[5] is 2...
 keppa[6] is 2 ..., b[6] is 7...
 keppa[7] is 8 ..., b[7] is 8...
 keppa[8] is 2 ..., b[8] is 1...
 keppa[9] is 12 ..., b[9] is 3...
 keppa[10] is 9 ..., b[10] is 13...
 keppa[11] is 11 ..., b[11] is 5...
 keppa[12] is 7 ..., b[12] is 9...
 keppa[13] is 8 ..., b[13] is 6...
 keppa[14] is 11 ..., b[14] is 12...
r is now 3 ...
delta is now 1... and temp is 1delta is now -1... and temp is 1delta
```

is now 4... and temp is 4delta is now 4... and temp is -1delta is now 4
... and temp is -1delta is now 2... and temp is 10delta is now 12...
and temp is 7delta is now 4... and temp is 6delta is now 6... and temp
is 12delta is now 13... and temp is 0delta is now 9... and temp is 10
delta is now 3... and temp is 1delta is now -1... and temp is 3
delta is now 2... and temp is 2delta is now 9... and temp is 11
delta is 9 ...
del is now 0 ....
keppa[0] is 1 ..., b[0] is 10...
keppa[1] is 2 ..., b[1] is 3...
keppa[2] is 2 ..., b[2] is 13...
keppa[3] is -1 ..., b[3] is 11...
keppa[4] is 4 ..., b[4] is 10...
keppa[5] is 5 ..., b[5] is 12...
keppa[6] is 4 ..., b[6] is 1...
keppa[7] is 1 ..., b[7] is 1...
keppa[8] is -1 ..., b[8] is 8...
keppa[9] is 10 ..., b[9] is 9...
keppa[10] is 8 ..., b[10] is 3...
keppa[11] is 5 ..., b[11] is 9...
keppa[12] is 10 ..., b[12] is 12...
keppa[13] is 0 ..., b[13] is 8...
keppa[14] is 8 ..., b[14] is 13...
r is now 4 ...
delta is now 2... and temp is 2delta is now 7... and temp is 12delta is
now 8... and temp is 11delta is now 8... and temp is -1delta is now 1
... and temp is 10delta is now 6... and temp is 11delta is now 13...
and temp is 0delta is now 0... and temp is 6delta is now 0... and
temp is -1delta is now 9... and temp is 7delta is now 14... and
temp is 4delta is now 8... and temp is 6delta is now 13... and
temp is 3delta is now 5... and temp is 7delta is now 13... and
temp is 7
delta is 13 ...
del is now 1 ....
keppa[0] is 10 ..., b[0] is 3...
keppa[1] is 8 ..., b[1] is 4...
keppa[2] is 11 ..., b[2] is 4...
keppa[3] is 6 ..., b[3] is -1...
keppa[4] is -1 ..., b[4] is 6...
keppa[5] is -1 ..., b[5] is 7...
keppa[6] is 5 ..., b[6] is 6...
keppa[7] is 14 ..., b[7] is 3...
keppa[8] is 13 ..., b[8] is -1...
keppa[9] is 9 ..., b[9] is 12...
keppa[10] is 14 ..., b[10] is 10...
keppa[11] is 3 ..., b[11] is 7...
keppa[12] is 9 ..., b[12] is 12...
keppa[13] is 2 ..., b[13] is 2...
keppa[14] is 9 ..., b[14] is 10...
r is now 5 ...
delta is now 11... and temp is 11delta is now 13... and temp is 4
delta is now 5... and temp is 7delta is now 11... and temp is 3delta
is now 11... and temp is -1delta is now 11... and temp is -1delta is now
8... and temp is 7delta is now 7... and temp is 11delta is now 0...
and temp is 9delta is now -1... and temp is 0delta is now 5... and temp
is 5delta is now 10... and temp is 0delta is now 11... and temp is 14
delta is now 8... and temp is 7delta is now 11... and temp is 7
delta is 11 ...
del is now 1 ....
keppa[0] is 11 ..., b[0] is 14...
keppa[1] is 6 ..., b[1] is 12...
keppa[2] is 4 ..., b[2] is 0...
keppa[3] is 6 ..., b[3] is 10...
keppa[4] is 13 ..., b[4] is -1...
keppa[5] is 13 ..., b[5] is -1...
keppa[6] is 3 ..., b[6] is 9...
keppa[7] is 1 ..., b[7] is 3...
keppa[8] is 13 ..., b[8] is 2...
keppa[9] is 4 ..., b[9] is 13...
keppa[10] is 10 ..., b[10] is 3...
keppa[11] is 4 ..., b[11] is 7...
keppa[12] is 2 ..., b[12] is 13...
keppa[13] is 8 ..., b[13] is 6...
keppa[14] is 0 ..., b[14] is 13...
r is now 6 ...
delta is now 12... and temp is 12delta is now 10... and temp is 3delta
is now 4... and temp is 2delta is now 12... and temp is 6delta is now
-1... and temp is 12delta is now 14... and temp is 14delta is now 10...
and temp is 11delta is now 0... and temp is 5delta is now 8... and temp
is 2delta is now 5... and temp is 4delta is now 3... and temp is 11

delta is now 10... and temp is 12delta is now 2... and temp is 4delta
is now 9... and temp is 11delta is now 8... and temp is 12
delta is 8 ...
del is now 1 ....
keppa[0] is 8 ..., b[0] is 3...
keppa[1] is 12 ..., b[1] is 13...
keppa[2] is 12 ..., b[2] is 11...
keppa[3] is 13 ..., b[3] is 13...
keppa[4] is 13 ..., b[4] is 5...
keppa[5] is 13 ..., b[5] is 5...
keppa[6] is 5 ..., b[6] is 10...
keppa[7] is 0 ..., b[7] is 8...
keppa[8] is 14 ..., b[8] is 5...
keppa[9] is 6 ..., b[9] is 11...
keppa[10] is 8 ..., b[10] is 2...
keppa[11] is -1 ..., b[11] is 11...
keppa[12] is 11 ..., b[12] is 9...
keppa[13] is 10 ..., b[13] is 0...
keppa[14] is 9 ..., b[14] is 7...
r is now 7 ...
delta is now 9... and temp is 9delta is now 13... and temp is 10delta
is now 1... and temp is 12delta is now -1... and temp is 1delta is
now 1... and temp is 1delta is now 0... and temp is 4delta is now 1...
and temp is 4delta is now 6... and temp is 11delta is now 1... and
temp is 11delta is now 4... and temp is 0delta is now -1... and temp
is 4delta is now -1... and temp is -1delta is now 10... and temp is 10
delta is now 14... and temp is 11delta is now 12... and temp is 5
delta is 12 ...
del is now 1 ....
keppa[0] is 2 ..., b[0] is 11...
keppa[1] is 8 ..., b[1] is 0...
keppa[2] is 4 ..., b[2] is 0...
keppa[3] is 5 ..., b[3] is 1...
keppa[4] is -1 ..., b[4] is 1...
keppa[5] is 1 ..., b[5] is 1...
keppa[6] is 2 ..., b[6] is 8...
keppa[7] is 6 ..., b[7] is 3...
keppa[8] is 4 ..., b[8] is 2...
keppa[9] is 8 ..., b[9] is 9...
keppa[10] is 5 ..., b[10] is 11...
keppa[11] is 12 ..., b[11] is -1...
keppa[12] is 2 ..., b[12] is 14...
keppa[13] is 11 ..., b[13] is 13...
keppa[14] is 6 ..., b[14] is 12...
r is now 8 ...
delta is now 3... and temp is 3delta is now 4... and temp is 7delta is
now 12... and temp is 6delta is now 0... and temp is 11delta is now 0
... and temp is -1delta is now 11... and temp is 12delta is now 8...
and temp is 7delta is now 12... and temp is 9delta is now 8... and temp
is 9delta is now 7... and temp is 11delta is now 8... and temp is 11
delta is now 9... and temp is 12delta is now 10... and temp is 13
delta is now -1... and temp is 10delta is now 1... and temp is 1
delta is 1 ...
del is now 1 ....
keppa[0] is 7 ..., b[0] is 1...
keppa[1] is 2 ..., b[1] is 7...
keppa[2] is 9 ..., b[2] is 3...
keppa[3] is 12 ..., b[3] is 4...
keppa[4] is 13 ..., b[4] is -1...
keppa[5] is 13 ..., b[5] is 0...
keppa[6] is 6 ..., b[6] is 1...
keppa[7] is 4 ..., b[7] is 5...
keppa[8] is 2 ..., b[8] is 3...
keppa[9] is 10 ..., b[9] is 7...
keppa[10] is 1 ..., b[10] is 4...
keppa[11] is 12 ..., b[11] is 11...
keppa[12] is 6 ..., b[12] is 1...
keppa[13] is 6 ..., b[13] is 10...
keppa[14] is 8 ..., b[14] is 5...
r is now 9 ...
delta is now 8... and temp is 8delta is now 0... and temp is 2delta is
now 6... and temp is 13delta is now -1... and temp is 6delta is now 9
... and temp is 9delta is now 4... and temp is 14delta is now 10...
and temp is 2delta is now 11... and temp is 14delta is now 12... and
temp is 0delta is now 2... and temp is 7delta is now -1... and temp is
2delta is now 8... and temp is 8delta is now 6... and temp is 14
delta is now 2... and temp is 3delta is now -1... and temp is 2
delta is -1 ...
del is now 0 ....
keppa[0] is 7 ..., b[0] is 1...

keppa[1] is 2 ..., b[1] is 7...
keppa[2] is 9 ..., b[2] is 3...
keppa[3] is 12 ..., b[3] is 4...
keppa[4] is 13 ..., b[4] is -1...
keppa[5] is 13 ..., b[5] is 0...
keppa[6] is 6 ..., b[6] is 1...
keppa[7] is 4 ..., b[7] is 5...
keppa[8] is 2 ..., b[8] is 3...
keppa[9] is 10 ..., b[9] is 7...
keppa[10] is 1 ..., b[10] is 4...
keppa[11] is 12 ..., b[11] is 11...
keppa[12] is 6 ..., b[12] is 1...
keppa[13] is 6 ..., b[13] is 10...
keppa[14] is 8 ..., b[14] is 5...
delta is 8 ...delta is 13 ...delta is 6 ...delta is 5 ...delta is 7 ...delta is 3 ...delta is 13 ...delta is 0 ...delta is 2 ...delta is 5 ...delta is 14 ...delta is 9 ...delta is 2 ...delta is 5 ...delta is 2 ...
delta is 2 ...
e[0] is 5....
e[1] is 10....
e[2] is 13....
e[3] is 7....
e[4] is 13....
e[5] is 13....
e[6] is -1....
e[7] is -1....
e[8] is 13....
e[9] is 3....
e[10] is 14....
e[11] is 7....
e[12] is 1....
e[13] is 9....
e[14] is 10....
delta is 12 ...delta is 9 ...delta is 4 ...delta is 3 ...delta is 12 ...delta is 4 ...delta is 4 ...delta is 4 ...delta is 11 ...delta is 8 ...delta is 4 ...delta is 8 ...delta is 5 ...delta is 4 ...
delta is 3 ...
delta is 3 ...
e[0] is 11....
e[1] is 6....
e[2] is 4....
e[3] is 9....
e[4] is 11....
e[5] is 3....
e[6] is 12....
e[7] is 1....
e[8] is 7....
e[9] is 1....
e[10] is 2....
e[11] is 12....
e[12] is 11....
e[13] is 13....
e[14] is 11....
delta is 3 ...delta is 11 ...delta is 8 ...delta is 9 ...delta is 8 ...delta is 10 ...delta is 5 ...delta is 12 ...delta is 11 ...
delta is 10 ...delta is 12 ...delta is 4 ...delta is 13 ...delta is 9 ...delta is 0 ...
delta is 0 ...
e[0] is 12....
e[1] is 2....
e[2] is 12....
e[3] is -1....
e[4] is 0....
e[5] is 14....
e[6] is 10....
e[7] is 11....
e[8] is 0....
e[9] is 13....
e[10] is 8....
e[11] is 10....
e[12] is 1....
e[13] is 10....
e[14] is 0....
delta is 4 ...delta is 10 ...delta is 4 ...delta is 4 ...delta is 8 ...delta is 0 ...delta is 1 ...delta is -1 ...delta is 1 ...
delta is 2 ...delta is 10 ...delta is 5 ...delta is 7 ...delta is 13 ...delta is 9 ...
delta is 9 ...
e[0] is 8....

```
e[1] is 9....
e[2] is 1....
e[3] is 0....
e[4] is 8....
e[5] is 9....
e[6] is 7....
e[7] is 7....
e[8] is 5....
e[9] is 1....
e[10] is 6....
e[11] is 8....
e[12] is 9....
e[13] is 0....
e[14] is 9....
delta is 0  ...delta is 5  ...delta is 4  ...delta is 14  ...delta is 13
  ...delta is 14  ...delta is 1  ...delta is 0  ...delta is 3  ...
delta is 6  ...delta is 4  ...delta is 14  ...delta is 0  ...
delta is 2  ...delta is 6  ...
delta is 6  ...
e[0] is 14....
e[1] is 0....
e[2] is 10....
e[3] is 7....
e[4] is 1....
e[5] is 2....
e[6] is 2....
e[7] is 5....
e[8] is 12....
e[9] is 4....
e[10] is 11....
e[11] is 0....
e[12] is 1....
e[13] is 1....
e[14] is 6....
e[0] is 14 ...
e[1] is 0 ...
e[2] is 10 ...
e[3] is 7 ...
e[4] is 1 ...
e[5] is 2 ...
e[6] is 2 ...
e[7] is 5 ...
e[8] is 12 ...
e[9] is 4 ...
e[10] is 11 ...
e[11] is 0 ...
e[12] is 1 ...
e[13] is 1 ...
e[14] is 6 ...
error[0] is -1 .... rec[0] is 1
error[1] is -1 .... rec[1] is 6
error[2] is -1 .... rec[2] is 1
error[3] is -1 .... rec[3] is 12
error[4] is -1 .... rec[4] is 5
error[5] is -1 .... rec[5] is 1
error[6] is -1 .... rec[6] is 2
error[7] is -1 .... rec[7] is 7
error[8] is -1 .... rec[8] is 1
error[9] is -1 .... rec[9] is 6
error[10] is -1 .... rec[10] is 1
error[11] is -1 .... rec[11] is 2
error[12] is -1 .... rec[12] is 5
error[13] is -1 .... rec[13] is 0
error[14] is -1 .... rec[14] is 3
  0 -- message  is 1  , encoded is 1 ,  corrected is 1
  1 -- message  is 6  , encoded is 4 ,  corrected is 6
  2 -- message  is 1  , encoded is 1 ,  corrected is 1
  3 -- message  is 2  , encoded is 12 ,  corrected is 12
  4 -- message  is 5  , encoded is 5 ,  corrected is 5
  5 -- message  is 0  , encoded is 1 ,  corrected is 1
  6 -- message  is 3  , encoded is 4 ,  corrected is 2
  7 -- message  is -1  , encoded is 7 ,  corrected is 7
  8 -- message  is -1  , encoded is 1 ,  corrected is 1
  9 -- message  is -1  , encoded is 6 ,  corrected is 6
  10 -- message  is -1  , encoded is 1 ,  corrected is 1
  11 -- message  is -1  , encoded is 2 ,  corrected is 2
  12 -- message  is -1  , encoded is 5 ,  corrected is 5
  13 -- message  is -1  , encoded is 0 ,  corrected is 0
  14 -- message  is -1  , encoded is 3 ,  corrected is 3
```

APPENDIX C

VHDL Coding of Reed-Solomon (*n*,*k*) Decoder

```
-- VHDL Model Created from SGE Symbol alpha.sym -- Sep 26 16:11:39 1995

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity ALPHA is
    Port (    IN1 : In    std_logic_vector (3 downto 0);
            OUT1 : Out   std_logic_vector (3 downto 0) );
end ALPHA;

architecture BEHAVIORAL of ALPHA is
  begin
out1(0) <= in1(3);
out1(1) <= in1(3) xor in1(0);
out1(2) <= in1(1);
out1(3) <= in1(2);
end BEHAVIORAL;

configuration CFG_ALPHA_BEHAVIORAL of ALPHA is
  for BEHAVIORAL

  end for;

end CFG_ALPHA_BEHAVIORAL;

-- VHDL Model Created from SGE Symbol and1to4.sym -- May  2 10:21:24 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity AND1TO4 is
    Port (    BI : In    std_logic;
            IN1 : In    std_logic_vector (3 downto 0);
           OUTP : Out   std_logic_vector (3 downto 0) );
end AND1TO4;

architecture BEHAVIORAL of AND1TO4 is

  begin
outp(3) <= bi and in1(3);
outp(2) <= bi and in1(2);
outp(1) <= bi and in1(1);
outp(0) <= bi and in1(0);

end BEHAVIORAL;

configuration CFG_AND1TO4_BEHAVIORAL of AND1TO4 is
  for BEHAVIORAL

  end for;

end CFG_AND1TO4_BEHAVIORAL;

-- VHDL Model Created from SGE Schematic galois_mult.sch -- May  3 09:40:12 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity GALOIS_MULT is
    Port (      B : In    std_logic_vector (3 downto 0);
          GAMMA : In    std_logic_vector (3 downto 0);
```

```
        RES : Out  std_logic_vector (3 downto 0) );
end GALOIS_MULT;

architecture SCHEMATIC of GALOIS_MULT is

  signal     W : std_logic_vector(3 downto 0);
  signal     Y : std_logic_vector(3 downto 0);
  signal     Z : std_logic_vector(3 downto 0);
  signal     R : std_logic_vector(3 downto 0);
  signal    R1 : std_logic_vector(3 downto 0);
  signal    R3 : std_logic_vector(3 downto 0);
  signal    Z1 : std_logic_vector(3 downto 0);
  signal    Z3 : std_logic_vector(3 downto 0);
  signal    Z2 : std_logic_vector(3 downto 0);

  component GALOIS_ADDER
    Port (     A : In   std_logic_vector (3 downto 0);
           B : In   std_logic_vector (3 downto 0);
           C : Out  std_logic_vector (3 downto 0) );
  end component;

  component AND1TO4
    Port (    BI : In   std_logic;
          IN1 : In   std_logic_vector (3 downto 0);
         OUTP : Out  std_logic_vector (3 downto 0) );
  end component;

  component ALPHA
    Port (   IN1 : In   std_logic_vector (3 downto 0);
         OUT1 : Out  std_logic_vector (3 downto 0) );
  end component;

begin

  I_1 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>Z3(3 downto 0),
           B(3 downto 0)=>Z2(3 downto 0),
           C(3 downto 0)=>RES(3 downto 0) );
  I_2 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>R3(3 downto 0),
           B(3 downto 0)=>Z1(3 downto 0),
           C(3 downto 0)=>Z2(3 downto 0) );
  I_3 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>R1(3 downto 0),
           B(3 downto 0)=>R(3 downto 0),
           C(3 downto 0)=>R3(3 downto 0) );
  A3 : AND1TO4
    Port Map ( BI=>B(3), IN1(3 downto 0)=>Z(3 downto 0),
           OUTP(3 downto 0)=>Z3(3 downto 0) );
  A2 : AND1TO4
    Port Map ( BI=>B(2), IN1(3 downto 0)=>Y(3 downto 0),
           OUTP(3 downto 0)=>Z1(3 downto 0) );
  A0 : AND1TO4
    Port Map ( BI=>B(0), IN1(3 downto 0)=>GAMMA(3 downto 0),
           OUTP(3 downto 0)=>R(3 downto 0) );
  A1 : AND1TO4
    Port Map ( BI=>B(1), IN1(3 downto 0)=>W(3 downto 0),
           OUTP(3 downto 0)=>R1(3 downto 0) );
  ALP2 : ALPHA
    Port Map ( IN1(3 downto 0)=>Y(3 downto 0),
           OUT1(3 downto 0)=>Z(3 downto 0) );
  ALP1 : ALPHA
    Port Map ( IN1(3 downto 0)=>W(3 downto 0),
           OUT1(3 downto 0)=>Y(3 downto 0) );
  ALP0 : ALPHA
    Port Map ( IN1(3 downto 0)=>GAMMA(3 downto 0),
           OUT1(3 downto 0)=>W(3 downto 0) );

end SCHEMATIC;

configuration CFG_GALOIS_MULT_SCHEMATIC of GALOIS_MULT is

  for SCHEMATIC
    for I_1, I_2, I_3: GALOIS_ADDER
      use configuration WORK.CFG_GALOIS_ADDER_BEHAVIORAL;
    end for;
    for A3, A2, A0, A1: AND1TO4
      use configuration WORK.CFG_AND1TO4_BEHAVIORAL;
    end for;
    for ALP2, ALP1, ALP0: ALPHA
```

```
      use configuration WORK.CFG_ALPHA_BEHAVIORAL;
    end for;
  end for;


end CFG_GALOIS_MULT_SCHEMATIC;

-- VHDL Model Created from SGE Symbol galois_adder.sym -- May  2 10:38:41 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity GALOIS_ADDER is
    Port (      A : In    std_logic_vector (3 downto 0);
                B : In    std_logic_vector (3 downto 0);
                C : Out   std_logic_vector (3 downto 0) );
end GALOIS_ADDER;

architecture BEHAVIORAL of GALOIS_ADDER is


begin

c<= a xor b;
end BEHAVIORAL;

configuration CFG_GALOIS_ADDER_BEHAVIORAL of GALOIS_ADDER is

  for BEHAVIORAL
  end for;

end CFG_GALOIS_ADDER_BEHAVIORAL;

-- VHDL Model Created from SGE Schematic summer.sch -- May  2 12:12:58 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity SUMMER is
    Port (      AX : In    std_logic_vector (31 downto 0);
                BX : Out   std_logic_vector (3 downto 0) );
end SUMMER;

architecture SCHEMATIC of SUMMER is

  signal      Z1 : std_logic_vector(3 downto 0);
  signal      Z2 : std_logic_vector(3 downto 0);
  signal      Z3 : std_logic_vector(3 downto 0);
  signal      Z4 : std_logic_vector(3 downto 0);
  signal      Z5 : std_logic_vector(3 downto 0);
  signal      Z7 : std_logic_vector(3 downto 0);

  component GALOIS_ADDER
    Port (      A : In    std_logic_vector (3 downto 0);
                B : In    std_logic_vector (3 downto 0);
                C : Out   std_logic_vector (3 downto 0) );
  end component;

begin

  I_1 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>Z5(3 downto 0),
               B(3 downto 0)=>Z7(3 downto 0),
               C(3 downto 0)=>BX(3 downto 0) );
  I_2 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>Z3(3 downto 0),
               B(3 downto 0)=>Z4(3 downto 0),
               C(3 downto 0)=>Z7(3 downto 0) );
  I_3 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>Z1(3 downto 0),
               B(3 downto 0)=>Z2(3 downto 0),
               C(3 downto 0)=>Z5(3 downto 0) );
  I_4 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>AX(27 downto 24),
```

```
            B(3 downto 0)=>AX(31 downto 28),
            C(3 downto 0)=>Z4(3 downto 0) );
  I_5 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>AX(19 downto 16),
            B(3 downto 0)=>AX(23 downto 20),
            C(3 downto 0)=>Z3(3 downto 0) );
  I_6 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>AX(11 downto 8),
            B(3 downto 0)=>AX(15 downto 12),
            C(3 downto 0)=>Z2(3 downto 0) );
  I_7 : GALOIS_ADDER
    Port Map ( A(3 downto 0)=>AX(3 downto 0),
            B(3 downto 0)=>AX(7 downto 4),
            C(3 downto 0)=>Z1(3 downto 0) );

end SCHEMATIC;

configuration CFG_SUMMER_SCHEMATIC of SUMMER is

  for SCHEMATIC
    for I_1, I_2, I_3, I_4, I_5, I_6, I_7: GALOIS_ADDER
      use configuration WORK.CFG_GALOIS_ADDER_BEHAVIORAL;
    end for;
  end for;

end CFG_SUMMER_SCHEMATIC;


-- VHDL Model Created from SGE Symbol in_port.sym -- Mar  4 15:47:29 1996

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity IN_PORT is
    Port (   CLK : In    std_logic := '0';
            RES : In    std_logic := '1';
              S : In    std_logic := '0';
            SYM : In    std_logic_vector (3 downto 0);
            AJR : Out   std_logic_vector (59 downto 0);
           OUT1 : Out   std_logic_vector (31 downto 0);
           OUT2 : Out   std_logic_vector (31 downto 0);
            ROW : Out   INTEGER );
end IN_PORT;

architecture BEHAVIORAL of IN_PORT is
type STATE_TYPE IS (S0,S1,S2,S3,S4,S5,S6,S7,
  S8,S9,S10,S11,S12,S13,S14);
signal  CS,NS : STATE_TYPE := S0;

  begin
  ---- Process to hold a combinational logic.
COMBIN : process(CS,CLK,S,RES)
begin
OUT2(31 downto 28) <= "0000";
case CS is
  when S0 =>
if (RES = '1') then
 NS <= S0;
else
NS <= S1;
end if;
  when S1 =>
          NS <= S2;
       when S2 =>
          NS <= S3;
       when S3 =>
          NS <= S4;
       when S4 =>
          NS <= S5;
       when S5 =>
          NS <= S6;
       when S6 =>
          NS <= S7;
       when S7 =>
          NS <= S8;
       when S8 =>
```

```
            NS <= S9;
        when S9 =>
            NS <= S10;
        when S10 =>
            NS <= S11;
        when S11 =>
            NS <= S12;
when S12 =>
            NS <= S13;
        when S13 =>
            NS <= S14;
when S14 =>
            NS <= S0;
end case;
end process;

---------- Process to hold synchronous elements

SYNCH : process(CLK,RES,SYM,S)
variable COUNT : INTEGER range 0 to 15 := 0;
variable rtemp : INTEGER range 0 to 15 := 0;
begin
      out2(31 downto 28) <= "0000";
      if (RES = '1') then
            CS <= S0;
            rtemp := 0;
            ROW <= 0;
      elsif (CLK'EVENT and CLK = '1') then
           if (CS = S0) then
             rtemp := 0;
             COUNT := 0;
           else
             rtemp := rtemp + 1;
           end if;
                CS <= NS;
           case rtemp is
           when 0 =>
             out1(3 downto 0) <= SYM;
             if (S = '1') then
                COUNT := COUNT + 1;
           case COUNT is
           when 1 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000001";
           when others =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000000";
           end case;
           end if;
           when 1 =>
                out1(7 downto 4) <= SYM;
            if (S = '1') then
                COUNT := COUNT + 1;
            case COUNT is
           when 1 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000010";
           when 2 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 4) <= "0000000000000010";
           when others =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000000";
           end case;
           end if;
           when 2 =>
                out1(11 downto 8) <= SYM;
           if (S = '1') then
                COUNT := COUNT + 1;
                case COUNT is
           when 1 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000100";
           when 2 =>
```

```
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000000100";
when 3 =>

        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000000100";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 3 =>
        out1(15 downto 12) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000001000";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000001000";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000001000";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00001000";

when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 4 =>
        out1(19 downto 16) <= SYM;
 if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000011";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000000011";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000000011";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00000011";
when 5 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 16) <= "0011";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 5 =>
        out1(23 downto 20) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
        case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
```

```
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000110";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000000110";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000000110";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00000110";
when 5 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 16) <= "0110";
when 6 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000110";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 6 =>
        out1(27 downto 24) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000001100";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000001100";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000001100";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00001100";
when 5 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 16) <= "1100";
when 6 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000001100";
when 7 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 24) <= "0000000000001100";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 7 =>
        out1(31 downto 28) <= SYM;
 if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000001011";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000001011";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
```

```
        AJR(19 downto 8) <= "000000001011";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00001011";
when 5 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 16) <= "1011";
when 6 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000001011";
when 7 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 24) <= "00000000000001011";
when 8 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 28) <= "000000001011";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 8 =>
        out2(3 downto 0) <= SYM;
 if (S = '1') then
     COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000101";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000000101";
when 3 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 8) <= "000000000101";
when 4 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 12) <= "00000101";
when 5 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 16) <= "0101";
when 6 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000101";
when 7 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 24) <= "0000000000000101";
when 8 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 28) <= "000000000101";
when 9 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 32) <= "00000101";
when others =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 9 =>
        out2(7 downto 4) <= SYM;
if (S = '1') then
     COUNT := COUNT + 1;
case COUNT is
when 1 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 0) <= "00000000000000001010";
when 2 =>
        AJR(59 downto 40) <= "00000000000000000000";
        AJR(39 downto 20) <= "00000000000000000000";
        AJR(19 downto 4) <= "0000000000001010";
```

```
when 3 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 8) <= "000000001010";
when 4 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 12) <= "00001010";
when 5 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 16) <= "1010";
when 6 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000001010";
when 7 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 24) <= "0000000000001010";
when 8 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 28) <= "000000001010";
when 9 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 32) <= "00001010";
when 10 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 36) <= "1010";
when others =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 10 =>
      out2(11 downto 8) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000000111";
when 2 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 4) <= "00000000000000111";
when 3 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 8) <= "000000000111";
when 4 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 12) <= "00000111";
when 5 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 16) <= "0111";
when 6 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000111";
when 7 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 24) <= "0000000000000111";
when 8 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 28) <= "000000000111";
when 9 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 32) <= "00000111";
when 10 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 36) <= "0111";
when 11 =>
      AJR(59 downto 40) <= "00000000000000000111";
when others =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000000000";
end case;
```

```
end if;
when 11 =>
     out2(15 downto 12) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 0) <= "00000000000000001110";
when 2 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 4) <= "0000000000001110";
when 3 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 8) <= "000000001110";
when 4 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 12) <= "00001110";
when 5 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 16) <= "1110";
when 6 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000001110";
when 7 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 24) <= "0000000000001110";
when 8 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 28) <= "000000001110";
when 9 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 32) <= "00001110";
when 10 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 36) <= "1110";
when 11 =>
     AJR(59 downto 40) <= "00000000000000001110";
when 12 =>
     AJR(59 downto 44) <= "0000000000001110";
when others =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 12 =>
     out2(19 downto 16) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 0) <= "00000000000000001111";
when 2 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 4) <= "0000000000001111";
when 3 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 8) <= "000000001111";
when 4 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 12) <= "00001111";
when 5 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000000000";
     AJR(19 downto 16) <= "1111";

when 6 =>
     AJR(59 downto 40) <= "00000000000000000000";
     AJR(39 downto 20) <= "00000000000000001111";
```

```
when 7 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 24) <= "0000000000001111";
when 8 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 28) <= "000000001111";
when 9 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 32) <= "00001111";
when 10 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 36) <= "1111";
when 11 =>
      AJR(59 downto 40) <= "00000000000000001111";
when 12 =>
      AJR(59 downto 44) <= "0000000000001111";
when 13 =>
      AJR(59 downto 48) <= "000000001111";
when others =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when 13 =>
      out2(23 downto 20) <= SYM;
if (S = '1') then
    COUNT := COUNT + 1;
case COUNT is
when 1 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000001101";
when 2 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 4) <= "00000000000001101";
when 3 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 8) <= "000000001101";
when 4 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 12) <= "00001101";
when 5 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 16) <= "1101";
when 6 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000001101";
when 7 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 24) <= "0000000000001101";
when 8 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 28) <= "000000001101";
when 9 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 32) <= "00001101";
when 10 =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 36) <= "1101";
when 11 =>
      AJR(59 downto 40) <= "00000000000000001101";
when 12 =>
      AJR(59 downto 44) <= "0000000000001101";
when 13 =>
      AJR(59 downto 48) <= "000000001101";
when 14 =>
      AJR(59 downto 52) <= "00001101";
when others =>
      AJR(59 downto 40) <= "00000000000000000000";
      AJR(39 downto 20) <= "00000000000000000000";
      AJR(19 downto 0) <= "00000000000000000000";
end case;
end if;
when others =>
      out2(27 downto 24) <= SYM;
```

```
            if (S = '1') then
                COUNT := COUNT + 1;
            case COUNT is
            when 1 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000001001";

           when 2 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 4) <= "00000000000001001";
            when 3 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 8) <= "000000001001";
            when 4 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 12) <= "00001001";
            when 5 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 16) <= "1001";
            when 6 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000001001";
            when 7 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 24) <= "0000000000001001";
            when 8 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 28) <= "000000001001";
            when 9 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 32) <= "00001001";
            when 10 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 36) <= "1001";
            when 11 =>
                AJR(59 downto 40) <= "00000000000000001001";
            when 12 =>
                AJR(59 downto 44) <= "0000000000001001";
            when 13 =>
                AJR(59 downto 48) <= "000000001001";
            when 14 =>
                AJR(59 downto 52) <= "00001001";
            when 0 =>
                AJR(59 downto 40) <= "00000000000000000000";
                AJR(39 downto 20) <= "00000000000000000000";
                AJR(19 downto 0) <= "00000000000000000000";
            when others =>
                AJR(59 downto 56) <= "1001";
            end case;
            end if;
            end case;
            ROW <= COUNT;
        end if;
end process;
end BEHAVIORAL;

configuration CFG_IN_PORT_BEHAVIORAL of IN_PORT is
  for BEHAVIORAL

  end for;

end CFG_IN_PORT_BEHAVIORAL;


-- VHDL Model Created from SGE Schematic init.sch -- May  3 09:42:53 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity INIT is
    Port (    ALP : In    std_logic_vector (31 downto 0);
```

```
          ALP1 : In    std_logic_vector (31 downto 0);
           S1 : In    std_logic_vector (31 downto 0);
           S2 : In    std_logic_vector (31 downto 0);
         DELT : Out   std_logic_vector (3 downto 0);
          SO1 : Out   std_logic_vector (31 downto 0);
          SO2 : Out   std_logic_vector (31 downto 0) );
end INIT;

architecture SCHEMATIC of INIT is

  signal     O5 : std_logic_vector(31 downto 0);
  signal     SO1_DUMMY : std_logic_vector (31 downto 0);
  signal     SO2_DUMMY : std_logic_vector (31 downto 0);

  component SUMMER
    Port (     AX : In    std_logic_vector (31 downto 0);
               BX : Out   std_logic_vector (3 downto 0) );
  end component;

  component GALOIS_MULT_32
    Port (     IN1 : In    std_logic_vector (31 downto 0);
              IN2 : In    std_logic_vector (31 downto 0);
             OUT1 : Out   std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_ADD_32
    Port (     IN1 : In    std_logic_vector (31 downto 0);
              IN2 : In    std_logic_vector (31 downto 0);
             OUT1 : Out   std_logic_vector (31 downto 0) );
  end component;

begin

  SO1 <= SO1_DUMMY;
  SO2 <= SO2_DUMMY;

  SU1 : SUMMER
    Port Map ( AX(31 downto 0)=>O5(31 downto 0),
               BX(3 downto 0)=>DELT(3 downto 0) );
  M2 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>S2(31 downto 0),
               IN2(31 downto 0)=>ALP1(31 downto 0),
               OUT1(31 downto 0)=>SO2_DUMMY(31 downto 0) );
  M1 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>S1(31 downto 0),
               IN2(31 downto 0)=>ALP(31 downto 0),
               OUT1(31 downto 0)=>SO1_DUMMY(31 downto 0) );
  A1 : GALOIS_ADD_32
    Port Map ( IN1(31 downto 0)=>SO1_DUMMY(31 downto 0),
               IN2(31 downto 0)=>SO2_DUMMY(31 downto 0),
               OUT1(31 downto 0)=>O5(31 downto 0) );

end SCHEMATIC;

architecture BEHAVIORAL of INIT is

  begin

end BEHAVIORAL;

configuration CFG_INIT_SCHEMATIC of INIT is

  for SCHEMATIC
    for SU1: SUMMER
      use configuration WORK.CFG_SUMMER_SCHEMATIC;
    end for;
    for M2, M1: GALOIS_MULT_32
      use configuration WORK.CFG_GALOIS_MULT_32_SCHEMATIC;
    end for;
    for A1: GALOIS_ADD_32
      use configuration WORK.CFG_GALOIS_ADD_32_SCHEMATIC;
    end for;
  end for;

end CFG_INIT_SCHEMATIC;

-- VHDL Model Created from SGE Symbol inverse.sym -- Oct 22 15:45:17 1995

library IEEE;
  use IEEE.std_logic_1164.all;
```

```
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity INVERSE is
    Port (    ALP : In   std_logic_vector (3 downto 0);
           ALPI : Out  std_logic_vector (3 downto 0) );
end INVERSE;

architecture BEHAVIORAL of INVERSE is

  begin
process(alp)
     begin
if (alp = "0000") then
    alpi <= "0000";
elsif (alp = "0001") then
    alpi <= "0001";
elsif (alp = "0010") then
    alpi <= "1001";
elsif (alp = "0100") then
    alpi <= "1101";
elsif (alp = "1000") then
    alpi <= "1111";
elsif (alp = "0011") then
    alpi <= "1110";
elsif (alp = "0110") then
    alpi <= "0111";
elsif (alp = "1100") then
    alpi <= "1010";
elsif (alp = "1011") then
    alpi <= "0101";
elsif (alp = "0101") then
    alpi <= "1011";
elsif (alp = "1010") then
    alpi <= "1100";
elsif (alp = "0111") then
    alpi <= "0110";
elsif (alp = "1110") then
    alpi <= "0011";
elsif (alp = "1111") then
    alpi <= "1000";
elsif (alp = "1101") then
    alpi <= "0100";
elsif (alp = "1001") then
    alpi <= "0010";
end if;
end process;


end BEHAVIORAL;

configuration CFG_INVERSE_BEHAVIORAL of INVERSE is
  for BEHAVIORAL

  end for;

end CFG_INVERSE_BEHAVIORAL;

-- VHDL Model Created from SGE Symbol mux2_31.sym -- Feb  5 19:10:31 1996

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity MUX2_31 is
    Port (      A : In   std_logic_vector (31 downto 0);
             B : In   std_logic_vector (31 downto 0);
            EN : In   std_logic;
          OUTM : Out  std_logic_vector (31 downto 0) );
end MUX2_31;

architecture BEHAVIORAL of MUX2_31 is

  begin
  process(en,A,B)
  begin
    if en = '0' then
```

```
      OUTM <= A;
    elsif en = '1' then
      OUTM <= B;
   else NULL;
   end if;
  end process;

 end BEHAVIORAL;

configuration CFG_MUX2_31_BEHAVIORAL of MUX2_31 is
  for BEHAVIORAL

  end for;

end CFG_MUX2_31_BEHAVIORAL;

-- VHDL Model Created from SGE Schematic ajr.sch -- Nov  7 12:47:43 1995

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity AJR is
    Port (    CLK : In   std_logic;
              S : In   std_logic;
           ALPJR : Out   std_logic_vector (3 downto 0) );
end AJR;

architecture SCHEMATIC of AJR is


  component IN-PORT
    Port (    CLK : In   std_logic;
           RESET : In   std_logic;
              S : In   std_logic;
            SUM : In   std_logic_vector (3 downto 0);
            AJR : Out   std_logic_vector (59 downto 0);
           OUT1 : Out   std_logic_vector (31 downto 0);
           OUT2 : Out   std_logic_vector (31 downto 0);
            ROW : Out   std_logic );
  end component;

begin

  PORT1 : IN-PORT
    Port Map ( CLK=>CLK, RESET=>R, S=>SIG,
            SUM(3 downto 0)=>SYMB(3 downto 0),
            AJR(59 downto 0)=>DEL(59 downto 0),
            OUT1(31 downto 0)=>VO1(31 downto 0),
            OUT2(31 downto 0)=>VO2(31 downto 0), ROW=>ROW );

end SCHEMATIC;

configuration CFG_AJR_SCHEMATIC of AJR is

  for SCHEMATIC
    for PORT1: IN-PORT
      use configuration WORK.CFG_IN-PORT_BEHAVIORAL;
    end for;
  end for;

end CFG_AJR_SCHEMATIC;

-- VHDL Model Created from SGE Symbol comp_add.sym -- Jul  5 18:47:45 1995

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity COMP_ADD is
    Port (     A1 : In   std_logic_vector (3 downto 0);
              A2 : In   std_logic_vector (3 downto 0);
             CON : In   std_logic_vector (3 downto 0);
            OUT1 : Out   std_logic_vector (3 downto 0) );
end COMP_ADD;
```

```
architecture BEHAVIORAL of COMP_ADD is

  begin
OUT1 <= A1 xor A2 when CON = "0000"
else A1;

end BEHAVIORAL;

configuration CFG_COMP_ADD_BEHAVIORAL of COMP_ADD is
  for BEHAVIORAL

  end for;

end CFG_COMP_ADD_BEHAVIORAL;

-- VHDL Model Created from SGE Schematic bloc1.sch -- May  3 09:40:13 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity BLOC1 is
    Port (   ALPI : In    std_logic_vector (31 downto 0);
           ALPI1 : In    std_logic_vector (31 downto 0);
             B : In    std_logic_vector (31 downto 0);
            B1 : In    std_logic_vector (31 downto 0);
           DEL : In    std_logic_vector (31 downto 0);
             E : In    std_logic_vector (1 downto 0);
           LAM : In    std_logic_vector (31 downto 0);
          LAM1 : In    std_logic_vector (31 downto 0);
             Z : In    std_logic_vector (31 downto 0);
            LO : Out   std_logic_vector (31 downto 0);
           LO1 : Out   std_logic_vector (31 downto 0) );
end BLOC1;

architecture SCHEMATIC of BLOC1 is

  signal      O5 : std_logic_vector(31 downto 0);
  signal      O8 : std_logic_vector(31 downto 0);
  signal      O9 : std_logic_vector(31 downto 0);
  signal     O10 : std_logic_vector(31 downto 0);
  signal      OB : std_logic_vector(31 downto 0);
  signal     OB1 : std_logic_vector(31 downto 0);

  component MUX2_31
    Port (      A : In    std_logic_vector (31 downto 0);
                B : In    std_logic_vector (31 downto 0);
               EN : In    std_logic;
             OUTM : Out   std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_ADD_32
    Port (    IN1 : In    std_logic_vector (31 downto 0);
              IN2 : In    std_logic_vector (31 downto 0);
             OUT1 : Out   std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_MULT_32
    Port (    IN1 : In    std_logic_vector (31 downto 0);
              IN2 : In    std_logic_vector (31 downto 0);
             OUT1 : Out   std_logic_vector (31 downto 0) );
  end component;

begin

  MUX5 : MUX2_31
    Port Map ( A(31 downto 0)=>Z(31 downto 0),
               B(31 downto 0)=>DEL(31 downto 0), EN=>E(1),
               OUTM(31 downto 0)=>OB1(31 downto 0) );
  MUX4 : MUX2_31
    Port Map ( A(31 downto 0)=>Z(31 downto 0),
               B(31 downto 0)=>DEL(31 downto 0), EN=>E(0),
               OUTM(31 downto 0)=>OB(31 downto 0) );
  I_1 : GALOIS_ADD_32
    Port Map ( IN1(31 downto 0)=>LAM1(31 downto 0),
               IN2(31 downto 0)=>O10(31 downto 0),
               OUT1(31 downto 0)=>LO1(31 downto 0) );
  ADD1 : GALOIS_ADD_32
```

```
    Port Map ( IN1(31 downto 0)=>LAM(31 downto 0),
               IN2(31 downto 0)=>O9(31 downto 0),
               OUT1(31 downto 0)=>LO(31 downto 0) );
  MUL2 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>OB1(31 downto 0),
               IN2(31 downto 0)=>ALPI1(31 downto 0),
               OUT1(31 downto 0)=>O8(31 downto 0) );
  MUL1 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>OB(31 downto 0),
               IN2(31 downto 0)=>ALPI(31 downto 0),
               OUT1(31 downto 0)=>O5(31 downto 0) );
  MUL4 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>B1(31 downto 0),
               IN2(31 downto 0)=>O8(31 downto 0),
               OUT1(31 downto 0)=>O10(31 downto 0) );
  MUL3 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>O5(31 downto 0),
               IN2(31 downto 0)=>B(31 downto 0),
               OUT1(31 downto 0)=>O9(31 downto 0) );

end SCHEMATIC;

configuration CFG_BLOC1_SCHEMATIC of BLOC1 is

  for SCHEMATIC
    for MUX5, MUX4: MUX2_31
      use configuration WORK.CFG_MUX2_31_BEHAVIORAL;
    end for;
    for I_1, ADD1: GALOIS_ADD_32
      use configuration WORK.CFG_GALOIS_ADD_32_SCHEMATIC;
    end for;
    for MUL2, MUL1, MUL4, MUL3: GALOIS_MULT_32
      use configuration WORK.CFG_GALOIS_MULT_32_SCHEMATIC;
    end for;
  end for;

end CFG_BLOC1_SCHEMATIC;

-- VHDL Model Created from SGE Schematic bloc2.sch -- May  3 10:15:13 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity BLOC2 is
    Port (    ALP : In    std_logic_vector (31 downto 0);
            ALPI : In    std_logic_vector (31 downto 0);
             DEL : In    std_logic_vector (31 downto 0);
               E : In    std_logic_vector (5 downto 0);
          IN1BLOC : In    std_logic_vector (31 downto 0);
          INBLOC1 : In    std_logic_vector (31 downto 0);
               S : In    std_logic_vector (31 downto 0);
              S1 : In    std_logic_vector (31 downto 0);
               V : In    std_logic_vector (31 downto 0);
              V1 : In    std_logic_vector (31 downto 0);
               Z : In    std_logic_vector (31 downto 0);
               C : Out   std_logic_vector (31 downto 0);
              C1 : Out   std_logic_vector (31 downto 0);
           O1ADD : Out   std_logic_vector (31 downto 0);
          O2ADD1 : Out   std_logic_vector (31 downto 0);
              SO : Out   std_logic_vector (31 downto 0);
             SO1 : Out   std_logic_vector (31 downto 0) );
end BLOC2;

architecture SCHEMATIC of BLOC2 is

  signal      W : std_logic_vector(31 downto 0);
  signal     W1 : std_logic_vector(31 downto 0);
  signal     S3 : std_logic_vector(31 downto 0);
  signal     OM : std_logic_vector(31 downto 0);
  signal     S4 : std_logic_vector(31 downto 0);
  signal    OM3 : std_logic_vector(31 downto 0);
  signal     T1 : std_logic_vector(31 downto 0);
  signal     T2 : std_logic_vector(31 downto 0);
  signal     T3 : std_logic_vector(31 downto 0);
  signal     T4 : std_logic_vector(31 downto 0);
  signal  O1ADD_DUMMY : std_logic_vector (31 downto 0);
  signal  O2ADD1_DUMMY : std_logic_vector (31 downto 0);
```

```
  signal      SO_DUMMY : std_logic_vector (31 downto 0);
  signal     SO1_DUMMY : std_logic_vector (31 downto 0);

  component MUX2_31
    Port (      A : In   std_logic_vector (31 downto 0);
                B : In   std_logic_vector (31 downto 0);
               EN : In   std_logic;
             OUTM : Out  std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_ADD_32
    Port (    IN1 : In   std_logic_vector (31 downto 0);
              IN2 : In   std_logic_vector (31 downto 0);
             OUT1 : Out  std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_MULT_32
    Port (    IN1 : In   std_logic_vector (31 downto 0);
              IN2 : In   std_logic_vector (31 downto 0);
             OUT1 : Out  std_logic_vector (31 downto 0) );
  end component;

  component CO_ADDER_32
    Port (      A : In   std_logic_vector (31 downto 0);
                B : In   std_logic_vector (31 downto 0);
                C : In   std_logic_vector (31 downto 0);
                D : Out  std_logic_vector (31 downto 0) );
  end component;

begin

  O1ADD <= O1ADD_DUMMY;
  O2ADD1 <= O2ADD1_DUMMY;
  SO <= SO_DUMMY;
  SO1 <= SO1_DUMMY;

  MUX5 : MUX2_31
    Port Map ( A(31 downto 0)=>SO1_DUMMY(31 downto 0),
               B(31 downto 0)=>T3(31 downto 0), EN=>E(5),
               OUTM(31 downto 0)=>T4(31 downto 0) );
  MUX4 : MUX2_31
    Port Map ( A(31 downto 0)=>SO_DUMMY(31 downto 0),
               B(31 downto 0)=>T1(31 downto 0), EN=>E(4),
               OUTM(31 downto 0)=>T2(31 downto 0) );
  MUX3 : MUX2_31
    Port Map ( A(31 downto 0)=>S4(31 downto 0),
               B(31 downto 0)=>S1(31 downto 0), EN=>E(3),
               OUTM(31 downto 0)=>OM3(31 downto 0) );
  MUX2 : MUX2_31
    Port Map ( A(31 downto 0)=>S3(31 downto 0),
               B(31 downto 0)=>S(31 downto 0), EN=>E(2),
               OUTM(31 downto 0)=>OM(31 downto 0) );
  MUX1 : MUX2_31
    Port Map ( A(31 downto 0)=>Z(31 downto 0),
               B(31 downto 0)=>DEL(31 downto 0), EN=>E(1),
               OUTM(31 downto 0)=>W1(31 downto 0) );
  MUX0 : MUX2_31
    Port Map ( A(31 downto 0)=>Z(31 downto 0),
               B(31 downto 0)=>DEL(31 downto 0), EN=>E(0),
               OUTM(31 downto 0)=>W(31 downto 0) );
  I_3 : GALOIS_ADD_32
    Port Map ( IN1(31 downto 0)=>W1(31 downto 0),
               IN2(31 downto 0)=>OM3(31 downto 0),
               OUT1(31 downto 0)=>SO1_DUMMY(31 downto 0) );
  I_4 : GALOIS_ADD_32
    Port Map ( IN1(31 downto 0)=>W(31 downto 0),
               IN2(31 downto 0)=>OM(31 downto 0),
               OUT1(31 downto 0)=>SO_DUMMY(31 downto 0) );
  I_8 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>ALP1(31 downto 0),
               IN2(31 downto 0)=>S1(31 downto 0),
               OUT1(31 downto 0)=>S4(31 downto 0) );
  I_5 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>SO1_DUMMY(31 downto 0),
               IN2(31 downto 0)=>ALP1(31 downto 0),
               OUT1(31 downto 0)=>T3(31 downto 0) );
  I_6 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>SO_DUMMY(31 downto 0),
               IN2(31 downto 0)=>ALP(31 downto 0),
               OUT1(31 downto 0)=>T1(31 downto 0) );
```

```
  MUL5 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>S(31 downto 0),
               IN2(31 downto 0)=>ALP(31 downto 0),
               OUT1(31 downto 0)=>S3(31 downto 0) );
  MUL7 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>T2(31 downto 0),
               IN2(31 downto 0)=>INBLOC1(31 downto 0),
               OUT1(31 downto 0)=>O1ADD_DUMMY(31 downto 0) );
  MUL8 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>T4(31 downto 0),
               IN2(31 downto 0)=>IN1BLOC(31 downto 0),
               OUT1(31 downto 0)=>O2ADD1_DUMMY(31 downto 0) );
  I_9 : CO_ADDER_32
    Port Map ( A(31 downto 0)=>V(31 downto 0),
               B(31 downto 0)=>SO_DUMMY(31 downto 0),
               C(31 downto 0)=>O1ADD_DUMMY(31 downto 0),
               D(31 downto 0)=>C(31 downto 0) );
  ADD7 : CO_ADDER_32
    Port Map ( A(31 downto 0)=>V1(31 downto 0),
               B(31 downto 0)=>SO1_DUMMY(31 downto 0),
               C(31 downto 0)=>O2ADD1_DUMMY(31 downto 0),
               D(31 downto 0)=>C1(31 downto 0) );

end SCHEMATIC;

configuration CFG_BLOC2_SCHEMATIC of BLOC2 is

  for SCHEMATIC
    for MUX5, MUX4, MUX3, MUX2, MUX1, MUX0: MUX2_31
      use configuration WORK.CFG_MUX2_31_BEHAVIORAL;
    end for;
    for I_3, I_4: GALOIS_ADD_32
      use configuration WORK.CFG_GALOIS_ADD_32_SCHEMATIC;
    end for;
    for I_8, I_5, I_6, MUL5, MUL7, MUL8: GALOIS_MULT_32
      use configuration WORK.CFG_GALOIS_MULT_32_SCHEMATIC;
    end for;
    for I_9, ADD7: CO_ADDER_32
      use configuration WORK.CFG_CO_ADDER_32_SCHEMATIC;
    end for;
  end for;

end CFG_BLOC2_SCHEMATIC;

-- VHDL Model Created from SGE Schematic bloc3.sch -- May  3 09:58:02 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity BLOC3 is
    Port (   ALPI : In    std_logic_vector (31 downto 0);
           ALPI1 : In    std_logic_vector (31 downto 0);
             B : In    std_logic_vector (31 downto 0);
            B1 : In    std_logic_vector (31 downto 0);
          DELI : In    std_logic_vector (31 downto 0);
             E : In    std_logic_vector (5 downto 0);
         I1BLOC1 : In    std_logic_vector (31 downto 0);
         I2BLOC1 : In    std_logic_vector (31 downto 0);
           INM1 : In    std_logic_vector (31 downto 0);
           INM2 : In    std_logic_vector (31 downto 0);
            LAM : In    std_logic_vector (31 downto 0);
           LAM1 : In    std_logic_vector (31 downto 0);
             BO : Out   std_logic_vector (31 downto 0);
            BO1 : Out   std_logic_vector (31 downto 0);
           DOUT : Out   std_logic_vector (3 downto 0) );
end BLOC3;

architecture SCHEMATIC of BLOC3 is

  signal     O1 : std_logic_vector(31 downto 0);
  signal     O2 : std_logic_vector(31 downto 0);
  signal     O3 : std_logic_vector(31 downto 0);
  signal     O4 : std_logic_vector(31 downto 0);
  signal     O7 : std_logic_vector(31 downto 0);
  signal    O11 : std_logic_vector(31 downto 0);
  signal    O12 : std_logic_vector(31 downto 0);
```

```
  component GALOIS_ADD_32
    Port (    IN1 : In   std_logic_vector (31 downto 0);
              IN2 : In   std_logic_vector (31 downto 0);
             OUT1 : Out  std_logic_vector (31 downto 0) );
  end component;

  component SUMMER
    Port (     AX : In   std_logic_vector (31 downto 0);
               BX : Out  std_logic_vector (3 downto 0) );
  end component;

  component MUX2_31
    Port (      A : In   std_logic_vector (31 downto 0);
                B : In   std_logic_vector (31 downto 0);
               EN : In   std_logic;
             OUTM : Out  std_logic_vector (31 downto 0) );
  end component;

  component GALOIS_MULT_32
    Port (    IN1 : In   std_logic_vector (31 downto 0);
              IN2 : In   std_logic_vector (31 downto 0);
             OUT1 : Out  std_logic_vector (31 downto 0) );
  end component;

begin

  I_1 : GALOIS_ADD_32
    Port Map ( IN1(31 downto 0)=>INM1(31 downto 0),
               IN2(31 downto 0)=>INM2(31 downto 0),
               OUT1(31 downto 0)=>O7(31 downto 0) );
  SUM1 : SUMMER
    Port Map ( AX(31 downto 0)=>O7(31 downto 0),
               BX(3 downto 0)=>DOUT(3 downto 0) );
  MUX10 : MUX2_31
    Port Map ( A(31 downto 0)=>O12(31 downto 0),
               B(31 downto 0)=>I1BLOC1(31 downto 0), EN=>E(4),
               OUTM(31 downto 0)=>BO(31 downto 0) );
  MUX11 : MUX2_31
    Port Map ( A(31 downto 0)=>O11(31 downto 0),
               B(31 downto 0)=>I2BLOC1(31 downto 0), EN=>E(5),
               OUTM(31 downto 0)=>BO1(31 downto 0) );
  MUX9 : MUX2_31
    Port Map ( A(31 downto 0)=>B1(31 downto 0),
               B(31 downto 0)=>LAM1(31 downto 0), EN=>E(3),
               OUTM(31 downto 0)=>O4(31 downto 0) );
  MUX7 : MUX2_31
    Port Map ( A(31 downto 0)=>ALPI1(31 downto 0),
               B(31 downto 0)=>DELI(31 downto 0), EN=>E(1),
               OUTM(31 downto 0)=>O3(31 downto 0) );
  MUX8 : MUX2_31
    Port Map ( A(31 downto 0)=>B(31 downto 0),
               B(31 downto 0)=>LAM(31 downto 0), EN=>E(2),
               OUTM(31 downto 0)=>O2(31 downto 0) );
  MUX6 : MUX2_31
    Port Map ( A(31 downto 0)=>ALPI(31 downto 0),
               B(31 downto 0)=>DELI(31 downto 0), EN=>E(0),
               OUTM(31 downto 0)=>O1(31 downto 0) );
  MUL10 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>O3(31 downto 0),
               IN2(31 downto 0)=>O4(31 downto 0),
               OUT1(31 downto 0)=>O11(31 downto 0) );
  MUL9 : GALOIS_MULT_32
    Port Map ( IN1(31 downto 0)=>O1(31 downto 0),
               IN2(31 downto 0)=>O2(31 downto 0),
               OUT1(31 downto 0)=>O12(31 downto 0) );

end SCHEMATIC;

configuration CFG_BLOC3_SCHEMATIC of BLOC3 is

  for SCHEMATIC
    for I_1: GALOIS_ADD_32
      use configuration WORK.CFG_GALOIS_ADD_32_SCHEMATIC;
    end for;
    for SUM1: SUMMER
      use configuration WORK.CFG_SUMMER_SCHEMATIC;
    end for;
    for MUX10, MUX11, MUX9, MUX7, MUX8, MUX6: MUX2_31
      use configuration WORK.CFG_MUX2_31_BEHAVIORAL;
    end for;
```

```
    for MUL10, MUL9: GALOIS_MULT_32
      use configuration WORK.CFG_GALOIS_MULT_32_SCHEMATIC;
    end for;
  end for;

end CFG_BLOC3_SCHEMATIC;

-- VHDL Model Created from SGE Schematic blocked.sch -- May  3 09:39:09 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity BLOCKED is
    Port (   ALP : In    std_logic_vector (31 downto 0);
           ALPI : In    std_logic_vector (31 downto 0);
           ALPI : In    std_logic_vector (31 downto 0);
          ALPI1 : In    std_logic_vector (31 downto 0);
             B : In    std_logic_vector (31 downto 0);
            B1 : In    std_logic_vector (31 downto 0);
            CE : In    std_logic_vector (13 downto 0);
           DEL : In    std_logic_vector (31 downto 0);
          DELI : In    std_logic_vector (31 downto 0);
             L : In    std_logic_vector (31 downto 0);
            L1 : In    std_logic_vector (31 downto 0);
             S : In    std_logic_vector (31 downto 0);
            S1 : In    std_logic_vector (31 downto 0);
             V : In    std_logic_vector (31 downto 0);
            V1 : In    std_logic_vector (31 downto 0);
             Z : In    std_logic_vector (31 downto 0);
            BO : Out   std_logic_vector (31 downto 0);
           BO1 : Out   std_logic_vector (31 downto 0);
            CO : Out   std_logic_vector (31 downto 0);
           CO1 : Out   std_logic_vector (31 downto 0);
         DELTA : Out   std_logic_vector (3 downto 0);
            LO : Out   std_logic_vector (31 downto 0);
           LO1 : Out   std_logic_vector (31 downto 0);
            SO : Out   std_logic_vector (31 downto 0);
           SO1 : Out   std_logic_vector (31 downto 0) );
end BLOCKED;

architecture SCHEMATIC of BLOCKED is

  signal     C1 : std_logic_vector(31 downto 0);
  signal     C2 : std_logic_vector(31 downto 0);
  signal     LO_DUMMY : std_logic_vector (31 downto 0);
  signal    LO1_DUMMY : std_logic_vector (31 downto 0);

  component BLOC3
    Port (   ALPI : In    std_logic_vector (31 downto 0);
           ALPI1 : In    std_logic_vector (31 downto 0);
             B : In    std_logic_vector (31 downto 0);
            B1 : In    std_logic_vector (31 downto 0);
           DELI : In    std_logic_vector (31 downto 0);
             E : In    std_logic_vector (5 downto 0);
         I1BLOC1 : In    std_logic_vector (31 downto 0);
         I2BLOC1 : In    std_logic_vector (31 downto 0);
           INM1 : In    std_logic_vector (31 downto 0);
           INM2 : In    std_logic_vector (31 downto 0);
            LAM : In    std_logic_vector (31 downto 0);
           LAM1 : In    std_logic_vector (31 downto 0);
            BO : Out   std_logic_vector (31 downto 0);
           BO1 : Out   std_logic_vector (31 downto 0);
          DOUT : Out   std_logic_vector (3 downto 0) );
  end component;

  component BLOC2
    Port (   ALP : In    std_logic_vector (31 downto 0);
           ALPI : In    std_logic_vector (31 downto 0);
           DEL : In    std_logic_vector (31 downto 0);
             E : In    std_logic_vector (5 downto 0);
         IN1BLOC : In    std_logic_vector (31 downto 0);
         INBLOC1 : In    std_logic_vector (31 downto 0);
             S : In    std_logic_vector (31 downto 0);
            S1 : In    std_logic_vector (31 downto 0);
             V : In    std_logic_vector (31 downto 0);
            V1 : In    std_logic_vector (31 downto 0);
             Z : In    std_logic_vector (31 downto 0);
```

```
        C : Out  std_logic_vector (31 downto 0);
        C1 : Out  std_logic_vector (31 downto 0);
      O1ADD : Out  std_logic_vector (31 downto 0);
     O2ADD1 : Out  std_logic_vector (31 downto 0);
        SO : Out  std_logic_vector (31 downto 0);
        SO1 : Out  std_logic_vector (31 downto 0) );
  end component;

  component BLOC1
    Port (  ALPI : In   std_logic_vector (31 downto 0);
          ALPI1 : In   std_logic_vector (31 downto 0);
            B : In   std_logic_vector (31 downto 0);
           B1 : In   std_logic_vector (31 downto 0);
          DEL : In   std_logic_vector (31 downto 0);
            E : In   std_logic_vector (1 downto 0);
          LAM : In   std_logic_vector (31 downto 0);
         LAM1 : In   std_logic_vector (31 downto 0);
            Z : In   std_logic_vector (31 downto 0);
           LO : Out  std_logic_vector (31 downto 0);
          LO1 : Out  std_logic_vector (31 downto 0) );
  end component;

begin

  LO <= LO_DUMMY;
  LO1 <= LO1_DUMMY;

  BLO3 : BLOC3
    Port Map ( ALPI(31 downto 0)=>ALPI(31 downto 0),
          ALPI1(31 downto 0)=>ALPI1(31 downto 0),
          B(31 downto 0)=>B(31 downto 0),
          B1(31 downto 0)=>B1(31 downto 0),
          DELI(31 downto 0)=>DELI(31 downto 0),
          E(5 downto 0)=>CE(13 downto 8),
          I1BLOC1(31 downto 0)=>LO_DUMMY(31 downto 0),
          I2BLOC1(31 downto 0)=>LO1_DUMMY(31 downto 0),
          INM1(31 downto 0)=>C2(31 downto 0),
          INM2(31 downto 0)=>C1(31 downto 0),
          LAM(31 downto 0)=>L(31 downto 0),
          LAM1(31 downto 0)=>L1(31 downto 0),
          BO(31 downto 0)=>BO(31 downto 0),
          BO1(31 downto 0)=>BO1(31 downto 0),
          DOUT(3 downto 0)=>DELTA(3 downto 0) );
  BLO2 : BLOC2
    Port Map ( ALP(31 downto 0)=>ALP(31 downto 0),
          ALP1(31 downto 0)=>ALP1(31 downto 0),
          DEL(31 downto 0)=>DEL(31 downto 0),
          E(5 downto 0)=>CE(5 downto 0),
          IN1BLOC(31 downto 0)=>LO1_DUMMY(31 downto 0),
          INBLOC1(31 downto 0)=>LO_DUMMY(31 downto 0),
          S(31 downto 0)=>S(31 downto 0),
          S1(31 downto 0)=>S1(31 downto 0),
          V(31 downto 0)=>V(31 downto 0),
          V1(31 downto 0)=>V1(31 downto 0),
          Z(31 downto 0)=>Z(31 downto 0),
          C(31 downto 0)=>CO(31 downto 0),
          C1(31 downto 0)=>CO1(31 downto 0),
          O1ADD(31 downto 0)=>C2(31 downto 0),
          O2ADD1(31 downto 0)=>C1(31 downto 0),
          SO(31 downto 0)=>SO(31 downto 0),
          SO1(31 downto 0)=>SO1(31 downto 0) );
  BLO1 : BLOC1
    Port Map ( ALPI(31 downto 0)=>ALPI(31 downto 0),
          ALPI1(31 downto 0)=>ALPI1(31 downto 0),
          B(31 downto 0)=>B(31 downto 0),
          B1(31 downto 0)=>B1(31 downto 0),
          DEL(31 downto 0)=>DEL(31 downto 0),
          E(1 downto 0)=>CE(7 downto 6),
          LAM(31 downto 0)=>L(31 downto 0),
          LAM1(31 downto 0)=>L1(31 downto 0),
          Z(31 downto 0)=>Z(31 downto 0),
          LO(31 downto 0)=>LO_DUMMY(31 downto 0),
          LO1(31 downto 0)=>LO1_DUMMY(31 downto 0) );

end SCHEMATIC;

configuration CFG_BLOCKED_SCHEMATIC of BLOCKED is

  for SCHEMATIC
    for BLO3: BLOC3
```

```
    use configuration WORK.CFG_BLOC3_SCHEMATIC;
  end for;
  for BLO2: BLOC2
    use configuration WORK.CFG_BLOC2_SCHEMATIC;
  end for;
  for BLO1: BLOC1
    use configuration WORK.CFG_BLOC1_SCHEMATIC;
  end for;
end for;

end CFG_BLOCKED_SCHEMATIC;

-- VHDL Model Created from SGE Symbol control_unit.sym -- Mar  4 21:11:41 1996

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;


entity CONTROL_UNIT is
    Port (    AJR : In   std_logic_vector (59 downto 0);
              BI1 : In   std_logic_vector (31 downto 0);
              BI2 : In   std_logic_vector (31 downto 0);
              CLK : In   std_logic;
              DEL : In   std_logic_vector (3 downto 0);
             DELI : In   std_logic_vector (3 downto 0);
            DELTI : In   std_logic_vector (3 downto 0);
              DET : In   std_logic_vector (3 downto 0);
                K : In   INTEGER;
              LI1 : In   std_logic_vector (31 downto 0);
              LI2 : In   std_logic_vector (31 downto 0);
              RES : In   std_logic;
              ROW : In   INTEGER;
              SI1 : In   std_logic_vector (31 downto 0);
              SI2 : In   std_logic_vector (31 downto 0);
              VI1 : In   std_logic_vector (31 downto 0);
              VI2 : In   std_logic_vector (31 downto 0);
              BO1 : Out  std_logic_vector (31 downto 0);
              BO2 : Out  std_logic_vector (31 downto 0);
            DELIO : Out  std_logic_vector (31 downto 0);
             DELO : Out  std_logic_vector (31 downto 0);
              LO1 : Out  std_logic_vector (31 downto 0);
              LO2 : Out  std_logic_vector (31 downto 0);
          OUT_CON : Out  std_logic_vector (13 downto 0);
              SO1 : Out  std_logic_vector (31 downto 0);
              SO2 : Out  std_logic_vector (31 downto 0);
              VO1 : Out  std_logic_vector (31 downto 0);
              VO2 : Out  std_logic_vector (31 downto 0) );

end CONTROL_UNIT;

architecture BEHAVIORAL of CONTROL_UNIT is
type STATE_TYPE IS (ALPHA,BETA,DELTA,DELT,SIGMA,STOP);
signal  CS,NS :STATE_TYPE := STOP;
signal r : INTEGER range 0 to 16 := 0 ;
signal tempd : std_logic_vector (3 downto 0);
  begin
--------Process to hold a combinational logic.
COMBIN : process(CS,r,k,tempd,row)
variable l : INTEGER range 0 to 15 := 0;
variable rt : INTEGER range 0 to 16 := 0;
variable rowt : INTEGER range 0 to 15 := 0;
procedure Process_State is
begin
    if (r = 15 - k) then
            NS <= SIGMA;
            out_con(13 downto 0) <="------11000000";
        elsif
--------------- r <= 15 - k
        (tempd(3 downto 0) = "0000") then
            NS <= DELTA;
            out_con(13 downto 0) <= "00000000000000";
        elsif
------------- delta not equal to "0000"
        ( r <= 2*l - rowt) then
            NS <= ALPHA;
```

```
          else
---------- r > 2*l - row
               NS <= BETA;
          end if;
      end Process_State;
begin
rt := r;
       case CS is
        when ALPHA =>
          out_con(13 downto 0) <= "00000011000000";
          Process_State;
        when BETA =>
            l := rt - l - rowt;
            out_con(13 downto 0) <= "00111111000000";
            Process_State;
       when DELTA =>
       Process_State;
  when SIGMA =>
     if (r = 15) then
      rowt := row; l:= 0;
     end if;
     if ( r = 15 - k + 1) then
        out_con(13 downto 0) <= "------00111111";
     else
               out_con(13 downto 0) <= "------00110011";
          end if;
     if ((r > 15 - k) and (r < 15)) then
  NS <= SIGMA;
     else
        if (rowt = 0) then
              Process_State;
           else
             NS <= DELT;
           end if;
          end if;
  when DELT =>
          out_con(13 downto 0) <= "11----11000000";
     if (r < rowt ) then
              NS <= DELT;
          else
            Process_State;
          end if;
  when STOP =>
l := 0;
           if (rt <= rowt ) then
              NS <= DELT;
   out_con(13 downto 0) <= "11----11000000";
          else
   Process_State;
          end if;
        end case;
end process;
----------

----------
SYNCH : process(CLK,RES,r,row,del,deli,det,delti,s11,s12,li1,li2,bi1,bi2,si1,si2,ajr)
variable rtemp : INTEGER range 0 to 16 := 0;
variable ajrt : std_logic_vector (59 downto 0);
variable t_delt,t_deli,mysign : std_logic_vector(3 downto 0);
begin
if (RES = '1') then
r <= 1;
CS <= STOP;
      elsif (CLK'EVENT and CLK = '1') then
        if (CS = STOP)   then
           r <= 1;
           rtemp := r;
         else
           rtemp := rtemp + 1;
           r <= r + 1;
         end if;
         if ((rtemp = 1) or (rtemp > 15))then
           lo1(19 downto 0) <= "00010001000100010001";
           lo1(31 downto 20) <= "000100010001";
           lo2(19 downto 0) <= "00010001000100010001";
           lo2(31 downto 20) <= "000000010001";
           bo1(19 downto 0) <= "00010001000100010001";
           bo1(31 downto 20) <= "000100010001";
           bo2(19 downto 0) <= "00010001000100010001";
           bo2(31 downto 20) <= "000000010001";
```

```
            so1 <= s11;
            so2 <= s12;
vo1 <= vi1;
vo2 <= vi2;
        else
            lo1 <= li1;
            lo2 <= li2;
            bo1 <= bi1;
            bo2 <= bi2;
            so1 <= si1;
            so2 <= si2;
        end if;
case rtemp is
            when 1 =>
                mysign := ajrt(3 downto 0);
            when 2 =>
                mysign := ajrt(7 downto 4);
            when 3 =>
                mysign := ajrt(11 downto 8);
            when 4 =>
                mysign := ajrt(15 downto 12);
            when 5 =>
                mysign := ajrt(19 downto 16);
            when 6 =>
                mysign := ajrt(23 downto 20);
            when 7 =>
                mysign := ajrt(27 downto 24);
            when 8 =>
                mysign := ajrt(31 downto 28);
            when 9 =>
                mysign := ajrt(35 downto 32);
            when 10 =>
                mysign := ajrt(39 downto 36);
            when 11 =>
                mysign := ajrt(43 downto 40);
            when 12 =>
                mysign := ajrt(47 downto 44);
            when 13 =>
                mysign := ajrt(51 downto 48);
            when 14 =>
                mysign := ajrt(55 downto 52);
            when 15 =>
                mysign := ajrt(59 downto 56);
                ajrt := ajr;
            when others =>
                r <= 1;
                rtemp := 1;
                mysign := ajrt(3 downto 0);
    end case;
    CS <= NS;
    if ((row = 0) and (rtemp = 1))  then
       t_delt := det ;
       t_deli := delti;
    elsif (rtemp <= row) then
  t_delt := mysign;
       t_deli := deli;
    else
       t_delt := del;
       t_deli := deli;
    end if;
    delo(3 downto 0) <= t_delt(3 downto 0);
    delo(7 downto 4) <= t_delt(3 downto 0);
    delo(11 downto 8) <= t_delt(3 downto 0);
    delo(15 downto 12) <= t_delt(3 downto 0);
    delo(19 downto 16) <= t_delt(3 downto 0);
    delo(23 downto 20) <= t_delt(3 downto 0);
    delo(27 downto 24) <= t_delt(3 downto 0);
    delo(31 downto 28) <= t_delt(3 downto 0);
    delio(3 downto 0) <= t_deli(3 downto 0);
    delio(7 downto 4) <= t_deli(3 downto 0);
    delio(11 downto 8) <= t_deli(3 downto 0);
    delio(15 downto 12) <= t_deli(3 downto 0);
    delio(19 downto 16) <= t_deli(3 downto 0);
    delio(23 downto 20) <= t_deli(3 downto 0);
    delio(27 downto 24) <= t_deli(3 downto 0);
    delio(31 downto 28) <= t_deli(3 downto 0);
    tempd <= t_delt;
end if;
end process;
```

```
end BEHAVIORAL;

configuration CFG_CONTROL_UNIT_BEHAVIORAL of CONTROL_UNIT is

  for BEHAVIORAL

  end for;

end CFG_CONTROL_UNIT_BEHAVIORAL;

-- VHDL Model Created from SGE Symbol ramalp.sym -- Jan  5 12:13:35 1996

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity RAMALP is
    Port (      A1 : Out   std_logic_vector (31 downto 0);
               A2 : Out   std_logic_vector (31 downto 0);
              AI1 : Out   std_logic_vector (31 downto 0);
              AI2 : Out   std_logic_vector (31 downto 0);
             ZERO : Out   std_logic_vector (31 downto 0) );
end RAMALP;

architecture BEHAVIORAL of RAMALP is

  begin
     a1(31 downto 0) <= "10111100011000111000010000100001";
     a2(31 downto 0) <= "00001001110111111111001111010010l";
     ai1(31 downto 0) <= "01011010011111101111110110010001";
     ai2(31 downto 0) <= "00000010010010000011O11011001011";
zero(31 downto 0) <= "00000000000000000000000O00000000";

end BEHAVIORAL;

configuration CFG_RAMALP_BEHAVIORAL of RAMALP is
  for BEHAVIORAL

  end for;

end CFG_RAMALP_BEHAVIORAL;

-- VHDL Model Created from SGE Schematic decoder.sch -- May  2 11:28:45 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;

entity DECODER is
    Port (      K : In    INTEGER;
              RES : In    std_logic;
              SIG : In    std_logic;
             SYMB : In    std_logic_vector (3 downto 0);
               CO : Out   std_logic_vector (31 downto 0);
              CO1 : Out   std_logic_vector (31 downto 0) );
end DECODER;

architecture SCHEMATIC of DECODER is

  signal     O1 : std_logic_vector(31 downto 0);
  signal     O2 : std_logic_vector(31 downto 0);
  signal      A : std_logic_vector(59 downto 0);
  signal     A1 : std_logic_vector(31 downto 0);
  signal     A2 : std_logic_vector(31 downto 0);
  signal    AI1 : std_logic_vector(31 downto 0);
  signal    AI2 : std_logic_vector(31 downto 0);
  signal     Z : std_logic_vector(31 downto 0);
  signal     S2 : std_logic_vector(31 downto 0);
  signal    DI1 : std_logic_vector(3 downto 0);
  signal     D2 : std_logic_vector(3 downto 0);
  signal    DI2 : std_logic_vector(3 downto 0);
  signal     B3 : std_logic_vector(31 downto 0);
  signal     B4 : std_logic_vector(31 downto 0);
  signal     L3 : std_logic_vector(31 downto 0);
  signal     L4 : std_logic_vector(31 downto 0);
```

```
signal      S5 : std_logic_vector(31 downto 0);
signal      S6 : std_logic_vector(31 downto 0);
signal      D1 : std_logic_vector(3 downto 0);
signal      B1 : std_logic_vector(31 downto 0);
signal      B2 : std_logic_vector(31 downto 0);
signal      L1 : std_logic_vector(31 downto 0);
signal      L2 : std_logic_vector(31 downto 0);
signal      S3 : std_logic_vector(31 downto 0);
signal      S4 : std_logic_vector(31 downto 0);
signal      DE : std_logic_vector(31 downto 0);
signal     DEI : std_logic_vector(31 downto 0);
signal      V1 : std_logic_vector(31 downto 0);
signal      V2 : std_logic_vector(31 downto 0);
signal      OC : std_logic_vector(13 downto 0);
signal      S1 : std_logic_vector(31 downto 0);
signal     N_1 : std_logic;
signal       R : INTEGER;

component CLOK
  Port (   OUTC : Out   std_logic );
end component;

component IN_PORT
  Port (    CLK : In    std_logic;
            RES : In    std_logic;
              S : In    std_logic;
            SYM : In    std_logic_vector (3 downto 0);
            AJR : Out   std_logic_vector (59 downto 0);
           OUT1 : Out   std_logic_vector (31 downto 0);
           OUT2 : Out   std_logic_vector (31 downto 0);
            ROW : Out   INTEGER );
end component;

component RAMALP
  Port (     A1 : Out   std_logic_vector (31 downto 0);
             A2 : Out   std_logic_vector (31 downto 0);
            AI1 : Out   std_logic_vector (31 downto 0);
            AI2 : Out   std_logic_vector (31 downto 0);
           ZERO : Out   std_logic_vector (31 downto 0) );
end component;

component INVERSE
  Port (    ALP : In    std_logic_vector (3 downto 0);
           ALPI : Out   std_logic_vector (3 downto 0) );
end component;

component INIT
  Port (    ALP : In    std_logic_vector (31 downto 0);
           ALP1 : In    std_logic_vector (31 downto 0);
             S1 : In    std_logic_vector (31 downto 0);
             S2 : In    std_logic_vector (31 downto 0);
           DELT : Out   std_logic_vector (3 downto 0);
            SO1 : Out   std_logic_vector (31 downto 0);
            SO2 : Out   std_logic_vector (31 downto 0) );
end component;

component CONTROL_UNIT
  Port (    AJR : In    std_logic_vector (59 downto 0);
            BI1 : In    std_logic_vector (31 downto 0);
            BI2 : In    std_logic_vector (31 downto 0);
            CLK : In    std_logic;
            DEL : In    std_logic_vector (3 downto 0);
           DELI : In    std_logic_vector (3 downto 0);
          DELTI : In    std_logic_vector (3 downto 0);
            DET : In    std_logic_vector (3 downto 0);
              K : In    INTEGER;
            LI1 : In    std_logic_vector (31 downto 0);
            LI2 : In    std_logic_vector (31 downto 0);
            RES : In    std_logic;
            ROW : In    INTEGER;
            S11 : In    std_logic_vector (31 downto 0);
            S12 : In    std_logic_vector (31 downto 0);
            SI1 : In    std_logic_vector (31 downto 0);
            SI2 : In    std_logic_vector (31 downto 0);
            VI1 : In    std_logic_vector (31 downto 0);
            VI2 : In    std_logic_vector (31 downto 0);
            BO1 : Out   std_logic_vector (31 downto 0);
            BO2 : Out   std_logic_vector (31 downto 0);
           DELO : Out   std_logic_vector (31 downto 0);
           DELO : Out   std_logic_vector (31 downto 0);
```

```
        LO1 : Out  std_logic_vector (31 downto 0);
        LO2 : Out  std_logic_vector (31 downto 0);
    OUT_CON : Out  std_logic_vector (13 downto 0);
        SO1 : Out  std_logic_vector (31 downto 0);
        SO2 : Out  std_logic_vector (31 downto 0);
        VO1 : Out  std_logic_vector (31 downto 0);
        VO2 : Out  std_logic_vector (31 downto 0) );
end component;

component BLOCKED
  Port (   ALP : In   std_logic_vector (31 downto 0);
          ALP1 : In   std_logic_vector (31 downto 0);
          ALPI : In   std_logic_vector (31 downto 0);
         ALPI1 : In   std_logic_vector (31 downto 0);
             B : In   std_logic_vector (31 downto 0);
            B1 : In   std_logic_vector (31 downto 0);
            CE : In   std_logic_vector (13 downto 0);
           DEL : In   std_logic_vector (31 downto 0);
          DELI : In   std_logic_vector (31 downto 0);
             L : In   std_logic_vector (31 downto 0);
            L1 : In   std_logic_vector (31 downto 0);
             S : In   std_logic_vector (31 downto 0);
            S1 : In   std_logic_vector (31 downto 0);
             V : In   std_logic_vector (31 downto 0);
            V1 : In   std_logic_vector (31 downto 0);
             Z : In   std_logic_vector (31 downto 0);
            BO : Out  std_logic_vector (31 downto 0);
           BO1 : Out  std_logic_vector (31 downto 0);
            CO : Out  std_logic_vector (31 downto 0);
           CO1 : Out  std_logic_vector (31 downto 0);
         DELTA : Out  std_logic_vector (3 downto 0);
            LO : Out  std_logic_vector (31 downto 0);
           LO1 : Out  std_logic_vector (31 downto 0);
            SO : Out  std_logic_vector (31 downto 0);
           SO1 : Out  std_logic_vector (31 downto 0) );
end component;

begin

  I_12 : CLOK
    Port Map ( OUTC=>N_1 );
  I_9 : IN_PORT
    Port Map ( CLK=>N_1, RES=>RES, S=>SIG,
           SYM(3 downto 0)=>SYMB(3 downto 0),
           AJR(59 downto 0)=>A(59 downto 0),
           OUT1(31 downto 0)=>O1(31 downto 0),
           OUT2(31 downto 0)=>O2(31 downto 0), ROW=>R );
  I_1 : RAMALP
    Port Map ( A1(31 downto 0)=>A1(31 downto 0),
           A2(31 downto 0)=>A2(31 downto 0),
           AI1(31 downto 0)=>AI1(31 downto 0),
           AI2(31 downto 0)=>AI2(31 downto 0),
           ZERO(31 downto 0)=>Z(31 downto 0) );
  I_11 : INVERSE
    Port Map ( ALP(3 downto 0)=>D1(3 downto 0),
           ALPI(3 downto 0)=>DI1(3 downto 0) );
  I_3 : INVERSE
    Port Map ( ALP(3 downto 0)=>D2(3 downto 0),
           ALPI(3 downto 0)=>DI2(3 downto 0) );
  I_5 : INIT
    Port Map ( ALP(31 downto 0)=>A1(31 downto 0),
           ALP1(31 downto 0)=>A2(31 downto 0),
           S1(31 downto 0)=>O1(31 downto 0),
           S2(31 downto 0)=>O2(31 downto 0),
           DELT(3 downto 0)=>D1(3 downto 0),
           SO1(31 downto 0)=>S2(31 downto 0),
           SO2(31 downto 0)=>S1(31 downto 0) );
  I_10 : CONTROL_UNIT
    Port Map ( AJR(59 downto 0)=>A(59 downto 0),
           BI1(31 downto 0)=>B3(31 downto 0),
           BI2(31 downto 0)=>B4(31 downto 0), CLK=>N_1,
           DEL(3 downto 0)=>D2(3 downto 0),
           DELI(3 downto 0)=>DI2(3 downto 0),
           DELTI(3 downto 0)=>DI1(3 downto 0),
           DET(3 downto 0)=>D1(3 downto 0), K=>K,
           LI1(31 downto 0)=>L3(31 downto 0),
           LI2(31 downto 0)=>L4(31 downto 0), RES=>RES, ROW=>R,
           S11(31 downto 0)=>S2(31 downto 0),
           S12(31 downto 0)=>S1(31 downto 0),
           SI1(31 downto 0)=>S5(31 downto 0),
```

```
            SI2(31 downto 0)=>S6(31 downto 0),
            VI1(31 downto 0)=>O1(31 downto 0),
            VI2(31 downto 0)=>O2(31 downto 0),
            BO1(31 downto 0)=>B1(31 downto 0),
            BO2(31 downto 0)=>B2(31 downto 0),
            DELIO(31 downto 0)=>DE(31 downto 0),
            DELO(31 downto 0)=>DEI(31 downto 0),
            LO1(31 downto 0)=>L1(31 downto 0),
            LO2(31 downto 0)=>L2(31 downto 0),
            OUT_CON(13 downto 0)=>OC(13 downto 0),
            SO1(31 downto 0)=>S3(31 downto 0),
            SO2(31 downto 0)=>S4(31 downto 0),
            VO1(31 downto 0)=>V1(31 downto 0),
            VO2(31 downto 0)=>V2(31 downto 0) );
  I_8 : BLOCKED
    Port Map ( ALP(31 downto 0)=>A1(31 downto 0),
            ALP1(31 downto 0)=>A2(31 downto 0),
            ALPI(31 downto 0)=>AI1(31 downto 0),
            ALPI1(31 downto 0)=>AI2(31 downto 0),
            B(31 downto 0)=>B1(31 downto 0),
            B1(31 downto 0)=>B2(31 downto 0),
            CE(13 downto 0)=>OC(13 downto 0),
            DEL(31 downto 0)=>DEI(31 downto 0),
            DELI(31 downto 0)=>DE(31 downto 0),
            L(31 downto 0)=>L1(31 downto 0),
            L1(31 downto 0)=>L2(31 downto 0),
            S(31 downto 0)=>S3(31 downto 0),
            S1(31 downto 0)=>S4(31 downto 0),
            V(31 downto 0)=>V1(31 downto 0),
            V1(31 downto 0)=>V2(31 downto 0),
            Z(31 downto 0)=>Z(31 downto 0),
            BO(31 downto 0)=>B3(31 downto 0),
            BO1(31 downto 0)=>B4(31 downto 0),
            CO(31 downto 0)=>CO(31 downto 0),
            CO1(31 downto 0)=>CO1(31 downto 0),
            DELTA(3 downto 0)=>D2(3 downto 0),
            LO(31 downto 0)=>L3(31 downto 0),
            LO1(31 downto 0)=>L4(31 downto 0),
            SO(31 downto 0)=>S5(31 downto 0),
            SO1(31 downto 0)=>S6(31 downto 0) );

end SCHEMATIC;

configuration CFG_DECODER_SCHEMATIC of DECODER is

  for SCHEMATIC
    for I_12: CLOK
      use configuration WORK.CFG_CLOK_BEHAVIORAL;
    end for;
    for I_9: IN_PORT
      use configuration WORK.CFG_IN_PORT_BEHAVIORAL;
    end for;
    for I_1: RAMALP
      use configuration WORK.CFG_RAMALP_BEHAVIORAL;
    end for;
    for I_11, I_3: INVERSE
      use configuration WORK.CFG_INVERSE_BEHAVIORAL;
    end for;
    for I_5: INIT
      use configuration WORK.CFG_INIT_SCHEMATIC;
    end for;
    for I_10: CONTROL_UNIT
      use configuration WORK.CFG_CONTROL_UNIT_BEHAVIORAL;
    end for;
    for I_8: BLOCKED
      use configuration WORK.CFG_BLOCKED_SCHEMATIC;
    end for;
  end for;

end CFG_DECODER_SCHEMATIC;
```

APPENDIX D

Synthesis Procedure and Optimization Results

D.1 Synthesis Procedure

```
/* Lectures des sources du groupe */
read -f vhdl alpha.vhd

current_design = ALPHA
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0.2 -from all_inputs() -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output ALPHA.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl and1to4.vhd

current_design = AND1TO4
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"

current_design = AND1TO4
set_max_delay 0.2 -from all_inputs() -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output AND1TO4.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl bloc1.vhd
read galois_mult_32.db
read GALOIS_ADD_32.db
read MUX2_31.db

current_design = BLOC1
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"


set_max_delay 0 -from all_inputs() -to all_outputs()

report_reference

compile -map_effort high -ungroup_all

write -f db -hierarchy -output bloc1.db
```

```
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl bloc2.vhd
read galois_mult_32.db
read GALOIS_ADD_32.db
read MUX2_31.db
read co_adder_32.db
current_design = BLOC2
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0 -from all_inputs() -to all_outputs()

report_reference

compile -map_effort high -ungroup_all

write -f db -hierarchy -output bloc2.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl bloc3.vhd
read galois_mult_32.db
read GALOIS_ADD_32.db
read MUX2_31.db
read SUMMER.db
current_design = BLOC3
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"


set_max_delay 0 -from all_inputs() -to all_outputs()

report_reference

compile -map_effort high -ungroup_all

write -f db -hierarchy -output bloc3.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl blocked.vhd
read bloc1.db
read bloc2.db
read bloc3.db

current_design = BLOCKED
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"


set_max_delay 0 -from all_inputs() -to all_outputs()
```

```
report_reference

compile -map_effort high -ungroup_all

write -f db -hierarchy -output blocked.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl control.vhd
remove_license VHDL-Compiler
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
current_design = CONTROL_UNIT
link
check_design


include control.iod
set_max_delay 0 -from all_inputs() -to all_outputs()
uniquify
compile -map_effort high -ungroup_all
report_area
write -f db -hierarchy -output control.db
check_design

foreach( inport, all_inputs() ) {
  report_timing -from inport
  }

foreach( outport, all_outputs() ) {
  report_timing -to outport
  }

report_constraints -all_violators -nosplit

exit


read -f vhdl decoder.vhd


current_design = DECODER
link
check_design
foreach (design,find(design,"*") ) {
  current_design = design
  reset_design
  }
current_design = DECODER
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
include decoder.iod
report_area

report_reference
write -f db -hierarchy -output decoder.db
check_design
report_design
report_constraints -all_violators -nosplit
foreach( inport, all_inputs() ) {
  report_timing -from inport
  }

foreach( outport, all_outputs() ) {
  report_timing -to outport
  }


exit
```

```
read -f vhdl galois_adder.vhd

current_design = GALOIS_ADDER
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0.2 -from all_inputs() -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output GALOIS_ADDER.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl galois_mult.vhd
read ALPHA.db
read AND1TO4.db
read GALOIS_ADDER.db
uniquify
current_design = GALOIS_MULT
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0 -from all_inputs() -to all_outputs()


compile -map_effort high -ungroup_all

write -f db -hierarchy -output GALOIS_MULT.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl in_port.vhd
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"

current_design = IN_PORT
link
check_design

include inport.iod
set_max_area 0
uniquify
compile -map_effort high -ungroup_all
report_area
write -f db -hierarchy -output IN_PORT.db
check_design

foreach( inport, all_inputs() ) {
  report_timing -from inport
  }

foreach( outport, all_outputs() ) {
  report_timing -to outport
  }

report_constraints -all_violators -nosplit

exit


read -f vhdl init.vhd
read galois_mult_32.db
read SUMMER.db
```

```
read GALOIS_ADD_32.db

current_design = INIT
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0 -from all_inputs() -to all_outputs()

report_reference

compile -map_effort high -ungroup_all

write -f db -hierarchy -output init.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl inverse.vhd
current_design = INVERSE
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
current_design = INVERSE
set_max_delay 0.2 -from all_inputs() -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output inverse.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit

read -f vhdl mux2_31.vhd
current_design = MUX2_31
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
current_design = MUX2_31
set_max_delay 0.44 -from all_inputs() -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output MUX2_31.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


read -f vhdl ramalp.vhd
current_design = RAMALP
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
current_design = RAMALP
set_max_delay 0.2  -to all_outputs()

compile -map_effort high -ungroup_all

write -f db -hierarchy -output RAMALP.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
```

```
report_timing

exit


read -f vhdl summer.vhd
read GALOIS_ADDER.db
uniquify
current_design = SUMMER
link
check_design
set_wire_load "05x05" -mode enclosed
set_operating_conditions "BCCOM"
set_max_delay 0 -from all_inputs() -to all_outputs()


compile -map_effort high -ungroup_all

write -f db -hierarchy -output SUMMER.db
check_design
report_design
report_area
report_constraints -all_violators -nosplit
report_timing

exit


This program is proprietary and confidential information of Synopsys, Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

Initializing...
read -f vhdl decoder.vhd
Loading db file '/cad1/synopsys/libraries/syn/standard.sldb'
Loading db file '/cad1/synopsys/libraries/syn/gtech.db'
Loading db file '/cad1/synopsys/libraries/syn/class.db'
Loading vhdl file '/grad/faculty/baher/shadia/shad/decoder.vhd'
Reading in the Synopsys vhdl primitives.
/grad/faculty/baher/shadia/shad/decoder.vhd:
Current design is now '/grad/faculty/baher/shadia/shad/DECODER.db:DECODER'
{"DECODER"}
read blocked.db
Loading db file '/grad/faculty/baher/shadia/shad/blocked.db'
Current design is now '/grad/faculty/baher/shadia/shad/blocked.db:BLOCKED'
{"BLOCKED"}
read control.db
Loading db file '/grad/faculty/baher/shadia/shad/control.db'
Current design is now '/grad/faculty/baher/shadia/shad/control.db:CONTROL_UNIT'
{"CONTROL_UNIT"}
read init.db
Loading db file '/grad/faculty/baher/shadia/shad/init.db'
Current design is now '/grad/faculty/baher/shadia/shad/init.db:INIT'
{"INIT"}
read RAMALP.db
Loading db file '/grad/faculty/baher/shadia/shad/RAMALP.db'
Current design is now '/grad/faculty/baher/shadia/shad/RAMALP.db:RAMALP'
{"RAMALP"}
read IN_PORT.db
Loading db file '/grad/faculty/baher/shadia/shad/IN_PORT.db'
Current design is now '/grad/faculty/baher/shadia/shad/IN_PORT.db:IN_PORT'
{"IN_PORT"}
read inverse.db
Loading db file '/grad/faculty/baher/shadia/shad/inverse.db'
Current design is now '/grad/faculty/baher/shadia/shad/inverse.db:INVERSE'
{"INVERSE"}
current_design = DECODER
Current design is 'DECODER'.
"/grad/faculty/baher/shadia/shad/DECODER.db:DECODER"
link
Linking design:
  DECODER
Information: Updating design information... (UID-85)
```

D.2 Optimization Results

```
****************************************
Report : area
Design : DECODER
Version: v3.4b
Date   : Tue Jan 14 12:33:36 1997
****************************************

Library(s) Used:

    class (File: /cad1/synopsys/libraries/syn/class.db)

Number of ports:          103
Number of nets:          1025
Number of cells:            7
Number of references:       6

Combinational area:   13993.000000
Noncombinational area:  4140.000000
Net Interconnect area:    undefined  (Wire load has zero net area)

Total cell area:      18133.000000
Total area:           undefined
1
report_reference

****************************************
Report : reference
Design : DECODER
Version: v3.4b
Date   : Tue Jan 14 12:33:41 1997
****************************************

Attributes:
    b - black box (unknown)
   bo - allows boundary optimization
    d - dont_touch
   mo - map_only
    h - hierarchical
    n - noncombinational
    r - removable
    s - synthetic operator
    u - contains unmapped logic

Reference        Library     Unit Area  Count   Total Area  Attributes
--------------------------------------------------------------------------
BLOCKED                   9671.000000      1  9671.000000  h
CONTROL_UNIT              5095.000000      1  5095.000000  h, n
INIT                   1256.000000      1  1256.000000  h
INVERSE                     58.000000      2   116.000000  h
IN_PORT                   1995.000000      1  1995.000000  h, n
RAMALP                       0.000000      1     0.000000  h
--------------------------------------------------------------------------
Total 6 references                                18133.000000
1
write -f db -hierarchy -output decoder1.db
Writing to file /grad/faculty/baher/shadia/shad/decoder1.db
1
check_design
report_design

****************************************
Report : design
Design : DECODER
Version: v3.4b
Date   : Tue Jan 14 12:38:50 1997
****************************************

Library(s) Used:

    class (File: /cad1/synopsys/libraries/syn/class.db)

Flip-Flop Types:

    No flip-flop types specified.
```

Latch Types:

No latch types specified.

Operating Conditions:

No operating conditions specified.

Wire Loading Model:

Selected manually by the user.

Name : 05x05
Location : class
Resistance : 0
Capacitance : 1
Area : 0
Slope : 0.186

Fanout	Length	Points	Average Cap	Std Deviation
1	0.39			

Wire Loading Model Mode: enclosed.

report_constraints -all_violators -nosplit

****
Report : constraint
-all_violators
Design : DECODER
Version: v3.4b
Date : Tue Jan 14 12:38:51 1997
****

max_capacitance

Net	Required Capacitance	Actual Capacitance	Slack
RES	0.15	2.58	-2.43 (VIOLATED)
K[0]	0.15	1.39	-1.24 (VIOLATED)
K[1]	0.15	1.39	-1.24 (VIOLATED)
K[2]	0.15	1.39	-1.24 (VIOLATED)
K[3]	0.15	1.39	-1.24 (VIOLATED)
K[4]	0.15	1.39	-1.24 (VIOLATED)
K[5]	0.15	1.39	-1.24 (VIOLATED)
K[6]	0.15	1.39	-1.24 (VIOLATED)
K[7]	0.15	1.39	-1.24 (VIOLATED)
K[8]	0.15	1.39	-1.24 (VIOLATED)
K[9]	0.15	1.39	-1.24 (VIOLATED)
K[10]	0.15	1.39	-1.24 (VIOLATED)
K[11]	0.15	1.39	-1.24 (VIOLATED)
K[12]	0.15	1.39	-1.24 (VIOLATED)
K[13]	0.15	1.39	-1.24 (VIOLATED)
K[14]	0.15	1.39	-1.24 (VIOLATED)
K[15]	0.15	1.39	-1.24 (VIOLATED)
K[16]	0.15	1.39	-1.24 (VIOLATED)
K[17]	0.15	1.39	-1.24 (VIOLATED)
K[18]	0.15	1.39	-1.24 (VIOLATED)
K[19]	0.15	1.39	-1.24 (VIOLATED)
K[20]	0.15	1.39	-1.24 (VIOLATED)
K[21]	0.15	1.39	-1.24 (VIOLATED)
K[22]	0.15	1.39	-1.24 (VIOLATED)
K[23]	0.15	1.39	-1.24 (VIOLATED)
K[24]	0.15	1.39	-1.24 (VIOLATED)
K[25]	0.15	1.39	-1.24 (VIOLATED)
K[26]	0.15	1.39	-1.24 (VIOLATED)
K[27]	0.15	1.39	-1.24 (VIOLATED)
K[28]	0.15	1.39	-1.24 (VIOLATED)
K[29]	0.15	1.39	-1.24 (VIOLATED)
K[30]	0.15	1.39	-1.24 (VIOLATED)
K[31]	0.15	1.39	-1.24 (VIOLATED)
SIG	0.15	1.39	-1.24 (VIOLATED)
SYMB[0]	0.15	1.39	-1.24 (VIOLATED)
SYMB[1]	0.15	1.39	-1.24 (VIOLATED)
SYMB[2]	0.15	1.39	-1.24 (VIOLATED)
SYMB[3]	0.15	1.39	-1.24 (VIOLATED)

```
1
report_timing -true
****************************************
Report : timing
        -path full
        -delay max
        -true
        -max_paths 1
Design : DECODER
Version: v3.4b
Date   : Tue Jan 14 12:39:20 1997
****************************************

Operating Conditions:
Wire Loading Model Mode: enclosed

Design          Wire Loading Model      Library
------------------------------------------------
DECODER             05x05           class
IN_PORT             05x05           class
RAMALP              05x05           class
INVERSE             05x05           class
INIT                05x05           class
CONTROL_UNIT        05x05           class
BLOCKED             05x05           class


  A True path:

  Startpoint: I_10/rtemp_reg[1]
              (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: I_10/VO2_reg[28]
              (rising edge-triggered flip-flop clocked by CLK)
  Path Group: CLK
  Path Type: max

  Point                                   Incr       Path
  ----------------------------------------------------------
  clock CLK (rise edge)                   0.00       0.00
  clock network delay (propagated)        0.00       0.00
  I_10/rtemp_reg[1]/CP (FD1)              0.00       0.00 r
  I_10/rtemp_reg[1]/Q (FD1)               1.57       1.57 f
  I_10/U752/Z (ND2I)                      0.34       1.91 r
  I_10/U753/Z (OR2)                       1.14       3.05 r
  I_10/U939/Z (EO)                        1.15       4.20 f
  I_10/U940/Z (AO2)                       2.42       6.62 r
  I_10/U758/Z (OR2P)                      0.71       7.33 r
  I_10/U773/Z (OR2P)                      0.71       8.03 r
  I_10/U1716/Z (IV)                       0.23       8.26 f
  I_10/U1117/Z (ND2I)                     0.25       8.52 r
  I_10/U1119/Z (ND2I)                     15.01      23.53 f
  I_10/U928/Z (AN2P)                      5.70       29.23 f
  I_10/U868/Z (MUX21L)                    0.85       30.08 r
  I_10/VO2_reg[28]/D (FD1)                0.00       30.08 r
  data arrival time                                  30.08

  clock CLK (rise edge)                   34.00      34.00
  clock network delay (propagated)        0.00       34.00
  clock uncertainty                       -0.30      33.70
  I_10/VO2_reg[28]/CP (FD1)               0.00       33.70 r
  library setup time                      -0.80      32.90
  data required time                                 32.90
  ----------------------------------------------------------
  data required time                                 32.90
  data arrival time                                  -30.08
  ----------------------------------------------------------
  slack (MET)                                        2.82

exit
1
dc_shell>
Thank you...
```