INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

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

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

A Distributed Arithmetic Based CORDIC Algorithm and its Use in the FPGA Implementation of the 2-D IDCT

Yi Yang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

April 2002

ABSTRACT

A Distributed Arithmetic Based CORDIC Algorithm and its Use in the FPGA Implementation of the 2-D IDCT

Yi Yang

The discrete cosine transform (DCT) based image compression techniques play an important role in today's digital applications, such as high definition television (HDTV) and teleconferencing, which require high speed transmission of digital image/video signals. The dedicated video codec is widely used to perform this task because of its low cost and high performance compared to the general-purpose processors. A video codec chip requires an integration of high-speed DCT and inverse DCT (IDCT) hardware units in a limited silicon space. In video codecs, the error in the IDCT computation is more critical than that in the DCT computation due to the cumulative nature of its computation error within a reconstruction loop of the codec. Thus, the design and implementation of IDCT algorithms poses a greater challenge.

This thesis presents a distributed arithmetic based CORDIC algorithm for the computation of the 1-D IDCT and an FPGA implementation of a cost-effective architecture for a 2-D IDCT processor using the proposed algorithm. The processor consisting of two 1-D IDCT cores, a transpose memory and a control logic block performs the 2-D IDCT computation by using the row-column decomposition approach. The basis of the proposed scheme is a combined use of the distributed arithmetic and the CORDIC algorithm in order to provide a small access time to the lookup tables and a reduced complexity for its

architecture. In the proposed design, the deep pipeline structure of an existing CORDIC based architecture is replaced by much smaller DA-based ROM accumulators. By adopting the CORDIC approach, the size of the ROMs becomes relatively small and independent of the size of the image block. In the proposed design, a bit-level digit-serial structure based on the redundant number system using an on-line algorithm is employed in order to provide a good compromise between the area and the speed of the processor.

An 8×8 2-D IDCT processor has been simulated and implemented on a Xilinx Virtex VC2V1000 FG256-4 FPGA board. The accuracy compliance with the IEEE 1180-1990 standard has been verified by a simulation scheme provided by the standard. The processor operates on blocks of 8×8 pixels, with 12-bit and 9-bit precisions for inputs and outputs, respectively. Comparing with the existing designs using similar technology, the proposed IDCT processor is found to provide a better performance when the area and the speed of operation are considered together.

Acknowledgements

I would like to express my sincere gratitude to my Supervisors, Dr. Chunyan Wang and Dr. M. Omair Ahmad, for their guidance, encouragement, and support throughout my graduate studies.

I would like to thank Yintao Jiang, Ning Guo and Wei Wang for their help and the fruitful discussions on my research.

Finally, I would like to express my sincere thanks to my wife, Jinglei, and to my parents for their support and understanding during the course of this research, and for their unselfish love.

Table of Contents

List of Figures

List of Tables

List of Acronyms

1-D	One-dimensional
2-D	Two-dimensional
AorS	Add or Subtract signal
ASIC	Application specific integrated circuit
BSD	Binary signed digit
CCITT	Commité Consultatif International de Telecommunications et Telegraphy
CORDIC	Coordinate rotation digital computer
CRA	Carry ripple adder
CSA	Carry-select adder
DA	Distributed arithmetic
DCT	Discrete cosine rransform
DFT	Discrete Fourier transform
DHT	Discrete Hadamard transform
DSP	Digital signal processing
DST	Discrete sine transform
DUT	Device under test
FPGA	Field programmable gate array
GOPS	Giga operations per second
HDL	Hardware design language
HDTV	High definition television
IC	Integrated circuit
IDCT	Inverse discrete cosine transform
IEEE	Institute of Electrical and Electronic Engineers

JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve transform
LSDF	Least significant digit first
MMP	Minus-minus-plus
MOPS	Million operations per second
MPEG	Moving Pictures Expert Group
MSB	Most significant bit
MSDF	Most significant digit first
PISO	Parallel-in-serial-out
PPM	Plus-plus-minus
R-NNC	Redundant-to-nonredundant number convertor
RAC	ROM-accumulator
RAM	Random access memory
RISC	Reduced instruction set computer
ROM	Read only memory
RTL	Register transfer level
SIPO	Serial-in-parallel-out
VLSI	Very large scale integration

List of Primary Symbols

a_m	Fixed coefficient of the sum of products
A	Matrix with the cosine basis functions of 1-D IDCT
B	Pixel wordlength
$C(u)$	Constant factor of the IDCT
d_i	Direction of the CORDIC rotation
E	Loopback data of the radix-2 ROM accumulator
$F_N(a, x_m^{(j)})$	Intermediate result of the sum of products using the distributed arithmetic approach
$g(i, u)$	Intermediate result of the IDCT using CORDIC algorithm
I	$N \times N$ identity matrix
K	Scaling factor of CORDIC algorithm
N	Size of the image block
O	Output data of the ROM in ROM accumulator
p	Horizontal coordinates of a 2-D plane
q	Vertical coordinates of a 2-D plane
$R_P(\phi, p^{(j)}, q^{(j)})$	First of the two pre-computed intermediate results of 1-D IDCT using DA-based CORDIC algorithm.
$R_Q(\phi, p^{(j)}, q^{(j)})$	Second of the two pre-computed intermediate results of 1-D IDCT using DA-based CORDIC algorithm
s_i	Sum digit in one-digit radix-2 signed-digit addition
t_i	Transfer digit in one-digit radix-2 signed-digit addition
u_i	Interim sum in one-digit radix-2 signed-digit addition

v_i	Transfer digit in one-digit radix-2 signed-digit subtraction
w_i	Interim sum in one-digit radix-2 signed-digit subtraction
x_m	Input data of the operation of sum of products
$x_m^{(j)}$	jth bit of x_m
$x(i)$	Output data of IDCT
$X(i)$	Input data for IDCT
$xc(i)$	Output digit of the CORDIC rotators
$xr(i)$	Output digit of the butterfly adder array
$xs(i)$	Output digit of the PISO register
$xt(i)$	Output results of the redundant-to-nonredundant number convertor
Y	Intermediate product matrix of the 2-D IDCT
z_i	Digit of the intermediate result of radix-4 signed-digit addition

Chapter 1

Introduction

In recent years, there has been a significant increase in the demand of multimedia applications. These applications require dealing with a large amount of data, and/or image, video and audio signals, resulting in high storage and transmission costs. Without compression, most of these applications would not be feasible. In the case of signal transmission, compression is a process intended to minimize the bit rate of the digital representation of a signal stream. As an example, in a facsimile image transmission, scanning and digitizing an 8.5×11 inch page at 200 dpi results in 3.74 Mbits data that takes typically 5.62 minutes of transmission over a 14.4 kbts/s modem. With compression, the transmission time can be reduced to only 17 seconds, thus resulting in substantial savings in the transmission cost.

Transform coding has been a dominant method of video and still image compression. Transform coding is based on the characteristic that image data tend to have a high degree of spatial redundancy. Furthermore, often image data are eventually presented to a human viewer. Thus, if one views a lossy still-image compression system from end to end, that is, from the creation of the visual information to the eventual display of the information after decompression, then it is prudent to exploit the characteristics within each com-

ponent of this system so as to generate a compressed stream with the least number of bits. Within such a system, compression is achieved by exploiting both the spatial redundancies within the image and the perceptual characteristics of the human visual system so that the loss due to the compression may not be discernible to the viewer.[3]

Some of the most commonly used transform coding functions include the basis functions of the discrete Fourier transform (DFT), the discrete cosine transform (DCT), the discrete sine transform (DST), the discrete Hadamard transform (DHT), and the Karhunen-Loeve transform (KLT). Among these transforms, the DCT is the basis of most of the image and video compression standards since its conception in 1974. Because the DCT uses cosine as the basis function as opposed to a complex exponential, it exhibits a computational complexity lower than that of the discrete Fourier transform by almost a factor of 2. Besides, the DCT virtually produces optimally decorrelated frequency domain coefficients. With respect to this property, the DCT is only slightly surpassed by the theoretically optimal and computationally intensive Karhunen-Loeve transform which unlike the DCT has several implementation-related deficiencies, including the fact that the basis functions are image dependent. Because of these most desirable properties, the DCT-based image coding has been adopted as a standard in many applications, such as JPEG, MPEG-2, MPEG-4, and CCITT Recommendation H. 26x [13].

1.1 Hardware Implementation of Video Compression Standards

Until recently, the general-purpose processors have provided enough computing power for the software implementation of some of the video standards. For example, MPEG-2 compression/decompression for a 720×480 frame size at 30 frames per second (fps) requires approximately 132 MOPS (million operations per second) and that is now easily achievable by several general-purpose processors such as Intel Pentium microprocessor [29]. However, for a high definition television (HDTV) application in which the frame size is assumed to be 1440×1152, the codec requires more than 22 GOPS (giga operations per second), which is beyond the present capabilities of a general-purpose processor [3]. Besides, the dedicated video compression chips are normally much cheaper than the general-purpose processors, thus making it quite appealing to the industry.

Recent hardware implementations of the video compression standards fall into three main design categories: video signal processors, multimedia coprocessors, and dedicated coders/decoders. Video signal processors are programmable processors with a digital signal processing (DSP) core or reduced instruction set computer (RISC) core and coprocessing units for compute-intensive operations, such as motion estimation. Multimedia coprocessors are also programmable; however, they also support multitasking for simultaneous acceleration of multimedia tasks, including video, audio, and graphics. Dedicated coders provide limited, if any, programmability and have a dedicated architecture and control logic for a specific video encoding or decoding standard. There is an expectation that applications using video compression, such as set-top boxes for interactive TV,

HDTV, digital cameras, and desktop video teleconferencing, which all require dedicated video codecs, will experience a huge growth in the next few years.

1.2 Motivation for Research in the Hardware Design of the Inverse DCT

Single-chip video codecs need to integrate in a limited silicon space the DCT or inverse DCT (IDCT) hardware with a core processor, an entropy coder, and other circuitry. For such a restricted size requirements for the implementation, efficient designs of DCT/ IDCT algorithms have received considerable attention in the research community [11]-[22]. Figure 1. shows the block diagram of H.261 video coder and decoder. The three key functions included in a video codec are: (1) the computation of the DCT and IDCT, (2) quantization of the DCT coefficients, and (3) entropy coding. Among these operations, the computation of the DCT/IDCT requires a substantial portion of the overall computation time.

In video codec applications, the error in the IDCT computation requires much more attention than that in the DCT computation, due to the cumulative nature of its computation error within the reconstruction loop of the code. As shown in Figure 1.1, both the encoder and the decoder contain a reconstruction loop. Any error in the computation of the IDCT simply gets accumulated because this error is added to the previous reconstructed frame. Therefore, if the error in the IDCT computation is not properly taken care of, it may result in a substantial quality degradation of the reconstructed pictures. Since the forward discrete cosine transform is outside the reconstruction loop and is not needed in the

decoder, error in its computation is not as critical. Thus, because of the greater need of the IDCT computation in a codec and the cumulative nature of the error in its computation, the design and implementation of IDCT algorithms poses a greater challenge.

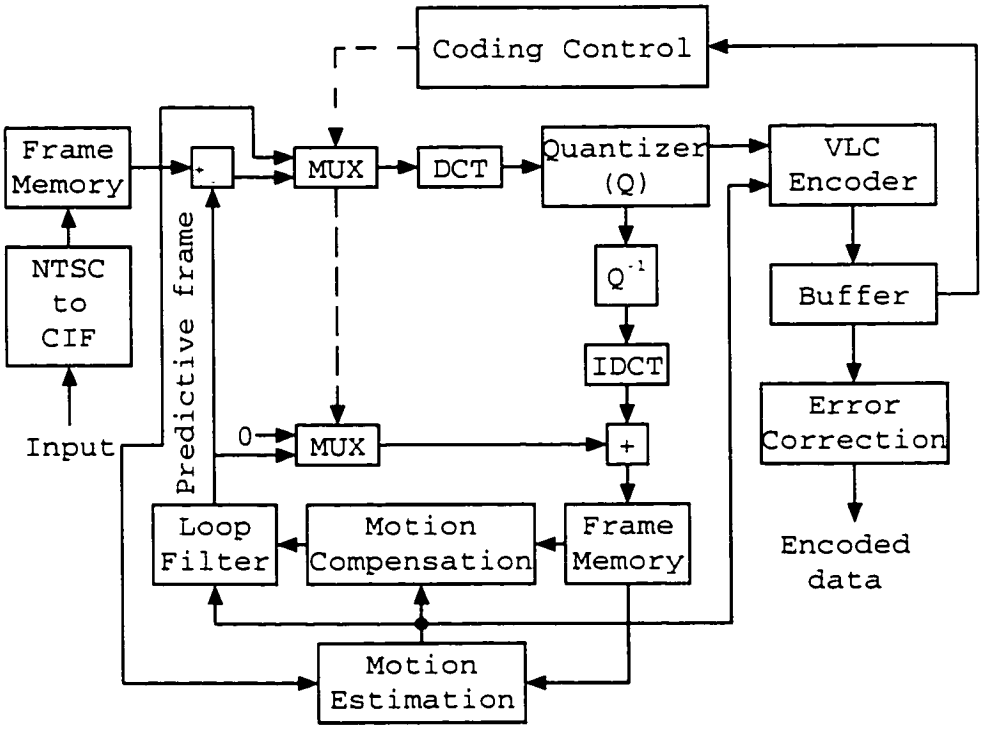

Figure 1.1 Block diagram of H. 261 video (a) coder and (b) decoder.

Many 2-D DCT/IDCT structures have been proposed with an objective of accelerating the computation speed by reducing the number of multipliers and adders. Among the existing architectures, many are based on distributed arithmetic (DA) [23]. The property of the DA that allows the implementation of a sum of products without using multipliers makes it ideally suitable for integrating a DCT processor into a video codec. However, in such an implementation, the size of ROM increases exponentially with the size of image blocks and the bit-serial nature of DA limits the processing speed and causes a large latency. In [33], a pipelined CORDIC algorithm has been employed in a DCT/IDCT architecture in order to achieve the high speed requirement of HDTV applications. However, deeply pipelined structure of this scheme occupies a large implementation area that could even be comparable to the direct multiplier-based implementations.

1.3 Scope and Organization of the Thesis

In this thesis, a new architecture for a 2-D inverse discrete cosine transform processor based on a modified radix-4 on-line CORDIC algorithm and the distributed arithmetic in the framework of the redundant number system is proposed. The architecture is designed to take advantage of the "carry-free" addition property of the redundant number representation and the multiplierless property of the DA. The carry-free property leads to a high-speed operation and makes the delay independent of the internal wordlength of the processor. The objective of employing the radix-4 on line CORDIC algorithm is to reduce

the latency and to control the ROM size without compromising the good feature of the DA.

The thesis is organized as follows. In Chapter 2, some background material on the principle of the DCT/IDCT computation and some of the widely used schemes for its design and implementation are presented. The advantages and disadvantages of these designs are also discussed in this chapter. The algorithm and the conceptual architecture for the design of a 1-D IDCT core are developed in Chapter 3. In Chapter 4, details of the hardware architecture of an 8×8 2-D IDCT processor, which employs the algorithm and architecture proposed in the previous chapter, are presented. In Chapter 5, the design of the proposed IDCT processor is validated through simulation. An exact-bit simulation is carried out to demonstrate the accuracy compliance of the proposed design with the IEEE standard 1180-1990. The designed processor is also implemented using Xilinx Virtex XC2V1000 FPGA board and tested for its speed, space requirement and the functional behavior. Finally in Chapter 6, the results of this investigation are summarized, the contributions highlighted, and the direction for some further study suggested.

Chapter 2

Background

Normally, there is a high degree of correlation between the intensity values of adjacent pixels of an image. Image compression can be achieved by removing such a redundant information.The discrete cosine transform (DCT) is widely used in most applications for the compression of digital image and video signals. After performing the DCT operation on the image/video signals, most of the energy is found to be concentrated in the low frequency region. Thus, it is possible to compress the image by neglecting the high-frequency components of the DCT. At the receiving end, an inverse DCT operation is performed to reconstruct the image. In this operation, the high frequency components, that were discarded before the signal transmission, are assumed to be zero. This simple technique yields good compression of the input image, without much compromise on the picture quality of the received signal.

Due to the importance of the two-dimensional (2-D) DCT/IDCT in digital image processing, particularly in video compression, various algorithms and architectures have been proposed for its implementation. Most of them can be classified into two categories. The first category of algorithms employ the techniques of matrix analysis and decomposition. For example, the fast DCT algorithms in [16] and [17] belong to this category. The

second category includes those algorithms that use polynomials and employ number theory. The algorithms given in [18] and [19] are such algorithms. Each of the categories can be further divided into two groups. In the first group of algorithms, the row-column decomposition is used in order to transform the problem of a 2-D DCT computation into that of a 1-D DCT computation. In the second group, the 2-D DCT computation is carried out directly on the 2-D data. The algorithms given in [16] and [19] belong to the first group, whereas those in [17] and [18] belong to the second group.

In this chapter, some background material directly related to the work to of this thesis is presented. First, the mathematical definition of the discrete cosine transform and the description of the inverse transform are given in Section 2.1. The distributed arithmetic (DA) technique and its use for the computation of IDCT is introduced in Section 2.2. In Section 2.3, a brief description of CORDIC algorithm and its application for the IDCT computation is presented. Finally, a discussion on the advantages and disadvantages of these two approaches are given in Section 2.4.

2.1 The Discrete Cosine Transform and its Inverse

The computation of the discrete cosine transform is one of the processes of transforming a block of data from the spatial domain to the frequency domain [2]. The inverse process of restoring the spatial domain data from the frequency domain is carried out through the inverse discrete cosine transform.

2.1.1 Definition of 1-D DCT and its inverse

The one-dimensional (1-D) DCT of a sequence of input data with N elements is defined as

$$X(u) = \sqrt{\frac{2}{N}} \cdot \sum_{i=0}^{N-1} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)x(i) \tag{2.1}$$

where $u = 0, 1, \ldots, N-1$, and $C(0) = \frac{1}{\sqrt{2}}$, and $C(n) = 1$ for $n \neq 0$.

The 1-D Inverse DCT of a sequence of N points can be expressed as

$$x(i) = \sqrt{\frac{2}{N}} \cdot \sum_{u=0}^{N-1} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)X(u) \tag{2.2}$$

where $i = 0, 1, \ldots, N-1$.

2.1.2 Definition of 2-D DCT and its inverse

The two-dimensional (2-D) DCT of an array x of $N \times N$ elements is defined as

$$X(k, l) = \frac{2}{N}C(k)C(l)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(n, m)\cos\frac{(2m+1)\pi k}{2N}\cos\frac{(2n+1)\pi l}{2N} \tag{2.3}$$

where $k, l = 0, 1, 2, \ldots, N-1$, $C(0) = \frac{1}{\sqrt{2}}$, and $C(n) = 1$ for $n \neq 0$.

Then, 2-D inverse DCT can be expressed as

$$x(n, m) = \frac{2}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} C(k)C(l)X(k, l)\cos\frac{(2n+1)\pi k}{2N}\cos\frac{(2m+1)\pi l}{2N} \tag{2.4}$$

where $m, n = 0, 1, 2, \ldots, N-1$.

2.1.3 Row-column decomposition of 2-D DCT/IDCT

A straightforward implementation of (2.3) and (2.4) requires N^4 multiplications for the evaluation of the DCT/IDCT. There are several alternative methods to realize the $N \times N$ two-dimensional DCT. These methods can be divided into two main categories, fast DCT algorithms which implement the 2-D DCT using the given 2-D data and the algorithms that use row-column decomposition approach in which the 2-D DCT is implemented by using two one-dimensional DCT and a matrix transpose operation [4].

The row-column decomposition, which has the advantage of regularity for VLSI implementation, is the most popular and effective approach of 2-D DCT computation in image and video coding applications. The 2-D DCT is a separable transform in that it can be expressed in a matrix form involving two 1-D DCT's (the row-column decomposition) as follows.

$$X = A \cdot x \cdot A^T \tag{2.5}$$

where $x = [x(i, j), \quad i, j = 0, 1, \ldots, N-1]$, $X = [X(x, j), \quad i, j = 0, 1, \ldots, N-1]$ and A are $N \times N$ arrays, representing the spatial input data, the frequency domain output data, and the matrix with the cosine basis functions respectively. By the virtue of the orthogonality of A, $A \cdot A^T = I_N$, where I is an $N \times N$ identity matrix. For example, for $N = 8$, we have

$$A = \begin{bmatrix} \alpha & \alpha & \alpha & \alpha & \alpha & \alpha & \alpha & \alpha \\ \beta & \gamma & \Delta & \varepsilon & -\varepsilon & -\Delta & -\gamma & -\beta \\ \Gamma & \delta & -\delta & -\Gamma & -\Gamma & -\delta & \delta & \Gamma \\ \gamma & -\varepsilon & -\beta & -\Delta & \Delta & \beta & \varepsilon & -\gamma \\ \alpha & -\alpha & \alpha & \alpha & \alpha & -\alpha & -\alpha & \alpha \\ \Delta & -\beta & \varepsilon & \gamma & -\gamma & -\varepsilon & \beta & -\Delta \\ \delta & -\Gamma & \Gamma & -\delta & -\delta & \Gamma & -\Gamma & \delta \\ \varepsilon & -\Delta & \gamma & -\beta & \beta & -\gamma & \Delta & -\varepsilon \end{bmatrix} \tag{2.6}$$

where

$$\begin{bmatrix} \alpha \\ \beta \\ \Gamma \\ \gamma \\ \Delta \\ \delta \\ \varepsilon \end{bmatrix} = \sqrt{\frac{2}{N}} \cdot \begin{bmatrix} \cos\frac{\pi}{4} \\ \cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{3\pi}{8} \\ \cos\frac{7\pi}{16} \end{bmatrix} \tag{2.7}$$

Using the row-column decomposition, $\boldsymbol{X}$ can be computed using two 1-D DCT transforms given by

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{x}^T \tag{2.8a}$$

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Y}^T \tag{2.8b}$$

where $\boldsymbol{Y}$ is an intermediate product matrix. The above decomposition to a matrix product results in a reduction in computational complexity to $2N^3$ multiplications and has been used in the implementations of most of the image compression algorithm. A standard

block diagram of computing DCT/IDCT using row-column decomposition is shown in Figure 2.1.

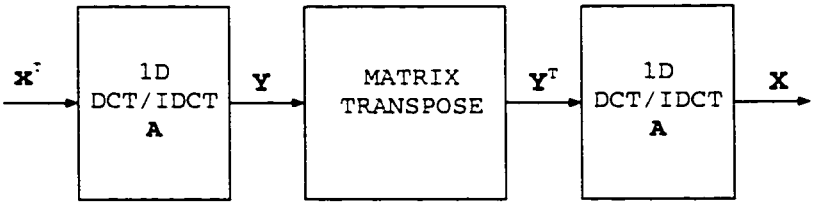

Figure 2.1 2-D DCT/IDCT computation using row-column decomposition.

We can easily see that the first 1-D DCT operates on each row of x while the second one operates on the transpose of Y.

2.2 Distributed Arithmetic for the Computation of the IDCT

Distributed arithmetic is a technique that allows the computation of a sum of products without using multiplication operations. By storing first a finite number of intermediate results, a sum of products can be obtained through repeated additions and shifting operations without the use of explicit multiplication operations. This technique allows the design of signal processors with a reduced gate count and a very regular structure. Hence, it is ideally suited for integrating a DCT/IDCT processor into a video codec.

2.2.1 An overview of the distributed arithmetic [20],[26]

Consider the evaluation of the following sum of products

$$y = \sum_{m=1}^{N} a_m x_m \tag{2.9}$$

where x_m denotes the input data and a_m is fixed coefficient. Without loss of generality, x_m can be assumed to be a two's complement binary number with B-bits of precision and that $|x_m| < 1$. Then, x_m can be expressed as

$$x_m = -x_m^{(0)} + \sum_{j=1}^{B-1} x_m^{(j)} 2^{-j} \tag{2.10}$$

where $x_m^{(j)}$ is the jth bit of x_m and has a value of either 0 or 1. This formulation implies a binary point next to the most significant bit $x_m^{(0)}$. Substituting (2.10) into (2.9) yields

$$y = \sum_{m=1}^{N} a_m \left[-x_m^{(0)} + \sum_{j=1}^{B-1} x_m^{(j)} 2^{-j} \right] \tag{2.11}$$

After an interchange of the order of summations, we have

$$y = \sum_{j=1}^{B-1} \left[\sum_{m=1}^{N} a_m x_m^{(j)} \right] \cdot 2^{-j} - \sum_{m=1}^{N} a_m x_m^{(0)} \tag{2.12}$$

Let

$$F_N(\boldsymbol{a}, x_m^{(j)}) = \sum_{m=1}^{N} a_m x_m^{(j)} \tag{2.13}$$

where $\boldsymbol{a} = [a_1, a_2, \ldots, a_N]^T$. Since $x_m^{(j)}$ can assume a value of either 0 or 1, $F_N(\boldsymbol{a}, x_m^{(j)})$ can assume only 2^N possible values. These values can be precomputed and stored in a lookup-table. Substituting (2.13) into (2.12), yields

$$y = \sum_{j=1}^{B-1} F_N(\boldsymbol{a}, x_m^{(j)}) 2^{-j} - F_N(\boldsymbol{a}, x^{(0)}). \tag{2.14}$$

This is the key expression for the implementation of a sum of products using distributed arithmetic. Multiplication by 2^{-1} corresponds to a right shift by one bit, hence, y in (2.14) can be computed by repeated use of the table look-up, additions, and shift operations.

Figure 2.2 shows a block diagram for the implementation of (2.14). Data is processed bit-serially, the least significant bit first. After each cycle, the output of the accumulator is shifted by one bit. The final sum is computed in B cycles. This circuit is commonly referred to as a ROM-Accumulator (RAC).

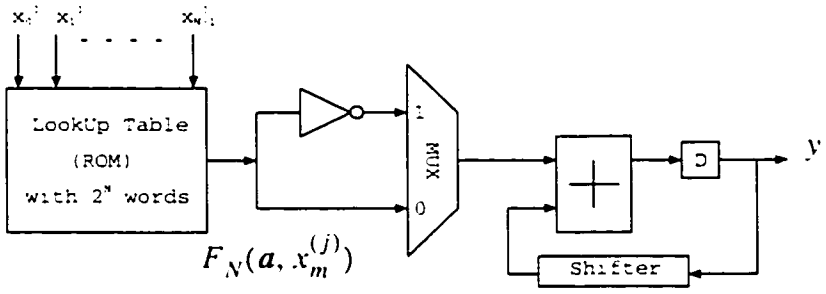

Figure 2.2 Block Diagram of a RAC.

2.2.2 DA-based implementation of the IDCT [16]

The technique described in Section 2.2.1 can now be directly applied to the hardware implementation of the 1-D IDCT. For a B-bit of data precision of the pixel, X can be expressed as

$$X(u) = -X_u^{(0)} + \sum_{j=1}^{B-1} X_u^{(j)} 2^{-j}. \tag{2.15}$$

where $X_u^{(j)}$ denotes the jth bit of $X(u)$ and has a value of either 0 or 1. Substituting (2.15) into (2.2) yields

$$x(i) = \sum_{u=0}^{N-1} \sqrt{\frac{2}{N}} \cdot C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)\left(-X_u^{(0)} + \sum_{j=1}^{B-1} X_u^{(j)} 2^{-j}\right) \tag{2.16}$$

After an interchange of the order of summations, we have

$$x(i) = \sum_{j=1}^{B-1}\left[\sum_{u=0}^{N-1} \alpha_u X_u^{(j)}\right] \cdot 2^{-j} - \sum_{u=0}^{N-1} \alpha_u X_u^{(0)} \tag{2.17}$$

where $\alpha_u = \sqrt{\frac{2}{N}} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)$.

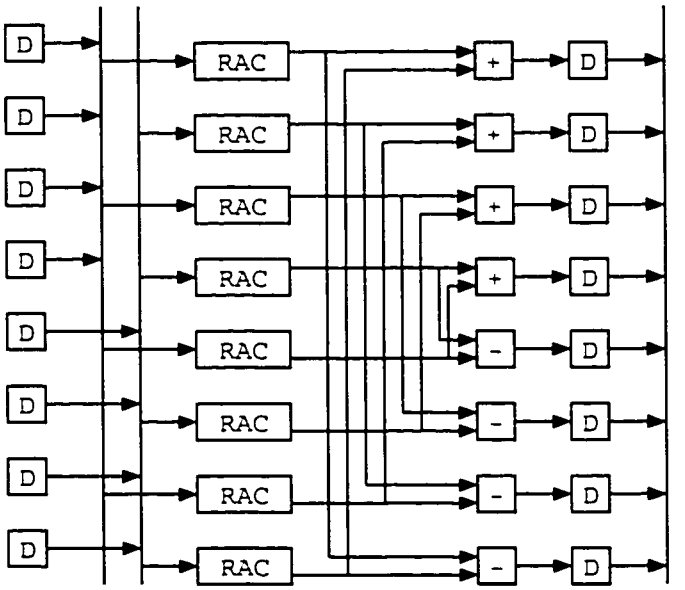

Figure 2.3 Parallel implementation of an eight-point IDCT using distributed arithmetic

Comparing (2.17) with (2.12), it is seen that $x(i)$ can be computed using the ROM-accumulator unit shown in Figure 2.2, where the size of the ROM is 256 words. The size of the ROM can be reduced to only 16 words if we take into consideration that an eight-point IDCT can be computed using two four-point inner-products. The output of the DA based IDCT will be available after B cycles. Figure 2.3 shows a parallel implementa-

tion of the eight-point IDCT using distributed arithmetic. In this implementation each RAC unit, which has 16 words of memory, is realized by the structure shown in Figure 2.2. The '*D*' blocks and '+' blocks in the figure represent the D flip-flops and adders, respectively.

2.2.3 Doubling the speed of distributed arithmetic

To increase the processing rate of the DA-based implementation, the RAC unit can be modified to process two bits from the data at the input of RAC. If we can access two F_N in (2.14) simultaneously, then the calculation of y can be performed in one-half of the number of clock cycles. Equation (2.14) can be expressed as

$$y = \sum_{k=1}^{B/2} F_N(\boldsymbol{a}, x^{(2k-1)})2^{-(2k-1)} + \sum_{k=1}^{B/2-1} F_N(\boldsymbol{a}, x^{(2k)})2^{-2k} - F_N(\boldsymbol{a}, x^{(0)}). \quad (2.18)$$

The above formulation allows odd-numbered and even-numbered bits to be processed in parallel. Figure 2.4 shows the architecture for a modified RAC unit that processes data at double the speed. This design requires an additional adder and twice the ROM size of the original design.

Correspondingly, for the DA-based implementation of the IDCT, (2.17) can be rewritten as

$$x(i) = \sum_{j=1}^{B/2}\left[\sum_{u=0}^{N-1} \alpha_u X_u^{(2j-1)}\right] \cdot 2^{-(2j-1)} + \sum_{j=1}^{B/2}\left[\sum_{u=0}^{N-1} \alpha_u X_u^{(2j)}\right] \cdot 2^{-(2j)} - \sum_{u=0}^{N-1} \alpha_u X_u^{(0)}. \quad (2.19)$$

Thus, the RACs in Figure 2.3 can be realized by the structure shown in Figure 2.4.

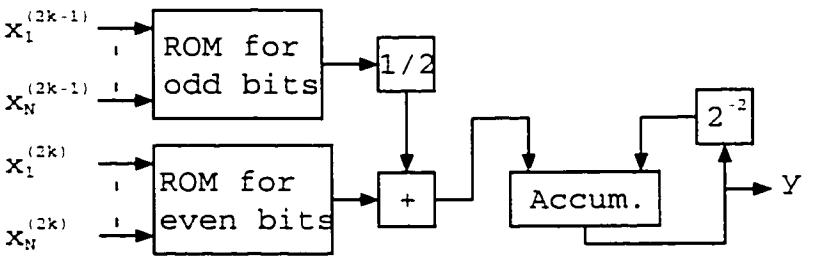

Figure 2.4 Block diagram of an RAC for doubling the speed.

2.3 1-D DCT/IDCT Computation using CORDIC Algorithm

In order to be feasible for high speed applications such as HDTV compression, a new approach in [33] using the CORDIC algorithm has been introduced for computing DCT/IDCT. Only simple adders and registers are needed in this implementation and the computing speed can be very high. In this section, this approach [33] is briefly reviewed.

2.3.1 CORDIC algorithm

COordinate Rotation DIgital Computer (CORDIC) was proposed in [25] to perform vector rotations (and therefore to compute sine, cosine, and arctangent functions) and to multiply or divide numbers using only shift and add elementary operations.

For easy reference, the original CORDIC algorithm is now briefly described. Consider the problem of rotating a vector (V, β) through an angle ϕ. The original vector is expressed in terms of its rectangular components p and q. After rotating, the corresponding components p' and q' of the rotated vector shown in Figure 2.5 are given by

$$\begin{aligned} p' &= p\cos\phi - q\sin\phi \\ q' &= q\cos\phi + p\sin\phi \end{aligned} \tag{2.20}$$

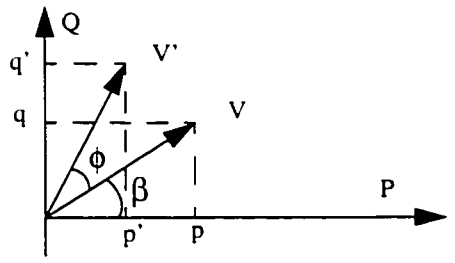

Figure 2.5 Vector (V, β) rotated through an angle ϕ.

Equation (2.20) can be rearranged as:

$$\begin{aligned} p' &= \cos\phi \cdot [p - q\tan\phi] \\ q' &= \cos\phi \cdot [q + p\tan\phi] \end{aligned} \tag{2.21}$$

The CORDIC algorithm computes the components of the vector V' by rotating the vector V through successively smaller angles $\phi_i = \text{atan}\,2^{-i}$, $i = 1,2,....,N$. With each iteration of rotation, a sum is accumulated which, by the final stage of rotation, yields the proper result. The iterative rotation can now be expressed as:

$$p_{i+1} = K_i \cdot [p_i - q_i \cdot d_i \cdot 2^{-i}] \tag{2.22a}$$

$$q_{i+1} = K_i \cdot [q_i + p_i \cdot d_i \cdot 2^{-i}] \tag{2.22b}$$

$$\phi_{i+1} = \phi_i - d_i \cdot \text{atan}\,2^{-i} \tag{2.22c}$$

where

$$d_i = \begin{cases} 1 & when\ \phi_i \geq 0 \\ -1 & when\ \phi_i < 0 \end{cases}.$$

and $K_i = \cos(\text{atan}\,2^{-i})$. It is noted that K_i can be removed from the recursive relations given by (2.22) and the final values of p_i (i.e. p') and q_i (i.e. q') can be multiplied by

$K = \prod_i K_i$. The multiplication by the tangent term in (2.21) is reduced to simple shift operations.

2.3.2 1-D IDCT computation using the CORDIC algorithm

The one-dimensional inverse discrete cosine transform was defined in Section 2.2.1 as

$$x(i) = \sqrt{\frac{2}{N}} \cdot \sum_{u=0}^{N-1} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)X(u) \tag{2.23}$$

where $i = 0,1,...,N-1$, $C(0) = \frac{1}{\sqrt{2}}$, and $C(n) = 1$ when $n \neq 0$.

By using some trigonometric identities of cosine basis functions, we can rewrite (2.23) by decomposing the summation of the right side of the equation into two parts as,

$$\begin{aligned} x(i) &= \sum_{u=0}^{N/2-1} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)X(u) + \sum_{u=N/2}^{N-1} C(u)\cos\left(\frac{2i+1}{2N}u\pi\right)X(u) \\ &= \sum_{u=0}^{N/2-1} g(i,u) \end{aligned} \tag{2.24}$$

where $i = 0, 1, \ldots, N-1$, and

$$g(i,u) = \cos\left(\frac{2i+1}{2N}u\pi\right)X(u) + (-1)^i \sin\left(\frac{2i+1}{2N}u\pi\right)X(N-u)$$

for $u = 1, 2, \ldots, (N/2)-1$, $i = 1, 2, \ldots, N-1$ and

$$g(i,0) = \cos\left(\frac{\pi}{4}\right)X(0) + (-1)^{\left\lceil\frac{i+1}{2}\right\rceil}\sin\left(\frac{\pi}{4}\right)X\left(\frac{N}{2}\right)$$

for $u = 0$. $i = 1, 2, \ldots, N-1$ and

$$g(0,0) = \cos\left(\frac{\pi}{4}\right)X(0) + \sin\left(\frac{\pi}{4}\right)X\left(\frac{N}{2}\right)$$

By using the symmetries in the trigonometric functions, the above function $g(i,u)$ can be expressed in different forms. Letting $K = 1, 2, 4, \ldots, N/2$, and $u = K, 2K, 3K, \ldots$, we can have

$$\begin{aligned} g\left(\frac{N}{K}-1-i,u\right) &= (-1)^{u/K}\left[\cos\left(\frac{2i+1}{2N}u\pi\right)X(u) + (-1)^i\sin\left(\frac{2i+1}{2N}u\pi\right)X(N-u)\right] \\ &= (-1)^{u/K}g(i,u) \\ &= a(i,u) \end{aligned} \tag{2.25}$$

For if $K = 2, 4, 8, \ldots, N/2$, and $u = K/2, 3K/2, 5K/2, \ldots$, we can have

$$\begin{aligned} g\left(\frac{N}{K}-1-i,u\right) &= (-1)^{\frac{\left(u-\frac{K}{2}\right)}{K}}\left[\sin\left(\frac{2i+1}{2N}u\pi\right)X(u) - (-1)^i\cos\left(\frac{2i+1}{2N}u\pi\right)X(N-u)\right] \\ &= b(i,u) \end{aligned} \tag{2.26}$$

Let

$$\begin{aligned} g(1,0) &= -\sin\left(\frac{\pi}{4}\right)X(0) + \cos\left(\frac{\pi}{4}\right)X\left(\frac{N}{2}\right) \\ &= -b(0,0) \end{aligned} \tag{2.27}$$

Exploiting the symmetries of the trigonometric functions included in (2.25) to (2.27), it can be shown that a complete set of values $g(i,u)$, which is needed for the computation of $x(i)$ according to (2.24), can be calculated from only $(N^2+8)/12$ values of the pair $(a(i,u), b(i,u))$. For example, for $N = 8$, only 6 different pairs $(a(i,u), b(i,u))$ are

needed for the computation of the $g(i, u)'$s. The following equation gives the mapping of $g(i,u)$ and $(a(i, u), b(i, u))$ for $N = 8$.

$$\begin{bmatrix} g(0,0) & g(0,1) & g(0,2) & g(0,3) \\ g(1,0) & g(1,1) & g(1,2) & g(1,3) \\ g(2,0) & g(2,1) & g(2,2) & g(2,3) \\ g(3,0) & g(3,1) & g(3,2) & g(3,3) \\ g(4,0) & g(4,1) & g(4,2) & g(4,3) \\ g(5,0) & g(5,1) & g(5,2) & g(5,3) \\ g(6,0) & g(6,1) & g(6,2) & g(6,3) \\ g(7,0) & g(7,1) & g(7,2) & g(7,3) \end{bmatrix} = \begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & a(0,3) \\ b(0,0) & -a(1,1) & b(0,2) & -a(1,3) \\ b(0,0) & b(1,1) & b(0,2) & -b(1,3) \\ a(0,0) & b(0,1) & -a(0,2) & b(0,3) \\ a(0,0) & b(0,1) & -a(0,2) & -b(0,3) \\ b(0,0) & -b(1,1) & b(0,2) & b(1,3) \\ b(0,0) & -a(1,1) & a(1,3) & -b(0,2) \\ a(0,0) & -b(0,1) & a(0,2) & -b(0,3) \end{bmatrix} \tag{2.28}$$

Comparing (2.25) and (2.26) with the vector rotation transform (2.20) and (2.20) respectively, we see that only 6 CORDIC computations are needed for an 8-point 1-D IDCT. A block diagram for the computation of the 8-point 1-D IDCT using CORDIC rotators is shown in Figure 2.6.

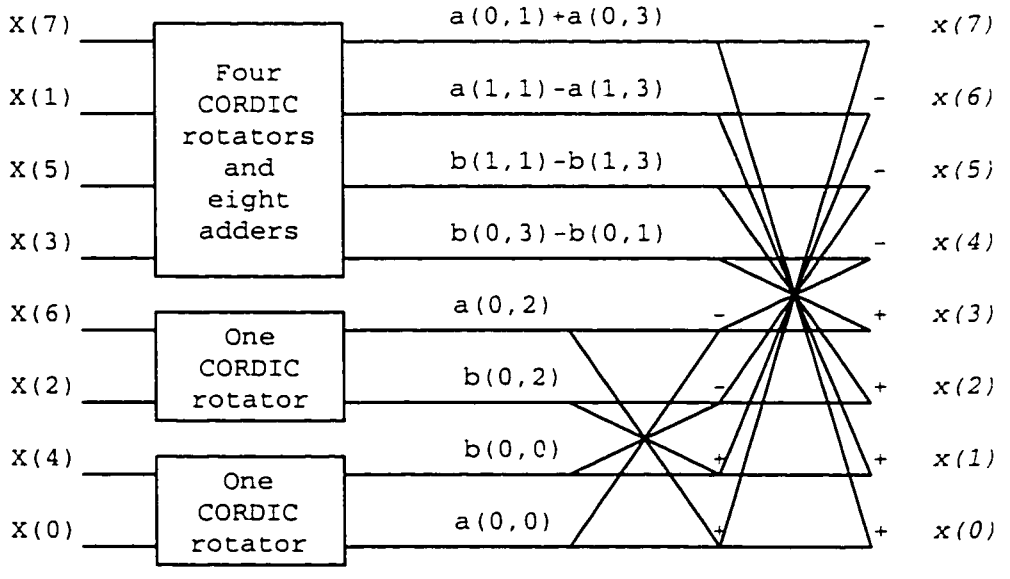

Figure 2.6 A block diagram of the 8-point 1-D IDCT using the CORDIC rotators.

2.4 Comparison of the Existing Design Approaches

In general, the basis of comparison for the various DCT/IDCT algorithms is the required number of multiplications and additions. However, there are also many other factors, such as complexity of control logic, requirements for memory size, power consumption, complexity of interconnect, etc., that can be taken into consideration. Although fast DCT algorithms, which directly implement 2-D DCT, are superior to the row-column DCT in the sense that the numbers of multiplication-accumulation operations are reduced to one-half, the irregular routing cost and the use of additional reording circuit make it to require even more area in actual VLSI implementations.

The distributed arithmetic is widely used in DSP integrated circuit design because of its ability of providing regular structures with smaller gate counts. In the context of the architecture of Figure 2.3, there can be three potential bottlenecks limiting the speed of IDCT computation using the DA. The first one is the ROM access time, especially when the ROM size is large. The ROM size in the conventional DA approach increases exponentially with the increment of the size of the image sub-block. The second one is the speed of the accumulators. In the most applications of image/video compressions, the internal wordlength is normally large in order to meet the critical accuracy requirement of the standards. Thus, the accumulators with large wordlength can dominate the speed of the whole system. The last bottleneck is the speed of the butterfly adder array. Normally, in order to improve the performance of the circuits, a large area is a compensation for achieving a higher speed. For example, a 16-bit carry-select adder (CSA) is 50% faster than a 16-bit

carry ripple adder (CRA), whereas the former is also 50% larger in area than the latter [24]. Besides, since DA is essentially a bit-serial arithmetic structure, and in the least significant bit first processing format, the sign bit is processed last, the accumulators can provide results only after every B clock cycles, where B is the wordlength of the input samples, for the subsequent operations.

In the CORDIC approach discussed in Section 2.3, the 1-D IDCT/DCT is implemented by six CORDIC rotators and two stages of butterfly adders in order to take advantage of the multiplierless property of the CORDIC algorithm. Although a direct implementation of the CORDIC algorithm for the computation of the IDCT presented in Section 2.3 is better than the multiplier-based approach, it does not provide the results that are superior to that of the DA scheme. It is because the outputs of the CORDIC rotator need to be scaled by a constant to obtain the correct results. Thus, the additional hardware to perform this multiplication eliminates the advantage of the CORDIC algorithm.

2.5 Summary

In this chapter, some background material on the mathematical description of the IDCT algorithms and their implementations necessary for the study undertaken in this thesis has been provided. In Section 2.1, the definition of the discrete cosine transform and the representation of its inverse have been presented. The row-column decomposition approach for the 2-D DCT computation has been given. The distributed arithmetic and its implementation for 1-D IDCT was described in Section 2.2. In Section 2.3, the CORDIC algorithm and the CORDIC based approach for the IDCT implementation has been briefly

reviewed. Finally, the advantages and disadvantages of the existing design approaches based on the distributed arithmetic and the CORDIC algorithm have been discussed in Section 2.4.

By using a lookup table (ROM) to store a finite number of intermediate results, the DA technique allows the design of an IDCT algorithm with a reduced gate count and a regular structure. However, this advantage is diminished because of the requirement of a large ROM size and the slow speed due to the large access time of the ROM and the bit serial operation of the DA. By employing the CORDIC algorithm, the IDCT computation can also be implemented without using multipliers, but unlike the DA implementation, it does not require the use of a ROM. However, the deep pipeline structure employed in the implementation of the CORDIC rotator due to its iterative property makes this approach require a large silicon area. In the next chapter, these problems of the existing approaches of the IDCT design and implementation will be addressed as a part of the investigation undertaken in this thesis.

Chapter 3

Algorithm Design of IDCT Core

In the previous chapter, various algorithms and architectures proposed for the IDCT implementation were presented. Among these algorithms, the distributed arithmetic (DA) based one is the most widely used technique for the DCT/IDCT implementation because of its simple and multiplierless structure. By storing the pre-computed intermediate results of multiplication in a lookup table, the IDCT computation is implemented simply by a structure containing only ROMs, adders and registers. However, the size of the ROM employed in DA approach is an exponential function of the size of the image block. Thus, the speed of operation of the DA based IDCT implementation is mainly determined by the large access time of the ROM. The CORDIC algorithm, also discussed in proceeding chapter, was introduced in [33] for the IDCT implementation in order to meet the high- speed requirement of applications such as HDTV. In this approach, the image pixels are divided into groups of two and processed in parallel by the CORDIC rotators to achieve a high-speed operation. However, due to the iterative property of the CORDIC algorithm, a deep pipeline structure must be employed in the CORDIC rotator, requiring large area for its implementation. Thus, it would seem that by combining the DA-based approach with that of the CORDIC algorithm, it should be possible to reduce the hardware

complexity and also achieve a reasonable speed of operation, if one could replace the deep pipeline structure of the latter by ROMs of smaller size.

In this chapter, a new DA-based CORDIC algorithm for the implementation of the IDCT computation is proposed [30]. In this scheme, the IDCT of the transform values, which are divided into groups of two, are carried out by several CORDIC rotators and butterfly adders. The distributed arithmetic technique is employed in the CORDIC rotator to avoid the usage of a deep pipeline structure. The proposed IDCT core employs the redundant number system and on-line digit-serial architecture to achieve a high operation speed.

In Section 3.1, the algorithm design of the DA-based CORDIC algorithm for a 1-D IDCT implementation is presented. In order to improve the speed of the overall system, a scheme [31] of doubling the speed of the implementation is given in Section 3.2. In Section 3.3, in order to further optimize the design at the bit-level architecture, the redundant number system is employed in the IDCT core. The carry-free property of the redundant number addition makes it possible to process data serially in the manner of most significant digit first (MSDF), which has been referred as on-line algorithm in [10].

3.1 A DA Based CORDIC Algorithm for the 1-D IDCT Computation

The expression of the 8-point 1-D IDCT can be represented by the following equation

$$x(i) = \frac{1}{2} \sum_{u=0}^{7} C(u) \cos\left(\frac{2i+1}{2\times 8} u\pi\right) X(u), \quad i = 0, 1, 2, \ldots, 7 \tag{3.1}$$

where $c(0) = \frac{1}{\sqrt{2}}$, and $c(n) = 1$ for $n \neq 0$. By using the technique presented in Section 2.3, the various value of $x(i)$ given by this equation can be computed as

$$\begin{aligned}
x(0) &= a(0,0) + a(0,1) + a(0,2) + a(0,3) \\
x(1) &= b(0,0) + a(1,1) - b(0,2) - a(1,3) \\
x(2) &= b(0,0) + b(1,1) + b(0,2) - b(1,3) \\
x(3) &= a(0,0) - b(0,1) - a(0,2) + b(0,3) \\
x(4) &= a(0,0) + b(0,1) - a(0,2) - b(0,3) \\
x(5) &= b(0,0) - b(1,1) + b(0,2) + b(1,3) \\
x(6) &= b(0,0) - a(1,1) - b(0,2) + a(1,3) \\
x(7) &= a(0,0) - a(0,1) + a(0,2) - a(0,3)
\end{aligned} \tag{3.2}$$

where

$$\begin{aligned}
b(0,0) &= X(0)\cos\left(\frac{\pi}{4}\right) - X(4)\sin\left(\frac{\pi}{4}\right) \\
a(0,0) &= X(4)\cos\left(\frac{\pi}{4}\right) + X(0)\sin\left(\frac{\pi}{4}\right)
\end{aligned} \tag{3.3}$$

$$\begin{aligned}
b(0,2) &= X(6)\cos\left(\frac{\pi}{8}\right) - X(2)\sin\left(\frac{\pi}{8}\right) \\
a(0,2) &= X(2)\cos\left(\frac{\pi}{8}\right) + X(6)\sin\left(\frac{\pi}{8}\right)
\end{aligned} \tag{3.4}$$

$$\begin{aligned}
b(0,3) &= X(5)\cos\left(\frac{3\pi}{16}\right) - X(3)\sin\left(\frac{3\pi}{16}\right) \\
a(0,3) &= X(3)\cos\left(\frac{3\pi}{16}\right) + X(5)\sin\left(\frac{3\pi}{16}\right)
\end{aligned} \tag{3.5}$$

$$\begin{aligned}
b(1,3) &= X(3)\cos\left(\frac{\pi}{16}\right) - X(5)\sin\left(\frac{\pi}{16}\right) \\
a(1,3) &= X(5)\cos\left(\frac{\pi}{16}\right) + X(3)\sin\left(\frac{\pi}{16}\right)
\end{aligned} \tag{3.6}$$

$$\begin{aligned} b(0,1) &= X(1)\cos\left(\frac{3\pi}{16}\right) - X(7)\sin\left(\frac{3\pi}{16}\right) \\ a(0,1) &= X(7)\cos\left(\frac{3\pi}{16}\right) + X(1)\sin\left(\frac{3\pi}{16}\right) \end{aligned} \tag{3.7}$$

$$\begin{aligned} b(1,1) &= X(7)\cos\left(\frac{\pi}{16}\right) - X(1)\sin\left(\frac{\pi}{16}\right) \\ a(1,1) &= X(1)\cos\left(\frac{\pi}{16}\right) + X(7)\sin\left(\frac{\pi}{16}\right) \end{aligned} \tag{3.8}$$

As shown in Figure 2.6, (3.3) to (3.8) can be computed by six CORDIC rotators using the rotation transform given by

$$\begin{aligned} p' &= p\cos\phi - q\sin\phi \\ q' &= q\cos\phi + p\sin\phi \end{aligned} \tag{3.9}$$

The p and q coordinates in (3.9) can represent pixel values. This equation can be computed using the distributed arithmetic techniques. For this purpose, the coordinates p and q can be expressed as

$$\begin{aligned} p &= -p^{(0)} + \sum_{j=1}^{B-1} p^{(j)} 2^{-j} \\ q &= -q^{(0)} + \sum_{j=1}^{B-1} q^{(j)} 2^{-j} \end{aligned} \tag{3.10}$$

where the pixel values p and q are represented by a B-bit precision, and $p^{(j)}$ and $q^{(j)}$ represent the *jth* bit of p and q, respectively. Substituting for p and q from (3.10) into (3.9) yields

$$
\begin{aligned}
p' &= \left(-p^{(0)} + \sum_{j=1}^{B-1} p^{(j)} 2^{-j}\right)\cos\phi - \left(-q^{(0)} + \sum_{j=1}^{B-1} q^{(j)} 2^{-j}\right)\sin\phi \\
q' &= \left(-q^{(0)} + \sum_{j=1}^{B-1} q^{(j)} 2^{-j}\right)\cos\phi + \left(-p^{(0)} + \sum_{j=1}^{B-1} p^{(j)} 2^{-j}\right)\sin\phi
\end{aligned}
\tag{3.11}
$$

The above equation can be rewritten as

$$
\begin{aligned}
p' &= -(p^{(0)}\cos\phi - q^{(0)}\sin\phi) + \sum_{j=1}^{B-1} (p^{(j)}\cos\phi - q^{(j)}\sin\phi)\cdot 2^{-j} \\
&= -R_P(\phi, p^{(0)}, q^{(0)}) + \sum_{j=1}^{B-1} R_P(\phi, p^{(j)}, q^{(j)}) \\
q' &= -(q^{(0)}\cos\phi + p^{(0)}\sin\phi) + \sum_{j=1}^{B-1} (q^{(j)}\cos\phi + p^{(j)}\sin\phi)\cdot 2^{-j} \\
&= -R_Q(\phi, p^{(0)}, q^{(0)}) + \sum_{j=1}^{B-1} R_Q(\phi, p^{(j)}, q^{(j)})
\end{aligned}
\tag{3.12}
$$

where

$$
\begin{aligned}
R_P(\phi, p^{(j)}, q^{(j)}) &= p^{(j)}\cos\phi - q^{(j)}\sin\phi \\
R_Q(\phi, p^{(j)}, q^{(j)}) &= q^{(j)}\cos\phi + p^{(j)}\sin\phi \cdot \\
j &= 0, 1, \ldots, B-1
\end{aligned}
\tag{3.13}
$$

The equations given by (3.12) constitute the key expression for the computation of p' and q' using a distributed arithmetic based CORDIC approach. Since each $p^{(j)}$ and $q^{(j)}$ may assume the value of either 0 or 1, the intermediate results $R_P(\phi, p^{(j)}, q^{(j)})$ and $R_Q(\phi, p^{(j)}, q^{(j)})$ can take only 2^2 possible values. These values can be precomputed and

stored in a look-up table. The equation given by (3.12) can be computed in B clock cycles by two ROM accumulators (RAC), as shown in Figure 3.1.

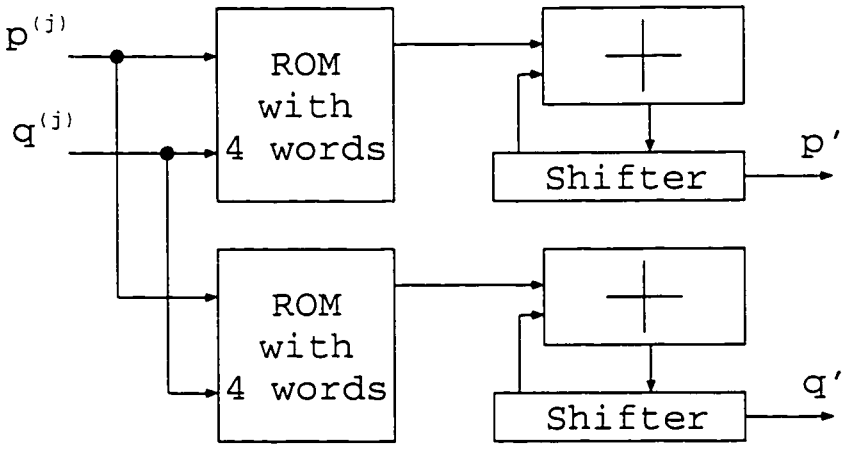

Figure 3.1 Block diagram of DA-based rotator.

The 1-D IDCT as expressed in (3.1) can be computed in two steps. First, the equations (3.3) to (3.8) are calculated by six rotators each as shown in Figure 3.1. Then (3.2) is implemented by two stages of butterfly adders. Figure 3.2 shows a block diagram for the implementation of the 1-D IDCT using the proposed DA-based CORDIC algorithm. It can be seen from this structure that a total of six 8-word ROMs are needed for the proposed IDCT design, compared with eight 16-word ROMs required by the conventional DA-based IDCT presented in Section 2.2.2. The ROM requirement is thus reduced by 63%. For each rotator as shown in Figure 3.1, there can be only two inputs, p and q coordinates respectively. Thus the size of the ROM in each rotator in the IDCT implementation is independent of the size of the image block, as compared to the DA approach presented in Section 2.2.2 in which the size of the ROM in each RAC increases exponentially with the increment of the image block size. However, the proposed approach still has the main dis-

advantage of slow processing speed of the DA-based approach due to the bit-serial nature of the distributed arithmetic. As the RAC of the DA-based approach, the rotator in the proposed design also needs B clock cycles to perform one accumulation operation. In the next section, an improved scheme of CORDIC rotator that doubles the processing speed of the accumulation is presented.

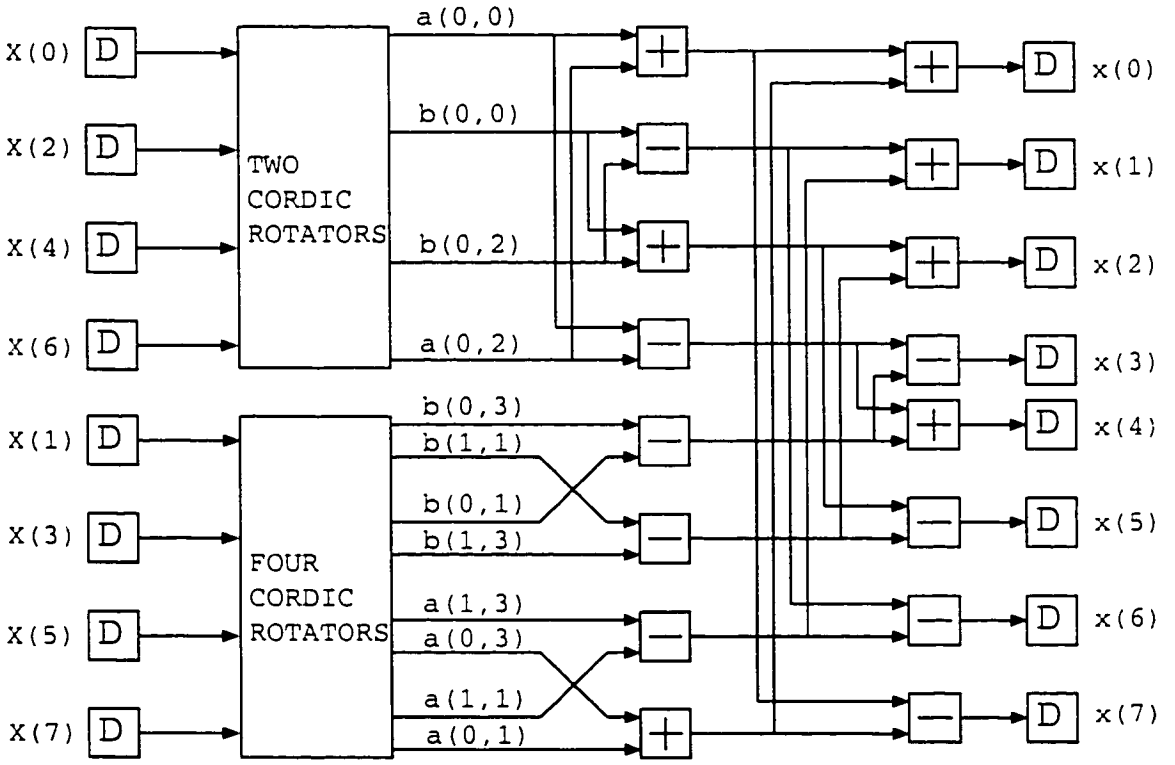

Figure 3.2 Implementation of 1-D IDCT using the proposed DA-based CORDIC algorithm.

3.2 Speed Improvement of the DA Based CORDIC Algorithm

In order to meet the accuracy requirements specified by the standards, the DCT/IDCT architectures normally employ a large internal wordlength in most video applica-

tions. The bit-serial DA approach presented in the previous section suffers from the low throughput and large latency. To increase the processing speed of the distributed arithmetic, a modified DA scheme [3] was presented in Section 2.2.3 to process the odd-numbered and even-numbered bits in parallel. This, however, resulted in an increase in hardware of one adder and doubling the size of the ROM for each RAC. Further, the bit-parallel structure must be employed for the additional adder, since it is followed by an accummulator. Depending on the accuracy requirement of an application standard, a parallel adder with a large wordlength may be needed. Thus, the speed of the additional adder may determine the overall speed of the entire IDCT core.

Benefiting from the small ROM size requirement of the DA-based CORDIC approach as described by (3.12), we now propose to double the speed of the DA architecture without requiring of an additional adder [31]. In (3.12), instead of processing one bit of p and one bit of q at a time, we take two bits each from p and q in each clock cycle. Thus, the latency of our proposed RAC is reduced to $B/2$, where B is the wordlength of p and q. Equation (3.12) can be rewritten as

$$\begin{aligned} p' &= (p^{(1)}\cos\phi - q^{(1)}\sin\phi) - 2(p^{(0)}\cos\phi - q^{(0)}\sin\phi) \\ &\quad + \sum_{k=1}^{B/2-1} (p^{(2k)}\cos\phi - q^{(2k)}\sin\phi)2^{-(2k)} \cdot 2 \\ &\quad + \sum_{k=1}^{B/2-1} (p^{(2k+1)}\cos\phi - q^{(2k+1)}\sin\phi)2^{-(2k+1)} \\ &= R_Q(\phi, p^{(0)}, q^{(0)}) + \sum_{k=1}^{B/2-1} R_Q(\phi, p^{(k)}, q^{(k)}) \end{aligned}$$

$$
\begin{aligned}
q' &= (q^{(1)}\cos\phi + p^{(1)}\sin\phi) - 2(p^{(0)}\cos\phi - q^{(0)}\sin\phi) \\
&\quad + \sum_{k=1}^{B/2-1} (q^{(2k)}\cos\phi + p^{(2k)}\sin\phi)2 \cdot 2^{-(2k+1)} \\
&\quad + \sum_{k=1}^{B/2-1} (q^{(2k+1)}\cos\phi + p^{(2k+1)}\sin\phi)2^{-(2k+1)} \\
&= R_P(\phi, p^{(0)}, q^{(0)}) + \sum_{k=1}^{B/2-1} R_P(\phi, p^{(k)}, q^{(k)})
\end{aligned}
\tag{3.14}
$$

where

$$
\begin{aligned}
R_P(\phi, p, q, k) &= 2(p^{(2k)}\cos\phi - q^{(2k)}\sin\phi) + p^{(2k+1)}\cos\phi - q^{(2k+1)}\sin\phi \\
R_Q(\phi, p, q, k) &= 2(q^{(2k)}\cos\phi + p^{(2k)}\sin\phi) + q^{(2k+1)}\cos\phi + p^{(2k+1)}\sin\phi \\
R_P(\phi, p, q, 0) &= (p^{(1)}\cos\phi - q^{(1)}\sin\phi) - 2 \cdot (p^{(0)}\cos\phi - q^{(0)}\sin\phi) \\
R_Q(\phi, p, q, 0) &= (q^{(1)}\cos\phi + p^{(1)}\sin\phi) - 2 \cdot (q^{(0)}\cos\phi + p^{(0)}\sin\phi)
\end{aligned}
\tag{3.15}
$$

In the set of equations given by (3.15), $R_P(\phi, p, q, k)$ and $R_Q(\phi, p, q, k)$ can take only 2^4 possible values, since $p^{(k)}$ and $q^{(k)}$ may assume a value of either 0 or 1 only. Two 16-word ROMs are used in each rotator to store the intermediate results of (3.14) as given in (3.15). A total of twelve 16-word ROMs are required by the 1-D IDCT architecture. Compared with the scheme presented in Section 3.1, the doubling of the speed is achieved without an increase in the number of additional adders in the rotator. Moreover, the total ROM size of 192 words employed in this design is still smaller than the ROM size of 256 words needed in the conventional DA-based IDCT design described in Section 2.2.3.

3.3 Bit-Level Architectural Details of the Proposed IDCT Core

In this section, the bit-level architecture for the general DA-based CORDIC structure for the 1-D IDCT computation proposed in Sections 3.1 and 3.2 is presented. In order to fully understand the realization of the bit-level architecture of the proposed design, the concept of the redundant number system and its implementation in the IDCT core are also addressed in this section.

There are three kinds of implementations of bit-level architectures in DSP algorithm designs, bit-parallel, bit-serial, and digit-serial. Bit-parallel systems process one whole word of the input sample each clock cycle and are ideal for high-speed applications. Bit-serial systems process one bit of the input sample every clock cycle. Such systems are area-efficient and suitable for low-speed applications [9]. Digit-serial systems [12] process multiple number of bits every clock cycle and are best suited for applications requiring moderate sample rate, where area and power consumption are critical. For the DA-based CORDIC algorithm presented in the previous section, the digit-serial architecture is obviously the best choice of implementations in view of the critical area requirement of the video codec applications. This choice of the architecture also conforms with the rotators which also has a digit-serial structure.

A digit-serial system can be designed for processing either the most significant digit first (MSDF) or the least significant digit first (LSDF). For a two's complement representation, the most significant bit (MSB) represents the sign of a number. In the rotators shown in Figure 3.1, if LSDF mode of operation is adopted, the sign of the accumulator's

content cannot be determined until the last bit of the input operand (the most significant bit) is processed. This technique has the disadvantage of presenting the results in a parallel form and thus requires re-serialization, since the subsequent stage to which the results need to be passed on are bit-serial adders as shown in Figure 3.2. The re-serialization can result in a complex control logic, additional buffers and lower performance. On the contrary, if the on-line algorithm which processes the data in an MSDF mode is adopted, the results of the accumulator can be passed on to the adder array before completing the entire accumulation operation. Thus, the accumulators can be chained with the subsequent adders. Because of these features of the MSDF processing, we adopt this processing mode in our design. An algorithm based on the MSDF processing has been referred to as on-line algorithm in the literature [10].

In order to facilitate the computation of a task using on-line algorithm, the data must be represented using the redundant number system. The carry-free property of the redundant number system and the MSDF mode of operation provide a fast operation speed and short latency and thus allow the subsequent calculations to start earlier. Binary signed digit (BSD) representation is a special case of redundant data representation and widely used in on-line algorithms. The basic concepts of the binary signed digit representation is explained in Section 3.3.1. The carry-free addition, which is the main attraction of using the redundant signed-digit number systems, is presented in Section 3.3.2. The scheme of on-line addition, making use of some of the features of the parallel addition of redundant numbers is described in Section 3.3.3.

3.3.1 Binary signed digit representation

In the conventional nonredundant radix-r number system, a digit can take on values from the set $\{0, 1, \ldots, r-1\}$, and all the numbers can be represented in a unique way. A radix-r redundant signed-digit number system is given by the set $\{\bar{\beta}, \overline{\beta-1}, \ldots, \bar{1}, 0, 1, \ldots, \alpha\}$, where the notation $\bar{x}$ denotes $-x$, $1 \le \alpha, \beta \le r-1$, and the digit set contains more than r values. This number system allows multiple representations of any number, thus the name redundant number system [1]. It has the advantage over the conventional number representations in that the addition of two numbers may be performed without (or with a limited) carry-propagation. In our design, the binary signed digit representation is used for the accumulation operation. A number in the BSD format can be written as

$$X = \sum_{i=1}^{N} x_i \cdot 2^{-i}, \quad x_i \in \{-1, 0, 1\} \tag{3.16}$$

In the implementation of the proposed IDCT core, an element of the digit set $\{-1, 0, 1\}$ is constructed using a two-bit binary code, as shown in Table 3.1. Each digit of a BSD number can be -1, 0 or 1, and the most significant non-zero digit of this number represents the sign of the number. If this MSD is +1, the BSD number is positive, otherwise it is a negative number.

Table 3.1 Binary Signed-Digit

Digit	Binary Code	
x_i	x_i^+	x_i^-
0	0	0
-1	0	1
1	1	0
0	1	1

3.3.2 Carry-free addition

With the redundant number system, it is possible to perform addition with or without limited carry propagation. The procedure of the carry-free hybrid addition of the binary and BSD number is now explained.

Consider the addition of a BSD number X and an unsigned conventional binary number Y, each with the wordlength of B:

$$S = X + Y \tag{3.17}$$

where $X = x_1x_2x_3\ldots x_B$, $Y = y_1y_2y_3\ldots y_B$, $x_i \in \{\bar{1}, 0, 1\}$, and $y_i \in \{0, 1\}$. This addition can be carried out in 2 steps. The first step is carried out in parallel for all the digit positions $i(1 \le i \le B)$: an intermediate sum $p_i = x_i + y_i$ is computed, which belongs to the set $\{\bar{1}, 0, 1, 2\}$. This step is expressed as

$$x_i + y_i = p_i = 2t_i + u_i \tag{3.18}$$

where $t_i \in \{0, 1\}$ is the transfer digit and, $u_i \in \{-1, 0\}$ is the interim sum. The least significant transfer digit t_{W+1} is assigned a value of zero. The most significant interim sum digit u_0 is also assigned a value of zero. In the second step, the sum digit s_i is formed by combining t_{i+1} and u_i into one digit as follows:

$$s_i = t_{i+1} + u_i. \tag{3.19}$$

As shown in (3.18) and (3.19), the sum digit s_i depends on x_i, x_{i+1}, y_i and y_{i+1} only. As a result of this, the carry propagation is limited to only one digit position. Table 3.2 summarizes the digit sets used in the redundant hybrid radix-2 addition.

Table 3.2 Digit Sets Involved in Hybrid Radix-2 Addition

Digit	Radix 2 Digit Set	Binary Code	Digit Value
x_i	$\{\bar{1}, 0, 1\}$	$x_i^+ x_i^-$	$x_i^+ - x_i^-$
y_i	$\{0, 1\}$	y_i^+	y_i^+
$p_i = x_i + y_i$	$\{\bar{1}, 0, 1, 2\}$	...	...
u_i	$\{\bar{1}, 0\}$	u_i^-	$-u_i^-$
t_i	$\{0, 1\}$	t_i^+	t_i^+
$s_i = u_i + t_{i+1}$	$\{\bar{1}, 0, 1\}$	$s_i^+ s_i^-$	$s_i^+ - s_i^-$

In the DA-based rotator shown in Figure 3.1, a hybrid radix-2 BSD parallel adder, which performs an addition of a BSD number and an unsigned binary number, is used in the accumulator. The carry-free property of the hybrid BSD adder carries two advantages.

First, the speed of this accumulator is comparable with that of the carry-look-ahead adder with a much smaller area requirement. Second, since the propagation of the carry is limited to only one digit position, the most significant digit of the accumulation result can be shifted out from the rotator in every clock cycle while the accumulation operation is still being performed. In order to conform with the operation of the rotator, the downstream adder array shown in Figure 3.2 need to process the output digit of the rotators in the MSDF mode. Thus, adders using on-line algorithm are required for the implementation of the adder array.

3.3.3 Addition using an on-line algorithm

The one-digit on-line adders, which perform the addition of two signed-digit numbers serially in the MSDF mode, are adopted in the butterfly adder array shown in Figure 3.2. The signed-digit addition using on-line algorithm can be derived from the parallel addition of two signed-digit numbers. In this section, the scheme of the parallel addition of two binary signed-digit numbers is first explained. Then, by taking advantage of the carry-free property of the addition of redundant numbers, the scheme of parallel addition is reformed to achieve the on-line addition of the two BSD numbers.

As shown in Table 3.1, a BSD digit x_i can be considered as the subtraction $x_i^+ - x_i^-$ of the components of its binary code. Consequently, a BSD number X can also be considered as the subtraction of X^+ and X^-, where $X^+ = x_1^+ x_2^+ x_3^+ \ldots x_B^+$ and $X^- = x_1^- x_2^- x_3^- \ldots x_B^-$.

Therefore, the addition of two binary signed digit numbers X and Y can be computed by performing hybrid addition and hybrid subtraction sequentially as

$$S = X + Y = X + Y^+ - Y^- \tag{3.20}$$

$$\Rightarrow \begin{aligned} Z &= X^+ - X^- + Y^+, \\ S &= Z^+ - Z^- - Y^- \end{aligned}$$

where $X = x_1x_2x_3\ldots x_B$, $Y = y_1y_2y_3\ldots y_B$, $x_i \in \{\bar{1}, 0, 1\}$ and $y_i \in \{\bar{1}, 0, 1\}$. The above operations of addition and subtraction can be carried digit by digit in parallel. The addition can be carried out in two steps. First, the hybrid addition of the signed digit number x_i and the unsigned binary number y_i^+ is performed to generate the intermediate result z_i as

$$\begin{aligned} x_i + y_i^+ &= 2t_i + u_i \\ z_i &= t_{i+1} + u_i \end{aligned} \tag{3.21}$$

where z_i is also a BSD digit and $Z = z_1z_2z_3\ldots z_W$. Then, the hybrid subtraction of the BSD digit z_i and the unsigned binary digit y_i^- is carried out as

$$z_i - y_i^- = 2v_i + w_i \tag{3.22a}$$

$$s_i = v_{i+1} + w_i \tag{3.22b}$$

where $w_i \in \{0, 1\}$, $v_i \in \{0, -1\}$. Note that the sum digit s_i depends on x_i, x_{i+1}, x_{i+2}, y_i, y_{i+1} and y_{i+2} only. Thus, the carry propagation is limited to only two digit positions.

For the conventional nonredundant number system, it is impossible to perform the operation of the on-line addition, which is in digit-serial MSDF mode, due to the factor that the carry propagation is from the least significant digit to the most significant digit. On the other hand, the carry-free property of the redundant number addition makes it possible to perform the on-line addition. By observing the operation of the parallel addition of the two BSD number, it would seem that by memorizing the intermediate results u_{i+2}, w_{i+1} and y_{i+2}^{-}, it is possible to compute the result of digit s_i without the information of any other less significant digit. Thus, the on-line addition can be shown as the state diagram in Figure 3.3. The first circle represents the computation of (3.21). The second circle represents the computation of (3.22a). The third circle represents the combination of the intermediate results w_i and v_{i+1} to form the result s_i, as given by (3.22b). The squares in the figure represents the memory cells which could be D flip flops in an actual VLSI implementations. To generate the first digit of the result s_i, 2 digits of the input operands x_{i+2}, y_{i+2} are needed. That is, after the two most significant digits of the operands are received, the most significant digits of the result is generated.

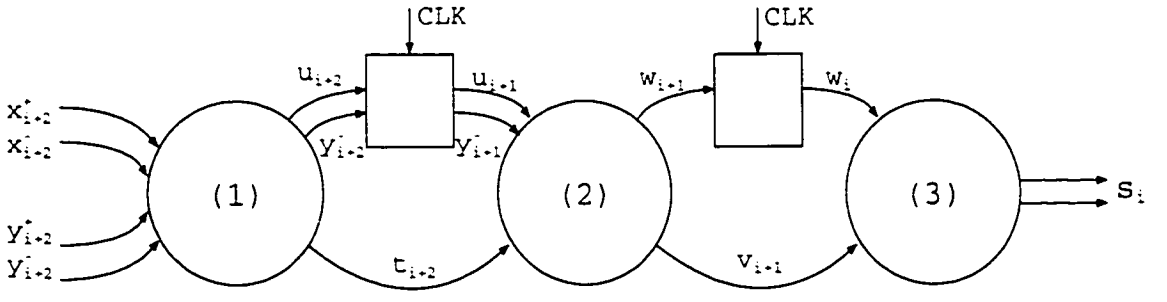

Figure 3.3 The scheme for the on-line addition.

For the signed-digit on-line algorithm, it is quite easy to implement a subtractor because negation can be performed by just exchanging positions of the respective positive bit and negative bit. For example, the scheme shown in Figure 3.3 can performs the subtraction $x_{i+2} - y_{i+2}$ by simply exchanging the position of y^{+}_{i+2} and y^{-}_{i+2}.

A total of 16 one-bit on-line adders are needed in each of the two 1-D IDCT cores. By using the on-line adder, the data stream can flow through the rotators and the butterfly adders without any buffering and re-serialization.

3.4 Summary

In this chapter, a new algorithm for the 1-D IDCT computation and its bit-level architecture employing distributed arithmetic and CORDIC rotators for its implementation have been presented. The DA-based CORDIC algorithm combines the DA-based approach for the IDCT computation with that of the CORDIC algorithm. In this scheme, the 8-point 1-D IDCT is decomposed and computed by six CORDIC rotators and butterfly adders. By employing the distributed arithmetic in the CORDIC rotators, the requirement of a deep pipeline structure of the conventional CORDIC algorithm approach has been avoided. By using the CORDIC algorithm, the ROM size of the RAC has also been reduced, and it has been made independent of the image block size. A scheme of doubling the speed of the implementation of the DA-based CORDIC algorithm has also been presented. The details of the bit-level architecture of the proposed design has been discussed. In the proposed IDCT core, the redundant number system has been employed in order to

further optimize the design at the bit-level architecture. The on-line algorithm, which is a redundant number algorithm with the processing mode of most significant digit first, has been described. By taking advantage of the carry-free addition and most significant digit first property of the on-line algorithm, the new design achieves a faster processing speed with the requirement of a smaller silicon area.

In the next chapter, details on the hardware architecture of an entire 2-D IDCT processor are presented.

Chapter 4

Hardware Architecture

In the previous chapter, the design of an algorithm for the computation of the 1-D IDCT and the corresponding architecture employing the distributed arithmetic (DA) and the CORDIC rotators for its implementation were presented. By applying the distributed arithmetic in the implementation of the rotators using CORDIC algorithm [30], the proposed design provided the advantage of a small and regular structure and a reduced size of ROM required by the rotators to store the intermediate results of the IDCT computation. The redundant number system along with the on-line algorithm was adopted in the bit-level architecture of the proposed design.

In this chapter, the entire hardware architecture for the computation of a 2-D IDCT using the 1-D IDCT core described in previous chapter is given. Since the data flow in our design is in the digit-serial MSDF mode, some special arithmetic cells and control logic different from the design using bit-parallel structure are needed. The hardware architecture of the proposed design is described starting from simple modules such as 1-digit on-line adder, plus-plus-minus (PPM) and minus-minus-plus (MMP) adders to more complex modules such as DA-based rotator and redundant-to-nonredundant number convertor.

This chapter begins with an overall description in Section 4.1 of the hardware processor, that makes use of the 1-D IDCT core proposed in Chapter 3, for the computation of 2-D IDCT. In Section 4.2, the structure of the parallel-in-serial-out buffer, which stores and serializes the incoming pixels, is presented. The rotators, which are based on the distributed arithmetic and perform the vector rotation of two pixels, is described in Section 4.3. The one-digit on-line adder, which forms the butterfly adder array, is presented in Section 4.4. A redundant-to-nonredundant number convertor, which is needed to convert the data suitable either for column-wise IDCT processing or for final outputting, is described in Section 4.5. Finally, the transpose memory, which stores and transposes the intermediate results of the 1-D IDCT, is presented in Section 4.6.

4.1 Hardware Architecture of the 2-D IDCT Processor

The hardware architecture of the entire processor is based on the algorithm design and the conceptual architecture for the 1-D IDCT computation proposed in the previous chapter. However, some additional hardware, such as input buffer register, output buffer register, transpose memory and process control logic, is required for the actual implementation of the processor. By including these essential units along with the IDCT proposed in the Chapter 3, the top-level hardware architecture of the entire IDCT processor is shown in Figure 4.1. As discussed in Chapter 2, in order to achieve a regular structure for the proposed IDCT core, the row-column decomposition approach for the 2-D IDCT implementation has been adopted in our design. The processor is composed of two 1-D IDCT cores

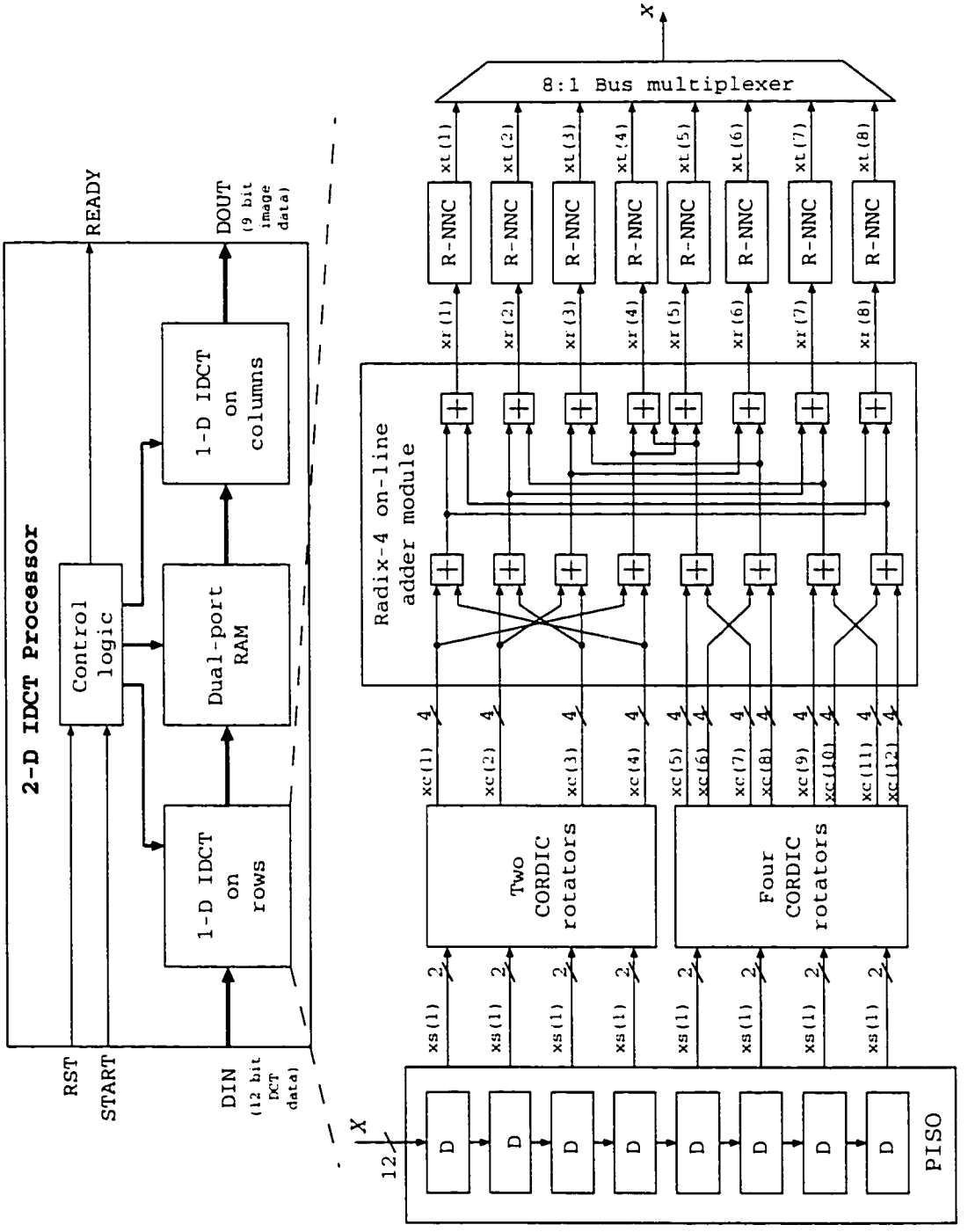

Figure 4.1 Top-level hardware architecture of the 2-D IDCT processor

(one 1-D IDCT (R) and one 1-D IDCT (C)) to carry out the IDCT operation for rows and columns for an image block, a transpose memory to transpose and store the intermediate results of IDCT operation, and a control logic block to control and supervise the overall operation of the processor.

In our design, the input pixel is represented by a signed 12-bit word with two's complement format. In each 1-D IDCT block, the input sample shifts into an 8-word parallel-in-serial-out (PISO) serialization register. In each clock cycle, the PISO register receives one 12-bit word and serially outputs eight pairs of 2-bit data ($xs(i)$) to the six downstream rotators. In each rotator, radix-2 signed-digit number representation is used to perform accumulation and shift operations. In each clock cycle, the two most significant digits of the accumulation result ($xc(i)$) in radix-2 signed-digit format are generated and shifted out from each rotator. These two radix-2 BSD numbers form one radix-4 signed-digit number and passed on to the subsequent 2-stage butterfly adder array, which is composed of a total of 16 radix-4 on-line adders. Meanwhile the accumulator continues to process. Each radix-4 on-line adder can process two radix-2 signed-digit number in an MSDF manner. Eight redundant number to nonredundant number convertors (R-NNC) following the butterfly adder array perform two tasks. First, they convert the incoming digit stream ($xr(i)$) in radix-4 signed-digit format into two's complement number serially in the MSDF mode. Second, each of the eight R-NNCs operate as a serial-in-parallel-out (SIPO) buffer. It receives the result of a butterfly adder serially and outputs them in parallel to the transpose memory. The last component in the 1-D IDCT core is an 8:1 bus multiplexer. It

selects one of the eight intermediate results ($xt(i)$) from the R-NNCs and passes it on to the transpose memory.

The transpose memory stores the 8×8 intermediate results of the 1-D IDCT (R), performs the transpose of the array, and generates the input for the 1-D IDCT (C). The control logic block of Figure 4.1 controls the computation process of the entire 2-D IDCT processor. It generates signals for the two PISO registers to control the serialization process and selection signals for the 8:1 bus multiplexers for the two 1-D IDCT cores. For the transpose memory, the control logic block controls the read and write operations and provides the access addresses.

The overall circuit starts with a '*START*' signal. A pulse at the '*START*' terminal indicates that the first sample is available at the input and the IDCT computation is started. The '*READY*' signal is asserted high when the computation of a block is completed and the data on the output terminal is valid. The '*RST*' is the active low reset signal, which clears all the registers in the IDCT core.

4.2 Parallel-in-Serial-out Register

In most applications, an image is divided into blocks of 8×8 pixels, and the 2-D IDCT is computed for each of the blocks. The pixels in a block are fed serially row by row starting from the top left pixel. This process is repeated for every block. The six DA-based rotators shown in Figure 4.1 process the data of eight pixels simultaneously, digit by digit (2 bits). Thus, a buffer is needed to store and transform the incoming pixel stream for the

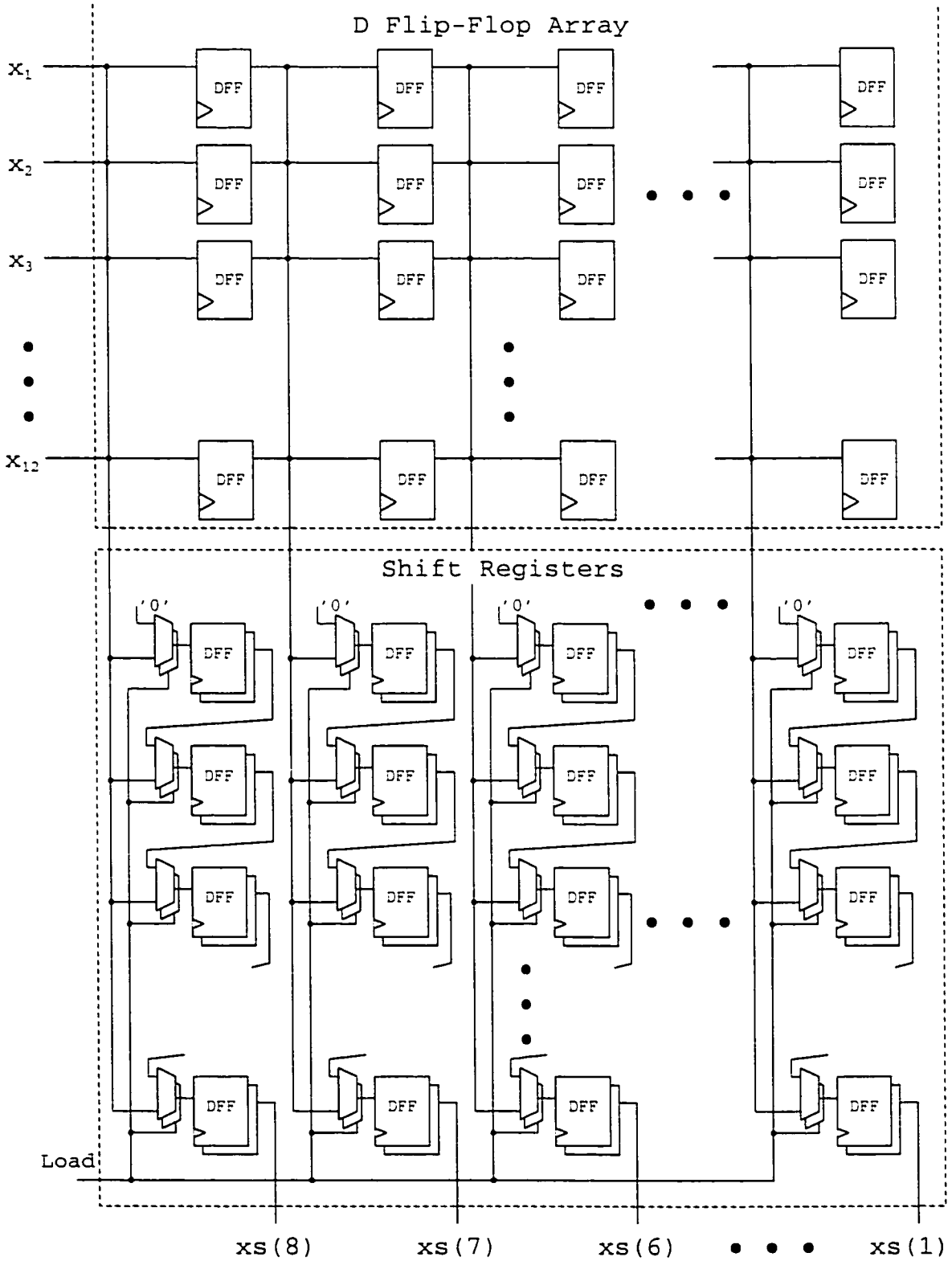

Figure 4.2 Parallel in serial out (PISO) register

downstream rotators. A parallel-in-serial-out (PISO) register, which can perform the operations of inputting and serialization of the data at the same time, is employed in each of the two 1-D IDCT cores. The PISO register, as shown in Figure 4.1, is composed of an 8×12 D flip-flop array to latch eight pixels, and eight 12-bit shift registers to serialize the data.

The operation of the PISO register can be described as follows. First, pixels are pipelined into the upper D flip-flop array word by word. After this array is loaded with eight words, a pulse on '*LOAD*' triggers the eight shift registers at the bottom to fetch all the eight words, so that the upper D flip-flop array is ready to accept the new pixels. Meanwhile the data in the shift registers are shifted out serially two bits a time to the rotators.

4.3 DA Based Rotator

The DA based rotator, as shown in Figure 4.3, is composed of five different submodules, the address decoder, the operation decoder, the 16-word ROM, radix-2 hybrid signed-digit parallel adder and shifter In each clock cycle, the rotator processes a two-bit digit from the PISO and shifts out two digits in radix-2 signed digit format from the shifter. An active high '*First_digit*' signal indicate the position of the most significant digit of the serial incoming data. An address decoder is employed to generate the addresses for the ROM accumulators (RAC) based on the combinations of x, y and '*First_digit*'. The operation decoder generates the signal 'AorS' which controls the adders in RAC to perform either addition or subtraction. Two RACs are used to compute the rotated coordinates

x' and y', respectively. As mentioned in Section 3.3.1, four kinds of rotators with rotation angles $\pi/4$, $\pi/16$, $\pi/8$ and $3\pi/16$ are deployed in this design. Each ROM contains 16 unsigned words of intermediate results computed using (3.15).

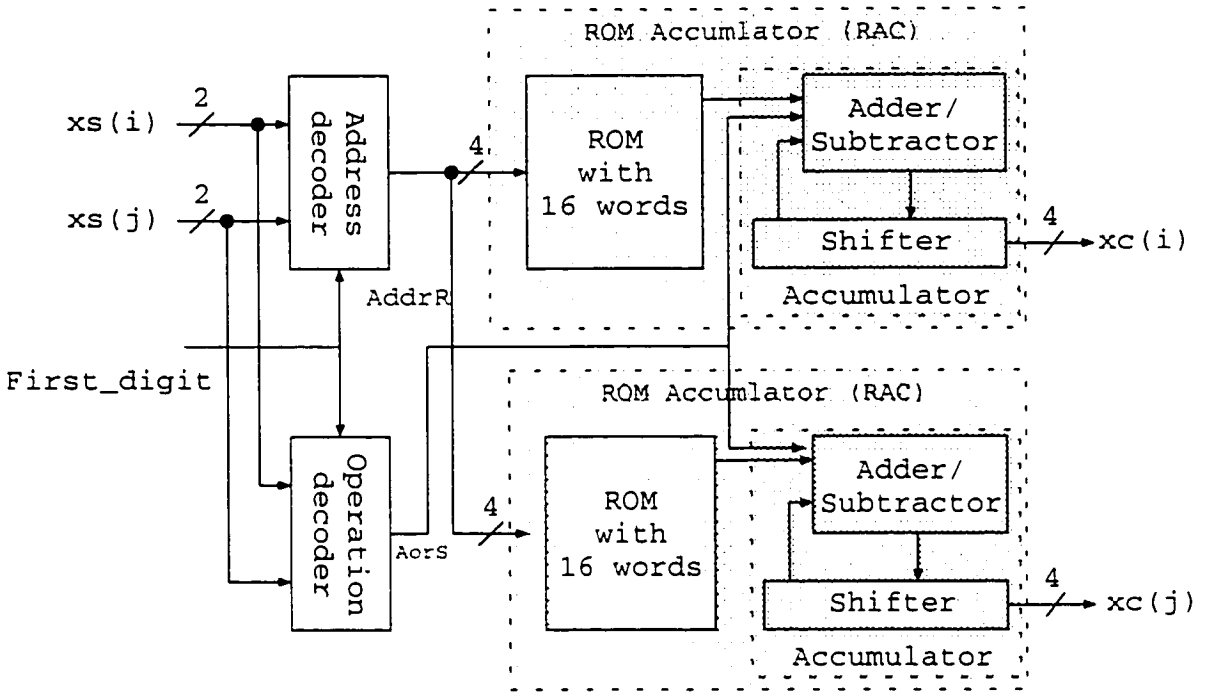

Figure 4.3 Hardware architecture of the DA-based rotator.

4.3.1 Address decoder

As mentioned in Section 3.2, the two 16-word ROMs store the results computed by (3.15). These contents of the ROMs are addressed by the combination of the incoming two bits of x and y from the PISO register. However, since the most significant bit is a sign bit in a two's complement number, the two most significant bits of the data being received from the PISO register need to be treated differently from the rest pairs of bits on each line of the register. Thus, an address decoder is employed to produce the address based on the

incoming bit pair and the signal 'First_digit' which indicates the arrival of the two most significant bits. The truth table of the address decoder is shown in Table 4.1.

Table 4.1 Truth table of the address decoder

$y(j)y(j-1)x(j)x(j-1)$	$ys(j)ys(j-1)xs(j)xs(j-1)$	
	First_digit = '0'	First_digit = '1'
0000	0000	0000
0001	0001	0001
0010	0010	0010
0011	0011	0001
0100	0100	0100
0101	0101	0101
0110	0110	1001
0111	0111	0101
1000	1000	1000
1001	1001	0110
1010	1010	1010
1011	1011	1001
1100	1100	0100
1101	1101	0101
1110	1110	0110
1111	1111	0101

4.3.2 Operation decoder

The results of (3.15), which should be stored in the ROMs, could be a either positive number or negative number. In order to reduce the size of the ROM, only the absolute value of the these numbers are stored in the ROMs in an unsigned binary format. Thus, an operation decoder is needed to control the downstream accumulators shown in Figure 4.3 to perform the operation of either addition or subtraction. In each clock cycle, the operation of the accumulator is controlled by an 'AorS' signal generated by the operation decoder. If a low-level signal is generated, the accumulator carries out an addition, otherwise, a subtraction is performed. The operation control signal '*AorS*' is decoded from the inputs of the rotator and '*First_digit*' provided by the control logic.

4.3.3 Hybrid radix-2 signed-digit accumulator

The addition/subtraction and shifting operations required by the accumulator shown in Figure 4.3 can be carried out by the hardware structure contained in the accumulator of Figure 4.4. The accumulators receive the data from the ROMs and perform the accumulation using radix-2 redundant signed-digit numbers. The hardware structure of the accumulator, as shown in Figure 4.4, is composed of a hybrid radix-2 redundant adder/subtractor which contains PPM adder, multiplexer and GUARD combinational logic, and a shifter which consists of 28 D flip-flops. In each clock cycle, the hybrid radix-2 signed-digit adder performs the parallel addition of the data from the ROM and the feedback data from the shifter and shifts out two digits (four bits) to the downstream module of the butterfly adder array as the final result.

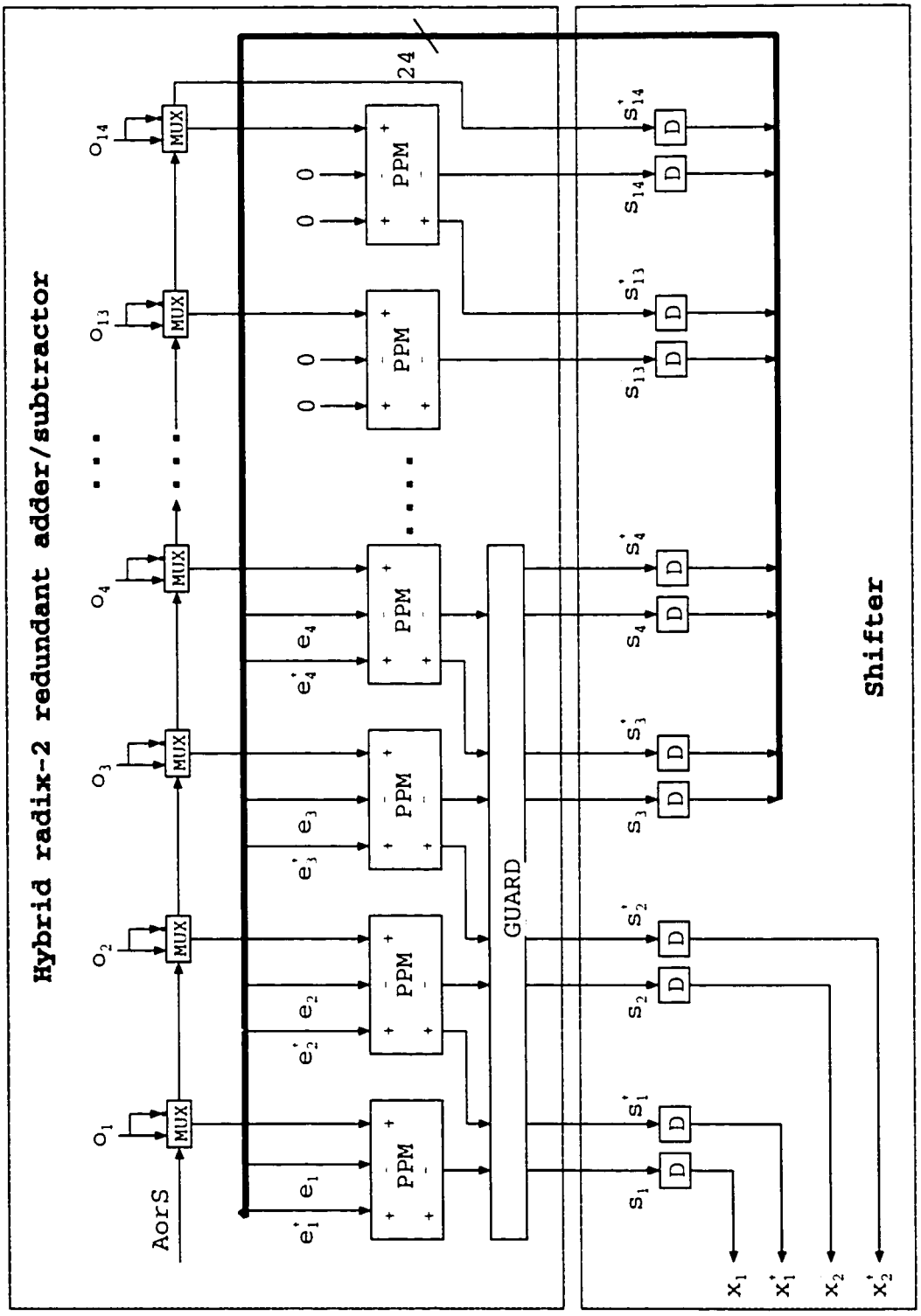

Figure 4.4 Hybrid radix-2 signed-digit accumulator.

The critical path of our design is located in the adders shown in Figure 4.4, since they consist of the largest combinational logic in the IDCT core. There are several possible designs for the adder/subtractor. Among them, the hybrid radix-2 redundant adder is the best choice, due to its propagation delay-free property and an area as efficient as that of a carry ripple adder. The hybrid radix-2 redundant adder/subtractor performs the addition of the data from the ROM in unsigned binary format and the intermediate results of the accumulation in signed-digit format.

The input $O = o_1 o_2 o_3 \ldots o_{14}$ from the ROM to the adder/subtractor is in binary unsigned format. The other input $E = e_1 e_2 e_3 \ldots e_{14}$ to the adder/subtractor is the feedback data from the output of the shift register in the binary signed digit format. In each clock cycle, the two most significant digits of S, namely s_i and s_{i+1}, in the BSD format is passed on to the subsequent butterfly adder. The rest of the digits of S are shifted two digits to the left and fed back as the input A of the adder/subtractor with two zero digits appended to the least significant digits. The purpose of the GUARD combinational logic block of the accumulator is to avoid an overflow in S by recoding the four most significant digits of S.

In the implementation of the redundant number system, the PPM adder and the MMP adder perform the same task as that of one-bit full adder in the nonredundant number system. Thus, the PPM adder shown in Figure 4.4 and the MMP adder which is used in the radix-4 on-line adder (Section 4.4) are presented in the following.

A. PPM and MMP

The plus-plus-minus (PPM) adder and the minus-minus-plus (MMP) adder are the most important basic building blocks in the implementation of the signed-digit redundant number adders. The PPM performs a one-digit addition $x_i^+ - x_i^- + y_i = 2t_i - u_i$ of a BSD digit x_i and an unsigned binary digit y_i. The MMP performs an one-digit subtraction $x_i^+ - x_i^- - y_i = -2t_i + u_i$. The area and complexity of these two units are similar to that of a 1-bit full adder. The truth tables of PPM and MMP are shown in Table 4.2.

Table 4.2 Truth table of PPM and MMP

Plus-Plus-Minus					*Minus-Minus-Plus*				
x_i^+	x_i^-	y_i	t_i	u_i	x_i^+	x_i^-	y_i	t_i	u_i
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	1	1
0	1	1	0	0	0	1	1	1	0
1	0	0	1	1	1	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	1	0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1

B. Overflow in the accumulator

As shown in Figure 4.4, in each cycle, the two most significant digits of the shifter are shifted out to be used by the butterfly adders. This means that each intermediate result

of the accumulation sum should not overflow in the shifter. This can be made possible using the redundancy property of the signed-digit number system. To guarantee this, two measures are taken to limit an accumulation result so that it always fits into the shifter.

- Two zeros are appended to the most significant positions of every word stored in the ROM. This means that $b_1 b_2$ are always equal to "00" in Figure 4.4.
- As shown in Figure 4.4, a combinational logic "$GUARD$" is designed to recode the four most significant digits to prevent the potential overflow.

4.4 Radix-4 On-line Adders

The radix-4 on-line adder module shown in Figure 4.1 consists of the two-stage butterfly adder arrays. The hardware structure of each of the radix-4 on-line adders in the module is shown in Figure 4.5. It is composed of two PPM adders, two MMP adders and eight D Flip Flops. This adder performs the addition of two radix-4 redundant numbers. The use of such an adder in our design is essential, since the DA-based rotator shifts out two digits in the radix-2 format in every clock cycle.

The radix-4 on-line addition can be carried out in a way similar to that of the on-line addition of two radix-2 redundant digits described in Section 3.3.3. As seen from the architecture in the Figure 4.5, the radix-4 on-line adder contains two stages of flip-flops, which introduce the latency of two clock cycles

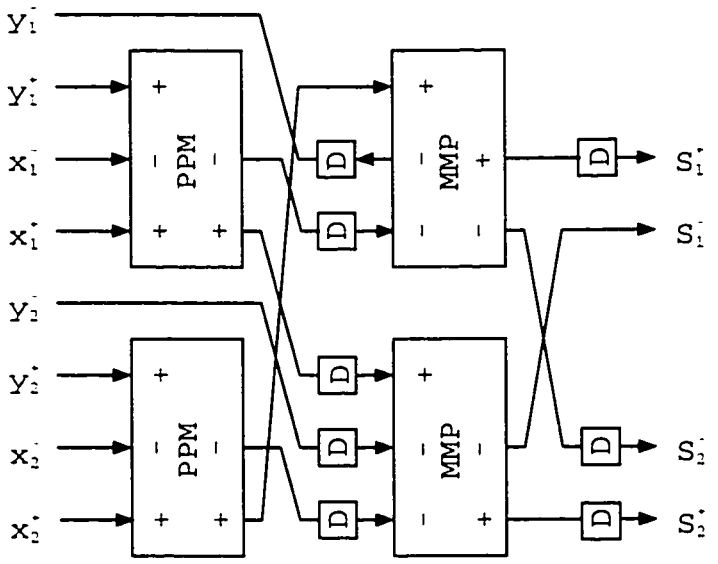

Figure 4.5 Digital-serial radix-4 signed digit redundant adders

4.5 Redundant-to-Nonredundant Number Convertor (R-NNC)

As shown in Figure 4.1, the results of the 1-D IDCT (R) in radix-4 signed-digit format need to be stored and transposed in the transpose memory. Directly memorizing the data in the redundant format requires the size of the transpose memory be twice the size when it is the two's complement format. Thus, a redundant to nonredundant number convertor (R-NNC) is employed to convert the results from the radix-4 redundant signed-digit format into the two's complement format. The conversion in MSDF mode is essential for the on-line algorithm. In [21], a scheme of redundant to nonredundant number conversion is given for the input digits in radix-2 redundant signed-digit format. Here we extend this technique to the input digits that are in radix-4 signed-digit format.

The most obvious scheme to realize such a convertor for a redundant number X would be to perform the subtraction $X^+ - X^-$. In this case, due to the carry propagation, it is not possible to know the value of any nonredundant digit of the conversion result, before the least significant digit of the redundant number becomes available. Thus, this scheme can not be used in our design

In order to perform the conversion in the MSDF mode, a priori knowledge of the carry-in signal at each digit is needed. Therefore, the two possible output digits are computed assuming carry inputs of -1 and 0, and the final output corresponding to the carry input 0 is selected as the output. The convertor takes two radix-2 redundant digits (x_i^{2+}, x_i^{2-}, x_i^{1+} and x_i^{1-}) and generates two-bit of a two's complement numbers in each clock cycle. It takes $B/2$ clock cycles to perform B-bit redundant number conversion. Table 4.3 shows how the output is obtained from the decoded values of the given input digits and two possible values of carry-in.

Table 4.3 Conversion mapping of a radix-4 redundant number to two's complement number

Carry-in \ Decoded value	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2	3
0	$\boxed{\bar{1}}01$	$\boxed{\bar{1}}10$	$\boxed{\bar{1}}11$	$\boxed{0}00$	$\boxed{0}01$	$\boxed{0}10$	$\boxed{0}11$
$\bar{1}$	$\boxed{\bar{1}}00$	$\boxed{\bar{1}}01$	$\boxed{\bar{1}}10$	$\boxed{\bar{1}}11$	$\boxed{0}00$	$\boxed{0}01$	$\boxed{0}10$

In this conversion, two partial results, X^0_{2i} and $X^{\bar{1}}_{2i}$ $(-1 \le i \le B/2)$, are obtained iteratively assuming the carry input to be 0 and -1, respectively, with the initial value $X^0_{-1} = 0$ and $X^{\bar{1}}_{-1} = 1$. Here, the superscript of the symbol of the partial result indicates the assumed value of the carry input and the subscript indicates the step of the conversion. The final result is given by X^0_B. During the conversion, in step i, X^0_{2i} and $X^{\bar{1}}_{2i}$ are updated by selecting $X^0_{2(i-1)}$ and $X^{\bar{1}}_{2(i-1)}$, and by appending two bits according to the rule given in Table 4.3. This procedure of updating is summarized in Table 4.4, where $X^0_{2(i-1)}$ is selected if $plus = 1$ and $X^{\bar{1}}_{2(i-1)}$ is selected if $minus = 1$. The values of $plus$ and $minus$ are determined based on the value of the input digit as shown in Table 4.4. If the input digit is $\bar{1}$, $\bar{2}$, or $\bar{3}$, it implies (Table 4.3) that the carry output is always equal to $\bar{1}$, no matter what the carry input is. Hence, $minus$ is set to 1 and $X^{\bar{1}}_{2(i-1)}$ is used for updating. Two bits are appended to $X^{\bar{1}}_{2(i-1)}$ to form X^0_{2i} and $X^{\bar{1}}_{2i}$, assuming the possible carry input of 0 or -1. If the input digit is 1, 2, or 3, it implies (Table 4.3) that the carry output is always equal to 0. Hence, $plus$ is set to 1 and $X^0_{2(i-1)}$ is used for updating. Two bits are appended to $X^0_{2(i-1)}$ to form X^0_{2i} and $X^{\bar{1}}_{2i}$, for possible carry input of 0 or -1. As shown in the fifth column in Table 4.3, when the input digit is equal to 0, it implies that both cases of carry output, -1 and 0, are possible. Hence, $minus$ and $plus$ are both set to 0 and $X^0_{2(i-1)}$

and $X^{\bar{1}}_{2(i-1)}$ are kept and updated as $X^{\bar{1}}_{2i}$ and X^{0}_{2i} by appending the two corresponding bits to $X^{0}_{2(i-1)}$ and $X^{\bar{1}}_{2(i-1)}$, respectively.

Table 4.4 Computation of the output digit during the conversion of a radix-4 redundant to a nonredundant number operating in the MSDF mode

Decoded value	X^{0}_{2i}	$X^{\bar{1}}_{2i}$	*Minus*	*Plus*
$\bar{3}$	$X^{\bar{1}}_{2(i-1)} + 0 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	$X^{\bar{1}}_{2(i-1)} + 0 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	1	0
$\bar{2}$	$X^{\bar{1}}_{2(i-1)} + 1 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	$X^{\bar{1}}_{2(i-1)} + 0 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	1	0
$\bar{1}$	$X^{\bar{1}}_{2(i-1)} + 1 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	$X^{\bar{1}}_{2(i-1)} + 1 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	1	0
0	$X^{0}_{2(i-1)} + 0 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	$X^{\bar{1}}_{2(i-1)} + 1 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	0	0
1	$X^{0}_{2(i-1)} + 0 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	$X^{0}_{2(i-1)} + 0 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	0	1
2	$X^{0}_{2(i-1)} + 1 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	$X^{0}_{2(i-1)} + 0 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	0	1
3	$X^{0}_{2(i-1)} + 1 \cdot 2^{2i-1} + 1 \cdot 2^{2i}$	$X^{0}_{2(i-1)} + 1 \cdot 2^{2i-1} + 0 \cdot 2^{2i}$	0	1

In order to design the converter, 2 basic cells, the *COPY* cell and the *APPEND* cell [19] are required. Depending on the values of the input digits, the *APPEND* cell generates the select signals *plus* and *minus*, and two pairs of output bits based on the input digits and the carry-in signal, as shown in Table 4.3. Depending on the values of the *plus* and *minus* signals, the *COPY* cell select either $X^{0}_{2(i-1)}$ or $X^{\bar{1}}_{2(i-1)}$. The logic equation

of the *COPY* and the *APPEND* cells can be derived from Table 4.3 and Table 4.4 as follows

$$plus = \overline{x_i^{1-}}\,\overline{x_i^{2-}} + x_i^{2+}\overline{x_i^{2-}} + x_i^{2+}x_i^{1+} + \overline{x^{2-}}\,x_i^{1+} + \overline{x_i^{1+}}\,\overline{x_i^{1-}}\,x_i^{2+} \tag{4.1a}$$

$$minus = \overline{x_i^{2+}}\,x^{2-} + \overline{x_i^{2+}}\,\overline{x_i^{1+}}\,x_i^{1-} + x^{2-}\overline{x_i^{1+}}\,x_i^{1-} \tag{4.1b}$$

$$x_{2i}^{0} = \overline{x_i^{1+}}\,x_i^{1-} + \overline{x_i^{1+}}\,x_i^{2+}\overline{x_i^{2-}} + x_i^{2-}x_i^{1+}\overline{x_i^{1-}} + \overline{x_i^{1-}}\,\overline{x_i^{2+}}\,\overline{x_i^{2-}} \tag{4.2a}$$

$$x_{2i-1}^{0} = \overline{x_i^{2+}}\,x_i^{2-} + x_i^{2+}x_i^{2-} \tag{4.2b}$$

$$x_{2i}^{-1} = \overline{x_i^{1+}}\,\overline{x_i^{1-}} + x_i^{1+}x_i^{1-} \tag{4.3a}$$

$$x_{2i-1}^{-1} = x_i^{2+}x_i^{2-}\overline{x_i^{1+}} + \overline{x_i^{2+}}\,\overline{x_i^{2-}}\,\overline{x_i^{1+}} + \overline{x_i^{2+}}\,\overline{x_i^{2-}}\,x_i^{1-} \tag{4.3b}$$

Based on the above equations, the *COPY* and the *APPEND* cells are designed as shown in Figure 4.6.

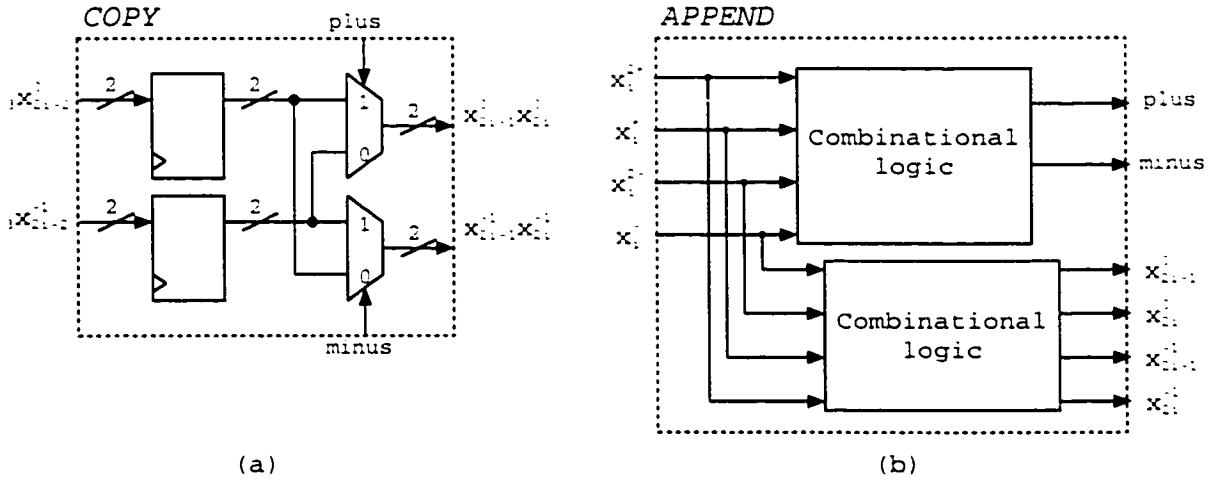

Figure 4.6 (a) COPY cell (b) APPEND cell used in the conversion of a radix-4 redundant number to a nonredundant number.

An MSDF converter to convert a radix-4 signed-digit number of wordlength B can be constructed by serially connecting an $APPEND$ cell and B $COPY$ cells as shown in Figure 4.7.

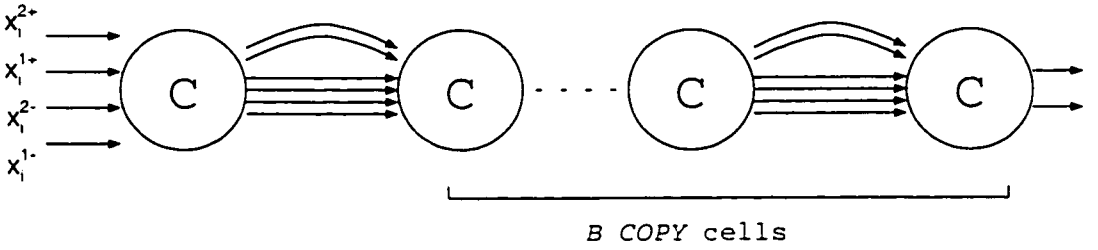

Figure 4.7 Block diagram of B-bit MSDF redundant to nonredundant converter.

As shown in Figure 4.1, a total of eight R-NNCs are needed to perform eight conversions in parallel in the proposed design. Note that since the transpose memory employed in our 2-D IDCT processor is a dual-port RAM (the details of transpose memory design is give in Section 4.6), only one word can be written in or read out in every clock cycle. It means that we need additional registers to buffer the conversion results. In order to avoid the additional delay with minimum extra hardware, eight unique converters and an 8:1 bus multiplexer is employed to perform the eight conversions and the multiplexing of the eight data, respectively. As shown in Figure 4.8, no buffer registers are needed for the first converter on the top. From the top to bottom, each converter employs one additional stage of buffer registers than the one just above it. Thus, the conversion result $xt(i-1)$ will be ready one clock cycle earlier than that of $xt(i)$, $2 \le i \le 8$.

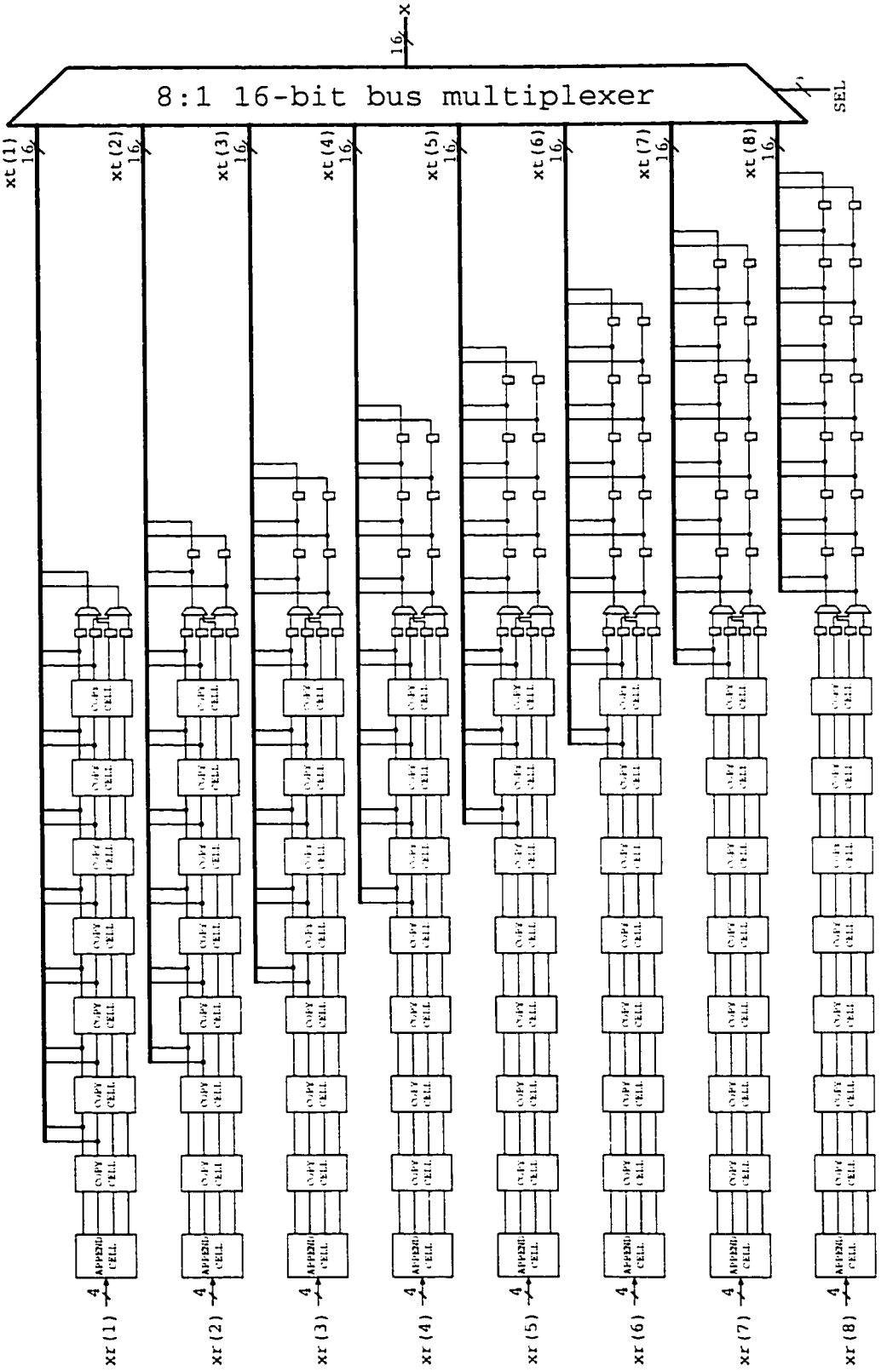

Figure 4.8 Eight redundant-to-nonredundant converters and the RAM write-in multiplexer.

4.6 Transpose Memory

The transpose memory module shown in Figure 4.1 stores and transposes the results of the IDCT (R) and provides the inputs for the IDCT (C). There are several ways to implement the transpose memory. The trivial way is to use two matrices for read and write alternatively. Another way [29], is to use a single matrix composed of $8 \times 8 \times W$ D flip-flops to transpose the data on-the-fly. However, since the proposed design is FPGA based, the transpose memory can be implemented with a 64-word dual-port block RAM taking advantage of the abundance of RAM in the FPGA and still using only one-half of the RAM required by the trivial method. A dual port block memory has two independent ports that enable shared access to a single memory space. Both ports are functionally identical, with each port providing read and write access to the memory [28]. The symbol of such a dual-port RAM is depicted in Figure 4.9.

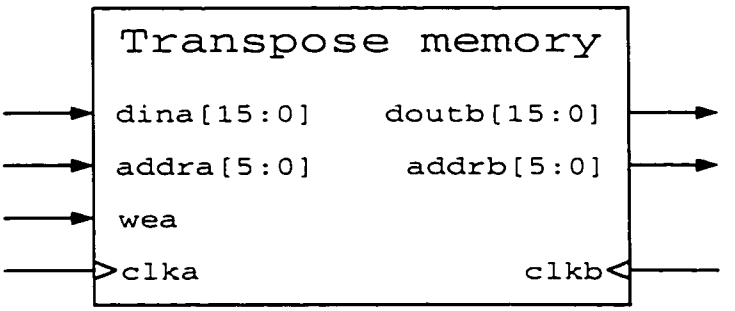

Figure 4.9 Schematic symbol of a dual-port memory.

The ports on the left side are in write-only mode. "*wea*" is the write enable signal. The ports on the right side are in read-only mode. Simultaneous reading-from and writing-to the same memory location will result in a correct data being written into the memory.

but invalid data being read out of it. Therefore, in our design, "$clka$" and "$clkb$" are driven by two clocks which are complementary so that the operations are non-overlapping.

4.7 Summary

In this chapter, a detailed design of the hardware architecture of the proposed 2-D IDCT processor using the DA-based CORDIC algorithm introduced in Chapter 3 has been given. The 2-D IDCT processor is composed of two 1-D IDCT cores, a transpose memory and a process control logic block. Since the most significant digit first (MSDF) operation mode is adopted in the proposed design, some special arithmetic and control logic units, particularly for the implementation of the redundant number system and on-line algorithm, have been employed.

In each 1-D IDCT core, the incoming pixels are buffered and serialized into two-bit digit streams by a parallel-in-serial-out (PISO) register in the MSDF mode for the six downstream DA-based rotators in order to perform the operations of loading and serialization simultaneously. In the rotators, the ROMs take the two-bit digits from the PISO register as the address and provide an unsigned binary number for the accumulators which are composed of an adder/subtractor and a shifter. By employing the hybrid radix-2 signed-digit adder, the accumulator can operate with a high-speed and requires a small area. Furthermore, the accumulators can also shift out to the butterfly adder array the two most significant digits of the intermediate results of accumulation in radix-2 signed-digit format in each clock cycle. This way, the butterfly adder array which employs sixteen on-line adders

is able to perform the operation of addition in radix-4. Eight redundant-to-nonredundant number convertors (R-NNC), which can convert a redundant number into the two's complement format in the MSDF mode on-the-fly, are adopted to convert the final results of 1-D IDCT computation without the use of an otherwise double size transpose memory. The 8:1 bus multiplexer at the last stage has been used to provide a stream of output pixels serially for each image block.

In the next chapter, the problem of implementation of the hardware architecture for the proposed 2-D IDCT processor will be undertaken.

Chapter 5

Implementation

The overall hardware architecture of the 2-D IDCT processor based on CORDIC algorithm and the distributed arithmetic was presented in the previous chapter. The hardware designs of the various building blocks employed in the processor were also presented. An implementation of the proposed scheme is essential in order to evaluate the performance and the hardware cost of the proposed 2-D IDCT processor. There are two approaches of the hardware implementation, the application specific integrated circuit (ASIC) and the field programmable gate array (FPGA). Among the two approaches, the FPGA is widely used for the evaluation of a prototype design because of its low cost and short development cycle. Thus, in this chapter, this approach is used for the implementation of the proposed design.

In this chapter, an exact-bit simulation scheme recommended by the IEEE standard is chosen to test the specification of the accuracy of the 2-D IDCT processor. The proposed design is also implemented and tested on a Xilinx Virtex XC2V1000 FG256-4 FPGA board [32]. A test is carried out for the validation of the FPGA implementation of the proposed IDCT processor.

In Section 5.1, a brief introduction of the IEEE standard 1180-1990, which has been specifically proposed for the accuracy compliance of the IDCT implementation, is explained. The setup of the exact-bit simulation and the simulation results of the proposed design are also given. The procedure of the FPGA implementation and the scheme for the validation of the FPGA implementation of the proposed IDCT processor are presented in Section 5.2. Finally, the results of the FPGA implementation of the proposed IDCT processor and a comparison with the existing dedicated designs are presented in Section 5.3.

5.1 IEEE 1180-1990 Compliance Standard for the 8×8 IDCT

Because of the continuous growth of audiovisual services since the late 1970s, customers have expected and demanded standards with which the video terminal equipments can be compatible. The IEEE standard 1180-1990 is one such standard developed by Commité Consultatif International de Telecommunications et Telegraphy (CCITT). This standard specifies the numerical characteristics of the 8×8 inverse discrete cosine transform for use in visual telephony and similar applications where the IDCT results are used in a reconstruction loop [13].

5.1.1 IDCT accuracy test procedure [13]

The setup for measuring the accuracy of an IDCT processor is shown in Figure 5.1. The procedure to test the accuracy of the IDCT consists of the following steps: (1) generate random data in a specified range, and form 8×8 data blocks, (2) perform a DCT on

each blocks, (3) perform an IDCT on the result of the DCT using both the proposed IDCT and a mathematical IDCT model with a 64-bit floating point accuracy, and (4) measure the peak, mean, and mean square errors between the output of the IDCT processor under consideration and the mathematical IDCT model.

The errors as mentioned in item (4) above can be defined as follows. Let $e_k(i, j)$ be the error between the output of the mathematical IDCT model and that of the IDCT processor under consideration, where $i, j = 0, \ldots, 7$ and $k = 1, \ldots, 10000$. The peak error $ppe(i, j)$ at pixel location (i, j) is defined as the peak value of $e_k(i, j)$. The mean square error $pmse(i, j)$ is defined as

$$pmse(i, j) = \frac{1}{10000} \sum_{k=1}^{10000} e_k^2(i, j), \quad i, j = 0, \ldots, 7. \tag{5.1}$$

The overall mean square error, $omse$, is defined as

$$omse = \frac{1}{64 \times 10000} \sum_{i=0}^{7} \sum_{j=0}^{7} \sum_{k=1}^{10000} e_k^2(i, j). \tag{5.2}$$

The mean error, $pme(i, j)$, is defined as

$$pme(i, j) = \frac{1}{10000} \sum_{k=1}^{10000} e_k(i, j), \quad i, j = 0, \ldots, 7. \tag{5.3}$$

The overall mean error, ome, is defined as

$$ome = \frac{1}{64 \times 10000} \sum_{i=0}^{7} \sum_{j=0}^{7} \sum_{k=1}^{10000} e_k(i, j). \tag{5.4}$$

To be the standard compliant, the following conditions must be met:

- For any (i, j) $(i, j = 0, \ldots, 7)$, the peak error ($ppe(i, j)$) should be less than or equal to 1.0, the average error ($pme(i, j)$) (over 10,000 blocks) should be less than or equal to 0.015, and the mean square error ($pmse(i, j)$) should be less than or equal to 0.06.
- Overall, *ome* should be less or equal than 0.0015, and the *omse* should be less equal than 0.02.
- All-zero input must produce all-zero output.

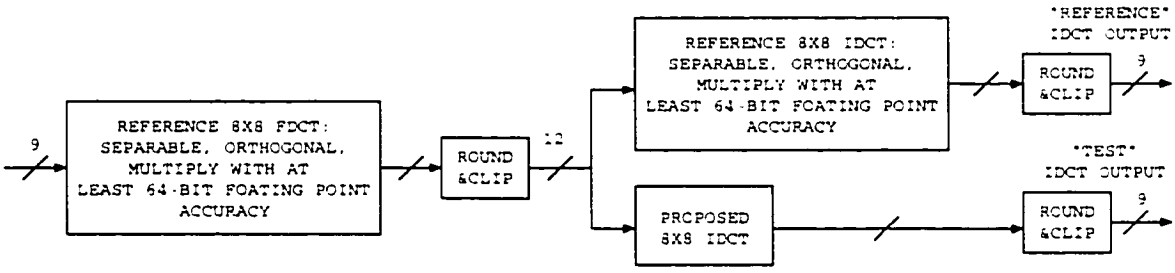

Figure 5.1 Setup for measuring the accuracy of a proposed 8x8 IDCT [13].

5.1.2 Exact-bit simulation results

According to the IEEE 1180-1990, the test should be carried out by an exact-bit simulation. A C program is developed for modeling the proposed IDCT processor with exactly the same data path bandwidth, truncation, and operation precision as those of the design. As described in Section 5.1.1, several error figures defined in the standards must be computed from the simulation result and compared with the given specification. For the sake of the completeness of this thesis, the definition of them error figures are given below.

Based on the simulation scheme described in Section 5.1.1, the error figures of the proposed design are calculated and compared with the specifications of the standard. As seen from Table 5.1, all the error figures given in the third column which are computed from the simulation results meet the specifications of IEEE 1180-1990 standard given in the second column.

Table 5.1 Compliance of IEEE 1180-1990 Standard

Item	Specification	Simulation results
$ppe(i, j)$ *(max)*	1	1
$pmse(i, j)$ *(max)*	0.06	0.013
$pme(i, j)$ *(max)*	0.015	0.008
omse	0.02	0.0084
ome	0.0015	0.0008
All-zero input	All-zero output	All-zero output

5.2 FPGA Implementation of the Proposed Design

As mentioned earlier, there are two available choices for the hardware implementation of the proposed design, the ASIC and the FPGA. The FPGA is particularly attractive because of its capability of providing rapid prototyping of systems and low-cost development. The implementation procedure of the IDCT processor follows the standard FPGA design flow with the HDL (hardware design language) entry [32]. The first step of the implementation is developing an RTL (register transfer level) circuit description of the

IDCT processor using VHDL language. Then, the VHDL description is used as the input to the Synopsys tools which synthesize the design into a netlist of logical gates based on the library of the target technology. The last step is loading the netlist to the target FPGA device.

The development and test of the proposed design was carried out on an INSIGHT VIRTEX II reference board, which contains a Virtex XC2V1000 FG256-4 FPGA chip, user switches, on-board 100 MHz clock generator, LED displays and other peripheral components [32]. Based on the static timing analysis of the design, the FPGA implementation of the 2-D IDCT processor can be assumed to run with an operation speed of 100 MHz. For such a high speed clock, the setup of the testbench becomes a very challenging task, due to the strict speed requirement for the interface between the device under test (DUT) and the testbench. In order to avoid the interference of the interconnection of the testbench, and the fact that in most applications the IDCT processor is encapsulated inside a video processing IC, a built-in testbench, which is implemented inside the same FPGA of the IDCT processor, is developed for the testing of the proposed design. The block diagram of the testbench along with the DUT is shown in Figure 5.2.

The test control logic block controls the procedure of the test. The RAM-T stores the testvectors randomly generated by the C-program provided in the IEEE standard. The RAM-E stores the corresponding results generated by the exact-bit model discussed in Section 5.1.2. The comparator performs the comparison of the actual results generated by the proposed 2-D IDCT processor, and the expected ones stored in the RAM-E. The user switch triggers the commencing of the test. The LED displays are used to show the results

of the test. The clock generator provides two complementary clocks for the testbench. In order to avoid potential timing violation, the DUT is driven by CLOCKB, which is a complementary version of CLOCKA driving the testbench.

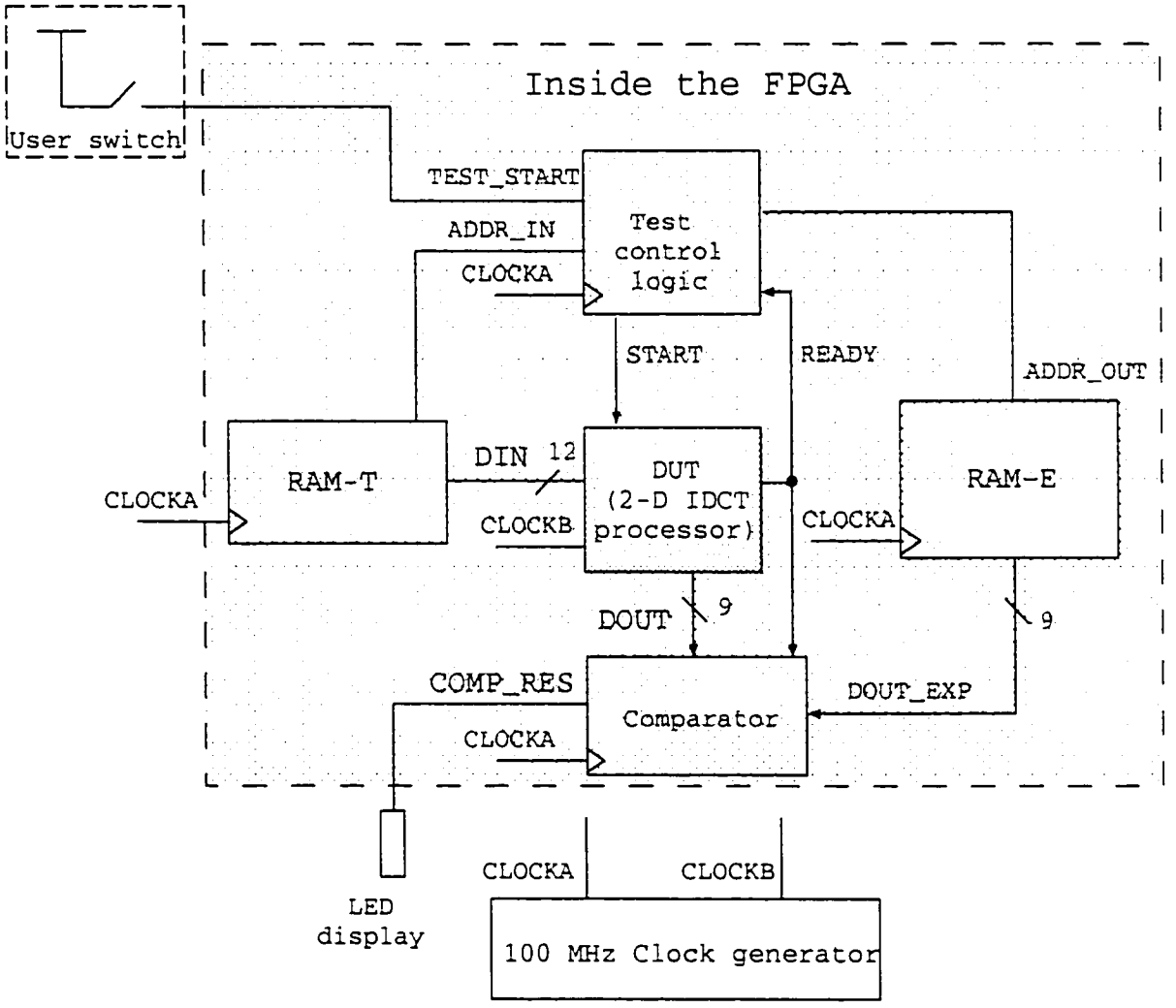

Figure 5.2 Blockdiagrm of the testbench for the 2-D IDCT processor.

The test procedure, with the timing diagram shown in Figure 5.3, can be described as follows. After a positive pulse is given from the user switch, 'TEST_START', the test

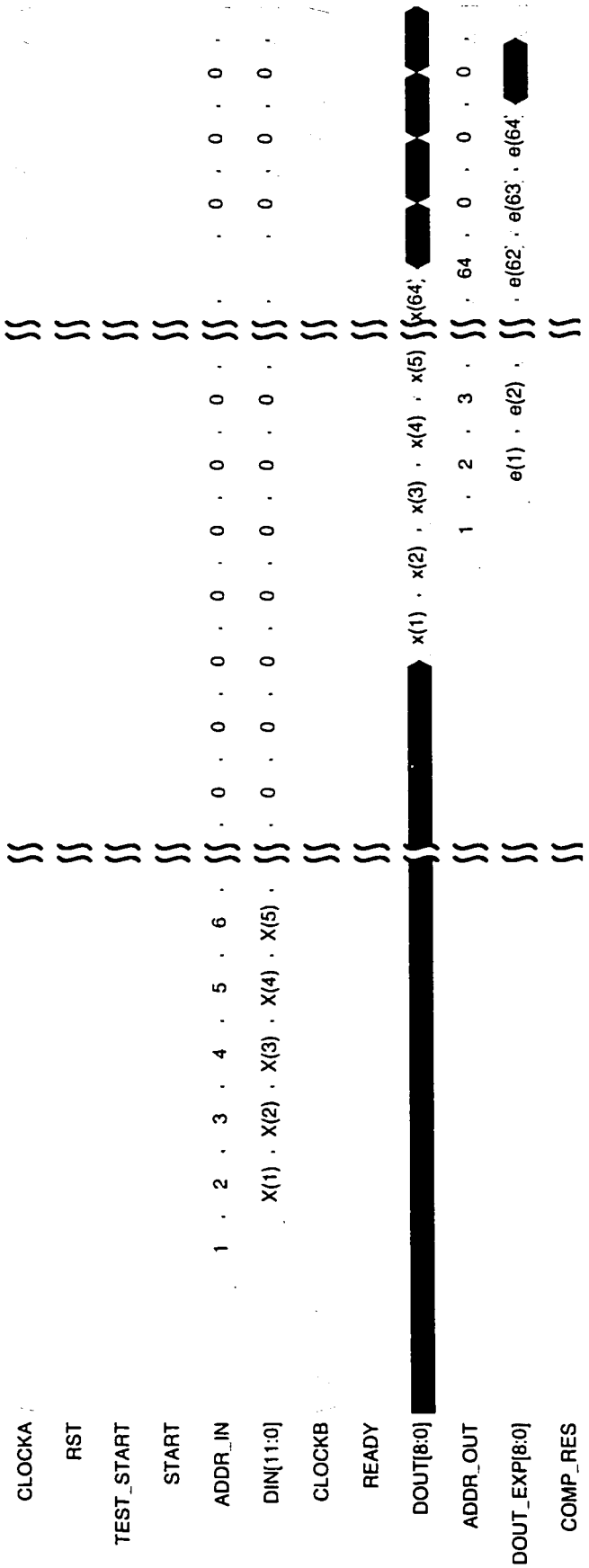

Figure 5.3 Timing diagram of the test procedure.

control block generates the read enable signal along with the 'ADDR_IN' to the RAM-T. It also generates a positive pulse 'START' which is aligned with the first pixel value on the DIN port provided by the RAM-T. The 'ADDR_IN' continues to increment until all the pixels of an 8×8 block are loaded into the DUT. A high-level signal on the 'READY' port of DUT is asserted when the first data of the 2-D IDCT results is available on the DOUT port. This signal also triggers both the test control block to generate the 'ADDRB' for the RAM-E as well as the comparator to start the comparison operation. The comparator compares each pair of the actual and expected results of the IDCT computation of the image block. If no error were found, the external LED display is turned on by the comparator.

5.3 Implementation Results and Comparison with Dedicated Designs

The hardware cost and the test results of the FPGA implementation of the proposed 2-D IDCT processor using Xilinx XC2V1000 FP256-4 are shown in Table 5.2. The processor operates on image blocks of 8×8 pixels, with 12-bit and 9-bit precision for inputs and outputs, respectively. The test results [32] show that the proposed 2-D IDCT processor provides the functional behavior as expected at a frequency of 100 MHz, with a throughput of 80 Mpixel/s. The total FPGA resource usage of the implementation of the proposed processor is 1580 slices. A block RAM, which is embedded inside the FPGA, is used for the implementation of the transpose memory.

Table 5.2 Characteristic of the 2-D IDCT processor

Inputs	*12 bits*
Outputs	*9 bits*
Internal wordlength	*16 bits*
Device	*Xilinx XC2V1000 FG256-4*
No. of slice	*1580*
Special features	*1 Block RAM*
Clock rates	*100 MHz*
Throughput	*80 Mpixels/sec*
Block size	8×8

There have been many FPGA DCT/IDCT implementations reported in the literature. The comparison of different implementations may not be as realistic as one may desire due to the different technology adopted in different designs. Also, since in some cases, not all specifications of a specific implementation are available from the reported literature, the task of comparison becomes even more difficult. Thus, only the designs with the similar implementation approach as ours are chosen for comparison. The specifications of several recent high-performance DCT/IDCT FPGA implementations, which all employ the Xilinx FPGA, are summarized in Table 5.2. All the designs use row-column decomposition approach. Chaudhary's IDCT processor [6] is based on the distributed arithmetic and it is a highly parallel structure. Data corresponding to 64 pixels are processed at a time. It has the highest throughput but requires the largest area. The parallel vector multiplier is adopted in the DCT/IDCT processor of CSELT [7]. This design pro-

cesses 8 pixels at a time and is implemented on the fastest family of Xilinx FPGA. The bit-serial multiplier based DCT/IDCT core of XENTEC [27] uses the smallest resource of the FPGA. However, its operation speed is relatively low. Comparing our design with these other designs, it can be seen from Table 5.3 that by taking into account both the area and the speed, the proposed design provides a better performance.

Table 5.3 Comparison with dedicated designs

Designs	Functions	FPGA type	Number of slices	Operation speed (MHz)	Throughput (Mpixels/sec)
Chaudhary [6]	IDCT	Xilinx XCV600 BG560-5	6140	55.6	271.36
CSELT [7]	DCT/ IDCT	Xilinx Virtex V200-6	1802	78	78
XENTEC [27]	DCT/ IDCT	Xilinx Virtex V100-6	1140	32	28
Proposed design	IDCT	Xilinx Virtex2 XC2V1000 FP256-4	1580	100	80

5.4 Summary

This chapter has started with a discussion on the issues concerning the implementation of the proposed 2-D IDCT processor, including a simulation on the compliance of the IEEE 1180-1990 standard and an FPGA implementation. An exact-bit model of the processor required by the IEEE standard which is especially defined for the IDCT accuracy by the CCITT has been developed in C language. The simulation results have shown

that the accuracy of the proposed 2-D IDCT processor meets the requirement of the standard. The proposed design has also been implemented and tested on a Xilinx XC2V1000 FP256-4. In order to validate the FPGA implementation of the processor, a built-in test-bench, which is downloaded to the same FPGA chip as the designed IDCT processor, has been developed and implemented. The implementation and test results have shown that considering the area and speed together, the proposed design provides a better performance than other comparable designs and FPGA implementations found in the literature.

Chapter 6

Conclusion

6.1 Concluding Remarks

In this thesis, a novel algorithm for the computation of the 1-D inverse discrete cosine transform (IDCT) using the distributed arithmetic (DA) and the CORDIC algorithm, and an FPGA implementation of a 2-D IDCT processor that employs the proposed algorithm have been presented.

Two of the existing schemes, the DA based DCT/IDCT algorithm and the CORDIC based DCT/IDCT algorithm, have constituted the basis of the study carried out in this thesis. The former is the DA-based method in which the IDCT computation has been implemented using ROM accumulators (RAC) by pre-computing the intermediate results of IDCT and storing them in lookup tables. However, the size of the ROMs used in each RAC is an exponential function of the size of the image block. Thus, the overall speed of the DA-based IDCT implementation could be limited by the access time of the ROMs. In the latter scheme, a high speed IDCT implementation has been proposed using CORDIC rotators, each realized by a structure comprising a number of stages. In this approach, the use of a deep pipeline structure needed for the implementation of the

CORDIC rotators makes it require a large silicon area. The proposed method of developing and implementing a scheme for the IDCT computation has been an attempt to reduce the large access time of the ROMs as well to decrease the area requirement by bringing together the distributed arithmetic and the CORDIC approach within the same scheme.

With a view of achieving the above objective, a 1-D IDCT core comprsing DA-based CORDIC rotators and butterfly adders are designed. In the proposed design, the complexity of each CORDIC rotator has been made independent of the size of the image block. The rotators have been implemented by ROM accumulators based on the distributed arithmetic. The intermediate results of the rotation operation are stored in lookup tables and accumulated in each clock cycle. Thus, no deep pipeline structure is needed. In contrast to the conventional DA-based approach, in which the data from each pixel of a row of the image block are needed to access the ROM, in the proposed design, each rotator needs data only from two pixels to access the ROM. Therefore, ROMs with a size of only four words are needed in each rotator. As a consequence, the ROM access time is no longer a dominating factor for the speed of the proposed scheme. Furthermore, the proposed design has been modified to process two bits of a pixel at a time by taking advantage of the requirement of only small-sized ROMs. Thus, an architecture for the IDCT computation to double the processing speed and yet to require an area which is smaller than that in the conventional DA-based approach has been achieved.

Based on the 1-D IDCT core, a hardware architecture for a 2-D IDCT processor has been designed. The processor consisting of two 1-D IDCT cores, a transpose memory and a control logic block computes the 2-D IDCT by using a row-column decomposition

approach. In order to achieve a good compromise between the performance and the hardware complexity of the processor, an implementation scheme using a digit-serial bit-level architecture employing the redundant number system and an on-line structure has been developed.

In terms of the area, two advantages have been achieved by such an architecture. First, no additional hardware for parallel to serial conversion, as needed in the bit-parallel architecture, is required here. Second, since each adder in the butterfly adder array inside the IDCT core has been implemented by a one-digit on-line adder, unlike the bit-parallel adder, the size of the butterfly adder array is independent of the internal wordlength.

In terms of the speed, since the digit-serial architecture processes multiple bits each clock cycle, a higher throughput has been achieved compared with the design of the bit-serial architecture. By using the redundant number system, the carry propagation delay of the adders used in the IDCT core has been limited to a few bit positions with almost no hardware increment. Thus, a high-speed operation has been achieved in the proposed design . Furthermore, the on-line algorithm makes it possible to pass on part of the results from the accumulator of a rotator to the butterfly adder array before completing the accumulation operation. Thus, the accumulators can be directly connected to the subsequent adder array without requiring an in-between buffer that carries out the operation of re-serialization.

In order to verify the compliance of the proposed design of the 2-D 8×8 IDCT processor with the IEEE 1180-1990 standard, which is a dedicated standard for the accu-

racy requirement of the implementation of an 8×8 IDCT, an exact-bit model of the IDCT processor has been developed in C language and simulated based on the scheme provided by the standard. The simulation results have shown that the accuracy specifications of the proposed design meets the requirement of the standard.

The 2-D 8×8 IDCT processor has been implemented and tested on a Xilinx Virtex XC2V1000 FG256-4 FPGA board. A testbench, which is implemented in the same FPGA as the designed IDCT processor, was developed and used to validate the proposed design. By comparing the results of the proposed design with those of the recent ones using similar technology, it has been shown that the design and the implementation undertaken in this thesis provides a better performance in terms of area and speed considered together.

6.2 Suggestions for Future Investigation

It has been shown that the implementation of the IDCT computation using the DA-based CORDIC algorithm provides a better performance than the designs using the conventional approaches employing the distributed arithmetic or the CORDIC algorithm individually. A study on the implementation of a processor combining the functions of both DCT and IDCT using the proposed scheme could be undertaken as a future study.

It is known that the trigonometric functions of some orthogonal transforms, such as discrete sine transform (DST), have also the symmetry property. Consequently, the pro-

posed algorithm can be modified for the implementation of such other orthogonal transforms.

Finally, it may be worthwhile to undertake a study for the ASIC design and implementation of the proposed 2-D IDCT processor, in view of the significance of DCT and IDCT functions in audio, video and communication applications.

References

[1] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Trans. on Electronic Computers*, vol. EC-10, pp.389-400, Sept. 1961

[2] N. Ahmed, et al., "Discrete Cosine Transform", *IEEE Trans. on Computer*, Vol, C-23, pp.90-93, 1974.

[3] V. Bhaskaran, K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Pub., 1995.

[4] W. H. Chen, C. H. Smith, and S.C. Fralick., "A fast computational algorithm for the discrete cosine transform", *IEEE Trans. on Communications*, September 1977.

[5] N. I. Cho, S. U. Lee,J. Duprat, "Fast algorithm and implementation of 2-D disrete cosine transform", *IEEE Trans. Circuits Syst.*, Vol. CAS-38, pp.297-305, Mar. 1991.

[6] K. Chaudhary, H. Verma and S. Nag, "An Inverse Discrete Cosine Transform (IDCT) Implementation in Virtex for MPEG Video Applications," Xilinx Application Note, 1999.

[7] "FIDCT Forward/Inverse Discrete Cosine Transform," CSELT S.p.A's product specification, Sept. 2000.

[8] T. Chang, C. Kung, C. Jen, "A Simple Processor Core Design for DCT/IDCT", *IEEE Trans. Circuits Syst. Video Tech*, Vol. 10, No. 3, pp.439-447, April. 2000

[9] P. B. Denyer and D. Renshaw, *VLSI Signal Processing: A Bit-Serial Approach*, Addison-Wesley, 1986.

[10] Milos D. Ercegovac and thomas Lang, "On-Line Arithmetic, A Design Methodology and Applications in Digital Signal Processing", *VLSI Signal Processing, III*, 1988.

[11] C.-Y.Hung and P. Landman, "Compact inverse discrete cosine transform circuit for MPEG video decoding," in *Proc. IEEE Workshop Signal Processing Systems*, 1997, pp. 364-373.

[12] R. I. Hartley and K. K. Parhi, *Digit-Serial Computation*. Kluwer, 1995.

[13] "IEEE Standard Specifications for the Implementation of 8×8 Inverse Discrete Cosine Transform", *IEEE Standard 1180-1990*, March, 1991.

[14] S. C. Knowles, J. G. McWhirter, R. F. Woods, and J. V. McCanny, "Bit-level systolic architectures for high performance IIR filtering," *Journal of VLSI signal Processing*, pp.207-212, 1980.

[15] Y. Katayama, T. Kitsuki, and Y, Ooi, "A block processing unit in a single-chip MPEG-2 video encoder LSI," in *Proc. IEEE Workshop Signal Processing Systems*, 1997, pp.459-468.

[16] Weiping Li, "A New Algorithm to Compute the DCT and its Inverse", *IEEE Trans. on Signal Processing*, Vol. 39, No. 6, pp.1305-1313, June 1991.

[17] Yung-Pin Lee, T. Chen, L. Chen, M. Chen, C. Ku, "A Cost-Effective Architecture for 8 x 8 Two-Dimensional DCT/IDCT Using Direct Method", *IEEE Trans. Circuits Syst. video Tech.*, Vol. 7, No. 3, June 1997

[18] A. Madisetti, A. N. Willson, "A 100 MHz 2-D 8 x 8 DCT/IDCT Processor for HDTV Applications," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 5, No.2, pp.158-164, April. 1995.

[19] L. Montalvo and A. Guyot, "Combinational digit-set converters for hybrid radix-4 arithmetic," in *Proc. of IEEE International Conference on Computer Design ICCD'94*, pp.498-503, Oct. 1994.

[20] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters", *IEEE Acoustics Speech and Signal Processing*, ASSP-22, December 1974.

[21] Keshab K. Parhi, VLSI digital signal processing systems: design and implementation, Wiley, 1999.

[22] R. Rambaldi, A. Ugazzoni, and R. Guerrieri, "A 3.5 uW 1.1V gate array 8x8 IDCT processor for video-telephony," *Proc. IEEE ICASSP*, 1998, vol. 5, pp. 2993-2996.

[23] D. Slawecki and W. Lee, "DCT/IDCT Processor Design for High Data Rate Image Coding", *IEEE Tran. on Circuits and Systems for Video Tech.*, vol. 2, No. 2, pp. 135-146, June 1992.

[24] M. J. S. Smith, *Application-Specific Integrated Circuits*, Addison Wesley, 1997.

[25] J.E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Computers*, Vol. EC-8, No.3, pp.330-334, Sept. 1959.

[26] Stanley A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review", *IEEE ASSP Magazine*, July 1989.

[27] "X_DCT_IDCT Forward and Inverse Discrete Cosine Transform," Xentec's product specification, Feb. 2000.

[28] Logicore product specification, Xilinx Inc., 2000.

[29] T. Xanthopoulos, A. Chandrakasan, "A low-power IDCT macrocell for MPEG2 MP@ML exploiting data distribution propertier for minimal activity," in *Proc. Symp. VLSI Circuits*, 1998, pp. 38-39

[30] Y. Yang, C. Wang, M.O. Ahmad, M.N.S. Swamy, "An On-Line CORDIC Based 2-D IDCT using Distributed Arithmetic," *Proc. Sixth International Symposium on Signal Processing and its Applications*, Kuala-Lumpur, Malaysia, Aug. 2001.

[31] Y. Yang, C. Wang, M.O. Ahmad, M.N.S. Swamy, "An On-Line Radix-4 CORDIC 2-D IDCT Core," *The 1st IEEE International Symposium on Signal Processing and Information Technology*, pp. 56-59, Cairo, Egypt, Dec. 2001.

[32] Y. Yang, C. Wang, M.O. Ahmad, M.N.S. Swamy, "An FPGA Implementation of an On-Line Radix-4 CORDIC 2-D IDCT Core," *Proc. IEEE International Symposium on Circuits and Systems,* Scottsdale, Arizona, May 2002.

[33] F. Zhou and P. Kornerup, "High Speed DCT Using a pipelined CORDIC Algorithm," *Proceedings of the 12th IEEE Symposium on Computer Arithmetic,* Bath, UK, July, 1995, pp.180-187.