Optimum Partitioning of Globally Asynchronous Locally Synchronous Processor Arrays

Adhir Upadhyay

A Thesis

In

The Department of

Electrical and Computer Engineering

Presented in Partial fulfillment of the Requirements
For the Degree of Master of Applied Science
(Electrical and Computer Engineering)
Concordia University
Montreal, Quebec, Canada

January 2004

Your file Votre référence
ISBN: 0-612-91130-6
Our file Notre référence
ISBN: 0-612-91130-6

ABSTRACT

Optimum Partitioning of Globally Asynchronous Locally Synchronous Processor Arrays

Adhir Upadhyay

Globally Asynchronous Locally Synchronous design style has evolved as a solution to increasing problems of distributing clocks at high frequency and with lower power consumption in DSM technologies. In GALS designs, partitioning a system into more locally synchronous sub-blocks reduces the size of each sub-block and allows higher clock frequency. Also, smaller sub-blocks reduce the capacitance in the clock networks because they need less H-tree levels. However, this implies a large number of sub-blocks, which increases the asynchronous power overhead. The thesis considers a 16x16 array of identical processors to evaluate GALS tradeoffs with different partitioning scenarios. Three different configurations of the array have been studied. This is, to our knowledge, the first work to propose closed form models for the optimum number of sub-blocks that accomplish minimum power for GALS design with passive clock distribution networks. Experimental results verify the effectiveness of the models. The potential increase in the clock frequency with partitioning and its effect on total power consumption has also been investigated for one of the array configurations.

For large VLSI designs, inserting repeaters in the clock network is an alternative to boost the clock frequency that often is limited by the interconnect bandwidth. The thesis also investigates the GALS tradeoffs for an array with active clock networks. An algorithm has been proposed to evaluate the optimum partitioning in this case.

The Delay-Insensitive (DI) asynchronous protocols offer a promising solution to the timing closure problem with long global interconnects in synchronous designs. A novel asynchronous wrapper using 1-of-4 DI protocol has been introduced. Simulation results show that the scheme achieves 66% higher throughput than previous designs.

ACKNOWLEDGEMENTS

This thesis would never have been possible without the support and inspiration from many of my colleagues and teachers. Special thanks are due to a few individuals:

I would like to express my sincere gratitude to Dr. Mohamed Nekili for his excellent guidance and support during my graduate career.

I gratefully acknowledge the collaboration of Syed Rafay Hasan and his encouraging words when things were not moving as expected.

A special thanks to my brothers, Maulik and Nishad, for their invaluable support. I also express my deepest gratitude to my parents, Gautam Upadhyay and Pragna Upadhyay, who have always supported me in all decisions.

Dedicated to my parents

Gautam Upadhyay

And

Pragna Upadhyay

For their endless devotion

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ε_{OX}	Dielectric constant for ILD material
λ_m	probability of exit per unit time
A_1	Constant for clock distribution parameters
Ack	Acknowledge
AR	Aspect ratio
B	Set of communicating sub-block pairs
C_0	Input capacitance of a minimum sized inverter
CAD	Computer-Aided-Design
C_{Driver}	Input capacitance of clock driver
C_{FF}	Capacitance of flip- flops
C_{int}	Distributed capacitance of interconnects per unit length
C_{inv}	Capacitive load due to one inverter
C_L	Total wiring and register input capacitance within the processor
C_{Light}	Speed of light in free space
ΔC_L	Clock driver load mismatch
C_{Leaf}	Interconnect capacitance per unit length of clock network assuming the network has uniform width equal to the width of the leaf of H-tree
CMOS	Complementary Metal Oxide Semiconductor
C_{reg}	Capacitance registers driven by asynchronous signals
C_{Rep}	Input capacitance of a repeater
C_w	Wire capacitance per unit length within asynchronous wrappers
D	Die size
DI	Delay Insensitive
d_{Proc}	Size of each processor

D_s	Die size of sub-block
DSM	Deep Sub-Micron
$f_{b,i}$	Frequency at which sub-blocks communicate
f_c	sampling clock frequency
f_d	expected number of data transitions per unit time
f_{Eff}	Effective clock frequency of sub-blocks
FF_{Proc}	Number of clocked flip-flops within each processor
FIFO	First In First Out
GALS	Globally Asynchronous Locally Synchronous
Gbps	Giga Bits Per Second
h	Factor by which the W/L ratio of repeater-transistors is increased
H_{int}	Height of interconnect
ΔH_{int}	Variations in height of interconnect
ILD	Inter Level Dielectric
IP	Intellectual Property
K	Constant that changes with the size of array
K_i	Number of repeaters in the i^{th} H-tree level.
l	Length of wire within processor
$l_{b,i}$	Wire length of the signal that communicates the asynchronous signal
L_{eff}	Effective channel length
ΔL_{eff}	Variations of effective channel length
L_{int}	Length of interconnect
l_w	Wire length for asynchronous communication signals
M	Number of repeated segments
Mbps	Mega Bits Per Second
ME	Mutual Exclusion Element

$MTBF$	Mean Time Between synchronization Failures
n	Number of H-tree levels
N_{Async}	Number of asynchronous wrappers required with each partition
N_{FF}	Number of clocked flip-flops in the design
N_{invb}	Number of inverters used by the ring oscillator in sub-block number b
n_{reg}	Number of registers driven by the asynchronous control signals
$N_{rep/Path}$	Number of repeaters in each clock path
$N_{rep/S}$	Number of repeaters in each sub-block
NRZ	Non-Return-to-Zero
n_{sig}	Number of signals in the communication protocol
P	Number of processors in the design
P_{Async}	Power consumed in asynchronous wrappers
P_{Async_1}	Constant that represents the power consumed in one asynchronous wrapper
PCB	Printed Circuit Board
PCC	Pausible Clocking Control
P_{Clk}	Power consumption in clock distribution network
P_{GALS}	Total power consumption in GALS
P_{Osc}	Power consumption in clock generation
P_{Osc_1}	Constant that represents power consumed in one ring oscillator
P_{Rep}	Power consumed in repeaters
P/S	Processors per Sub-block
R_0	Output resistance of a minimum sized inverter
RC	Resistance Capacitance time constant
Req	Request
R_{int}	Distributed resistance of interconnects per unit length

R_{Leaf}	Interconnect resistances per unit length of clock network assuming the network has uniform width equal to the width of the leaf of H-tree
R_{Rep}	Resistance of repeater
R_{sh}	Sheet resistance for metal layer,
R_{tr}	output resistance of clock driver within each processor
RTZ	Return-to-Zero
S	Number of sub-blocks
SOC	System-on-Chip
S_{Opt_Bi}	Optimum number of sub-blocks for an array with bidirectional links
S_{Opt_Linear}	Optimum number of sub-blocks for an array with linear communication links
S_{Opt_Uni}	Optimum number of sub-blocks for an array with unidirectional links
ST	Single Track
$STBF$	Single-Track Full-Buffer
T	Temperature in degrees Kelvin
ΔT	Temperature variations
T_{Clk}	Total clock period
T_{ILD}	Thickness of interlevel dielectric
ΔT_{ILD}	Variations in thickness of interlevel dielectric
t_{ox}	Gate oxide thickness
Δt_{ox}	Variations in gate oxide thickness
t_r	Time allotted to resolve metastability
t_{Rep}	Propagation delay of a repeater
T_{Strch}	Stretched negative half period of the clock cycle
V_{DD}	Supply voltage
ΔV_{DD}	Variations in supply voltage
V_T	Transistor threshold voltage

ΔV_T	Variations in transistor threshold voltage
$VLSI$	Very Large Scale Integration
W	Window of time around a clock edge where a data transition would trigger a metastable condition
W_{int}	Width of interconnect
ΔW_{int}	Variations in width of interconnect
W_{Leaf}	Width of the leaf of H-tree
X_b	Set of communication instances for a particular pair – b - of communicating sub-blocks

TABLE OF CONTENTS

CHAPTER 1

Introduction

Most digital designs today use the clock signal to define a time reference for correct movement of data within the system. These synchronous designs are now facing challenges because with each technology generation, the number of devices per chip is increasing by reducing the minimum feature size and enlarging the chip area. Designing clock distribution networks for such large chips with frequency in the GigaHertz range is a daunting task. The problems with synchronous designs will require some radical changes in design styles and significant improvements in process technology.

The following section reviews some of the issues with synchronous designs for future large System-on-Chip (SOC) Designs. Section 1.2 evaluates asynchronous design as a possible solution. Section 1.3 discusses advantages and disadvantages of Globally Asynchronous Locally Synchronous (GALS) design style. Section 1.4 is related work and thesis overview. Section 1.5 summaries the chapter.

1.1 Issues with Synchronous Designs

1.1.1 Clock skew

Clock skew, defined as difference in arrival times of clock signal at different clock nets in synchronous designs, can severely limit the performance and may create race conditions. The clock skew is usually composed of the following parts; mismatch in RC delays along the various

paths of distributed clock wires, disparity in the clock repeater delays along the path, difference in capacitive load, and mismatch due to process parameter variations. The trend in VLSI is that the chips get larger, the clock gets faster and everything gets more complicated. As the clock cycle shrinks, we see a corresponding drop in allowable clock skew. However, larger die sizes mean that a larger overall clock distribution network must be provided. These two points will make clock distribution a most difficult aspect of computer design. Also, the process variations are not scaling with feature size or clock speed [27]. If clock skew becomes large part of clock cycle time, it decreases the time available for computation. To manage global clock skew will become a major challenge in chip design due to large die sizes expected in future designs. Clock skew and its components are further analyzed in section 3.1.

1.1.2 Interconnect delay and bandwidth

The foremost problem posed to the synchronous designs in Deep Sub-Micron (DSM) technologies is the reverse scaling properties of global wiring like the clock networks and the power distribution lines. The function of the interconnects or wiring system is to distribute clock and other signals and to provide power/ground to and among the various circuits/systems functions on a chip. The fundamental development requirement for the interconnects is to meet the high-speed transmission needs of chips despite further scaling of feature sizes. Distribution of the clock and signal functions is accomplished on three types of wiring (local, intermediate, and global). Local wiring, consisting of very thin lines, connects gates and transistors within an execution unit or a functional block (such as embedded logic, cache memory, address adder) on the chip. Local wires usually span a few gates and occupy first and sometimes second metal layers in a multi-level system. The lengths of these wires tend to scale down with technology. Intermediate wiring provides clock and signal distribution within a functional block. Intermediate

wires are wider and taller than local wires to provide lower resistance signal/clock paths. Global wiring provides clock and signal distribution between the functional blocks, and delivers power/ground to all functions on a chip. Global wires, which occupy the top one or two layers, are longer than 4 mm and can be as long as half of the chip perimeter.

Figure 1.1 shows relative gate and wire delays for DSM technologies as outlined in the 2001 International Technology Roadmap for Semiconductors (ITRS) [27]. It shows that the RC delay

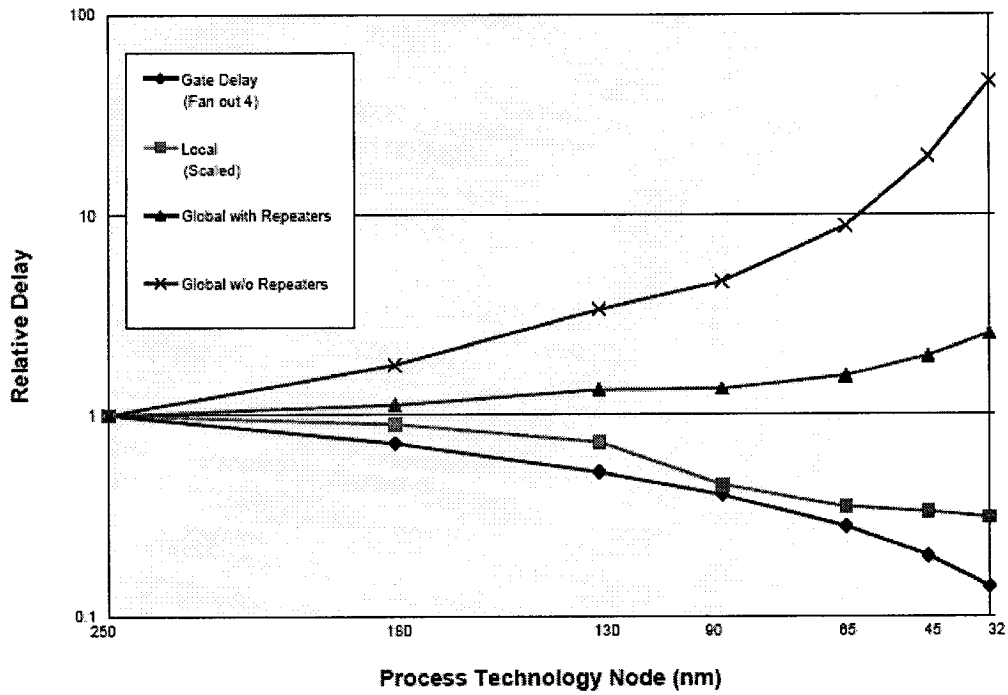


Figure 1.1. Relative delay for local and global wiring versus feature Size [27]

through fixed-length wire increases as the base fabrication technology scales to a smaller dimension. Since gate delays decrease under scaling, we see an ever-increasing disparity between wire and gate delays. With transistor switching speeds becoming faster while propagation delays along the interconnects are becoming slower, the system performance bottleneck is no longer the

delays within the logic blocks. Instead, communication among blocks has become a primary design challenge. It is no longer sufficient to simply wire two blocks together and expect them to communicate properly and efficiently. It may even be necessary to allocate multiple clock cycles to a signal's journey across the chip. So defining an optimal clock period of a common single clock for a large system is a difficult task, as this clock has to serve for everything from memory access to clocking pipelined datapaths.

Increasing resistance is the main reason behind the increased *RC* constant of interconnects in DSM. Resistance is inversely proportional to the cross-sectional area of the wire. Due to rising need for higher densities On-Chip, wiring pitches are dropping rapidly at about the same rate as gate length. Wiring capacitance is also increasing in scaled processes due to the higher densities needed to route modern chips. For instance, line-to-line spacing and insulator thickness are both shrinking, resulting in an overall increase in line capacitance. Also, one of the methods to reduce resistance has been to slowly scale line thickness, resulting in taller, thinner wires. These wires with high aspect ratio (AR = height/width) tend to have more parallel plate capacitance to neighboring lines resulting into higher overall capacitance despite smaller line widths. The increase in the *RC* constant limits the bandwidth of the clock distribution networks in DSM. In the simplest case, an interconnect can be modeled as an *RC* low-pass filter. The clock signal is degraded as it passes through the interconnect because of its limited bandwidth. If the interconnections have large *RC* constants, the waveform will have long rise times, and a high frequency clock signal will not be possible. Increasing a wire's width will monotonically increase that wire's bandwidth, since it decreases the wire *RC* product. However, making wires excessively fat will reduce the number of wires available, and hence potentially reduce bandwidth over that area.

Repeaters can be incorporated to increase the bandwidth of the clock networks. However, utilizing active repeaters in the global clock distribution networks introduce additional clock skew because of the device parameter variations. The process variations are not scaling with feature size or clock speed. Also, repeaters consume large area and power.

Local, intermediate, and global wiring pitches/aspect ratios are differentiated to highlight a hierarchical scaling methodology that has been broadly adopted. Implementation of copper and low k materials allows scaling of the intermediate wiring levels and minimizes the impact on wiring delay. Local wiring levels are relatively unaffected by traditional scaling. RC constant, however, is dominated by global interconnect and the benefit of materials changes alone is insufficient to meet overall performance requirements [27].

1.1.3 Timing closure

The increase in wire delays in DSM technologies is also causing the so-called timing closure problem, which is a challenging task of estimating and accounting for wire delays. The timing assumptions both within and across synchronous blocks must be verified pre and post layout in custom designs. The increase in coupling capacitance in DSM technologies is a potential timing hazard in that delay becomes a function of neighboring signal activity, making static timing analysis difficult [5]. Iterations of design and verification required after each change, waste time and engineering resources and increase time to market. The issue is further elaborated in section 2.5.

1.1.4 Power consumption

With ongoing advances of semiconductor technology, the power dissipation has been moving higher on the list of VLSI design constraints. Today it is at or near the top of this list, notably for ICs in portable equipment where battery lifetime is of major concern. In most high-performance synchronous VLSI designs, the distribution of low-skew global clock signals approaching the GigaHertz range is the single largest source of power consumption as confirmed by a study shown in Figure 1.2 [28]. In another study of Alpha 21064 processor, the global clock consumed 40% of the 30W power at 200MHz [29]. The reason for clock nets being the largest contributor in total power is that they have very large fanout and they span the entire chip. The interconnect capacitance of clock distribution network has significant contribution in clock power. Thus, it can be concluded that improvements in clock distribution techniques, especially local clock distribution, have the potential to lead to major power saving overall.

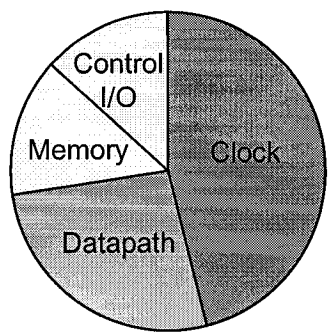


Figure 1.2. Power breakdown in a high-performance CPU [28]

1.1.5 Heterogeneous timing

We are also entering the SOC era where circuit building blocks from a number of design houses (intellectual property blocks or IP-blocks) are purchased by a systems builder and integrated onto a single chip. In some ways, this is similar to building systems by purchasing off the shelf ICs and integrating them via a PCB. However, SOC results in lower cost mass-market products with much lower power requirements. Communication between building blocks of a SOC is a complex problem particularly when a range of clocking strategies have to be tailored to each building block in order to obtain the required performance within a power budget [20].

1.2 Is Asynchronous the Solution?

Asynchronous circuits are fundamentally different from synchronous designs in a sense that there is no common and discrete time. The timing is managed locally by handshaking signals (Request and Acknowledge) between components in order to perform the necessary synchronization, communication and sequencing of operation as shown in Figure1.3. Data transactions are controlled by handshake protocols via channels. These channels are a bundle of wires upon which a protocol controls the communication of data, also called a "token". The handshaking protocol guarantees that the input tokens are processed only when valid data exists and output tokens are sent only when the output channels have reset. Numerous forms of channels have been developed that trade off robustness to timing variations and power/performance [30]. The channels have been further discussed in Section 2.3.

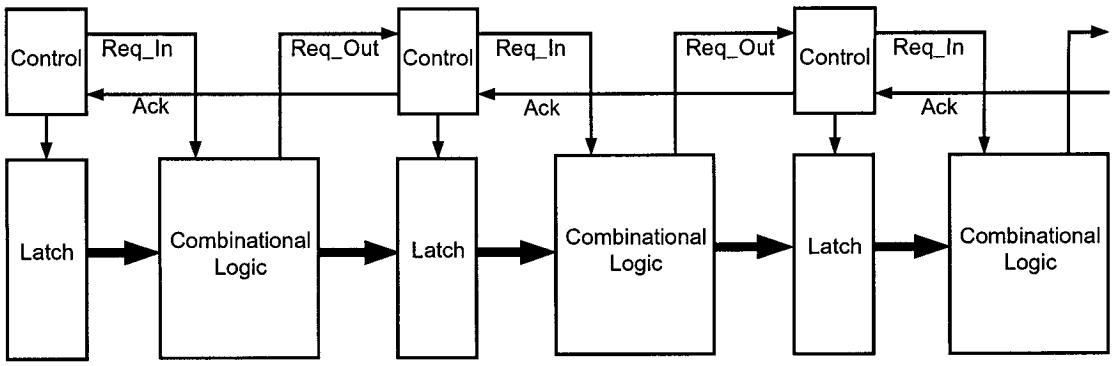


Figure 1.3. An asynchronous circuit [21]

The following sub-sections discuss some merits and demerits of the asynchronous designs.

1.2.1 Avoidance of clock skew

As mentioned before, in synchronous designs, managing clock skew is vital for the design to function correctly. Through careful engineering of the clock distribution networks, it is possible to mitigate the clock-skew problem, but solutions such as balanced clock trees are expensive in silicon area and power consumption and require extensive delay modeling and simulation [22].

The absence of a global clock in an asynchronous circuit avoids the problem of clock skew and the complexity and design efforts of the clock distribution network.

1.2.2 Robustness towards process variations

Asynchronous designs can serve to improve the process variation problem. Usually, in a synchronous design, a balanced clock tree (like H-tree) is used to globally distribute a clock

throughout the chip. In the case of long communication channel, the problem lies in the fact that the clock must travel a completely different path to the register inputs than the data must travel. If the data is on a critical path, this difference could result in a timing failure. However, in the case of an asynchronous design, all data and control bits travel a similar path. As a result, any process variation they experience should be more localized hence correlated, thereby reducing the potential timing variations and errors.

The asynchronous circuits with bundled data protocol that rely on matched delay suffer from the same timing closure problem as synchronous designs. However, delay insensitive asynchronous circuits (circuits insensitive to logic and wire delays) offer a solution to this problem at the cost of area and power consumption. The issue is discussed in detail in section 2.5.

1.2.3 Low power consumption

Asynchronous designs can reduce power consumption by avoiding two of the problems of the synchronous designs [22]:

- All parts of a synchronous design are clocked, even if they perform no useful function. A quiescent circuit only consumes a leakage current. For most CMOS circuits this leakage current is negligible compared to the dynamic current for that circuit in an active mode. A synchronous circuit is either quiescent (i.e. the clock is turned "off") or active entirely (i.e. clock is "on"). An asynchronous circuit, in contrast, only consumes energy when and where active. Any sub-circuit is quiescent until activated. After completion of its task, it returns to a quiescent, almost non-dissipating state until a next activation [31].

- The clock line itself, as discussed before, is a heavy load, requiring large drivers, and a significant amount of power is wasted in driving the clock line. Asynchronous circuits save power in clock distribution.

There are synchronous solutions to these problems, such as clock-gating, but the solutions are complex and the problems can often be avoided with no extra effort or complexity when using asynchronous design [22].

However, the asynchronous control logic that implements the handshaking consumes additional power.

1.2.4 Average case performance

In synchronous designs, the worst-case delay in combinational logic, plus some margin for flip-flop delays and clock skew, is the lower bound for the clock period. Thus, the actual delay is always less (and sometimes much less) than the clock period. Therefore, during a typical clock cycle, the circuit may in fact become quiescent well before the next clock signal. In an asynchronous circuit the next computation step can start immediately after the previous step has completed; there is no need to wait for a transition of the clock signal. This leads, potentially, to a fundamental performance advantage for asynchronous circuits, an advantage that increases with the variability in delays associated with these computation steps. However, part of this advantage is canceled by the overhead required to detect the completion of a step. Furthermore, it may be difficult to translate local timing variability into a global system performance advantage [31].

1.2.5 Modularity and heterogeneous timing

The performance of an asynchronous design can be improved by modifying only the most active parts of the circuit, the only constraint being that the communication protocol used on the interface to the module must still be obeyed. In contrast, for a synchronous design, improved performance can often only be achieved by increasing the global clock frequency, which will usually require most of the design to be reimplemented [22].

With SOC designs that use IP blocks designed by different vendors and optimized to run at different clock speeds, heterogeneous system timing will offer enormous design challenge. Asynchrony makes it easier to deal with interconnecting variety of different operating frequencies and timings, without worrying about synchronization failures, difference in clock phase and frequencies, and clock skew [31].

1.2.6 Disadvantages of asynchronous designs

Asynchronous has numerous disadvantages as compared to synchronous designs that make switching to completely asynchronous designs impractical in near future. The following are some of the demerits of asynchronous design.

- **Complexity**

  The clocked design paradigm has one simple fundamental rule; every processing stage must complete its activity in less than the duration of the clock period. Since there is no global clock to tell when outputs are stable, asynchronous circuits must prevent any

hazards, or glitches, on their outputs. A false transition on an output from one circuit can cause the next circuit to prematurely operate on meaningless results. Additional circuitry is used to prevent hazards. This circuitry can increase the area and delay of the asynchronous circuit. In order to achieve average-case performance, asynchronous circuits require additional circuitry to start each computation and detect the completion of operations. The additional circuitry required for asynchronous design can, in some cases, make the average-case delay of an asynchronous circuit become larger than the worst-case delay for the comparable synchronous circuit. Also, this added complexity results in larger circuits and a more difficult design process.

- **Testability**

  A synchronous circuit two features that simplify testing dramatically; it can be stopped during each clock cycle, and it is both simple and cheap to include a scan-chain through all flip-flops. Asynchronous circuits exhibit more autonomy, and given the large variety of isolated latch elements it is harder and more costly to connect them into scan chains. The extensive use of state-holding elements (such as Muller C-element), together with the self-timed behavior, makes it difficult to test the feedback circuitry that implements the state-holding behavior. Accordingly, testing asynchronous circuits is harder, and the cost overhead for design-for-testability measures is higher [31]. The non-deterministic behavior of arbiter element is a major obstacle in testing of fabrication faults [22].

- **Deadlock**

  Control logic designed using an asynchronous design technique is likely to deadlock if an event is either lost or incorrectly introduced, for example as a result of noise or ionizing radiation. Synchronous control circuits offer better tolerance of such problems where, for example, the extra event may cause an incorrect output, but will not normally cause a

complete system deadlock. Of course in some systems neither alternative can be tolerated [22].

- **Verification**

  Verification of synchronous designs requires the checking of the static timing constraint imposed by the clock and of the logical functionality of each module. For an asynchronous design, verification is difficult due to the non-deterministic behavior of arbiter elements, and deadlock is not easy to detect without exhaustive state space exploration [22].

- **Lack of inertia**

  Most commercial Computer-Aided-Design (CAD) tools are specialized for use in synchronous designs. The asynchronous design methodology is far from being mature. The academic research community has been very active in developing CAD tools, and many tools that support the design of asynchronous circuits are available on the Internet. So far, CAD tool vendors have monitored these developments, but they have not yet included such tools in their product portfolios [31]. Also, the design engineers in industry are not familiar with asynchronous designs and the benefits they offer. They are more skilled in pushing the limits of proven design methodologies that adapting new ones.

Even once efficient asynchronous design tools become available, a paradigm shift from a long tradition of synchronous design is not likely to happen in near future. A more gradual transition is already underway - the move toward Globally Asynchronous Locally Synchronous (GALS) designs, first suggested by Chapiro [4].

1.3 GALS Designs

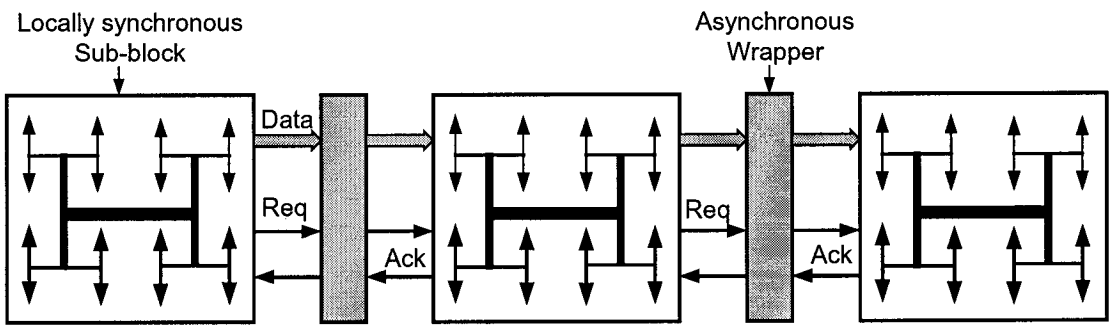


Figure 1.4. GALS system

GALS designs offer the best of both synchronous and asynchronous design styles while avoiding some of their disadvantages. The system is divided into locally synchronous sub-blocks and communication among sub-blocks is through asynchronous channels as shown in Figure 1.4. A GALS scheme combines the following features [9]:

- All major modules are designed in accordance to proven synchronous clocking disciplines.
- Data exchange between any two modules strictly follows an asynchronous handshake protocol.
- Each module is allowed to run from its own local clock.
- All asynchronous circuitry necessary for coordinating the clock-driven with the self-timed operation is confined to "asynchronous wrappers" arranged around each clock domain.

An anticipated advantage of GALS is that issues with asynchronous designs are limited to communication channels only. The sub-blocks can be designed using conventional CAD tools and design methodology. Smaller sub-blocks have relaxed design constraints and localized clock skew problem, as there is no global clock signal to distribute throughout the chip. Thus, GALS systems eliminate the need for careful design and fine-tuning of a global clock distribution network. The design efficiency achieved by divide-and-conquer is probably the most important benefit of GALS designs. Also, smaller skew allows each sub-block to be run at higher frequency. The absence of the global clock results in considerable power saving. GALS designs also offer possibility of power saving by allowing different speed/voltages for different sub-blocks. It is particularly suitable for SOC design methodology using predesigned functional blocks or IP blocks. An asynchronous wrapper allows locally synchronous IP blocks to communicate despite vast differences in internal clock frequencies and design styles.

The primary performance penalty for GALS systems is the delay due to signal arbitration and any metastability resolving time. The delay of the handshaking protocol can significantly affect the throughput of the system. Also, GALS systems require extra hardware for asynchronous wrappers and local clock generation that consumes area and power. Estimating optimum Partitioning for GALS systems to maximize benefits is the question designers face in an initial design stage. Smaller locally synchronous sub-blocks will allow higher clock frequency. It also saves power in global clock distribution but increases asynchronous overhead because increased number of partitions. This goal of this work is to study these GALS trade-offs and suggest an optimum partitioning strategy for a regular structure like an array of identical processors.

1.4 Related Work and Thesis Overview

The issue of optimum partitioning of GALS system was addressed by Hemani *et al.* [1] with the power as an objective. The suggested a design methodology for GALS systems but did not provide any models to evaluate optimum partitioning. Iyer *et al.* [2] studied effect on both power and performance on a hypothetical superscalar processor architecture. Optimum partitioning was not a criterion of their work. Our work investigates the GALS tradeoffs with different partitioning scenario. A hypothetical design of 16x16 array with 256 identical processors and 3.2 cm size has been considered. The example of processor arrays is a generic one, where we can easily illustrate the issue in question. We studied three configurations of the design; (1) an array with linear communication links among processors, (2) an array with unidirectional links among neighboring processors, and (3) an array with bidirectional links among neighboring processors. In this work, for the sake of simplicity, we considered a GALS array with identical processors, a uniform partitioning and identical local clock frequencies.

For the array with linear communication links, the clock frequencies for each partition has been calculated taking into account the interconnect bandwidth, the process parameter variations and the delay of the handshaking protocol. With change in frequency, the behavior of the power consumption was studied and the optimum number of sub-blocks (i.e. the number of partitions) that offers the maximum clock frequency at the lowest possible power consumption was evaluated.

We have also proposed closed form models for the optimum number of sub-blocks that achieves the least power at a constant frequency and with the passive clock distribution networks for all the three array configurations mentioned above.

The effect of inserting repeaters in clock network on clock frequency and power was also considered. An algorithm has been proposed to estimate optimum number of sub-blocks for an array with the active clock networks.

The remainder of this thesis is organized as follows; Chapter 2 describes tradeoffs of different asynchronous wrapper architectures and introduces a novel asynchronous wrapper architecture. Chapter 3 is estimation of system frequency with each partition for linear array of processors. Chapter 4 reviews power calculations and suggests optimum partitioning methodologies, and finally chapter 5 describes experimental setup and results.

1.5 Summary

The scaling of CMOS technology increases the *RC* delays and hampers long distance communication as well as the distribution of high-speed clock signal. Also, the wire delays are becoming harder to predict due to the process parameter variations, the DSM effects, and the dynamic effects such as cross talk. Another problem is the large power consumption in the clock distribution networks. In this chapter, these issues with the synchronous designs for Giga-scale integration have been discussed. Asynchronous and GALS design styles have been evaluated as an alternative to the synchronous designs.

CHAPTER 2

Asynchronous Wrappers

One of the principles of binary digital design is that the signals must only be sampled or observed when in one of the two distinct states. The digital circuit is really an analogue circuit approximating digital behavior. The approximation breaks down at the point when circuit switches from 0 to 1 or from 1 to 0, which takes a significant time as the signal passes through the analogue space between two digital threshold voltages. If the value is sampled during this time, which appears as a delay in digital model, its value is unpredictable and may give unexpected behavior [22]. Metastability, as defined in [7], is the state of a flip-flop in which neither binary state (a '0' or a '1') appears at the output within a time period consistent with the normal operation of the flip-flop. The primary concern with GALS is to avoid metastability while crossing clock domains at sub-block boundaries. When two synchronous systems are run from independent clocks and have to communicate with each other, they need to take special care in handling the signals received from each other. The reason is that since they do not share common time reference, the receiver may sample what sender sent precisely when the corresponding signal is changing. The receiver may get an intermediate value, which is digitally undefined. If it were to use that value without further ado, different components of the receiver might make inconsistent interpretations of the value, with the possible consequent failure of the receiver. [4].

The next section assesses the synchronization failure problem, section 2.2 explains different methods to avoid synchronization failures, section 2.3 is classification of asynchronous wrappers based on handshaking protocol, section 2.4 is survey of different asynchronous wrappers with

bundled data protocols, section 2.5 discusses the problems with bundled data protocols and compares various delay insensitive asynchronous architectures, section 2.6 justifies the choice of asynchronous wrapper for this work, and section 2.7 summaries the chapter.

2.1 Assessment of Synchronization Failures

In the late 1960's, designers of synchronous systems that engaged in high-speed communication between independent clock domains found a new class of problem related to accepting an unsynchronized signal into a clock domain. A device that can reliably and with bounded delay order two events in time cannot be constructed under the assumption of classical physics. The basic reason for this is that such a device would have to make a discrete decision – which event happened first – based on a continuous-valued input – time. Given an input that may change asynchronously, if we attempt to design a device that samples its value and returns it as a digital signal, we must accept that the device either may take unbounded time to make its decision or that it may sometimes produce values that are not legal ones or zeros but rather something in-between [14]. For example, a metastable flip-flop will remain in this tenuous state of equilibrium until some circuit parameter varies sufficiently so as to drive the state of the flip-flop into one of the two binary states. Therefore, theoretically, the time it takes a flip-flop to exit from the metastable region is unbounded. For conventional systems with fixed clocks, if the flip-flop is still metastable when the time allotted for synchronization is exhausted, we say there has been a synchronization failure.

*M*ean *T*ime *B*etween synchronization *F*ailures is given by [4],

$$MTBF = \frac{e^{\lambda_m t_r}}{f_c f_d W} \tag{2.1}$$

where, λ_m is the probability of exit per unit time, t_r is time allotted to resolve metastability, f_c is the sampling clock frequency, f_d is the expected number of data transitions per unit time, and W is the window of time around a clock edge where a data transition would trigger a metastable condition. As can be observed from (2.1), with increase in the clock speeds and integration, this failure problem becomes more relevant.

2.2 Methods to Avoid Synchronization Failures

A simpler method to alleviate the synchronization failures is the well-known Double-latching or an extension of double-latching called pipeline synchronization as shown in Figure 2.1.

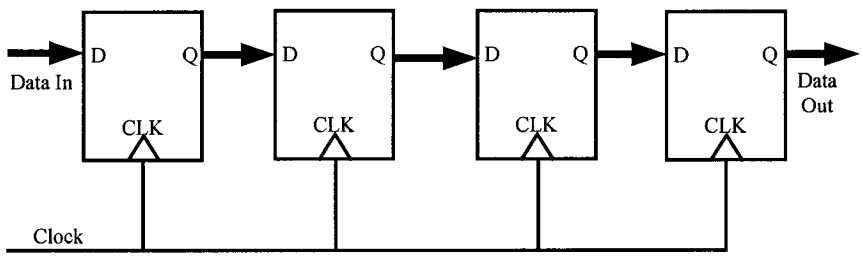


Figure 2.1. Pipeline synchronization

These methods reduce the probability of synchronization failure to an acceptable level by repeatedly synchronizing signals with back-to-back latches. They are inexpensive to implement but a major drawback is the latency of communication.

The methods in second category rely on stopping or stretching each synchronous module's local clock to guarantee that communication signal never violate setup and hold time constraints with respect to local clock. The idea of pausing the clock to avoid metastability issues was first proposed by Chapiro [4]. Such clocking systems manage asynchronous communication between two clock domains by stretching off-phase of both the clocks while the handshaking and data transfer takes place. This is typically done using a Mutual Exclusion (ME) element inside the loop of a ring oscillator as shown in Figure 2.2. The ME forces temporal separation of the

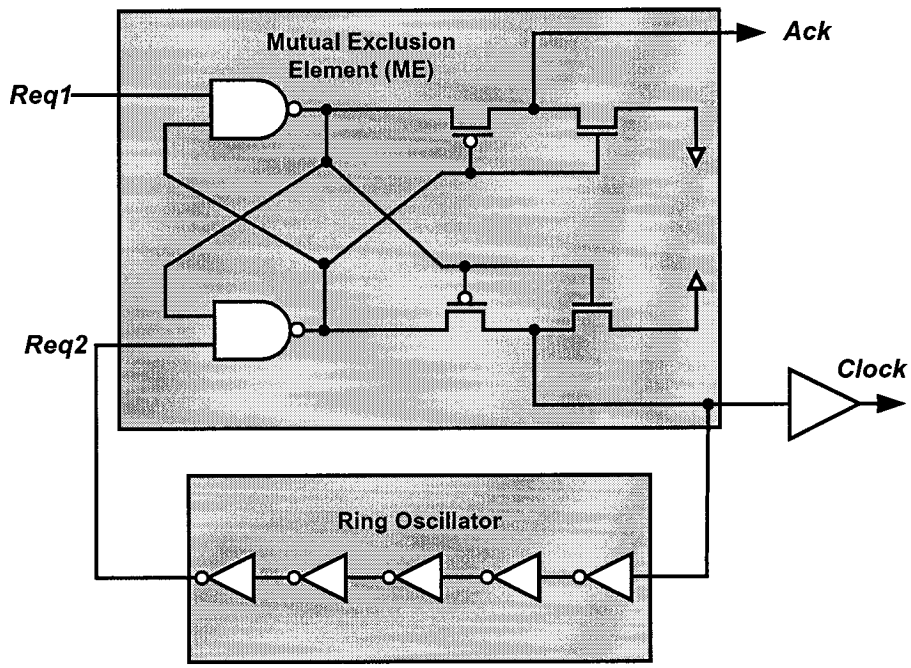


Figure 2.2. Pausible clock generator

sampling edges of the clock and external requests. Because MEs require that requests competing for shared resources must be persistent, the clock input to the ME must be "stretched" when it loses the arbitration. In the case where the rising edge of the clock and the request to stop the clock arrive simultaneously, the ME "tosses a coin" and grants one of the rising edge of the clock or the request to pause the clock. A ring oscillator, which is typically an odd number of inverters

in a chain, is used instead of a crystal in order to be able to adjust the duration of off-phase of the clock. The clock from ME normally has 50% duty cycle, except when it looses the arbitration, in which case the off-phase of the clock is stretched.

This scheme completely eliminates the synchronization failure problem. Furthermore, the asynchronously controlled stoppable clocks make the automatic power down operation of currently idle blocks. However, the stopping of clock may significantly affect the throughput of the system. The loss of long-term timing predictability is another consequence of stretching the clock. After pausing a clock, the first edge through ring oscillator and clock buffer will propagate slower than subsequent events [32].

2.3 Classification of Asynchronous Wrappers Based on Handshaking Protocols

To adapt synchronous blocks to an asynchronous environment they are equipped with an asynchronous wrapper. These wrappers manage all data transfers into and out of the respective blocks and deliver a locally generated clock using clock generator explained in section 2.2.

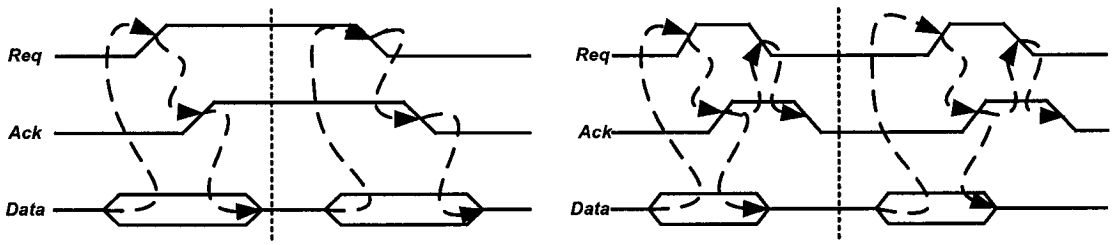
The asynchronous wrappers follow a handshaking protocol to exchange data safely. The handshake protocols can be classified as follows:

(1) 2-phase or non-return-to-zero (NRZ) or transition signaling

(2) 4-phase or return-to-zero (RTZ) or level signaling

(3) Single Track (ST) or 2−phase, return-to-zero handshaking

Figure 2.3 shows the difference in 2-phase and 4-phase protocols. The term 4-phase refers to the number of communication actions:



(a) 2-Phase Protocol (b) 4-Phase Protocol

Figure 2.3. 2-phase and 4-phase handshake protocols [21]

(1) the sender issues the data and sets request high,

(2) the receiver absorbs the data and sets acknowledge high,

(3) the sender responds by taking request low, and

(4) the receiver acknowledges this by taking acknowledge low.

The disadvantage of superfluous return-to-zero transition in 4-phase protocol is that it costs unnecessary time and energy. The 2-phase protocol shown in Figure 2.3(a) avoids this. The information on request and acknowledge wires is now encoded in signal transitions on the wires and there is no difference in 0-to-1 or 1-to-0 transition, they both represent a "signal event" [21]. Because only two transitions are required per handshake, 2−phase protocols are more time and energy efficient than 4-phase protocols. However, on the other hand, the absence of return-to-zero transitions of the 2-phase protocols incur event-driven logic, which frequently amount to more complex custom design than those implementing the various 4-phase protocols. Transition

signaling, which tends to result in large, slow circuits may be a good alternative to 4-phase RTZ signaling for long interconnects because each communication across the link only requires two link propagation delays, as opposed to four.

The Single Track (ST) handshake protocol shown in Figure 2.4 needs just one wire to signal both request and acknowledgement. The sender pulls the ST line high to indicate a request. The receiver pulls the ST line low to signal an acknowledgement.

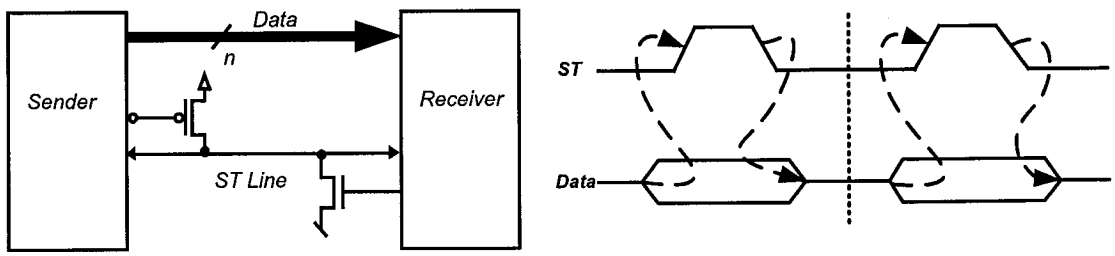


Figure 2.4. Single-track handshaking protocol

Since after an *REQ* transmission, the *ACK* signal returns the wire to its initial electrical state, the single-track signaling essentially combines the common 2-phase and 4-phase protocols to become a 2-phase, return-to-zero handshaking scheme. However, since it relies on momentarily high impedance states, the implemented circuit will run correctly only if it is neither exposed to heavy ambient noise nor through long wires without repeaters.

Asynchronous protocols can also be categorized based on the delay assumption in circuit and wires as follows:

(1) Bundled data protocols, and

(2) Delay insensitive (DI) protocols.

Both these protocols can employ either NRZ, or RTZ or ST protocols described before. A detailed classification of asynchronous circuits can be found in [21].

The following sections contain brief description of bundled data and DI protocols and some wrapper architectures that utilize these protocols with pausible clocking schemes.

2.4 Asynchronous Wrappers with Bundled Data Protocols

The term "bundled data" refers to a situation where the data signals use normal Boolean levels to encode information, and where separate request and acknowledge wires are bundled with the data signals as shown in Figure 2.5. The delay between *Req_In* and *Req_Out* has to be matched with the delay of combinational logic for the circuit to function properly.

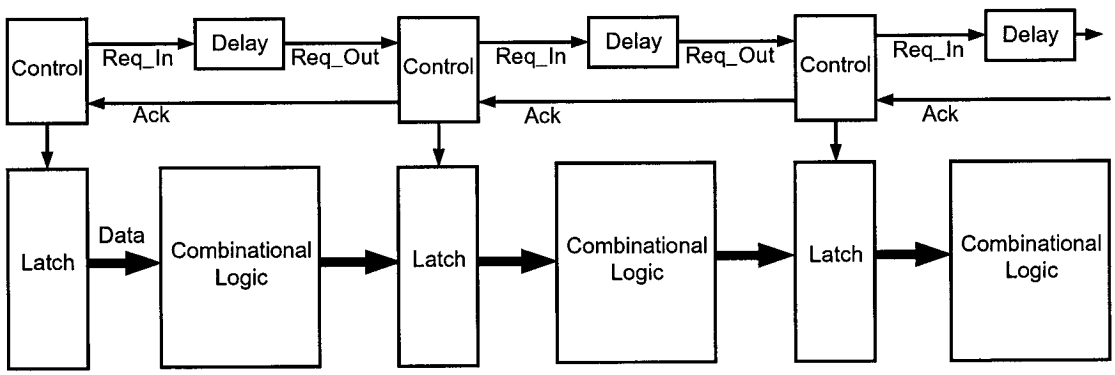
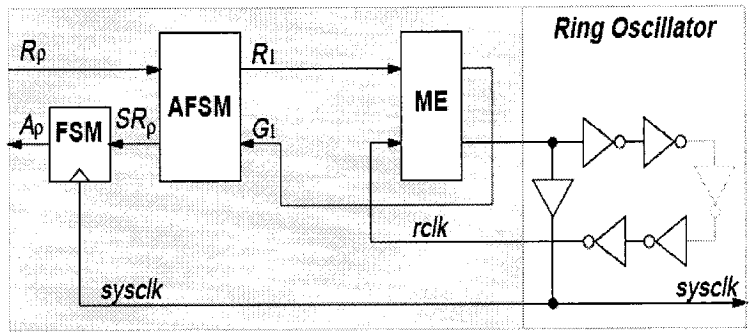


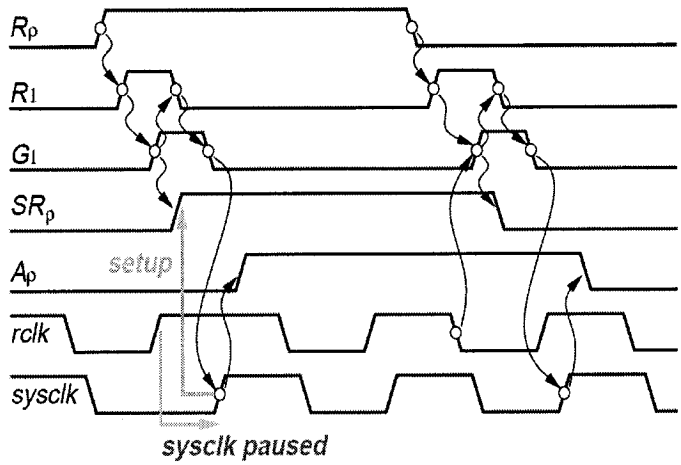
Figure 2.5. Bundled data protocol

The protocol is also referred to as "single rail " protocol. The term bundled data hints at the timing relationship between the data signals and the handshake signals, whereas the term single-rail hints at the use of one wire to carry one bit of data [21].

2.4.1 Asynchronous wrapper by Yun *et al.*



(a) Receiver PCC for one-way communication



(b) PCC timing (one-way communication)

Figure 2.6. Yun's asynchronous wrapper [11]

The bundled data scheme suggested in [11] uses an asynchronous FIFO channel to communicate between every pair of locally synchronous sub-blocks. The communication between a sub-block and the FIFO is done using a request/acknowledge handshaking. Synchronization of handshaking signals to the local sub-block clock is done with pausible clocking control (*PCC*). Figure 2.6(a) shows a receiver PCC circuit for one-way communication with all control signals and clock generator. The timing diagram of signals is illustrated in Figure 2.6(b).

A request event from FIFO on R_p is forwarded to ME via the Asynchronous Finite State Machine (AFSM). If *rclk* is low when R_l rises, then ME immediately raises G_l, which prompts AFSM to generate an even on SR_p. This event is effectively synchronized to *sysclk*, i.e., guaranteed not to introduce a synchronization failure when sampled by the finite state machine (FSM) under a reasonable timing assumption described in [11]. Note that *rclk* may rise before the ME lowers G_l, but the ME will not allow *sysclk* to rise until G_l becomes low. In order to prevent *sysclk* from stalling indefinitely (until the next toggling of the request, R_p), the AFSM lowers R_l immediately after G_l rises, which in turn causes G_l to fall allowing *sysclk* to rise.

The PCC does not differentiate rising edges of R_p from falling edges-both edges enable R_l to be asserted and G_l to be asserted as a result. In fact, the AFSM effectively performs a 2-phase to 4-phase conversion from R_p to R_l and a 4-phase to 2-phase conversion from G1 to SR_p.

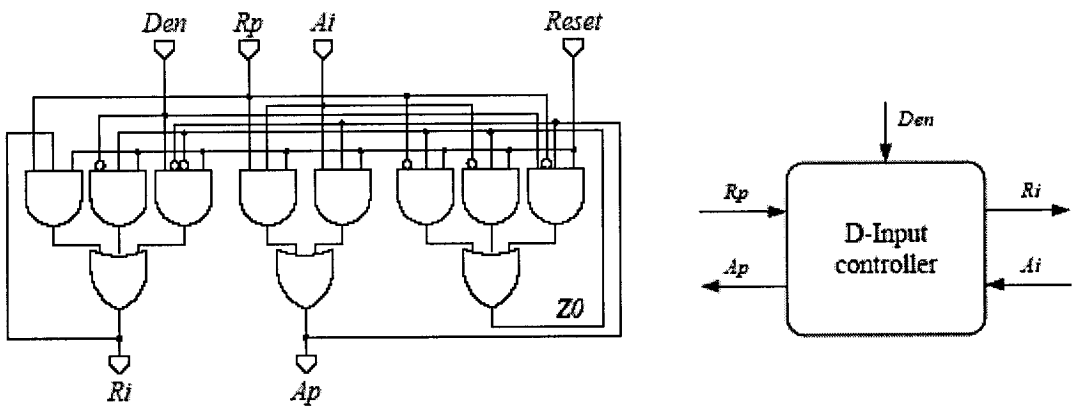
The scheme was tested on a MOSIS 1.2μm CMOS technology with two synchronous modules including the pausible clocking control and an asynchronous FIFO. The resulting system worked reliably up to the local clock frequency of 220MHz - the maximum clock rate is limited by the ring oscillator, not the pausible clocking control. The power consumption of the scheme was not mentioned.

The scheme requires at least two clock cycles to transfer data and at most one port per module can be active at a time because of the arbiter. Moreover, possible applications are confined to circular data flows, as increasing fan-ins and fan-outs make their arbiter block large and impractical [9].

2.4.2 Asynchronous wrapper by Muttersbach *et al.*

Muttersbach *et al.* [9] used the similar pausible clock generation scheme as shown in Figure 2.2 to prevent metastability. The 4-phase bundled data handshake protocol has been used to signal validity of the data.

Two types of port controllers were used in the work: demand-ports (D-ports) and poll-ports (P-ports). The AND-OR combinational circuit of D-port shown in Figure 2.7 is result of synthesis from Extended-burst-mode specifications [9].



(a) Circuit Implementation (b) Block Symbol

Figure 2.7. D-type input controller [9]

Upon activation by a switching event on *Den* it issues a request for a clock stretch R_i+, which gets acknowledged by A_i+. The (+) and (-) signs represent rising and falling edge of the signal respectively. When the clock is ensured to be and remain low, the external handshake cycle on R_p/A_p gets processed and subsequently the clock may resume again.

Poll-type (P-type) controllers differ from their demand-type counterparts in the way they influence the clock generation. After being activated by a transition on the *Pen* signal (Figure 2.8), a P-port polls the handshake lines it is attached to. If its communication partner reacts, R_i is set only during the processing of the handshake cycle. Thus, in many cases, the local clock is not affected at all.

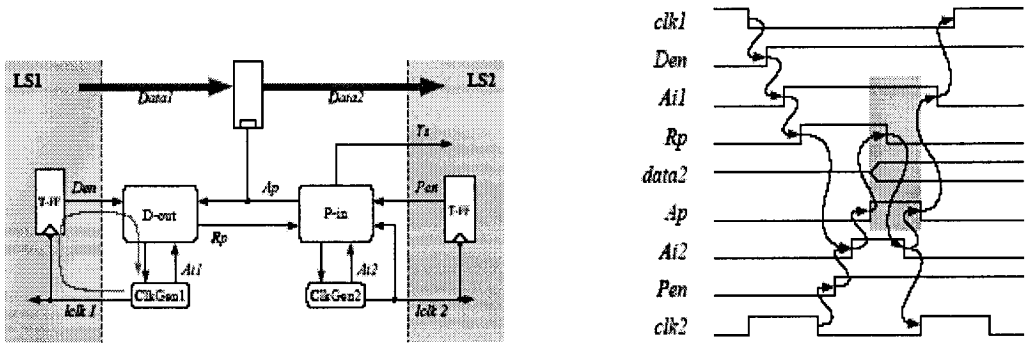


Figure 2.8. Configuration of a data exchange channel and corresponding waveforms [9]

Figure 2.8 shows two locally synchronous modules communicating via an asynchronous channel and the drawn waveforms of a data transfer. The sending module (*LS1*) is equipped with a D-type output, while the receiver (*LS2*) is a P-input. The cycle is started by *Den*+, whereupon clock generation 1 is stopped (A_i1+). In this situation the receiver has not been enabled so far. Immediately after this has happened the receiving P-port detects the pending transfer, ensures that clock 2 is stable low (A_i2+) and sets the acknowledge signal (A_p+). As soon as R_p has gone low

clock 2 may start again, while clk1 has to be kept stable low, until the handshake on R_p/A_p has completely finished. In a multi-port wrapper some other ports might keep the clock stretched for an unknown amount of time. Using A_p for latch control makes sure that the receiving port got enabled before the latch gets transparent and that clock 1 is paused during the transparency phase.

An AND-OR combinational circuit with delay elements is used to handle multiple requests from many ports of a sub-block. This resolves the disadvantage of Yun's scheme where increasing fan-ins and fan-outs from a sub-block make its arbiter block large and impractical. In 0.25 μm CMOS process, the scheme achieves data rates exceeding 300MHz on the channel. The disadvantage of 4-phase protocol used in this scheme is that the return-to-zero transition costs unnecessary time and energy. The four transitions required per handshake could result in significant performance penalty. The power consumption of the scheme was not mentioned in [9].

2.4.3 Asynchronous wrapper by De Clercq *et al.*

De Clercq *et al.* [12] invented a high-speed GALS communication structure using bundled data single-track (ST) handshaking described in section 2.3 and pausible clocking. They suggested two versions for send and receive ports. The WAIT version of the ports blocks the sender/receiver from the moment the transfer a request to send/receive data is issued till the moment the transfer is complete. The sample version of the ports allows the sender/receiver to keep on running till the transfer is to take place, at which time, their clock is stopped for the duration of the transfer.

Figure 2.9 shows a data communication channel with WAIT send and receive ports. Figure 2.10 illustrates corresponding waveforms for transmitting one data with WAIT send and receive ports.

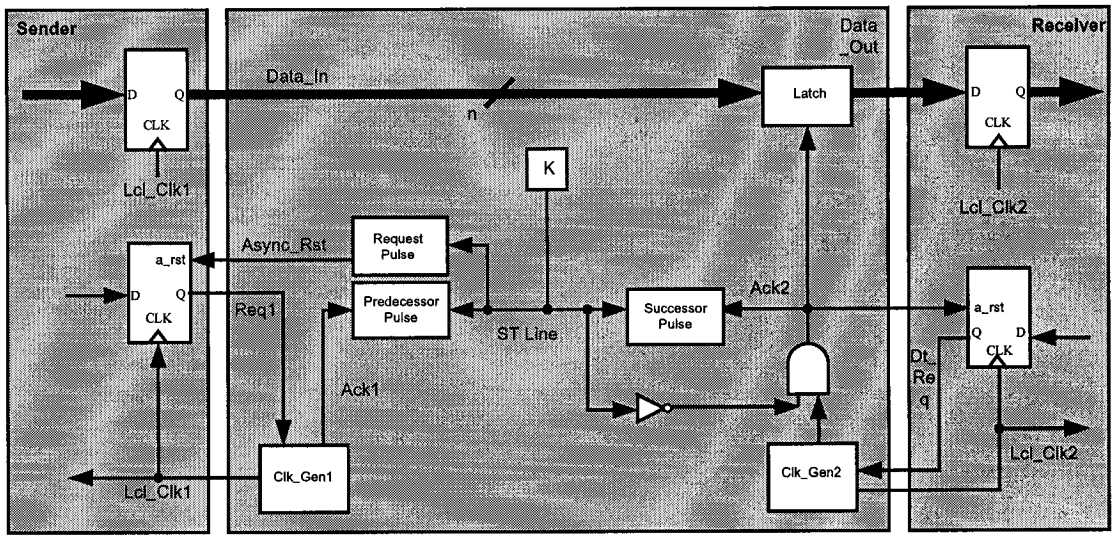


Figure 2.9. Data communication channel with WAIT send and receive ports [12]

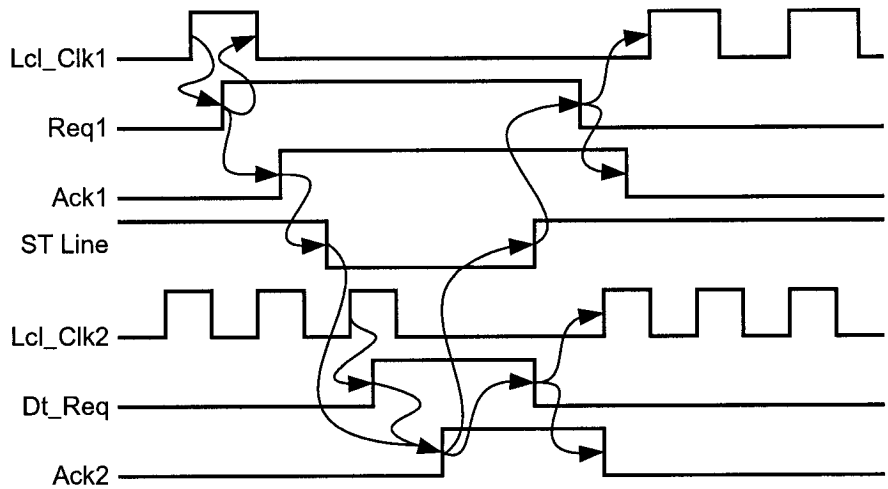


Figure 2.10. Waveforms for communication channel with WAIT send and receive ports [12]

Upon receiving *Req1* from sender, the ME in the clock generator pauses the *Lcl_Clk1* and raises *Ack1*. *Ack1* pulls down ST line with the help of *Predecessor Pulse* block. When *Dt_Req* is received from the receiver, *Ack2* goes high. *Ack2* has three functions: (1) to pull the ST line back high, (2) to latch the data from the sender, and (3) to reset *Dt_Req*. When ST line goes high, it generates a pulse on *Async_Rst* through *Request Pulse* block. *Async_Rst* resets *Req1* and

Lcl_Clk1 starts again. A keeper, a positive feedback loop of weak back-to-back inverters, represented by a box marked K, maintains the voltage of the ST line constant when it's not being driven.

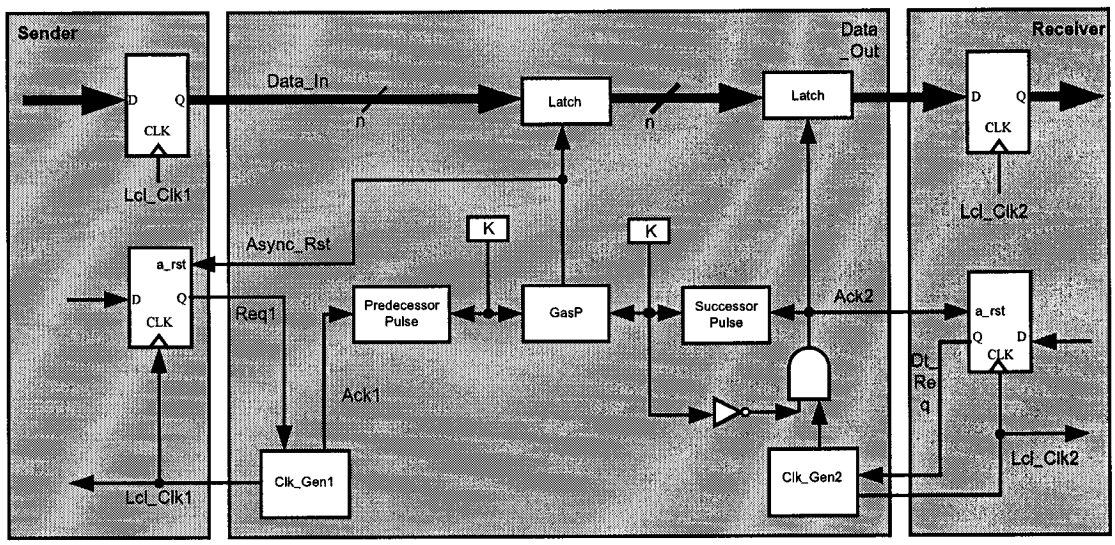


Figure 2.11. Data communication channel with GasP stage pipeline and WAIT ports [12]

A GasP pipeline stage [8] can be inserted between sender and receiver (Figure 2.11) to break the request-to-acknowledge delay in scheme of Figure 2.10. It increases the throughput of the system at the expense of increased latency. A GasP pipeline stage latches data when its predecessor is *FULL* (predecessor ST line is low) and its successor is *EMPTY* (successor ST line is high). A GasP stage is self-resetting and has an operation cycle equivalent to a three-inverter loop. Thus when *Lcl_Clk1* is stopped, the *Predecessor Pulse* block pulls the GasP stage's predecessor ST line low. The GasP stage then latches the data and pulls its successor ST line high. The latching signal of the GasP stage is also used to asynchronously reset *Req1*. Once the successor ST line of the GasP stage is low, the remainder of the circuit behaves as in ST adapter discussed before.

With 0.18 μm CMOS process, the scheme in Figure 2.12 achieves a throughput of 1.1 Giga-Data-Items per Second (GDI/S) with power consumption of 3.67 mW. Comparing the throughput using a full scaling model (to justify the difference in simulation technology), the scheme of Muttersbach *et al* in [9] gives a throughput of 300 Mega-Data-Items per Second (MDI/S). However, since the scheme in Figure 2.12 relies on momentarily high impedance states on the ST lines, the implemented circuit will run correctly only if it is not exposed to heavy ambient noise.

2.5 Asynchronous Wrappers with Delay Insensitive Protocols

All the bundled data protocols rely on delay matching (Figure 2.5), such that the order of signal events at the sender's end is preserved at receiver's end. On a push channel (where sender is the active party that initiates the data transfer on the channel), data is valid before request is set high. This order should also be valid at the receiver's end. Usually, the request signal is driven by a matched delay line that is larger than sander's computation delay plus some margin. Therefore, the bundled data protocols involve timing margins and assumptions, both in interconnects and logic, when physically implementing such circuits. As CMOS manufacturing technology scales into deep and ultra-deep sub-micron designs, for high-speed on-chip communication, the bundled data protocols will suffer the same drawbacks as the synchronous designs.

Timing closure is a challenging issue for GALS with bundled data protocols and synchronous designs with long global interconnects. This rise in coupling capacitance produces noise, which, for DSM interconnects, comes in two distinct forms: delay deterioration and crosstalk. Here, we concentrate on *delay deterioration*, which means that the total capacitance that a gate is subjected to, is no longer a constant value. The rising contribution of coupling capacitance to total load capacitance means that the *Miller effect* can significantly affect on-chip delays. The Miller effect

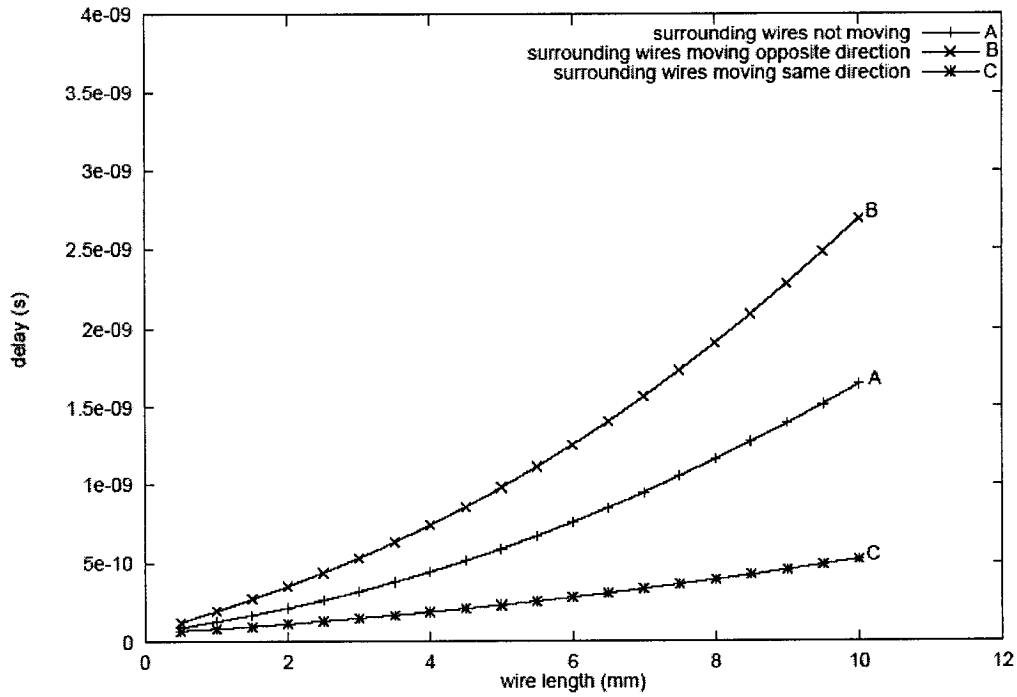
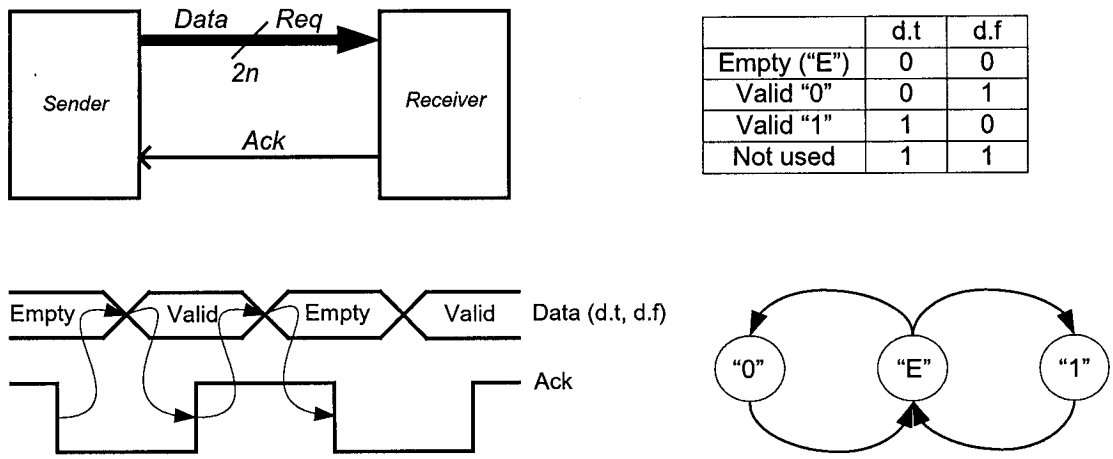


Figure 2.12. Signal propagation delay (transit time) for a close-packed wire [23]

- states that the simultaneous switching of both terminals of a capacitor will modify the effective-capacitance between the terminals. For instance, if wire *A* switches from *0* to V_{DD} while adjacent wire *B* switches from V_{DD} to *0*, the effective voltage swing between the two terminals is $2V_{DD}$. From *Q* equal *CV*, the charge *Q* needed to switch wire *A* is now double that of the case where wire *B* is static. Alternatively, we can interpret this change as a doubling of "effective" capacitance *C*. The increase in coupling capacitance is a potential timing hazard in that delay varies with neighboring signal activity, making static timing analysis difficult [6]. Iterations of design and verification required after each change, waste time and engineering resources and increase time to market. The severity of the problem can be observed in Figure 2.12. Bainbridge *et al.* [23] performed crosstalk analysis with group of wires with minimum dimension and spacing in 0.35 μm technology. In the worst case, neighboring signal activity can produce few neno-seconds difference in delay of a wire with lengths in mili-meters. Possible variations in delays in long wires, caused by crosstalk make bundled data systems difficult to design, requiring

the inclusion of large margins, which seriously affect the performance of the system. Synchronous designs face a similar timing closure problem since the clock frequency must be chosen such that the receiver is told that data is stable at the correct moment.

An alternative to bundled data protocol is a more sophisticated protocol that is robust to wire delays, called Delay Insensitive (DI) protocol. The DI protocols encode request on data lines and therefore avoids the need for timing analysis, giving designs that operate correctly whatever the delay in the interconnecting wires. The most common DI channel protocols are; dual rail channel and 1-of-N channel (or 1 hot channel).



	d.t	d.f
Empty ("E")	0	0
Valid "0"	0	1
Valid "1"	1	0
Not used	1	1

Figure 2.13. A delay insensitive channel using 4-phase dual rail protocol [21]

The dual rail protocol uses two wires per bit (as opposed to one wire per bit in bundled data and synchronous designs) of information that has to be communicated as shown in Figure 2.13. In essence it is a 4-phase protocol using two request wires per bit of information *d*; one wire *d.t* is used for signaling logic 1, and another wire *d.f* is used for signaling logic 0. As evident form Figure 2.13, no separate request wire is required. If the request signal is required for control

circuits, like in pausible clocking schemes to stop the clock, can be generated by OR-ing the data lines.

For 1-of-N channels, the data is encoded in such a way that only one out of N data wire changes its state. It uses N data wires to send log_2N bits of data. Since only one wire changes its state, it yields lower power than conventional dual-rail channels.

The bundled data and delay insensitive protocols trade off robustness to timing variations and area/power/performance. In the following sub-sections, we discuss tradeoffs of the DI scheme suggested by Bainbridge *et al.* and our DI design using single-track handshaking.

2.5.1 1-of-4 Delay insensitive interconnects by Bainbridge *et al.*

Table 2.1. 1-of-4 data encoding [23]

d3 wire	d2 wire	d1 wire	d0 wire	Information Transferred
1	0	0	0	Two-bit data value 11
0	1	0	0	Two-bit data value 10
0	0	1	0	Two-bit data value 01
0	0	0	1	Two-bit data value 00

The scheme by Bainbridge *et al.* [23] uses 1-of-4 encoding with 4-phase protocol. It uses four wires to send two bit data. Table 2.1 shows this 1-of-4 data encoding where only one wire in a group is allowed to signal data at any time. Since it is possible to detect the arrival of each

symbol at the receiver (with RTZ signaling, the wires are all low when no symbol is being transmitted) a 1-of-4 encoding is delay insensitive, as are all the other 1-of-N codes.

Further advantages of 1-of-4 RTZ signaling are: (1) the likelihood that two adjacent wires will switch at the same time is much less; (2) any crosstalk that does occur will be between wires switching their signals in the same direction (3) Since only one wire changes its state, it yields lower power than conventional dual-rail channels. However, beyond 1-of-4, one-hot 1-of-n codes are significantly more expensive than a dual-rail encoding, e.g. the 3-bits of data that can be carried in a 1-of-8 code requires only 6 wires in dual-rail, 4-bits requires a 1-of-16 code, but only 8 wires in dual-rail [25].

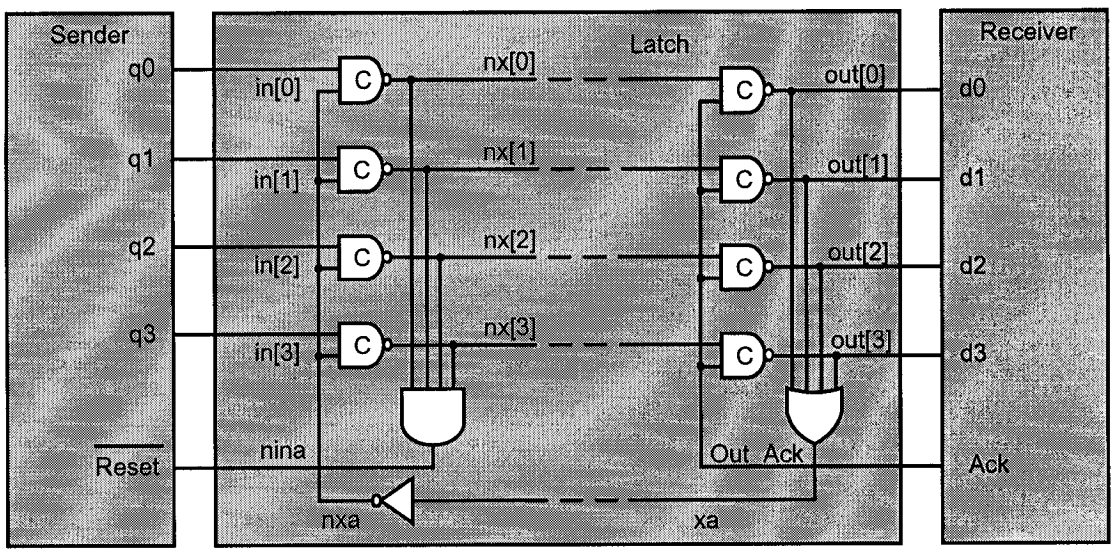


Figure 2.14. Bainbridge's 1-of-4 latch with sender and receiver

They invented different 1-of-4 modules like latch, merge element, select block, and switch, which can be used to build complete asynchronous multi-point interconnects. This work was confined to asynchronous designs and did not provide any GALS-specific asynchronous wrapper scheme.

In brief, the operation of the scheme in Figure 2.14 is that: (1) initially the input group is low; (2) an input symbol is presented by raising one of the inputs, *in[1]* say; (3) since *xa* is low at reset, *in[1]* passes to *nx[1]* causing an acknowledge on *nina*; (4) *in[1]* may then fall at any time, but this will not be passed to *nx[1]* until *xa* has risen, indicating that the next stage has accepted the value encoded on the *nx* lines; (5) after *nx[1]* rises, so does *nina*, returning the latch to its original idle state.

A link in Network-on-Chip (NOC) environment was designed using the above-mentioned modules in authors' other work [24]. On 0.35-micron technology, simulations show a throughput of around 700 megabits per second (Mbps) per link, with more than 1 Gbps per link projected for 0.18-μm CMOS technology—using suitable link lengths to minimize end-to-end latency. This corresponds to 120 Mbps per wire on 0.35-μm CMOS technology and 160 Mbps per wire on 0.18-micron CMOS technology.

2.5.2 A novel asynchronous wrapper using 1-of-4 encoding and ST handshaking

The delay insensitive scheme in Figure 2.15 uses a "*HI means FULL*" form of *GasP* templates and ST handshaking. The *GasP* family of asynchronous circuits were invented by Sutherland *et al.* [8] and used in a bundled data wrapper scheme by De-Clercq *et al* in [12].

The operation of the scheme in Fig. 4 can be briefly explained as follows: (1) initially all *IN* lines and *ST* lines are low and signals *A* and *B* are high, (2) the rising edge of *Clk_Send* makes one of the input line, say *IN[1]*, "1"; (3) rising of *IN[1]* sends a request to the sender clock generator by raising *Req_Send* and the clock will be paused until *Req_Send* is reset; (4) *Gasp[1]* drives *ST[1]*

high, thereby driving the signal *A* low and generating a pulse on *Rst_Send* through *Pls_Gen1*; (5) *Rst_Send* will make *IN[1]* "*0*" and remove *Req_Send*, thereby restarting *Clk_Send*. Any other input line can become "1" now but the "0" on signal *A* will stop it from propagating further; (6) when the receiver is ready to accept data, *Req_Recv* will go high on the rising edge of *Clk_Recv*; (7) After *Clk_Recv* going low, *Ack_Recv* will rise, generating a pulse on *Pull_Zero* through

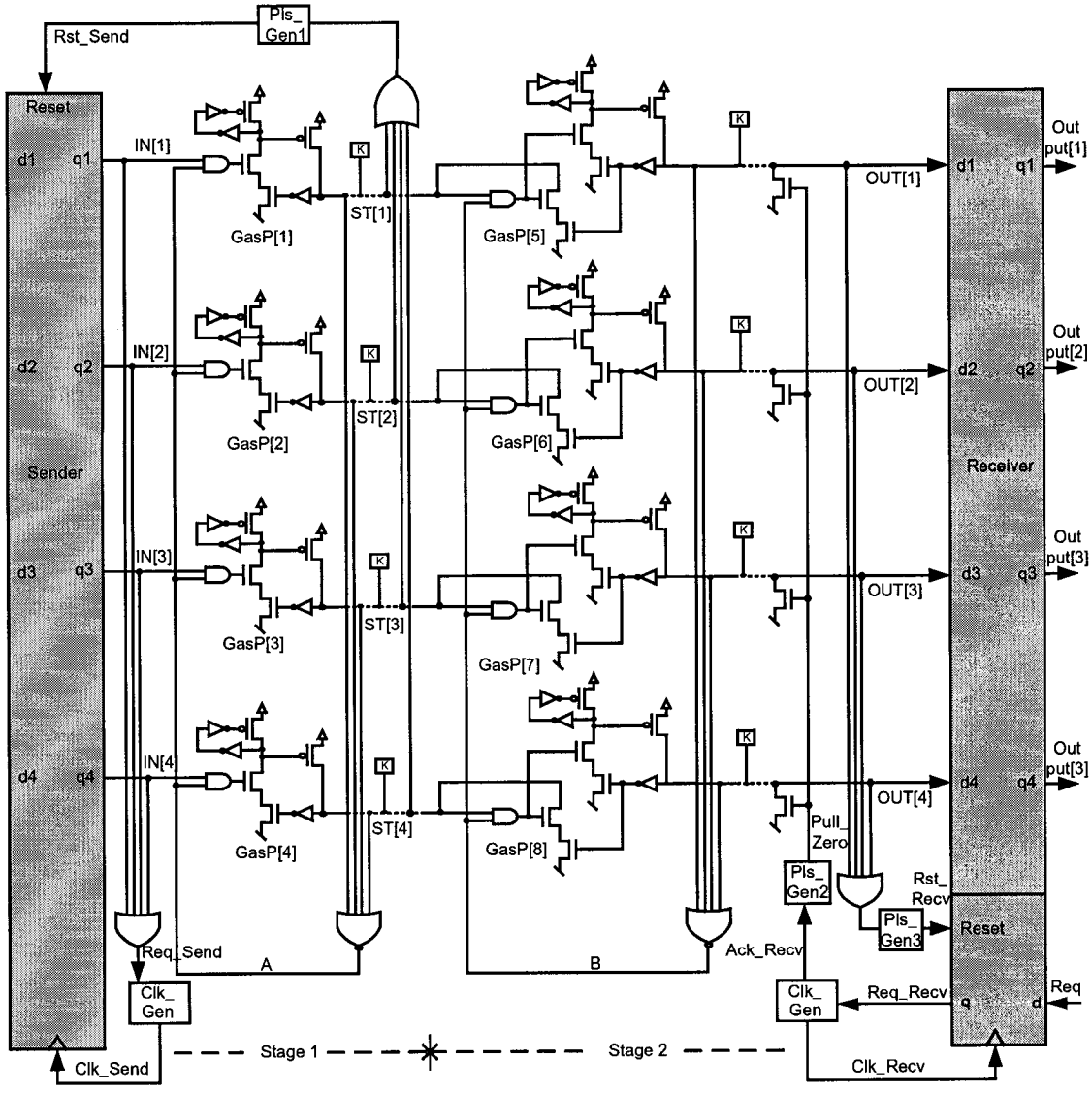


Figure 2.15. 1-of-4 delay insensitive scheme using ST handshake

Pls_Gen2, which drives all *OUT* lines low; (8) *GasP[5]* will now trigger driving *ST[1]* low and *OUT[1]* high, which makes signal *B* low; (9) *ST[1]* will reset signal *A* low and any other input, blocked by signal *A* before, can propagate through *GasPs* of stage-1 but will now be blocked by signal *B*; (10) *OUT[1]* going high will generate a pulse on *Rst_Recv* with the help of *Pls_Gen3*, that will reset *Req_Recv* and restart *Clk_Recv*.

A keeper, a positive feedback loop of weak back-to-back inverters, represented by a box marked K, maintains the voltage of the ST line constant when it's not being driven [12].

The *Pls_Gen* block in the scheme is shown in Figure 2.16. It produces a high pulse on *Output* when the *Input* rises. The width of the pulse is equivalent to the delay of three inverters.

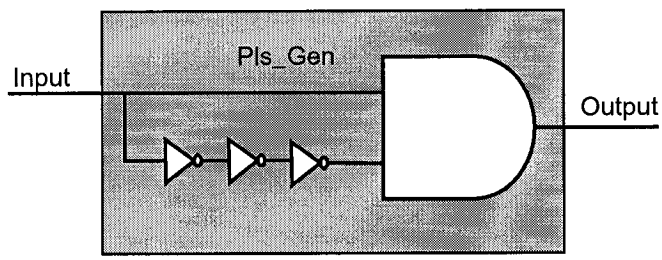


Figure 2.16. Pulse generator block of Figure 2.16

This scheme saves one wire as compared to the design in [23]. Moreover, because of ST handshaking, it requires just two transitions per handshake as opposed to four in [23]. The scheme was simulated in 0.18 μm CMOS technology. The result of simulation are shown in Figure 2.17 for five consecutive transfers on *IN[1]* and *IN [2]*. The simulations show a throughput of 1.66 Gbps per link, which is 66% higher than the projected throughput in the scheme by Bainbridge *et al* with the same technology. This corresponds to 415 Mbps per wire, which is approximately 2.6 times the throughput per wire predicted in [9]. The power

consumption was measured 3.8 mw per data transaction (excluding the encoder and decoder required at the sender's end and receiver's end respectively). Also, the susceptibility of ST handshaking to the ambient noise warrants a careful design.

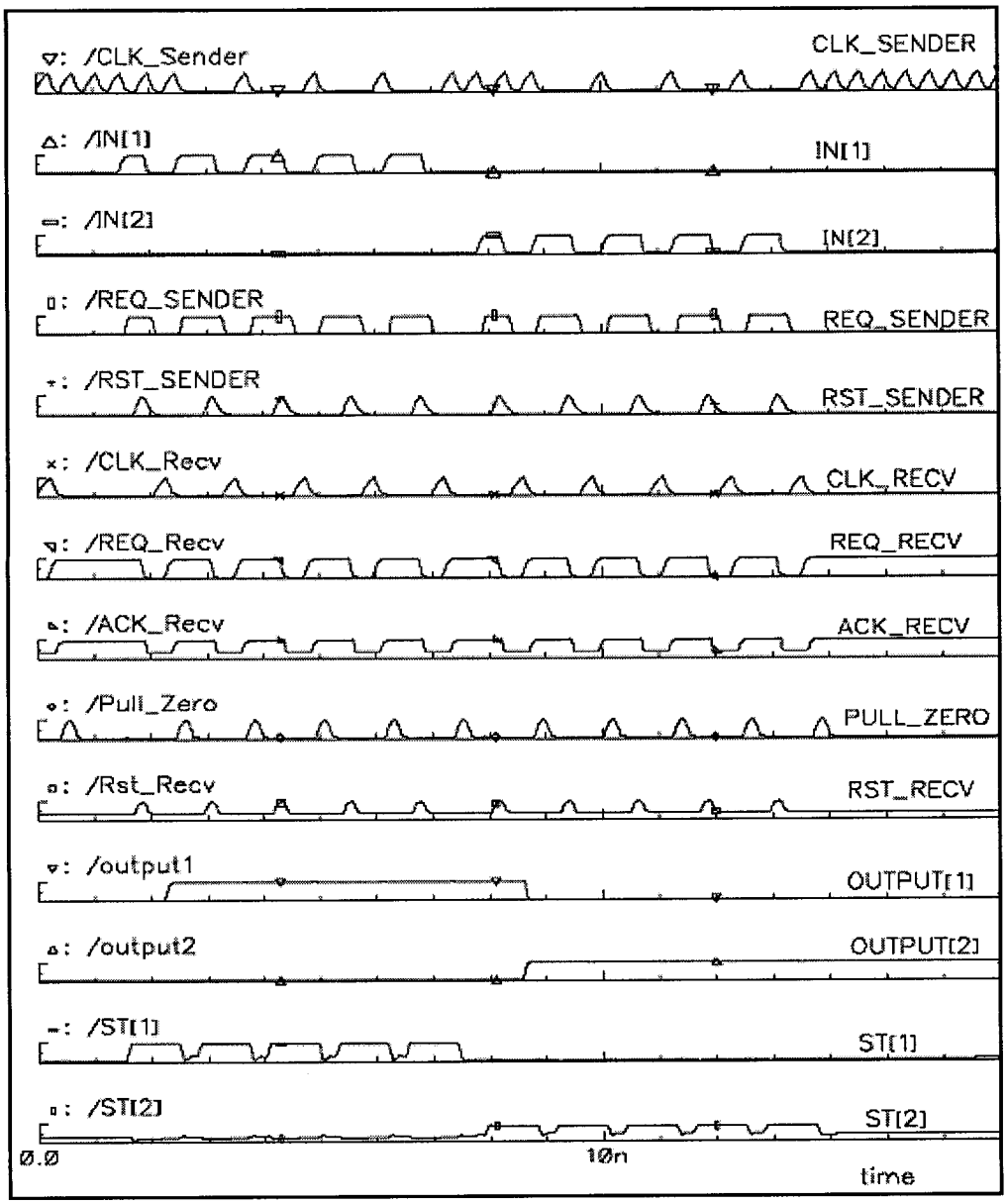


Figure 2.17. Simulation results of 1-of-4 delay insensitive scheme using ST handshake

The Single-Track Full-Buffer *STFB* suggested in [13] may be used in place of *GasP* in the proposed architecture. However, STFB has some tight timing constraints that, if violated, could result in significant short circuit current during the transitioning of the *IN, ST,* and *OUT* lines. Fig. 2.18 shows the schematic and State Transition Graph (STG) of the STFB.

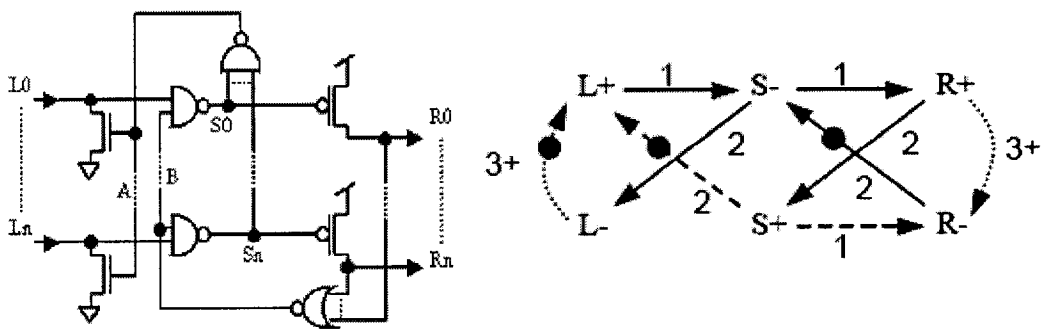


Figure 2.18. 1-of-N STFB buffer [13]

The notation "+", "↑" and "-", "↓" represent the rising and falling of the signals respectively. In Fig. 6, the timing margin between the tri-stating of an output wire and the earliest time the environment can reset the wire (R-) is zero [13]. In the proposed scheme, due to the self-resetting property of *GasPs* in stage-2, it stops driving the *OUT* lines after a delay of approximately three inverters. The delay between *OUT*+ and *Pull_Zero*+ is much greater than the delay in self-resetting the *GasPs.* This avoids short-circuit currents on *OUT* lines.

Another timing constraint in *STFB* is that the timing margin between tri-stating of an input wire and the earliest time the left environment can drive the wire (*L*+) is also zero. In the proposed scheme, the *IN* lines are driven low by resetting the sender to avoid short-circuit currents on *IN* lines. The *ST* lines between two consecutive stages have a similar timing restriction. The *GasPs* in stage-1 drives the ST lines high to indicate a request and the *GasPs* in stage-2 drives it low to

indicate an acknowledgement. Therefore, to avoid short circuit currents, *GasPs* in stage 1 must stop driving the ST lines before stage 2 starts driving it low. However, this one sided constraint can be easily satisfied by appropriate sizing of the AND gates of *GasPs* in stage 2. Moreover, after a transaction, signal *B* should close stage-2 before the following transaction appears on the ST lines. This can be achieved by appropriate sizing of the NOR gates driving the signals *A* and *B*.

2.6 Choice of Asynchronous Wrapper for This Work

Our work concentrates on GALS designs with array of identical processors. The incentive to use delay insensitive scheme is absent in this case as the communication among blocks is confined to its immediate neighbors. From all the bundled data schemes studied in this chapter, the scheme by De Clercq *et al.* described in section 2.3.1 (C) seems more appropriate because it offers the highest throughput.

2.7 Summary

In this chapter, the problem of synchronization failure with GALS systems is analyzed and methods to avoid it are discussed. Different asynchronous protocols are discussed and asynchronous wrappers were classified based on the handshaking protocol they use. Advantages and disadvantages of bundled data protocol and delay insensitive protocol are discussed. A novel asynchronous wrapper scheme using 1-of-4 delay insensitive based on single-track handshaking in also introduced and simulation results are presented.

CHAPTER 3

Clock Frequency Estimation in GALS

Synchronous digital designs use a clock signal to define a time reference for correct movement of data within the system. In processors, the clock frequency determines the rate of data processing. In I/O and memory buses, the clock frequency determines the rate of data transmission. Consequently digital systems designers strive to maximize the clock frequency in order to achieve high system performance [3]. There are three main parameters that affect the clock frequency in a GALS system: clock skew, clock signal degradation and delay in asynchronous handshaking.

We studied a simplified GALS system, which is essentially an array of identical, pre-designed processors with linear communication links as shown in Figure 3.1. The effects of each of the above mentioned parameters that affect the clock frequency have been elaborated in the following sections.

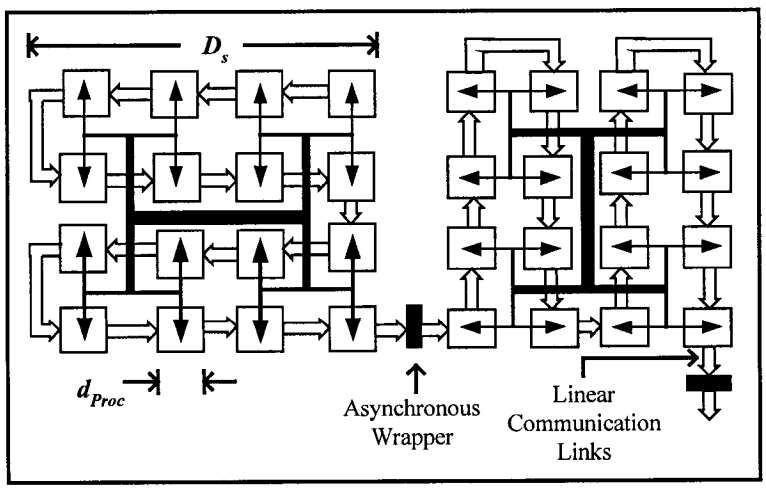


Figure 3.1. GALS array of processors with linear communication links

3.1 Clock Skew

Clock skew can severely limit the performance and may create race conditions in synchronous digital systems. Figure 3.2 shows two sequentially adjacent registers separated by combinational logic. Clock skew is defined as difference in clock signal arrival times between two sequentially adjacent registers as illustrated in Figure 3.3.

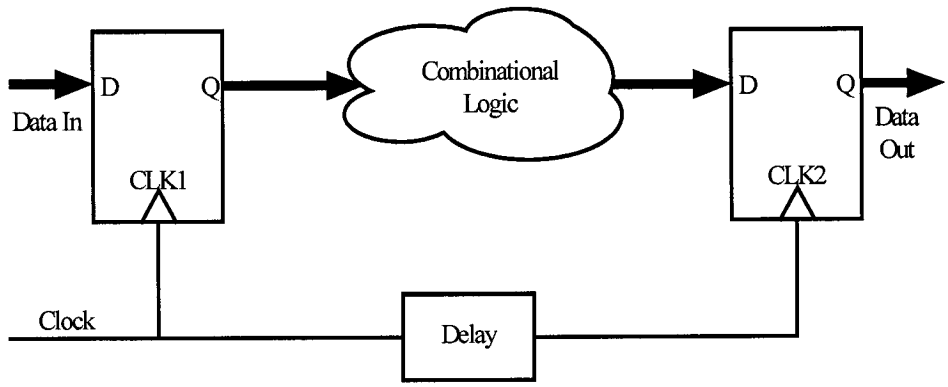


Figure 3.2. Sequentially adjacent clocked registers in a synchronous system

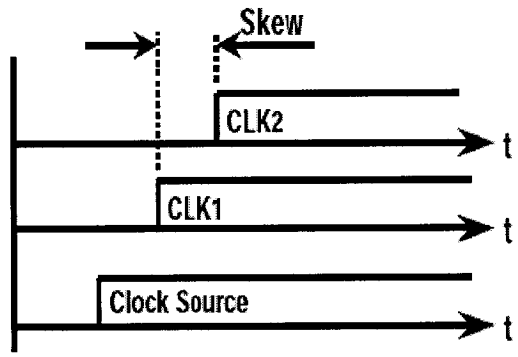


Figure 3.3. Clock skew between the two points, CLK1 and CLK2, in Fig. 3.1 [17]

Due to variations of the arrival times of the clock signal, one needs to add the worst case skew to the path delays and consequently to the machine cycle time in order to guarantee that the circuit

will function properly. Controlling the skew becomes harder as the circuits get faster, the chips become larger, and the minimum feature size is scaled down. For high-speed systems, a large amount of design effort is spent to minimize the clock skew and prevent it from becoming a significant portion of the cycle time [3].

The clock skew is usually composed of the following parts; mismatch in RC delays along the various paths of distributed clock wires, disparity in the clock buffer delays along the path, difference in capacitive load, and mismatch due to process parameter variations.

3.1.1 Skew due to mismatch in *RC* delays:

The interconnects used for clock signal distribution have intrinsic resistances and capacitances. Clock skew is caused by difference in *RC* time constants of different clock paths (such as the lines connecting point clock driver to points1 and 2 in Figure 3.4).

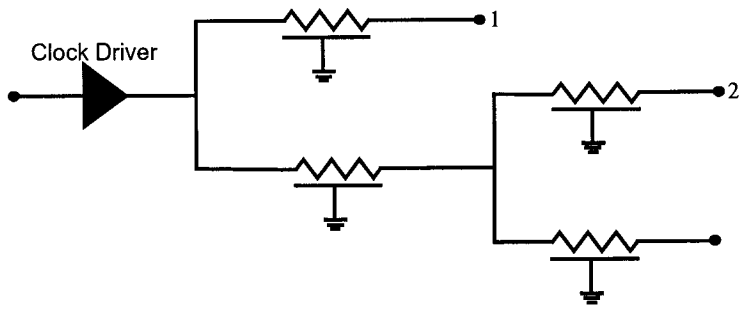


Figure 3.4. Clock distribution tree with *RC* delays [3]

The clock skew can be minimized by distributing the clock signal in such a way that the interconnections carrying the clock signal to the functional sub-blocks are of equal length. This can be achieved with hierarchy of planar symmetric H-tree as shown in Figure 3.1.

The H-tree delays the clock signal equally before arriving at the sub-block and therefore they are perfectly synchronous. The conductor widths in H-tree structure are designed to decrease progressively as the signal propagates to lower levels of hierarchy (also called binary H-tree). This strategy minimizes reflections of the high-speed clock signals at the branching points.

If the sub-blocks in the system are identical, as is the case in an array of processors, skew due to mismatch in *RC* delays can be effectively minimized with H-tree clock distribution scheme. Sub-blocks with large differences in size and capacitive load require a clock routing algorithm to achieve zero skew. However, clock skew due to process parameter variations as explained in next sub-section can still not be avoided.

3.1.2 Skew due to process parameters variations

The delay of each of the elements of a clock path is highly sensitive to geometric, material, and environmental variations that exist in the implementation technology [7]. With clock speed now approaching multi-GigaHertz, a skew of few Pico-seconds can significantly affect performance. Zarkesh-Ha [17] characterized the clock skew components resulting from process parameter variations and provided closed-form model for each clock skew component, which consists of both process and circuit parameter fluctuations. His work has been used as a basis to analyze the effect of clock skew components in GALS in the following sub-sections.

(A) Device parameter variations

Active devices in the clock paths are the primary source of clock skew in a well-balanced clock network because the active device characteristics vary much more greatly than the passive device characteristics. Variations in transistor parameters such as threshold voltage (ΔV_T), gate oxide thickness (Δt_{ox}), and effective channel length (ΔL_{eff}) can produce significant skew. The expressions for these clock skew components, as mentioned in [17] are summarized in Table 3.1.

Table 3.1. Skew components of device parameter variations

Threshold Voltage Fluctuation	$T_{CSK}(V_T)=0.7\,R_{tr}\,C_L\left(\frac{V_T}{V_{DD}-V_T}\right)\left(\frac{\Delta V_T}{V_T}\right)$ (3.1)
Transistor Channel Length Tolerance	$T_{CSK}(L_{eff})=0.7\,R_{tr}\,C_L\left(\frac{\Delta L_{eff}}{V_T}\right)$ (3.2)
Gate Oxide Thickness Tolerance	$T_{CSK}(t_{ox})=0.7\,R_{tr}\,C_L\left(\frac{\Delta t_{ox}}{t_{ox}}\right)$ (3.3)

Where R_{tr} is the output resistance of clock driver within each processor, C_L is the total wiring and register input capacitance within the processor, and V_{DD} and V_T are supply and transistor threshold voltage, respectively.

Skew due to device parameter variations does not change with partitioning because the variations in transistor parameters are technology dependent and do not scale with the sub-block size. If the system consists of predesigned and identical processors, each processor will have equal C_L and will require identical clock drivers. So, R_{tr} and C_L are also not affected by varying number of processors in each sub-block in this case.

(B) Interconnect parameter variations

Variations in interconnect width (ΔW_{int}), thickness (ΔH_{int}) and interlevel dielectric (*ILD*) thickness (ΔT_{ILD}) cause skew by changing *RC* time constants of different clock paths. With shrinking geometries of interconnects, these variations become much more important. The expressions for important clock skew components are summarized in Table 3.2.

Table 3.2. Skew components of interconnect parameter variations

ILD Thickness Variation	$T_{CSK}(T_{ILD}) = 0.4(R_{int} C_{int}) D_S^2 \left(1 - \frac{1}{2^{n/2}}\right)\left(\frac{\Delta T_{ILD}}{T_{ILD}}\right)$ (3.4)
Wire Thickness Variations	$T_{CSK}(H_{int}) = 0.4(R_{int} C_{int}) D_S^2 \left(1 - \frac{1}{2^{n/2}}\right)\left(\frac{\Delta H_{int}}{H_{int}}\right)$ (3.5)

where R_{int} and C_{int} are the distributed resistance and capacitance per unit length of interconnects, D_s is the die size of each sub-block, and n is the number of H-tree levels. The skew due to these components changes with number of partitions because it is proportional to the square of the size of each sub-block.

(C) System parameter variations

System-level variations such as power-supply voltage fluctuation (*IR* drop, ΔV_{DD}), temperature variations (ΔT), and non-uniform distribution of clocked registers (clock driver load mismatch, ΔC_L) may create significant clock skew. The expressions for these clock skew components are summarized in Table 3.3.

Table 3.3. Skew components of system parameter variations

IR Drop	$T_{CSK}(V_{DD})=0.7 R_{tr} C_L \left(\frac{V_T}{V_{DD}-V_T}\right)\left(\frac{\Delta V_{DD}}{V_{DD}}\right)$ (3.6)
Non-uniform Register Distribution	$T_{CSK}(C_L)=0.7 R_{tr} C_L \left(\frac{\Delta C_L}{C_L}\right)$ (3.7)
Temperature Gradient	$T_{CSK}(T)=0.7 R_{tr} C_L \left(\frac{E_g/q + V_T}{V_{DD}-V_T}\right)\left(\frac{\Delta T}{T}\right)$ (3.8)

Where, Eg/q=1.12 V is the energy gap of *Si* in volts and T is the temperature in degrees Kelvin.

IR drop will not change much with partitioning the system into more number of sub-blocks because the power distribution still requires global wires. In a system of identical processors as shown in Figure 3.1, non-uniform register distribution has insignificant contribution since all the processors present the same load. Temperature gradient will be less steep for smaller size of the sub-block.

(D) Internal clock skew

The internal clock skew is basically the internal wire delay within the processor from the clock driver to the registers. The wiring delay inside the processor is computed using Equation 3.9,

$$T_{CSK}(Internal) = 0.4 (R_{int} C_{int}) l^2 + \left(\frac{\sqrt{\varepsilon_{OX}}}{C_{Light}}\right) l \quad (3.9)$$

where, ε_{OX} is the relative dielectric constant of the *ILD* material, C_{Light} is the speed of light in free space, and length of wire, l, is the distance from center to the corner of the processor as shown in Figure 3.5.

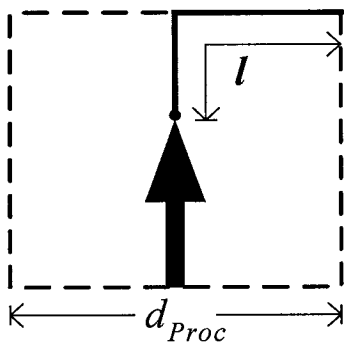


Figure 3.5. Longest clock path within a processor [17]

3.2 Clock Signal Degradation

In the simplest case, an interconnect can be modeled as an *RC* low-pass filter. The clock signal is degraded as it passes through the interconnect because of its limited bandwidth. If the interconnections have large *RC* constants, the waveform will have long rise times, and a high frequency clock signal will not be possible. Equation 3.10 gives the lumped model of interconnect bandwidth of an H-tree based clock distribution network,

$$f_{-3dB} = \frac{1}{2\pi(R_{int}\, C_{int}).\, D_s^{\,2}.\left(1-\frac{1}{2^{n/2}}\right)^2} \tag{3.10}$$

Where R_{int} and C_{int} are the distributed resistance and capacitance per unit length, D_s is die size of each sub-block, and n is number of H-tree levels.

The lumped *RC* model of Equation 3.10 underestimates the actual bandwidth of a distributed *RC* line. However, in the most typical cases, it gives a good first-order approximation for the interconnect bandwidth.

3.2.1 Effect of inserting repeaters

If the *RC* constant of the clock tree is not small enough, high-speed repeaters may be employed to increase the interconnect bandwidth. These distributed repeaters serve the double function of amplifying the clock signal degraded by the distributed interconnect impedances and isolating the local clock nets from upstream load impedances. We can estimate the bandwidth of un-repeated wire by asking how long we must wait between successive transitions on a wire. Repeated wires offer substantially increased bandwidth because after sending one signal down a wire, we only need wait until that signal fully transitions on the first repeater segment before we send the next signal. Repeater insertion also decreases the propagation delay of the wire, which is given by Equation 3.11.

$$t_p = 0.38\ R_{int}\ C_{int} \left(\frac{L_{int}}{M} \right)^2 M + (M - 1) t_{Re\ p} \tag{3.11}$$

where, L_{int} is length of interconnect, M is number of repeated segments, and t_{Rep} is the propagation delay of each repeater.

However, repeaters used in a clock distribution network may introduce considerable clock skew within a well-balanced clock network since the active device characteristics vary much more greatly than the passive device characteristics. This undesirable effect of inserting active repeaters is, however, very well offset by improvements gained in bandwidth if the interconnect has large *RC* constant. Therefore, although passive H-tree structures provide significantly less variation in clock skew, this advantage must be weighed against the increased signal degradation.

Usually, ideal repeater placement is a complicated issue involving many parameters that one needs to take into consideration in large VLSI designs. In a simplified case, for a clock tree with equal widths, source-end-terminated repeaters between H-tree hierarchy levels, as illustrated in Figure 3.6, can be used for matching at the branching points to avoid reflections. If binary H-tree is used to avoid reflections, repeaters can be inserted at equal distances as shown in Figure 3.7 because the resulting network is equivalent to a uniformly distributed *RC* line.

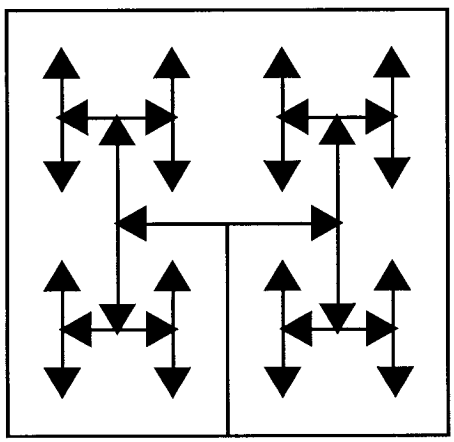


Figure 3.6. Repeaters between H-Tree hierarchy levels

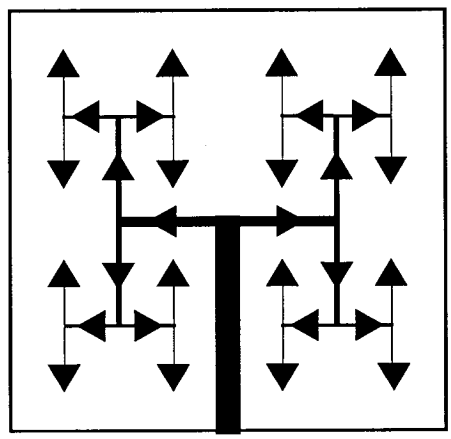


Figure 3.7. Repeaters with binary H-tree

3.3 Delay in Asynchronous Handshaking

As mentioned in section 2.2, in GALS systems, the interfacing between a functional sub-block and its self-timed environment is handled by the asynchronous wrappers to avoid the synchronization failures. These Asynchronous wrappers may introduce significant performance penalty. With pausible clock scheme, the clock is stretched whenever a clock-data conflict is detected. The probability of conflict increases with the frequency. At high frequencies, the clock may be stretched on every data transaction if the delay in handshaking in the asynchronous wrappers is larger than the low phase of clock period. Also, if there is large difference in clock frequencies of two communicating blocks, the faster clock will be paused for the time it is waiting the slower block to respond. The effective clock frequency also depends on the rate of communication among locally synchronous sub-blocks. The stretching of the clock beyond normal clock period brings down effective frequency of sub-blocks.

The duration of clock stretching is also very sensitive to the type of communication mechanism used, like multiplexed buses, star networks, rings, or dedicated point-to-point links. For example, shared bus architecture with central arbiter will be highly congested if there is large number of communicating sub-blocks. In this case, a sender's clock will be paused for long time waiting for a receiver to respond and gaining access to the shared communication resources. Predicting average period of clock stretching becomes probabilistic in this situation. However, for the simplified system with linear array of identical processors as shown in Figure 3.1, period of clock stretching is deterministic and can be easily computed.

3.4 Estimation of Effective Sub-block Clock Frequency

In GALS systems, the clock frequency can be evaluated by considering the clock skew, the interconnect bandwidth, and the clock stretching with partitioning. If clock skew is the factor that puts a limit on maximum frequency, clock frequency can be estimated assuming the skew to be 5 to 10 percent of total clock period.

The stretching of the clock beyond normal clock period brings down effective frequency of sub-blocks. The effective frequency is given by Equation 3.12,

$$f_{Eff} = \frac{1}{(0.5T_{Clk}) + T_{Strch}} \tag{3.12}$$

where, T_{Clk} is total clock period without stretching and T_{Strch} is stretched total negative half period of the clock cycle as shown in Figure 3.8.

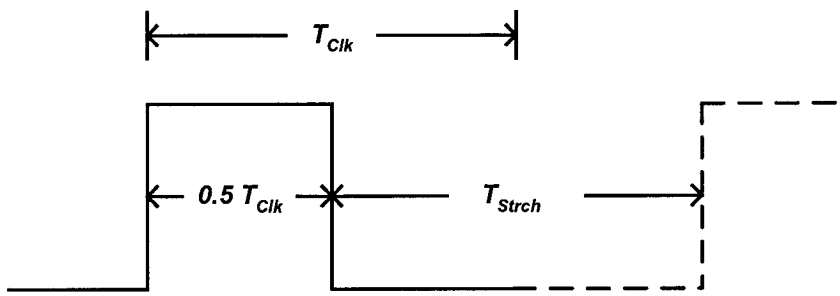


Figure 3.8. Clock period definitions

3.5 Summary

In this chapter, different parameters that affect the clock frequency of a GALS system have been elaborated. Section 3.1 explains various components of the clock skew such as mismatch in the *RC* delays, process parameter variations, and the internal clock skew. The closed form models of clock skew introduced by process parameter variations, as provided in [17], have also been summarized. It includes device, interconnect, and system parameter variations. Section 3.2 explains clock signal degradation due to limited bandwidth of interconnects. The effect of inserting repeaters in clock network has also been investigated. Section 3.3 discusses the delay in asynchronous wrappers and pausing of clock. Section 3.4 gives an equation to evaluate effective clock frequency considering all the parameters.

CHAPTER 4

Power Estimation and Optimum Partitioning in GALS

Power consumption in large high-performance synchronous VLSI designs has become designer's primary concern with increasing demand for portable electronic devices. Distribution of low-skew global clock signals, now approaching GigaHertz range, is often the single largest source of power consumption. Therefore elimination of the clock networks may result in substantial power savings. GALS designs offers an opportunity of considerable power saving by eliminating the global clock distribution networks. However, the GALS systems require extra hardware in terms of the asynchronous wrappers and the local clock generation that consumes additional power.

The power consumption in GALS has three main components:

(1) Power due to the clock networks within the synchronous sub-blocks,

(2) Power due to the asynchronous wrapper, and

(3) Power due to the local clock generation.

In this chapter, we provide power estimation models for GALS array of processors with three different configurations. We also provide models for evaluating optimum number of sub-blocks that leads to the least power consumption at a constant frequency. The power consumption of the computational logic of a GALS system is assumed to be identical for different partitions and thus irrelevant for the comparison purpose.

4.1 Power in Passive Clock Distribution Networks

There are three main components that affect power consumption in a passive clock network; the interconnect capacitance, the capacitive load of clocked flip-flops, and the clock drivers within each processor. Based on the model in [17], the power in an H-tree type clock network can be approximated by Equation 4.1.

$$P_{Clk} \approx f_C V^2_{DD} \left(C_{Leaf} . D . \left(2^{n+1} + \sqrt{N_{FF}} \right) + 2^n C_{Driver} + N_{FF} C_{FF} \right) \quad (4.1)$$

where, f_C is the effective clock frequency that includes the probable clock stretching, C_{Leaf} is the interconnect capacitance per unit length of clock network assuming the network has uniform width equal to the width of the leaf of H-tree, n is the number of H-tree levels, D is the die size, N_{FF} is the number of clocked flip-flops, C_{FF} is the input capacitance of flip-flops, and C_{Driver} is the input capacitance of clock driver, and V_{DD} is the supply voltage.

Figure 4.1 shows a GALS system as an array of 64 identical processors divided into 4 locally synchronous sub-blocks.

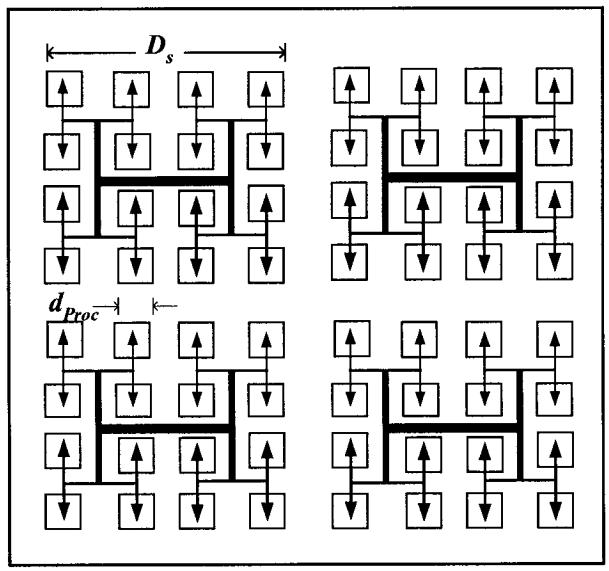


Figure 4.1. GALS array of processors ($S = 4$ and $P = 64$)

For the design of Figure 4.1, the relation between the number of H-tree levels n, the number of sub-blocks S, and the number of processors P, is given by Equation 4.2.

$$2^n = \frac{P}{S} \tag{4.2}$$

where, S is an integer that results in a symmetrical H-tree within the sub-blocks.

Therefore, Equation 4.1 can be modified as shown in Equation 4.3, to express the clock power in terms of the number of sub-blocks with each partitioning of the GALS array shown in Figure 4.1.

$$P_{Clk} \approx \left\{ \left[\left(C_{Leaf} \sqrt{\frac{P}{S}} d_{Proc} \right) \left(2\frac{P}{S} + \sqrt{\frac{P}{S} FF_{Proc}} \right) S \right] + [PC_{Driver}] + [P\, FF_{Proc} C_{FF}] \right\} f_C\, V^2{}_{DD} \tag{4.3}$$

Where, FF_{Proc} is the number of clocked flip-flops within each processor. The first term, which represents total wiring capacitance, varies with the number of partitions. Varying the number of partitions does not affect the power due to the clock driver and the clocked flip-flops.

4.2 Power in Asynchronous Wrappers

Asynchronous wrappers add extra logic and wires. There are four major factors that contribute to the power consumption asynchronous wrappers [1]:

(1) The number of logic gates required to implement the handshaking protocol.

(2) The frequency with which sub-blocks communicate with other sub-blocks, the worst case being that they communicate in every clock cycle.

(3) Number of sub-blocks that participate in communication, the worst case being every sub-block communicates with every other.

(4) The length and number of wires for control signals.

The power consumed in wrappers is given by Equation 4.4 [1],

$$P_{Async} = \left[\sum_{b}^{B} \sum_{i}^{X_b} \left(n_{Sig} C_w l_{b,i} + n_{Reg} C_{Reg} \right) \right] f_{b,i} V^2{}_{DD} \qquad (4.4)$$

where, B is the set of communicating sub-block pairs, X_b is the set of communication instances for a particular pair, b, of communicating sub-blocks, n_{Sig} is the number of signals in the communication protocol, C_w is the wire capacitance per unit length, $l_{b,i}$ is the wire length for a communication signal for a particular pair b, on communication instances i, n_{Reg} is the number of registers driven by the control signals and C_{Reg} is the register capacitance, $f_{b,i}$ is the frequency at which a sub-blocks pair b communicates on instance i.

In the following sub-sections, we adapt Equation 4.4 to different configurations of arrays of identical processors.

4.2.1 Case of a linear array of processors

Figure 3.1 shows a GALS system with linear arrays of identical processors. For this system, the number of asynchronous wrappers required with each partition is given by Equation 4.5.

$$N_{Async} = (S - 1) \tag{4.5}$$

Therefore, assuming the worst case where the sub-blocks communicate every clock cycle and all the sub-blocks have equal clock frequency, Equation 4.4 can be modified as shown in Equation 4.6, to express the power consumed in the asynchronous wrappers for GALS array of Figure 3.1.

$$P_{Async} = (S-1)\left(n_{Sig} C_w l_w + n_{Reg} C_{Reg}\right) f_C V^2{}_{DD} \tag{4.6}$$

where, l_w is the wire length for communication signals implementing the handshake protocol. With increase in the number of partitions, S will increase and so will P_{Async}.

4.2.2 Case of an array with unidirectional links

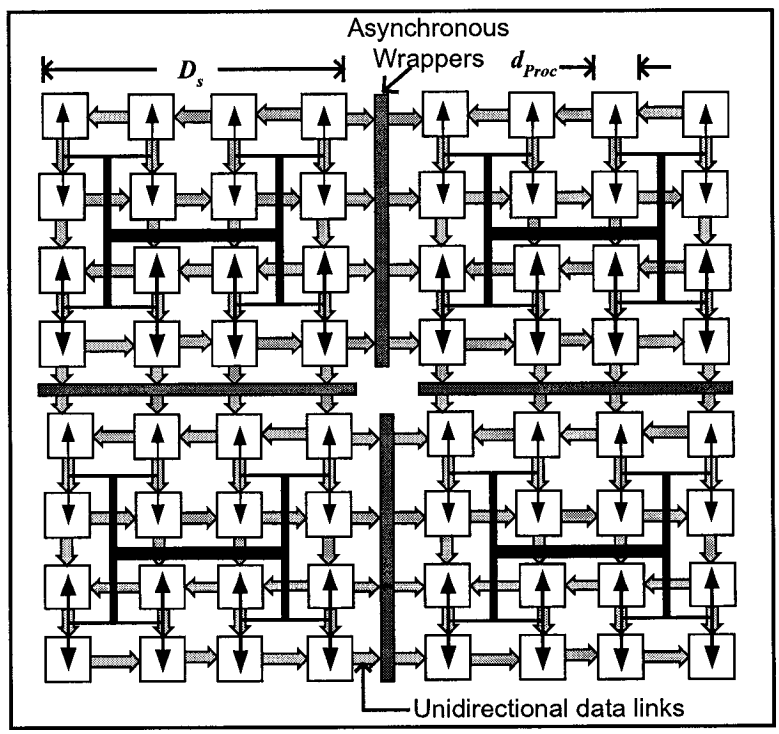


Figure 4.2. Square arrays of processors with unidirectional communication links

Figure 4.2 shows a GALS array of identical processors with unidirectional communication links among them. For this configuration, the number of asynchronous wrappers required with each partitioning is given by Equation 4.7.

$$N_{Async} = \frac{P.\sqrt{S}}{2^K} - 2\sqrt{P} \tag{4.7}$$

where, K is a constant that depends on the size of the array such that,

$$P = 2^{2(K+1)} \tag{4.8}$$

Assuming the worst case where the sub-blocks communicate every clock cycle and all the sub-blocks have equal clock frequencies, the power consumed in the asynchronous wrappers is given by Equation 4.9.

$$P_{Async} = \left(\frac{P.\sqrt{S}}{2^K} - 2\sqrt{P} \right) \left(n_{Sig} C_w l_w + n_{Reg} C_{Reg} \right) f_C V^2{}_{DD} \tag{4.9}$$

Equation 4.9 does not apply for $S=1$ because a fully synchronous system does not require any asynchronous wrapper.

4.2.3 Case of an arrays with bidirectional links

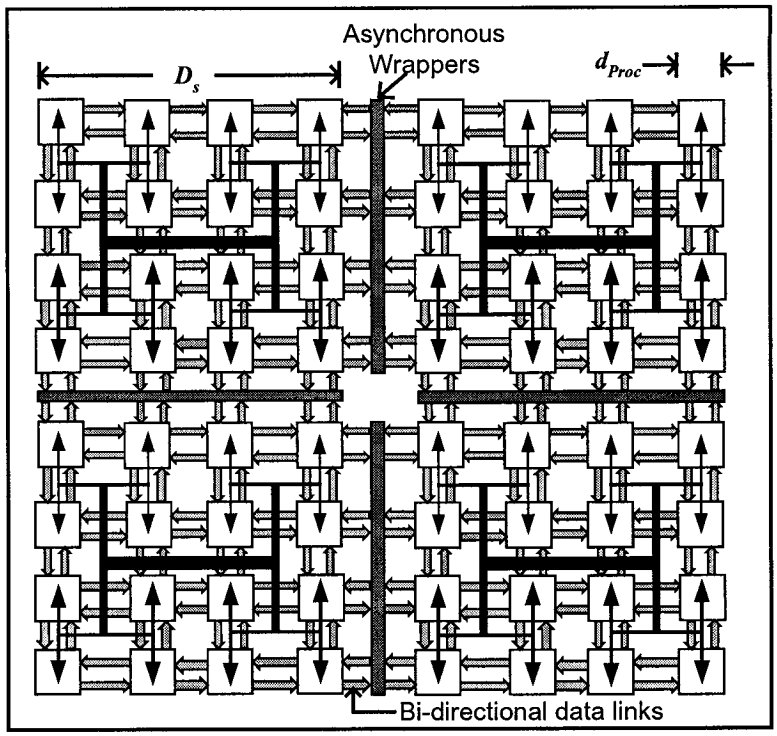


Figure 4.3. Square arrays of processors with bidirectional communication links

Figure 4.2 shows a GALS array of identical processors with bidirectional communication links among them. For this configuration, the number of asynchronous wrappers required with each partitioning is given by Equation 4.10.

$$N_{Async} = \frac{2 . P . \sqrt{S}}{2^K} - 4\sqrt{P} \tag{4.10}$$

Assuming the worst case where the sub-blocks communicate every clock cycle and all the sub-blocks have equal clock frequencies, the power consumed in the asynchronous wrappers is given by Equation 4.11.

$$P_{Async} = \left(\frac{2 . P . \sqrt{S}}{2^K} - 4\sqrt{P} \right) \left(n_{Sig} C_w l_w + n_{Reg} C_{Reg} \right) f_C V^2_{DD} \qquad (4.11)$$

Equation 4.11 does not apply for $S=1$ because a fully synchronous system does not require any asynchronous wrappers.

4.3 Power in Local Clock Generation

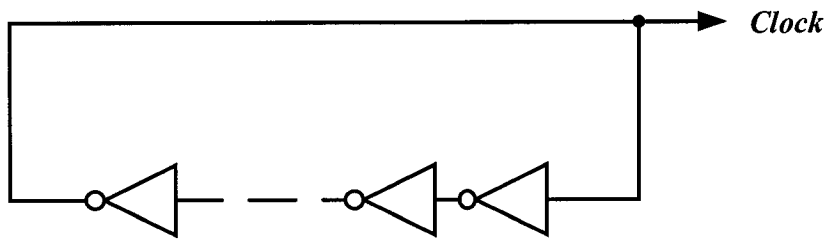


Figure 4.4. Ring oscillator

In the GALS architectures, the local clock generators are required for the sub-blocks. The simplest form of On-Chip oscillator is an odd number of inverters connected in a circular chain as shown in Figure 4.4. Such a circuit has no stable operation point and will therefore oscillate. The ring oscillator frequency is determined by the propagation time through the chain of inverters.

This scheme is particularly suitable for asynchronous wrappers with plausible clocking because such a clock is simple to stop by gating the clock pulse. Inevitably the performance of a ring oscillator is dependent on manufacturing tolerance and environmental conditions. However, On-Chip ring oscillator has an advantage that its frequency drift due to changing temperature or supply voltage closely tracks the delay of logic components. More precise schemes for On-Chip clock generation have been proposed in [26] and [19] but are more expensive in terms of power

consumption and area. Ring oscillators are low cost solutions and their power consumption for a GALS architecture can be estimated by Equation 4.12,

$$P_{Osc} = \sum_{b}^{B} N_{Invb} \cdot C_{Inv} \cdot f_b \cdot V^2_{DD} \tag{4.12}$$

where, B is the set of sub-blocks, C_{Inv} is the capacitive load due to one inverter, N_{Invb} is the number of inverters used by the ring oscillator in sub-block number b and f_b is its frequency of operation.

For all three configurations of the array discussed in section 4.2, each sub-block requires its own clock generator irrespective of the scheme of communication adapted among the sub-blocks. Since all sub-blocks are identical in size, if we assume all the sub-blocks have equal clock frequency, they will have identical clock generators. The power consumed in the clock generators is given by Equation 4.13 in this case.

$$P_{Osc} = S \cdot N_{Inv} \cdot C_{Inv} \cdot f_C \cdot V^2_{DD} \tag{4.13}$$

where, N_{Inv} is the number of inverters required in each ring oscillator. Equation 4.12 is not valid for $S = 1$ because it means a fully synchronous system which does not require any local clock generator.

4.4 Optimum Partitioning Methodology for GALS Array with Passive Clock Networks

Total power consumption in GALS (P_{GALS}), given by Equation 4.14, is composed of the power due to the clock distribution networks, the power due to the asynchronous wrappers, and the power due to the local clock generation.

$$P_{GALS} = P_{Clk} + P_{Async} + P_{Osc} \tag{4.14}$$

With the partitioning of a GALS designs, P_{GALS} is a tradeoff between the power saved in global clock and the power overhead in the asynchronous wrappers and the clock generation. Sections 4.1, 4.2, and 4.3 show all the components of P_{GALS} as functions of number sub-blocks (*i.e.* number of partitions) for different array configurations. To evaluate the optimum number of sub-blocks that gives the least power consumption for a constant frequency, we can take the derivative of P_{GALS} with respect to S and equate it to zero.

Since the equations for P_{Clk} and P_{Osc} are the same for all three array configurations, we will evaluate dP_{Clk}/dS and dP_{Osc}/dS first and later apply it to evaluate the optimum partitioning for different configurations.

The derivative of the power consumption in the clock networks with respect to S is computed from (4.3) as,

$$\frac{d\left(P_{Clk}\right)}{dS} = \left(C_{Leaf} \cdot d_{Proc} \cdot 2 \cdot P^{1.5} f_C \cdot V^2_{DD}\right)\left(-\frac{1}{2} S^{-1.5}\right) \tag{4.15}$$

Since the parameters in the first term do not vary with the partitioning, we can replace them by a constant to simplify Equation 4.15 as,

$$\frac{d\,(P_{Clk})}{dS} = -A_1 . S^{-1.5} \tag{4.16}$$

where, A_1 is,

$$A_1 = C_{Leaf} . d_{Proc} . P^{1.5} . f_C . V^2_{DD} \tag{4.17}$$

The derivative of the power in clock generation with respect to S is computed from (4.13) as,

$$\frac{d\,P_{Osc\_1}}{d\,S} = N_{Inv} . C_{Inv} . f_C . V^2_{DD} \tag{4.18}$$

All the terms in this derivative are independent of the number of sub-blocks S and they represent the power consumed in one ring oscillator. Therefore, these terms can be replaced by a constant as shown in Equation 4.19.

$$\frac{d\,P_{Osc}}{d\,S} = P_{Osc\_1} \tag{4.19}$$

where, P_{Osc_1} is a constant that represents the power consumed in one ring oscillator.

The number of wrappers required for each partition changes with the scheme of communication among processors. The following sub-sections give the optimum sub-blocks for different array configurations of section 4.2.

4.4.1 Case of an Array with Linear Communication links

For the GALS array with linear communication links shown in Figure 3.1, dP_{Async} / dS is computed from (4.6) as,

$$\frac{d\,P_{Async}}{d\,S} = \left(n_{Sig}\,C_w\,l_w + n_{Reg}\,C_{Reg}\right) f_C\,V^2{}_{DD} \tag{4.20}$$

All the terms in this derivative are independent of the number of sub-blocks S and they represent the power consumed in one asynchronous wrapper. Therefore these terms can be replaced by a constant as shown in Equation 4.21.

$$\frac{d\,P_{Async}}{d\,S} = P_{Async\_1} \tag{4.21}$$

where, P_{Async_1} is a constant that represents the power consumed in one asynchronous wrapper.

Therefore, the derivative of the total power with respect to S can be evaluated from (4.14) as,

$$\frac{d\,(P_{GALS})}{dS} = -A_1 . S^{-1.5} + P_{Async\_1} + P_{Osc\_1} = 0 \tag{4.22}$$

As a consequence, the optimum number of sub-blocks for the array with linear communication links is given by,

$$S_{Opt\_Linear} = \left[\frac{A_1}{P_{Async\_1} + P_{Osc\_1}} \right]^{\frac{1}{1.5}} \quad (4.23)$$

S_{Opt_Linear} gives the least P_{GALS} for a constant clock frequency.

4.4.2 Case of a Square Array with Unidirectional Links

For the GALS array shown in Figure 4.2, dP_{Async} / dS is computed from (4.9) as,

$$\frac{d\,P_{Async}}{d\,S} = \frac{P \cdot P_{Async\_1}}{2^{K+1}} . S^{-0.5} \quad (4.24)$$

Evaluating dP_{GALS} / dS gives the polynomial shown in (4.25).

$$\left(P_{Osc\_1}\right).S^{1.5} + \left(\frac{P \cdot P_{Async\_1}}{2^{K+1}}\right).S - A_1 = 0 \quad (4.25)$$

Substituting the values of the constants and solving Equation 4.25 for S will give the optimum number of sub-blocks (S_{Opt_Uni}) that leads to the minimum power at a constant frequency for the GALS array shown in Figure 4.2 with any size.

4.4.3 Case of a Square Array with Bidirectional Links

For the GALS array shown in Figure 4.3, dP_{Async} / dS is computed from (4.11) as,

$$\frac{d\ P_{Async}}{d\ S} = \frac{P\ .\ P_{Async\ \_1}}{2^{K}}.S^{\ -0.5} \tag{4.26}$$

Evaluating dP_{GALS} / dS gives the polynomial shown in (4.27).

$$\left(P_{Osc\ \_1}\right).S^{\ 1.5} + \left(\frac{P\ .\ P_{Async\ \_1}}{2^{K}}\right).S\ -\ A_1\ =\ 0 \tag{4.27}$$

Substituting the values of the constants and solving Equation 4.27 for S gives the optimum number of sub-blocks (S_{Opt_Bi}) that leads to the minimum power at a constant frequency for the GALS array shown in Figure 4.3 with any size.

4.5 Power in Clock Repeaters

If the *RC* constant of the clock tree is not small enough, high-speed repeaters may be employed to increase interconnect bandwidth as explained in section in 3.2.1. These active repeaters in clock network tend to be area- and power-hungry.

Figure 4.5 shows clock path from root to leaf in a binary H-tree. The numbers in the circles represent the H-tree levels. Since the clock network is equivalent to uniformly distributed *RC* line, we can insert identical repeaters at equal distances dividing the line into sections with equal *RC* constants for each section. The optimal number of sections in this case is given by (4.28).

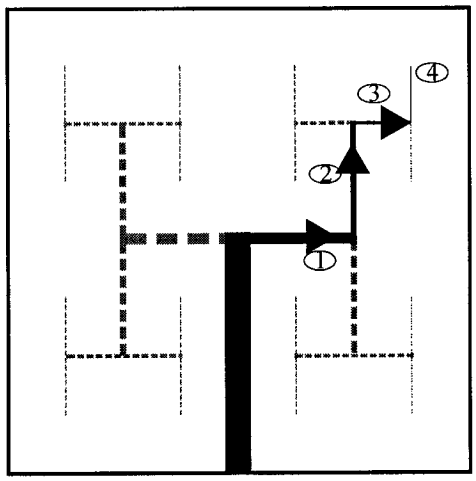


Figure 4.5. Clock path from root to leaf in a binary H-Tree

$$M = \left(\sqrt{\frac{P}{S}} - 1 \right) . d_{Proc} \cdot \sqrt{\frac{0.38 \; R_{Leaf} \; C_{Leaf}}{t_{Rep}}} \tag{4.28}$$

where, R_{Leaf} and C_{Leaf} are the interconnect capacitance and resistances per unit length of clock network assuming the network has uniform width equal to the width of the leaf of H-tree, and t_{Rep} is the propagation delay of a repeater. If M is a fractional number, it is rounded off to the next higher integer.

Neglecting the clock drivers at the source and within each processor, total number of repeaters in each clock path is given by Equation 4.29.

$$N_{Rep \; / \; Path} = M - 1 \tag{4.29}$$

Depending on the driving capability of each repeater and the length of the H-tree levels, we may need one or more repeaters in a particular H-tree level. If repeaters are large and the length of the H-tree level is small, we may not need repeaters at all in that H-tree level. Considering this, the number of repeaters in each sub-block is given by (4.30).

$$N_{Rep/S} = \sum_{i}^{n} K_i \cdot 2^i \tag{4.30}$$

where, n is the number of H-tree levels in a sub-block and K_i is the number of repeaters in i^{th} H-tree level.

Therefore, the power consumed in the repeaters in a GALS design is given by equation 4.31.

$$P_{Rep} = S \cdot N_{Rep/S} \cdot C_{Rep} \cdot f_C \cdot V_{DD}^2 \tag{4.31}$$

where, C_{Rep} is input capacitance of a repeater.

4.6 Optimum Partitioning Methodology for GALS with Active Clock Networks

Clock repeaters occupy large are and consume significant power in synchronous system. With increasing partitions in GALS system, each locally synchronous sub-block requires fewer repeaters. If the size of sub-block (D_S) is reduced significantly, we may not need repeaters. Since $N_{Rep/S}$ (and hence P_{Rep}) is not a continuous function of S, we suggest algorithm in Figure 4.6 for evaluating optimum partition for user-defined frequency.

For (S=1 to P);

Evaluate M;

If (M > 1)

$P_{GALS}(S) = P_{Clk} + P_{Rep} + P_{Async} + P_{Osc}$;

End if;

If (M ≤ 1)

$P_{GALS}(S) = P_{Clk} + P_{Async} + P_{Osc}$;

End if;

End for;

$P_{Opt} = min\ \{P_{GALS}(1), P_{GALS}(2) \ldots P_{GALS}(P)\}$;

S_{Opt} = *S corresponding to* P_{Opt}

Figure 4.6. Algorithm for optimum partitioning of GALS with clock repeaters

The same algorithm applies to all configurations of processor arrays since change the communication scheme among processors does not affect number or size of repeaters in each sub-block.

4.7 Summary

In this chapter, different components of the power consumption in the GALS processor array have been investigated. Three different components of the clock power have been considered and

models for evaluating optimum partitioning have been proposed. Section 4.1 explains the power consumed in the passive clock distribution networks. Section 4.2 discusses the power consumed in the asynchronous wrappers and provides models for three different array configurations. Section 4.3 details the power consumed in local clock generation. Section 4.4 provides the models used to predict the optimum number of sub-blocks. Section 4.5 discusses the power consumed in the clock repeaters. Section 4.6 introduces an algorithm to evaluate the optimum number of sub-blocks that give least power consumption at a constant frequency and with active clock networks.

CHAPTER 5

Experimental Setup and Results

To verify the concepts introduced in the previous sections, we studied a hypothetical 16x16 array of identical processors. The size of each processor was set to 0.2 cm resulting into a 3.2 cm die. The number of clocked flip-flops within each processor was set to 200. The H-tree based clock network was assumed to be routed by the fifth metal level shielded with the fourth and sixth metal levels of 0.18μm CMOS technology. Three different communication schemes were considered for the array: (1) linear communication links among processors (Figure 3.1), (2) unidirectional links among neighboring processors (Figure 4.2), and (3) bidirectional links among neighboring processors (Figure 4.3). To evaluate the GALS trade-offs, the designe was partitioned in different number of sub-blocks, starting with a fully synchronous system, as shown in Table 5.1.

Table 5.1. Partitioning of the GALS processor array

Number of Sub-Blocks S	**Processors/Sub-block** P/S	**H-Tree levels within each Sub-block** n
1	256	8
2	128	7
4	64	6
8	32	5
16	16	4
32	8	3
64	4	2
126	2	1
256	1	0

Table 5.2 summarizes the technology parameters and Table 5.3 shows all the process and the design parameters.

Table 5.2. Technology parameters

Technology	0.18 μm
Supply voltage, V_{DD}	1.8 V
Threshold voltage, V_T	0.45 V
Gate oxide thickness, t_{OX}	40.8 ^{o}A
Thickness of ILD, T_{ILD}	2000 ^{o}A
Dielectric constant for oxide, ε_{OX}	0.35 pF / cm
Sheet resistance for metal layer, R_{sh}	0.078 Ω

Table 5.3. Process and design parameters

Die size, D	3.2 cm
Number of processors, P	256
Size of a processor, d_{Proc}	0.2 cm
Number of F/F in a processor, FF_{Proc}	200
Input capacitance of a F/F, C_{FF}	10 fF
Width of the leaf of H-tree, W_{Leaf}	0.72 μm
Interconnect capacitance of leaf per unit length, C_{Leaf}	1050 fF / cm
Interconnect resistance of leaf per unit length, R_{Leaf}	1130 Ω / cm
Clock driver input capacitance, C_{Driver}	500 fF
Clock driver output resistance, R_{tr}	12.0 Ω

The interconnect resistance per unit length was calculated using Equation 5.1.

$$R_{int} = R_{sh} \cdot \frac{(L_{int} - \Delta L_{int})}{(W_{int} - \Delta W_{int})} \tag{5.1}$$

where, R_{sh} is the sheet resistance for the metal layer used, ΔL_{int} is variation in the interconnect length, and ΔW_{int} is variation in the interconnect width. The interconnect dimensions are illustrated in Figure 5.1.

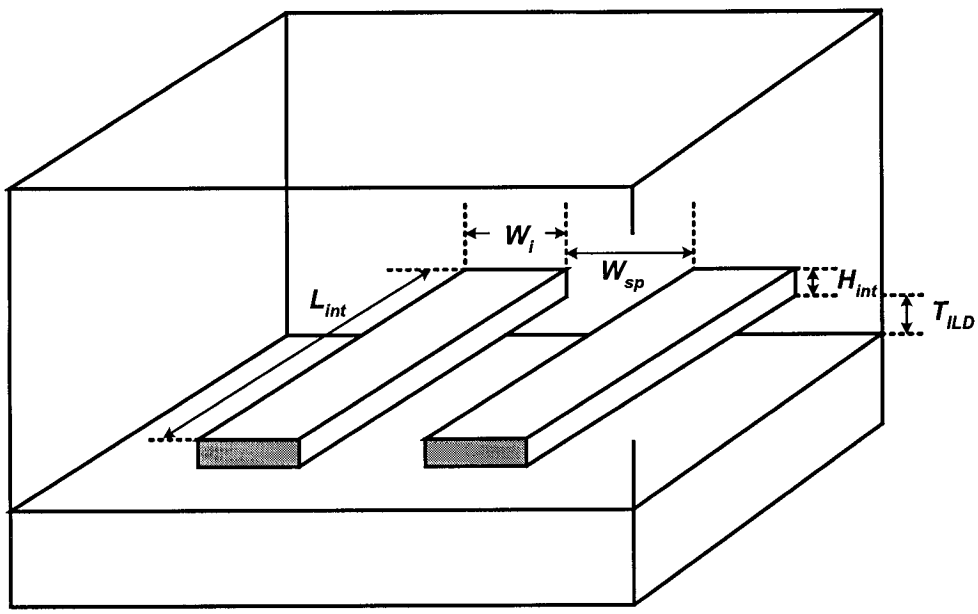


Figure 5.1. Interconnect dimensions [3]

The capacitance of interconnect per unit length can be approximated by Equation 5.2 [3].

$$\frac{c_{int}}{\varepsilon_{ox}} = 1.15\left(\frac{W_{int}}{T_{ILD}}\right) + 2.80\left(\frac{H_{int}}{T_{ILD}}\right)^{0.222} + \left[0.06\left(\frac{W_{int}}{T_{ILD}}\right) + 1.66\left(\frac{H_{int}}{T_{ILD}}\right) - 0.14\left(\frac{H_{int}}{T_{ILD}}\right)^{0.222}\right]\left(\frac{T_{ILD}}{W_{sp}}\right)^{1.34} \tag{5.2}$$

where, ε_{OX} is dielectric constant for ILD material.

5.1 Estimation of Effective Clock Frequency

The frequency for each sub-block was calculated considering the clock network bandwidth, the clock skew, and the clock stretching as described in chapter 3.

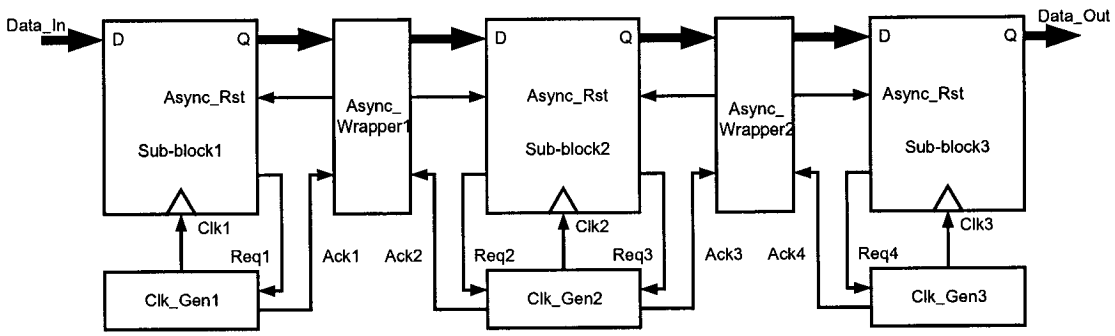


Figure 5.2. Scheme for simulation

To evaluate the average duration of clock stretching, we simulated the scheme shown in Figure 5.2. Three locally synchronous sub-blocks were connected with linear communication links. The asynchronous wrapper used was the data communication channel with GasP pipeline stage and WAIT ports as suggested by De Clercq *et al.* and described in section 2.4.3. The scheme was simulated 0.18 μm CMOS technology from TSMC (Taiwan Semiconductor Manufacturing Company). The simulations were carried out using the electrical simulator HSPICE under Cadence tool set. The design was optimized and tested for wide range of frequencies resulting from different partitioning scenarios to estimate average duration of clock stretching. The simulations showed that there was no stretching of the clock for the frequencies below 700 MHz. For higher frequencies, the average duration of the "off" phase of a stretched clock (T_{Strch}) for *Clk3* was measured 650 pS. The maximum effective clock frequency (f_{Eff}) achieved was 1.2 GHz with both the sender and the receiver running at 2.3 GHZ.

Two cases were considered for three array configurations: (1) Clock network without repeaters, and (2) Clock network with repeaters.

5.1.1 Frequency estimation for passive clock networks

The methodology adopted is explained below:

- The bandwidth of the clock network was evaluated from Equation 3.10 for the calculated values of C_{Leaf} and R_{Leaf} shown in Table 5.3.

- The clock skew resulting from the variations in each process parameter, as described in section 3.1.2, were evaluated for both, the interconnects and the clock drivers. The percentage variations in the process parameters were assumed the same as mentioned in [17]. The contributions of different process parameter were added together to find the total skew. The skew due to *RC* mismatch and clock driver load mismatch was not included because all the processors in the array are identical. The clock frequency limited by the process parameter variations was evaluated assuming that clock skew is 10% of the clock period.

- For each partitioning, the frequencies resulting from the bandwidth limitations and the process parameter limitations were compared and the smaller frequency was selected as the clock frequency of the sub-blocks.

- The effective clock frequency was computed from Equation 3.12 by incorporating the average duration of clock stretching as available from the simulations.

5.1.2 Frequency estimation for active clock networks

The additional skew resulting from inserting active devices into the clock network was taken into consideration to evaluate the effective clock frequency. The repeaters were sized according to the capacitive load offered by the clock network as described below.

(A) Sizing of repeaters

As described in [3], to get an optimal propagation time of a repeated wire, the *W/L* ratio of the repeater-transistors is increased by a factor *h* given by Equation 5.3,

$$h = \sqrt{\frac{R_0 \; C_{int}}{R_{int} \; C_0}} \tag{5.3}$$

where C_0 and R_0 are the input capacitance and the output resistance of a minimum sized inverter.

Table 5.4. Repeater parameters

Resistance of minimum sized inverter, R_0	7790 Ω
Capacitance of minimum sized inverter, C_0	2 fF
W/L ratio for PMOS, h_{PMOS}	60
Mobility ratio	4
Width of PMOS, W_{PMOS}	10 μm
Length of PMOS, L_{PMOS}	0.18 μm
Width of NMOS, W_{NMOS}	2.5 μm
Length of NMOS, L_{NMOS}	0.18 μm
Resistance of repeater, R_{Rep}	70 Ω
Capacitance of repeater, C_{Rep}	96 fF
Repeater propagation delay, t_{Rep}	60 pS

The PMOS of the repeater was sized using Equation 5.3 and the NMOS size was determined taking into account the mobility ratio, to get equal rise and fall times. Table 5.4 summaries various repeater parameters.

(B) Estimation of effective clock frequency

The repeater was simulated in 0.18 μm CMOS technology to measure the propagation delay (t_{Rep}). The number of repeaters required in the design was estimated using Equation 4.29. The capacitive load on each repeater was evaluated as the sum of the interconnect capacitance between two successive repeaters and the gate capacitance of a repeater. To evaluate the additional skew resulting from the repeaters, the skew from the device and interconnect parameter variations was computed for each repeated segment and added together. The same methodology as outlined in section 5.1.1 was adapted to calculated effective clock frequency.

5.1.3 Limitation

The frequency estimation was restricted to array with linear communication links among processors. More complex schemes like unidirectional links among neighboring processors (Figure 4.2), and bidirectional links among neighboring processors (Figure 4.3) will have multiple requests coming in the ME of the clock generator. For these configurations, the estimation of average duration of clock stretching requires the knowledge of the probability of each request being present at given time and the time before it is acknowledged.

5.2 Power Estimation and Optimum Number of Sub-blocks

The methodology adapted for power estimation is summarized below:

- For each partitioning scenario, the power consumption in the clock network was calculated using Equation 4.3 with parameters listed in Table 5.2 and Table 5.3.

- The power consumption in the asynchronous wrappers, the ring oscillators, and the repeaters, as described in Sections 4.2, 4.3, and 4.5 respectively, was evaluated from simulations at different frequencies using a 0.18 μm CMOS technology from TSMC. The simulations were carried out using the electrical simulator HSPICE under Cadence tool set.

- For the array with linear communication links, the behavior of total power (P_{GALS}) was studied in four cases; (1) with increasing clock frequency and passive clock network, (2) with increasing clock frequency and active clock network, (3) Constant clock frequency and passive clock network, and (4) Constant clock frequency active clock network.

- For the arrays with unidirectional and bidirectional links as discussed in Section 4.2.2 and 4.2.3 respectively, the behavior of total power (P_{GALS}) was studied in two cases; (1) Constant clock frequency and passive clock network, and (2) Constant clock frequency and active clock network.

- The optimum number of sub-blocks as established from the manual calculations mentioned above were compared with the results from the equations in Section 4.4 to verify the validity of the equations.

5.3 Experimental Results for an Array with Linear Communication Links

An array of processors with linear communication links is shown in Figure 3.1. The following sub-sections present the results for four cases discussed in Section 5.2 for this configuration.

5.3.1 Power with increasing frequency and passive clock networks

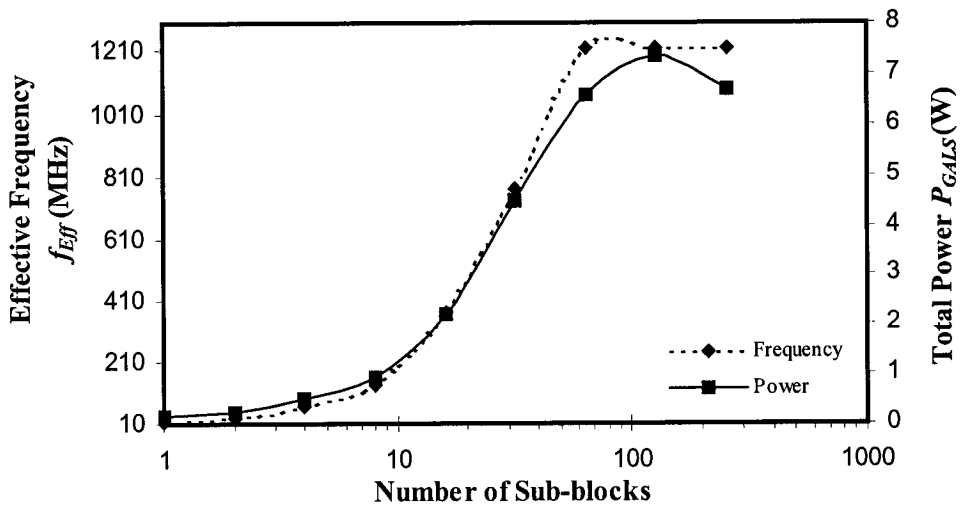


Figure 5.3. Power with increasing frequency and passive clock network for the linear array

As illustrated in Figure 5.3, the clock frequency of the fully synchronous system is lower bounded due to the interconnect bandwidth. The exponential increase in the frequency with the number of partitioning comes from an improved bandwidth due to smaller sizes of the sub-blocks. The simulation results showed that there was no stretching of the clock for frequencies below 700 MHz. Beyond this point, the skew due to the process parameter variations and the clock stretching have significant influence on the effective clock frequency. The average duration of clock stretching (T_{Strch}) for *Clk3* was measured 650 pS. The maximum effective clock frequency (f_{Eff}) achieved was 1.2 GHz with both the sender and the receiver running at 2.3 GHz. The maximum attainable frequency is constrained by the delay in asynchronous wrapper and the ring oscillator.

Keeping the supply voltage (V_{DD}) constant, there are two factors that affect the power consumption: (1) the capacitance, and (2) the frequency. With increasing number of sub-blocks, the capacitance of the clock network decreases. However, the exponential rise in frequency dominates the effect of decrease capacitance and the power increases. The frequency saturates with 64 number of sub-blocks and onwards. For 256 sub-blocks, complete elimination H-tree saves more power than increase in asynchronous power overhead at the same frequency. Therefore, the power curve takes a plunge after 128 sub-blocks. However, this power is still more that the power for 64 sub-blocks. Therefore, the optimum number of sub-blocks, in this case, is 64 that give the maximum frequency with least power consumption.

5.3.2 Power at constant frequency and passive clock networks

In Figure 5.4, the clock frequency is kept constant at 15 MHz, which is the clock frequency of fully synchronous system in Figure 5.3. The power decreases up to 32 sub-blocks due to

decreasing H-Tree capacitance. The power consumption in the asynchronous wrappers is negligible up to this point. Increasing the number of sub-blocks further raises power due to the asynchronous overhead.

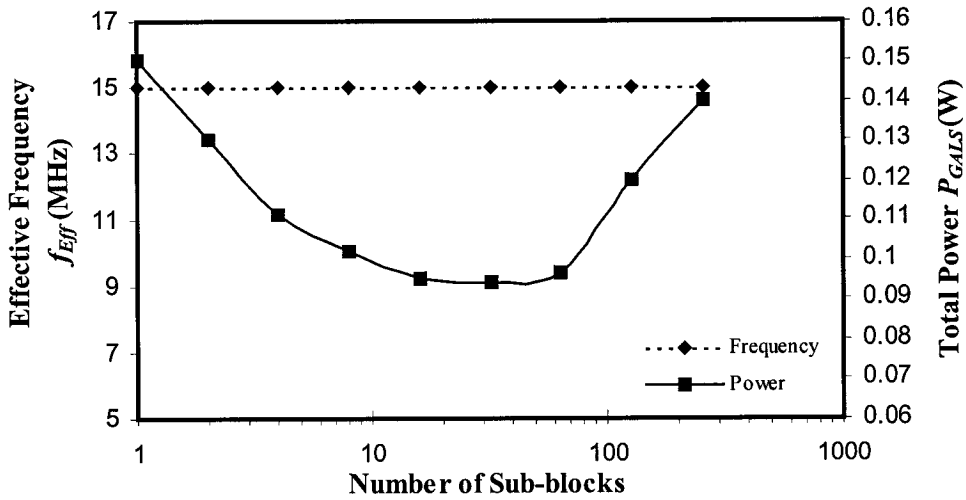


Figure 5.4. Power at constant frequency and passive clock network for the linear array

Table 5.5 shows various computed and simulated values of the constants required in evaluating the optimum number of sub-blocks from Equation 4.23.

Table 5.5. Values of the constants in Equation 4.23 at 15 MHz

Constant for clock power, A_1	0.0418
Power consumed in each wrapper, P_{Async_1}	0.117 mW
Power consumed in each oscillator, P_{Osc_1}	0.143 mW

Substituting all the constant values in Equation 4.23 gives the following result.

$$S_{Opt\_Linear} = 30$$

This value of optimum number of sub-blocks is close to 32 sub-blocks as predicted in Figure 5.4. The optimum partitioning, in this case results in 37% of power saving as compared to fully synchronous designs.

5.3.3 Power with increasing frequency and active clock networks

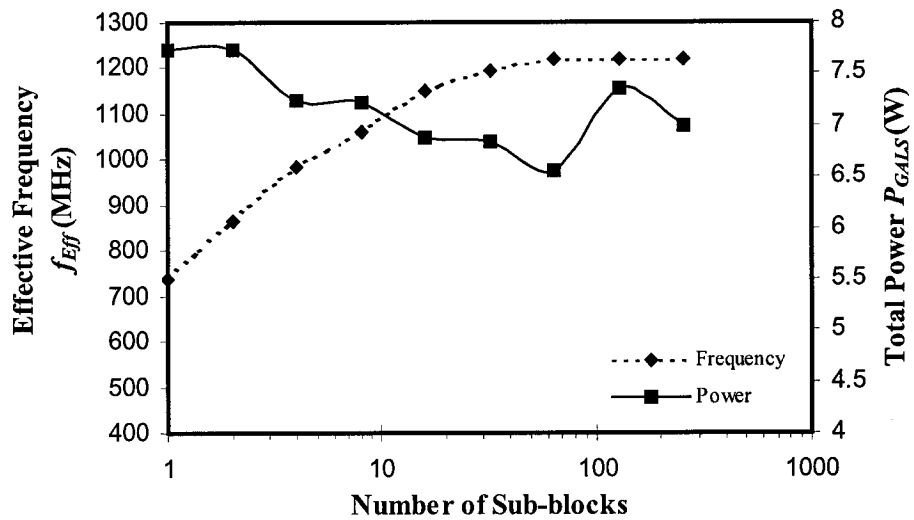


Figure 5.5. Power with increasing frequency and active clock network for the linear array

Inserting repeaters in the clock network removes the bandwidth constraints on the clock frequency. As can be observed form Figure 5.5, the undesirable effect (skew due to process variations) of inserting active repeaters is very well offset by improvements gained in the bandwidth. The clock frequency of a fully synchronous system, which was limited to 15MHz, is now close to 700*MHz*. Since the frequency is already high, the stretching of clock occurs right from the first partitioning and therefore the gains in frequency are not as steep as Figure 5.3. The frequency finally levels at 1.2 GHz that is limited by the asynchronous wrappers.

The major difference in Figure 5.3 and Figure 5.5 is the behavior of power. In Figure 5.5, the power decreases despite the rise in frequency and there are two reasons that can be attributed to this behavior; (1) the difference in the clock frequencies is not as large as Figure 5.3 for two successive partitions, and (2) the decrease in the capacitance now comes from two sources; the reduction in H-tree levels, and the diminishing number of large clock repeaters.

Table 5.6 gives the total number of repeaters required with each partitioning as evaluated from the methodology described in section 4.5.

Table 5.6. Number of repeaters required for different number of sub-blocks

Number of Sub-Blocks S	**Number of Repeaters Required** N_{Rep}
1	106
2	96
4	88
8	80
16	64
32	64
64	0
126	0
256	0

The power decreases up to 64 sub-blocks in Figure 5.5. As no repeaters are required after this point, the power consumed in the asynchronous wrappers now takes over and total power starts rising. The optimum number of sub-blocks is 64 in this case because it gives the maximum frequency for the least power consumption.

5.3.4 Power at constant frequency and active clock networks

For constant frequency as well, the power decrease up to 64 sub-blocks due to diminishing repeaters and decreasing H-tree capacitance. Since no repeaters have been used after this point, the power in asynchronous wrapper governs the total power and it starts rising. The manual methodology to evaluate the optimum number of sub-blocks for the active clock networks is expressed in from of an algorithm in Figure 4.6.

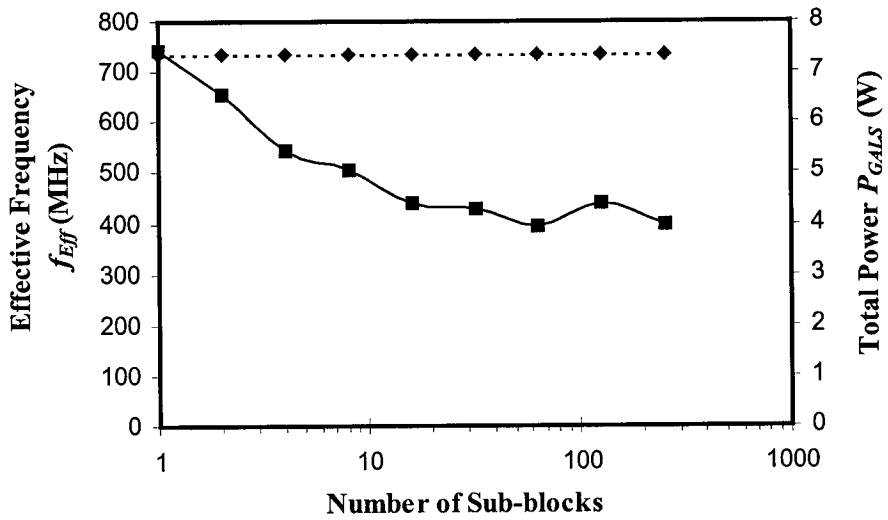


Figure 5.6. Power at constant frequency and active clock network for the linear array

5.4 Experimental Results for Array With Unidirectional Links

A GALS array of processors with unidirectional communication links is shown in Figure 4.2. The following sections present the power trade-offs for active and passive clock networks at a constant frequency.

5.4.1 Power at constant frequency and passive clock networks

In Figure 5.7, the clock frequency is kept constant at 15 MHz. The power decreases up to 16 sub-blocks due to decreasing H-Tree capacitance. The power consumption in the asynchronous wrappers is negligible up to this point. Increasing the number of sub-blocks further raises the total power due to increased contribution of the asynchronous overhead.

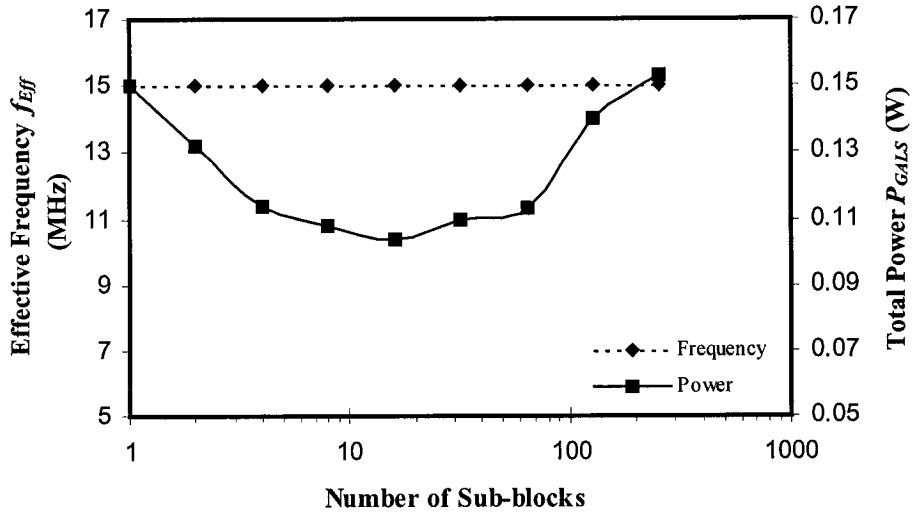


Figure 5.7. Power at constant frequency and with passive clock network for the array with unidirectional links

For this configuration, each partitioning requires significantly more number of asynchronous wrappers as compared to the linear array configuration of section 5.4.2. Therefore the optimum number of sub-blocks is smaller in this case compared to the case of linear array.

The polynomial in Equation 4.25 can be used to evaluate optimum number of sub-blocks for this configuration. The value of constant K is 3 for an array of size 16x16. The values of other constants will be the same as shown in Table 5.5 because the frequency is the same. Substituting all the constant values in Equation 4.25 and solving it with *Xmaple* gives the following result:

$$S_{Opt\_Uni} = 16.98$$

This value of is close to 16 sub-blocks as predicted in Figure 5.7 and it results in 31% of power saving as compared to fully synchronous designs.

5.4.2 Power at constant frequency and active clock networks

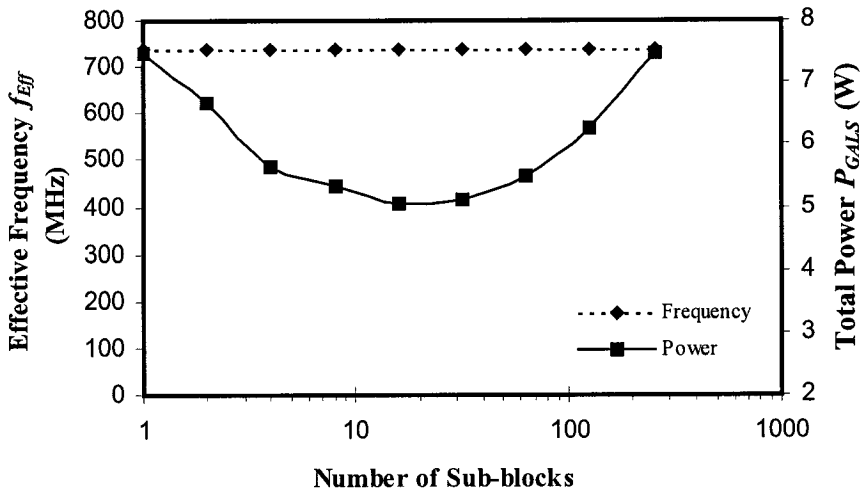


Figure 5.8. Power at constant frequency and with active clock network for the array with unidirectional links

In Figure 5.8, the clock frequency is kept constant at 736 MHz, which is the clock frequency of a fully synchronous system with repeaters. The total power decreases up to 16 sub-blocks due to the diminishing number of repeaters and decreasing H-tree capacitance. For this configuration, since each partitioning requires significantly larger number of wrappers as compared to the linear array, power consumption in wrappers starts dominating sooner in this case. The algorithm in Figure 4.6 can be used to evaluate the optimum sub-blocks for this configuration as well.

5.5 Experimental Results for Array with Bidirectional Links

This configuration requires twice the number of asynchronous wrappers as compared to the array with unidirectional links because all the processors communicate to neighbors in both directions as shown in Figure 4.3.

5.5.1 Power at constant frequency and passive clock networks

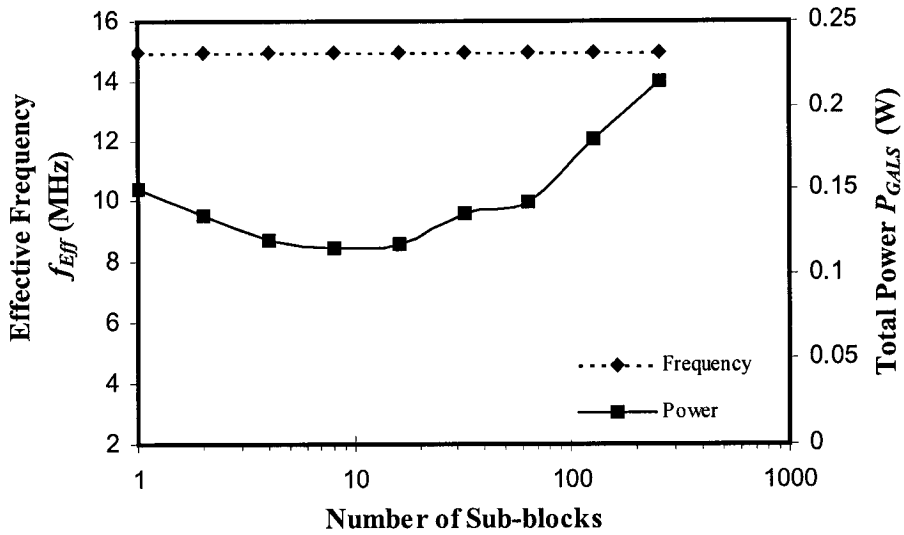


Figure 5.9. Power at constant frequency and with passive clock network for the array with bidirectional links

As illustrated in Figure 5.9, by keeping the clock frequency constant at 15 MHz, the total power decreases up to 8 sub-blocks due to decreasing H-Tree capacitance. The contribution of the power consumed in the asynchronous wrappers is comparatively smaller up to this point. Increasing the number of partitions further raises the total power due to increased contribution of the asynchronous overhead. The optimum number of sub-blocks is smaller in this case as compared

to the case of array with unidirectional links because each partitioning requires twice the number of asynchronous wrappers as compared to the case of array with unidirectional links.

The polynomial in Equation 4.27 can be used to evaluate the optimum number of sub-blocks for this configuration. The value of the constant K is 3 for an array of size 16x16. Other constants values will be the same as shown in Table 5.5 because the frequency is the same. Substituting all values in Equation 4.25 and solving it with *Xmaple* gives the following result:

$$S_{Opt\_Bi} = 9.96$$

This value of is close to 8 sub-blocks as predicted in Figure 5.9. It results in 23% of power saving as compared to fully synchronous designs.

5.5.2 Power at constant frequency and active clock networks

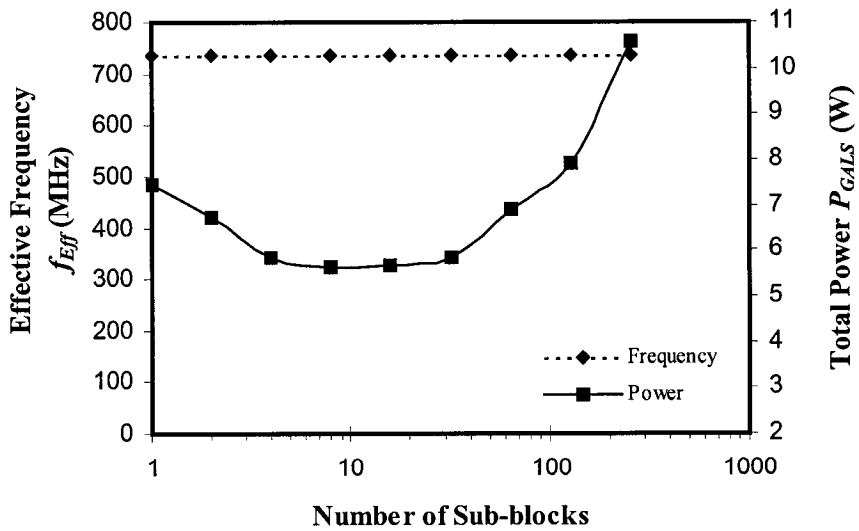


Figure 5.10. Power at constant frequency and with active clock network for the array with bidirectional links

The number of sub-blocks that gives least power consumption at 736 MHz clock frequency is 8 in this case. The reason for the optimum number of sub-blocks being such small is that each partitioning requires a large number of asynchronous wrappers. Such an increase in the power consumed in the asynchronous wrappers overrides the effect of decrease in the clock network capacitance and the number of the repeaters required.

CHAPTER 6

Conclusion and Future Work

The GALS design methodology is a potential solution to the growing problems of distributing a high frequency clock signal at low power in large VLSI designs. GALS designs offer a solution to these problems by reducing the clock network capacitance and localizing the clock skew within the boundaries of smaller, locally synchronous sub-blocks. An asynchronous wrapper with pausible clocking is used at the boundaries of two communicating blocks to avoid synchronization failures.

6.1 Does GALS Boost the Effective Clock Frequency of a System?

The answer to this question is both "yes" and "no" depending on the architecture. GALS does increase the frequency of individual, locally synchronous sub-blocks by containing the skew and the bandwidth problems. However, overall effective clock frequency (or throughput) of the system may degrade for some architectures. The investigation of an array with linear communication links reveals that the clock frequency increases from 15 MHz for a fully synchronous system to 1.2 GHz with GALS. However, the skew due to mismatch in *RC* delays of various clock paths and the skew due to clock load mismatch were absent in this study because of the type of architecture used. The maximum frequency was limited by the asynchronous wrapper and the ring oscillator. The frequency estimation was restricted the case of an array with linear communication links among processors. More complex schemes like unidirectional links among neighboring processors (Figure 4.2), and bidirectional links among neighboring processors (Figure 4.3), will have multiple requests coming in the asynchronous wrappers. Each request can

pause the clock for unbounded time until it is acknowledged. Therefore, for these configurations, estimation of the average duration of clock stretching requires the knowledge of the probability of each request being present in the asynchronous wrapper at any instance and the time before it is acknowledged.

Finding an optimum partitioning to achieve highest throughput is also very sensitive to the type of communication mechanism used for architectures with multiple links among all of its sub-blocks. For example, in shared-bus architecture with central arbiter, large number of partitions will increase the number of requests in the arbiter at any given time. The bus will be highly congested in this case. Therefore, a sender's clock will be paused for a long time waiting for a receiver to respond and gaining access to the shared communication resources. Predicting the average period of clock stretching becomes probabilistic in this situation. Large number of partitions may degrade the system throughput by lengthening the wait period of the sub-block clocks. A general solution to evaluate optimum GALS partitioning with clock frequency as an objective is not possible in this situation. However, an architecture-specific study will be addressed in a future work.

For large VLSI designs, inserting the repeaters in clock network is an alternative to boost the clock frequency otherwise limited by the interconnect bandwidth. As shown in chapter 5, for the fully synchronous design, inserting the repeaters increases the frequency form 15 MHz to 700 MHz. The undesirable effect of inserting active repeaters is very well offset by the improvements gained in the bandwidth. Therefore for the GALS architectures where increasing partitioning is not a feasible solution, a combined strategy of partitioning and repeater insertion may be adapted to achieve desired effective system throughput.

6.2 Does GALS Save Clock Power?

The answer to this question is "yes". In most high-performance VLSI designs, the distribution of low-skew global clock signals approaching GigaHertz range is the single largest source of power consumption. One of the motivations behind switching to GALS designs is to save power in the clock networks. In GALS designs, partitioning a system into more locally synchronous sub-blocks reduces the size of each sub-block, which diminishes the capacitance in the clock networks because they need less H-tree levels. However, this implies a large number of sub-blocks, which increases the asynchronous power overhead. These power tradeoffs have been studied for a GALS array of identical processors.

The thesis enriched previous works on GALS power consumption and partitioning in multiple dimensions. The model of GALS power components were adapted to suite GALS array of processors with three different configurations. We also proposed closed form models to predict optimum number of sub-blocks that gives the least power consumption. The validity of the models has been verified with experimental results. The models can provide a useful firsthand guideline for the designers in the initial design stages.

Inserting repeaters in the clock network adds one more aspect in the GALS power tradeoffs. The GALS power benefits are more evident in this case because the repeaters are often area and power hungry. Smaller sub-blocks require fewer repeaters but more wrappers. The proposed algorithm is a useful tool to estimate optimum number of sub-blocks in this case.

The behavior of total power consumption with increasing clock frequencies was also studied for each partitioning scenario and for active as well as passive clock networks. The results show that,

for the GALS array with passive clock networks, the total power increases with partitioning because the exponential rise in frequency dominates the effect of decrease in global clock capacitance. However, for the GALS array with active clock networks, the GALS power decreases with partitioning because of the decrease in the clock network capacitance and reduction in the number of repeaters required.

6.3 Does GALS Resolve the Timing Closure Problem?

The answer to this question is "yes", but at the cost of area and, in some cases, power. Timing closure is a challenging issue for the designs with bundled data protocols and the synchronous designs with long global interconnects as described in Chapter 2. The DI asynchronous protocols encode request on data lines and therefore avoids the need for timing analysis, giving designs that operate correctly whatever the delay in the interconnecting wires. A novel asynchronous wrapper architecture based on DI protocol and ST handshaking has been proposed in this work. In order to prove its feasibility, we simulated the design shown of Fig.4 with 0.18 μm TSMC CMOS technology. The perceived advantage of ST handshaking over 4-phase handshaking is visible in improved throughput. It achieves 66% higher throughput as compared to the DI templates proposed in [23]. Moreover, the scheme saves the fifth *ACK* wire in the scheme by Bainbridge *et al.* It avoids the timing constraints described in [13] by appropriate sizing of transistors.

Finally, since only one wire changes its state for each data transaction, the power consumption is comparable to the bundled data scheme in [12].

.

References

[1] A. Hemani "*Lower Power Consumption in Clock By Using Globally Asynchronous Locally Synchronous Design Style*," in Proc. of Design Automation Conference (DAC), 1999.

[2] A. Iyer *et al.*, "*Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors* ", In Proc. of Annual International Symposium on Computer Architecture, May 2002.

[3] H. Bakoglu, "*Circuits, Interconnections and Packaging for VLSI*", Reading MA: Addison-Wesley, 1990.

[4] D. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*, Ph.D. thesis, Stanford University, Oct. 1984.

[5] D. Sylvester *et al.*, "*Getting to the Bottom of Deep Submicron*," in Proc. of ICCAD, 1998, pp. 203-211.

[6] D. Sylvester *et al.*, "*Getting to the Bottom of Deep Submicron II: A Global Wiring Paradigm*," in Proc. of International Symposium on Physical Design, ACM Press, New York, 1999, pp. 193-200.

[7] E. Friedman, "*Clock distribution networks in VLSI circuits and systems*", IEEE Press, 1995.

[8] I. Sutherland *et al.*, "*GasP: A Minimal FIFO Control*", In Proc. of International Symposium on Asynchronous Circuits and Systems (ASYNC), 2001, pp: 46 – 53.

[9] J. Muttersbach *et al.*, "*Practical Design of Globally-Asynchronous Locally-Synchronous systems*", In Proc. of International Symposium on Advanced Research in Asynchronous Circuits and Systems, April 2000, pp: 52–59.

[10] J. M. Rabaey, "*Digital Integrated Circuits; A Design Perspective*", Prentice Hall, 1996.

[11] K. Y. Yun *et al.*, "*Pausible clocking: A First step toward heterogeneous systems*", in Proc. of International Conference on Computer Design (ICCD), Oct. 1996.

[12] M. De Clercq *et al.*, "*1.1-GDI/s Transmission Between Pausible Clock Domains*", in Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), 2002.

[13] M. Ferretti *et al.*, "*Single-Track Asynchronous Pipeline Templates Using 1-of-N Encoding* ", in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), March 2002

[14] M. Nyström *et al.*, "*Crossing the Synchronous-Asynchronous Divide*", in Proc. of Workshop on Complexity-Effective Design, 2002.

[15] N. H. E. Weste, K. Eshraghian, "*Principles of CMOS VLSI Design; A System Perspective*", second edition, Addison-Wesley Publ. Company 1993.

[16] P. J. Restle *et al.*, "*A Clock Distribution Network for Microprocessors*," IEEE Journal of Solid State Circuits (JSSC), May 2001.

[17] P. Zarkesh-Ha, "*Global Interconnect Modeling for a Gigascale System-on-a-Chip* ", Ph.D. thesis, Georgia Institute of Technology, February 2001.

[18] R. Kol *et al.*, " *Adaptive Synchronization*", in Proc. of International Conference on Computer Design, October 1998.

[19] S.W.Moore *et al.*, "*Self-calibrating clocks for globally asynchronous locally synchronous systems*", In Proc. International Conf. Computer Design (ICCD), September 2000 pp 73-78.

[20] S. Moore *et al.*, "*Point to Point GALS Interconnect*", In Proc. International Symposium on Asynchronous Circuits and Systems (ASYNC), 2002.

[21] J. Sparsϕ, S. Furber, "*Principles of Asynchronous Circuit Design*", Kluwer academic publishers, Boston, 2001.

[22] W. J. Bainbridge, "*Asynchronous System-on-Chip Interconnect* ", Ph.D. thesis, Department of computer science, University of Manchester, March 2000.

[23] W.J. Bainbridge *et al.*, "*Delay Insensitive System-on-Chip Interconnect using 1-of-4 Data Encoding*", in Proc. Async'01, Utah, April 2001 pp. 118-126.

[24] W.J. Bainbridge *et al.*, "*CHAIN: A Delay-Insensitive Chip Area Interconnect*", IEEE Micro, Sep/Oct 2002.

[25] W. J. Bainbridge *et al.*, " *Delay-Insensitive, Point-to-Point Interconnect Using M-of-N Codes* ", in Proc. of Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC), May 2003.

[26] T. Olsson *et al.*, " *A digitally Controlled on-chip clock multiplier for Globally Asynchronous Locally Synchronous Systems* ", in Proc. of the 42nd Midwest Symposium on Circuits and Systems, 2000, pp 84-87.

[27] Semiconductor Industry Association, "*International technology roadmap for semiconductors*", 2001.

[28] V. Tiwari *et al.*, "*Reducing Power in High-performance Microprocessors*", 35th DAC, June 1998.

[29] R. Chen *et al.*, "*Clock Power Issues in System-on-a-Chip Designs* ", IEEE Computer Society Workshop on VLSI, 1999.

[30] P. Beerel, "*Asynchronous Circuits: An increasingly Practical Design Solution*", International Symposium on Quality Electronic Design, March 2002.

[31] C.H. Van Berkel *et al.*, *"Scanning the Technology: Applications of Asynchronous Circuits"*, Proceedings of the IEEE, Volume 87, Issue 2, Feb. 1999, Pp. 223-233.

[32] A. Chakraborty *et al.*, **"***Efficient Self-Timed Interfaces for Crossing Clock Domains***"**, Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03), May 2003.