Efficient DCT blocks in sub-micron CMOS.

Tracy Ann. Franklin
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1880

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
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.

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

UMI®
EFFICIENT DCT BLOCKS IN SUB MICRON CMOS

By

Tracy A. Franklin

A Thesis
Submitted to the College of Graduate Studies and Research through Electrical Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada

2000

© 2000 Tracy A. Franklin
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52550-3
ABSTRACT

The DCT block transform is used every day in the compression of images, video and audio for the transmission of television signals, multimedia graphics, audio and video all over the world.

Image compression is the efficient coding of digital images to reduce the number of bits required, to represent an image, while maintaining a quality image. Data compression is the transmission of data over communication channels in an effective manner that uses as few bits as possible to minimize the bandwidth required, while maintaining distortions within limits.

The efficient implementation of the DCT block in sub-micron CMOS continues to be dependent on the degree of regularity of the chip’s architecture. Whereas the chip’s required speed is dependent on the degree of parallelism of the DCT transform algorithm-architecture. That is, the speed of the block DCT chip is dependent on the subband signals maintaining a parallel FIR filter bank form. The block DCT transform is efficiently implemented when all the multiplication operations are removed from the architecture, thus minimizing the size of the chip.

This thesis deals with the VLSI implementation of a multiply free approximate DCT architecture. The approximate DCT has the capability of mapping integers to integers in sub-micron CMOS with near perfect reconstruction. The approximate DCT is based on the algorithm-architecture found in [11], the binDCT. The forward binDCT chip and inverse binDCT chip designs assume the use of image data as their input. The use of an image as the input signal places the constraint that the chip is to be designed to implement unsigned binary arithmetic.
ACKNOWLEDGEMENTS

I would like to thank Dr. J. J. Soltis and Dr. T. D. Tran for their support, guidance and encouragement during the course of this research. I would also like to thank Dr. G. Jullien, Dr. Ahmadi, and the members of the University of Windsor, VLSI Research Group for their advice and comments throughout the research.

I would especially, like to thank CMC, Canadian Microelectronics Corporation supports staff for their advice in implementing the chips in 0.35\(\mu\) CMOS technology.

Finally, I would like to thank my parents, siblings, grandmother, aunts and uncles for their constant love, encouragement, and patience.
TABLE OF CONTENTS

ABSTRACT iii
ACKNOWLEDGEMENTS iv
LIST OF FIGURES viii
LIST OF ABBREVIATIONS x

CHAPTER

1. INTRODUCTION 1
   1.1 VLSI Design Constraints 1
   1.2 Review of the Literature 1
   1.3 Thesis Overview 4
   1.4 Thesis Organization 5

2. ALGORITHM-ARCHITECTURE TECHNIQUES FOR DSP TRANSFORMS 7
   2.1 Fast Convolution Algorithms 7
       2.1.1 Winograd Algorithm 7
       2.1.2 Iterated Convolution Algorithm 9
       2.1.3 Cyclic Convolution 10
   2.2 Algorithmic Strength Reduction Techniques for Parallel FIR Filters 11
       2.2.1 Block processing using polyphase decomposition 11
       2.2.2 Fast FIR Algorithms 14
           2.2.2.1 Low-complexity parallel FIR filters 14
           2.2.2.2 Parallel filtering algorithms from linear convolution 15
           2.2.2.3 Fast parallel FIR algorithms for large block sizes 16
   2.3 Numerical Strength Reduction Techniques 17
   2.4 Algorithm-Architecture Transformation of the 1-D DCT 18
   2.5 Summary 22

I. ONE-DIMENSIONAL BINDCCT ALGORITHM 23
   3.1 Introduction 23
   3.2 Summary of Algorithm 24
   3.3 Lifting Steps 26
   3.4 Matlab Simulations 29
   3.5 Summary 31

II. VLSI IMPLEMENTATION OF THE ONE-DIMENSIONAL BINDCCT ALGORITHM 31
III. CONCLUSIONS AND FUTURE STUDY
5.1 Conclusion 49
5.2 Future Work 49
  5.2.1 The use of the folding transformation in 2-D orthogonal transform algorithms 50

APPENDIX A: MATLAB CODE
A.1 Lifting Steps 51
A.2 The forward and inverse binDCT m-functions using dedicated lifting steps 52
A.3 The floating point DCT 54
A.4 The forward and inverse binDCT m-functions that use the general lifting step 56
A.5 The forward and inverse binDCT m-functions derived from Real number arithmetic 58

APPENDIX B: RTL DESIGN DESCRIPTION IN VERILOG HDL
B.1 One-dimensional 8-point binDCT 62
B.2 One-dimensional 8-point binDCT RTL description including I/O pads for 0.35μ CMOS technology 71
B.3 Stimulus for the one-dimensional forward binDCT chip RTL description 82
B.4 Stimulus for the one-dimensional forward binDCT chip RTL description 83
B.5 One-dimensional 8-point inverse binDCT RTL description 85
B.6 One-dimensional 8-point inverse binDCT RTL description with the 0.35μ CMOS technology pads included 97
B.7 Stimulus for the one-dimensional 8-point inverse binDCT RTL description 109
B.8 Stimulus for the one-dimensional 8-point inverse binDCT chip RTL description 111

APPENDIX C: VERILOG-XL SIMULATION RESULTS
C.1 The Verilog-XL simulation results for forwarddct rtl.v design 113
C.2 The Verilog-XL simulation results for
Inversedct_rtl.v design
C.3 The Verilog-XL simulation results for forwardbindctchip_rtl.v design
C.4 The Verilog-XL simulation results for inversebindctchip_rtl.v design
C.5 The Verilog-XL simulation results for forwardbindctchip_gates.v design
C.6 The Verilog-XL simulation results for inversebindctchip_gates_1_3.v design
C.7 The Verilog-XL simulation results for forwardbindctchip_gold1.v design
C.8 The Verilog-XL simulation results for inversebindctchip_gold1.v design
C.9 The Verilog-XL simulation results for bindct2d_rtl.v design

APPENDIX D: PEARL ANALYSIS RESULTS
D.1 Pearl analysis results from Design Planner for the 1-D Forward binDCT
D.2 Pearl analysis results from Design Planner for the 1-D Inverse binDCT
D.3 Pearl analysis results from Silicon Ensemble for the 1-D Forward binDCT
D.4 Pearl analysis results from Silicon Ensemble for the 1-D Inverse binDCT

APPENDIX E: LVS RESULTS
E.1 LVS results for the forward one-dimensional binDCT chip
E.2 LVS results for the inverse one-dimensional binDCT chip

APPENDIX F: THE BINDCT CLOCK TREE GENERATION RESULTS FROM DESIGN PLANNER
F.1 Forward one-dimensional binDCT chip clock tree initial analysis
F.2 Forward one-dimensional binDCT chip clock tree initial timing
F.3 Forward one-dimensional binDCT chip clock tree final analysis
F.4 Forward one-dimensional binDCT chip clock tree final timing
F.5 Inverse one-dimensional binDCT chip clock tree initial analysis
F.6 Inverse one-dimensional binDCT chip clock tree initial timing
F.7 Inverse one-dimensional binDCT chip clock
tree final analysis 134
F.8 Inverse one-dimensional binDCT chip clock
tree final timing 136

APPENDIX G: THE TWO-DIMENSIONAL BINDCT
RTL DESCRIPTION
G.1 The 8 x 8 two-dimensional binDCT chip RTL
description 137
G.2 The test bench for the 2-D binDCT chip 174

APPENDIX H: THE RESULTS FOR THE GATE-LEVEL
CONstrained DESIGN OF THE
BINDCT CHIPS
H.1 Initial no gate-level constraint results for the
forward binDCT 178
H.2 Initial no gate-level constraint results for the
inverse binDCT 185
H.3 Final results for the gate-level constrained
design of the forward binDCT chip 196
H.4 Final results for the gate-level constrained
design of the Inverse binDCT chip 216

REFERENCES 243

VITA AUCTORIS 246
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Traditional 2-parallel FIR filter implementation.</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Reduced-complexity 2-parallel FIR filter implementation.</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>Block diagram of the transposition of the linear convolution</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Step 1 of the 8-point DCT structure.</td>
<td>20</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>Functional units represented in the final block diagram.</td>
<td>21</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Final 8-point DCT structure for implementation.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>The fast forward floating-point DCT implementation.</td>
<td>25</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>The fast forward binDCT version B implementation.</td>
<td>26</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>The lifting step incorporation the floor operator for integer to Integer mapping.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Lifting step using binary bit shifting to perform division and flooring.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>The MATLAB results for the compression of the CROWN image and its reconstruction using the floating-point DCT transform and the binDCT transform.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>System level design of the 1-D 8-point binDCT chip.</td>
<td>32</td>
</tr>
<tr>
<td>Figure 13.</td>
<td>Inside the system level design of the 1-D binDCT chip.</td>
<td>33</td>
</tr>
<tr>
<td>Figure 14.</td>
<td>The 5/8 lifting step is implemented using two shifts and one addition operation.</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 15. Serial -to- parallel input transmission. 37

Figure 16. Parallel -to- serial output transmission. 38

Figure 17a. The forward binDCT chip after floor planning is complete. 42

The power pads have been added. The I/O pads and standard cells have been placed. The binDCT clock tree has been generated at this point and the forward binDCT chip design is re-routed.

Figure 17b. The inverse binDCT chip after the corner pads are inserted, the standard cells placed, the binDCT clock tree has been generated at this point and the inverse binDCT chip design is re-routed and re-placed. 43

Figure 18. The forward binDCT chip after routing is completed in Silicon Ensemble. 45

Figure 19. The inverse binDCT chip after power is routed and the standard cells are routed in Silicon Ensemble. 46

Figure 20. The final forward binDCT chip in DFII prior to filling. 47, 126

Figure 21. The final inverse binDCT chip in DFII prior to filling. 48, 129
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>CRT</td>
<td>Chinese Remainder Theorem</td>
</tr>
<tr>
<td>FFA</td>
<td>Fast Fir Algorithms</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Pictures Exchange Format</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Pictures Exchange Format</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 VLSI Design Constraints

For VLSI implementation, the discrete cosine transform (DCT) block must meet the necessary performance speed specified by the input-sampling rate, at a minimal cost to its fabrication. The performance of the final chip is based on the real-time constraints of the input data signal. Thus a floating-point algorithm needs to be developed efficiently and implemented accordingly.

From a general standpoint all of the algorithm-architecture techniques used in past DCT block transform designs, for VLSI implementation have consider the following constraints: the speed of the chip, minimizing the chip area, the regularity of the algorithm structure, and the size of the block to be processed, N. The degree of regularity of the architecture is highly dependent on the structure of the algorithm, whereas, the speed of the chip is highly dependent on the degree of parallelism of the final architecture.

1.2 A Review of Literature

Over the past two decades, the DCT block has been implemented in VLSI by using fast convolution algorithm techniques to reduce the number of multiplication operations. Algorithmic strength reduction transformations have also been used to implement the DCT block in VLSI by reducing the amount of silicon area used or the amount power consumed by the chip.

The fast convolution algorithms such as Winograd’s short convolution algorithm, the Iterated convolution algorithm and Cyclic convolution algorithm all reduce the number of
multiplication operations, while increasing the number of addition operations [1], [8]-
[10].

The algorithmic strength reduction transformations for parallel FIR filters increase the
effective throughput of the original filter by decomposing the FIR filter into its polyphase
components. The strength reduction transformations also reduce the original filters’
power consumption.

The combination of fast short-length convolution algorithms applied to small block sized
parallel FIR filters have been shown to generate a new class of fast FIR filtering
algorithms (FFAs), that are used to develop large block sized parallel FIR filters [1].
In [2]-[7], all of the two-dimensional DCT transforms were based on the row-column
decomposition technique that uses the separability property of two-dimensional
orthogonal transforms as a basis to design a 2-D DCT chip that consists of two 1-D
DCTs.

The reduction of the two-dimensional DCT transform into two one-dimensional DCT
transforms is shown below as a proof.

Proof: If we represent the digital picture as an M x N matrix, [g], and the two-
dimensional orthogonal transform of the matrix [g] as [G], then the two-dimensional
transform can be implemented by a series of one-dimensional transforms,

\[
[G] = [T_M][g][T_N]^T \quad (1.2.1),
\]

\[
[g] = [T_M]^T[G][T_N] \quad (1.2.1),
\]

where \([T_M]\) is an M x M and \([T_N]\) is an N x N real transformation matrices. Let \([G]\) be
the two-dimensional DCT of \([g]\), then

\[
G(u, v) = \frac{2e(u)e(v)}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m, n) \cos \left(\frac{(2m + 1)u\pi}{2M}\right) \cos \left(\frac{(2n + 1)v\pi}{2N}\right) \quad (1.2.2),
\]
where \( u = 0, \ldots, M - 1 \), \( v = 0, \ldots, N - 1 \),

\[
e(u) = \begin{cases} 
\frac{1}{\sqrt{2}}, & u = 0 \\
1, & \text{otherwise}
\end{cases}
\]

and,

\[
e(v) = \begin{cases} 
\frac{1}{\sqrt{2}}, & v = 0 \\
1, & \text{otherwise}
\end{cases}
\]

Using the separability property, the resulting two-dimensional DCT is

\[
G(u, v) = \frac{2}{\sqrt{M}} e(u) \sum_{m=0}^{M-1} \left( \sum_{n=0}^{N-1} \frac{2}{\sqrt{N}} e(v) \cos \left( \frac{(2n + 1)v\pi}{2N} \right) \right) \cos \left( \frac{(2m + 1)u\pi}{2M} \right) \quad (1.2.3),
\]

where \( u = 0, \ldots, M - 1 \), \( v = 0, \ldots, N - 1 \),

\[
e(u) = \begin{cases} 
\frac{1}{\sqrt{2}}, & u = 0 \\
1, & \text{otherwise}
\end{cases}
\]

and,

\[
e(v) = \begin{cases} 
\frac{1}{\sqrt{2}}, & v = 0 \\
1, & \text{otherwise}
\end{cases}
\]

The inner summation represents an \( N \)-point one-dimensional DCT of the rows of \( g \),

and the outer summation represents an \( M \)-point one-dimensional DCT of the columns of

the semi-transformed matrix [18].

The row-column decomposition implementation of the 2-D DCT requires an efficient

algorithm-architecture for the 1-D DCT, for the VLSI implementation to be effective.

The 1-D DCT must have a minimum number of multiplication operations, while keeping

a regular architecture structure for VLSI implementation.
The efficient design of the 1-D DCT block lies in the algorithm-architecture approach one chooses to use. The most efficient method would be a multiplication-free algorithm, implemented using:

1) an algorithm for encoding of algebraic integers [2] using the Feig-Winograd algorithm utilizing systolic arrays; or,

2) direct method implementing a fast-forward algorithm [3]; or,

3) or an algorithm-architecture based on cyclic convolutions [4]; or,

4) On the other hand, skew-circular convolution techniques utilizing parallel filters and the folding transformation [5].

Several other 1-D DCT algorithm approaches have also been developed in the past, that only reduce the number of multiplication operations, while increasing the number of addition operations. These approaches maintain a regular architecture structure. The use of polyphase decomposition to reduce the complexity of the 1-D DCT algorithm and increase the throughput was illustrated in [6], where the number of multiplication operations was reduced significantly from 64 to 12. The use of the direct method to design a cost-effective 8x8 2-D DCT/IDCT architecture is illustrated in [7]. By using a combination of polyphase decomposition and the folding transformation to reduce the size of chip, a time-multiplexed 2-D DCT architecture is implemented using a minimum number of registers, two 1-D DCT architectures and a control circuit.

1.3 Thesis Overview

The goal of this thesis is to design an efficient DCT block in VLSI, that could possibly be applied to video processing. Since the JPEG and MPEG standards are all based on the DCT transform for compression, an area-efficient, low power, high-throughput
implementation is highly desired. The binDCT architecture is multiply free, area-efficient, and consumes less power than the traditional fast forward floating-point DCT architecture. The binDCT is basically an Integer DCT transform, that maps integers to integers with almost perfect numerical reconstruction (i.e. a scale version of the original data is obtained).

The forward and inverse binDCT transforms are implemented in 0.35\(\mu\)m CMOS using the black box cell libraries for a full digital design. Based on the 0.35\(\mu\)m CMOS technology statistics an estimated die size of 2.154 mm x 1.314 mm for the forward binDCT transform is calculated, and an estimated die size of 1.818 mm x 1.818 mm for the inverse transform is calculated.

1.4 Thesis Organization

Starting with a brief introduction of the algorithm-architecture transformation techniques for any general transform, an algorithm-architecture transformation is applied to the one-dimensional DCT transform to derive an 8-point one-dimensional DCT architecture. The 8-point one-dimensional DCT architecture derived has undergone a significant decrease in the number of multiplication operations, from 56 to 13. The new architecture is then reduced further by the application of cascaded dyadic shears or dyadic lifting steps to replace the rotation angles within the butterfly pairs. The resulting architecture is multiply free.

The VLSI implementation of the one-dimensional binDCT architecture starts with the system-level approach, then the design of the Register Transfer Level (RTL) hardware description modules for the implementation of one-dimensional binDCT architecture is done. Verilog is used to model, analyze and simulate the RTL description of the chip
written in the HDL language. Synopsys Design Analyzer is used to synthesize the RTL
description into a gate-level netlist. Cadence’s Design Planner is used to plan the chip’s
floor plan, place the standard cells, power stripes and rings. The CTGEN clock
generator tool is used to generate the clock tree for the chip’s main clock. Silicon
Ensemble is used to route the power, and the nets of the chip. Cadence’s Design
Framework is used to perform the final chip testing prior to fabrication, such as (LVS)
layout versus schematic check to ensure the final chip matches the golden netlist, and the
(DRC) design rule check to ensure the final chip does not violate any design rules.
Chapter 2

Algorithm-Architecture Techniques for DSP Transforms

2.1 Fast Convolution Algorithms

Since an adder consumes less area on a chip, and its computation time is much smaller than a multiplier, a reduction in the number of multiplication operations is pre-requisite for VLSI implementation. The fast convolution algorithms reduce the algorithm-architecture complexity by reducing the number of multiplication operations. There are several fast convolution algorithms, the Winograd short convolution algorithm, the Iterated convolution algorithm and the cyclic convolution algorithm.

2.1.1 Winograd Algorithm

The Winograd short convolution algorithm is based on the Chinese Remainder Theorem (CRT) over a polynomial ring. First, the CRT theorem for polynomials is stated as:

Given

\[ c^{(i)}(p) = R_{m^{(i)}(p)}[c(p)], \]

for \( i=0, 1, \ldots, k \), where \( m^{(i)}(p) \) are relatively prime, then,

\[ c(p) = \sum_{i=0}^{k} c^{(i)}(p)N^{(i)}(p)M^{(i)}(p) \mod M(p) \quad (2.1.1.1), \]

where,

\[ M(p) = \prod_{i=0}^{k} m^{(i)}(p), \]

\[ M^{(i)}(p) = \frac{M(p)}{m^{(i)}(p)}, \]

and \( N^{(i)}(p) \) is the solution of
\[ N^{(i)}(p)M^{(i)}(p) + n^{(i)}(p)m^{(i)}(p) = GCD(M^{(i)}(p), m^{(i)}(p)) = 1 \quad (2.1.1.2), \]

provided that degree of \( c(p) \) is less than the degree of \( M(p) \). To solve (2.1.1.2) for \( n^{(i)}(p) \) and \( N^{(i)}(p) \), use the Euclidean greatest common divisor (GCD) algorithm for polynomials. Consider the computation of

\[ s(p) = h(p)x(p) \mod m(p) \quad (2.1.1.3), \]

where, \( m(p) = m^{(0)}(p)m^{(1)}(p)...m^{(k)}(p) \) and \( m^{(i)}(p) \), are pair-wise relatively prime.

Let

\[ s^{(i)}(p) = R_{m^{(i)}(p)}[s(p)], \]

for \( i=0,1...k \). As long as, \( \deg s(p) < \deg m(p) \), the polynomial \( s(p) \) can be uniquely determined by

\[ s(p) = \sum_{i=0}^{k} s^{(i)}(p)N^{(i)}(p)M^{(i)}(p) \mod m(p) \quad (2.1.1.4). \]

Note that when \( \deg h(p)x(p) < \deg m(p), \)

\[ s(p) = h(p)x(p) \mod m(p) = h(p)x(p) \quad (2.1.1.5). \]

The structure of the Winograd convolution algorithm is summarized as follows:

1) Choose a polynomial \( m(p) \) with a degree higher than the degree of \( h(p)x(p) \) and factor it into \( k+1 \) relatively prime polynomials with real coefficients, i.e.

\[ m(p) = m^{(0)}(p)m^{(1)}(p)...m^{(k)}(p). \]

2) Let

\[ M^{(i)}(p) = \frac{M(p)}{m^{(i)}(p)}, \]

use the Euclidean GCD algorithm to solve (2.1.1.2) for \( N^{(i)}(p) \).

3) Compute
\[ h^{(i)}(p) = h(p) \mod m^{(i)}(p), \quad (2.1.1.6) \]
\[ x^{(i)}(p) = x(p) \mod m^{(i)}(p), \]
for \( i=0,1,\ldots,k \).

4) Compute
\[ s(i)(p) = h^{(i)}(p)x^{(i)}(p) \mod (i)(p), \quad (2.1.1.7), \]
for \( i=0,1,\ldots,k \). Note that the short convolutions, represented as \( h^{u}(p)x^{u}(p) \), require multiplication.

5) Compute \( s(p) \) using (2.1.1.4).

The number of multiplication operations, in the Winograd algorithm is dependent on the degree of each \( m^{(i)}(p) \). Thus, the degree of \( m(p) \) should be as small as possible [1][2].

2.1.2 Iterated Convolution Algorithm

The iterated convolution algorithm uses efficient short-length convolution algorithms iteratively to build longer convolutions. Resulting in an architecture structure that possesses a good balance between multiplication complexity and addition complexity.

The procedure for the iterated convolution is summarized as:

1) Decompose the long convolution into several levels of short convolutions.

2) Construct fast convolution algorithms for the short convolutions.

3) Use the short convolution algorithms to hierarchically implement the long convolution. Starting with the lower order short convolutions, and proceeding to the next order short convolution. [8] [9]
2.1.3 Cyclic Convolution

Cyclic convolution is also known as circular convolution. Let $h = \{h_0, h_1, \ldots, h_{n-1}\}$ be the filter coefficients and $x = \{x_0, x_1, \ldots, x_{n-1}\}$ be the data sequence. Then cyclic convolution can be expressed as

$$s(p) = h \circ_a x = h(p)x(p) \mod(p^n - 1), \quad (2.1.3.1)$$

and the output samples are given by

$$s_i = \sum_{k=0}^{n-1} h_{(i-k)\mod(n)} x_k, \quad (2.1.3.2),$$

for $i=0, 1, \ldots, n-1$. The cyclic convolution can also be computed as a linear convolution reduced by modulo $p^n - 1$, or the cyclic convolution can be computed using the CRT with $m(p) = p^n - 1$, which is simpler.[1][3]

2.2 Algorithmic Strength Reduction Techniques for Parallel FIR Filters

Algorithmic strength reduction leads to a reduction in the complexity of the architecture used by exploiting the presence of substructure sharing. The algorithmic strength reduction transformation can lead to a reduction in silicon area and power consumption in the VLSI implementation.

Since the FIR filter is one of the main processing elements in any DSP system, it is used in applications such as image and video processing, etc. In video processing, the FIR filter is required to operate at high frequencies. Block processing techniques are applied to the digital FIR filters to increase its throughput, or reduce its power consumption.

Since traditional applications of block processing to a FIR filter, increases the circuit size linearly by a factor, $L$, the block size of the parallel FIR filters. The circuit area for $L$-parallel FIR filters becomes: $L$-parallel FIR filters area = $L \times$ original FIR filter area.
Due to fabrication costs, and size limits for VLSI implementation the traditional approach is deferred. Instead parallel FIR filtering architectures that consume less area are chosen. The algorithmic strength reduction techniques employ several fast filtering algorithms for smaller block sizes to design parallel FIR filters for larger block sizes.

2.2.1 Block processing using Polyphase Decomposition

Using the polyphase decomposition technique used in multirate signal processing the parallel FIR filters can be formulated. An N-tap FIR filter can be expressed in the time domain as

\[ y(n) = h(n) * x(n) = \sum_{i=0}^{N-1} h(i)x(n-i), n = 0,1,2,\ldots, \infty, \quad (10), \]

where \( \{x(n)\} \) is an infinite length input sequence and the sequence \( \{h(n)\} \) contains FIR filter coefficients of length \( N \), or in the z-domain as

\[ Y(z) = H(z)X(z) = \sum_{n=0}^{N-1} h(n)z^{-n} \sum_{n=0}^{\infty} x(n)z^{-n}, \quad (2.2.1.1). \]

The input sequence \( \{x(0), x(1), x(2), \ldots\} \) can be decomposed into an even-numbered part, and an odd-numbered part as follows,

\[
X(z) = x(0) + x(1)z^{-1} + x(2)z^{-2} + x(3)z^{-3} + ... \\
= x(0) + x(2)z^{-2} + x(4)z^{-4} + ... + z^{-l}[x(l) + x(3)z^{-2} + x(5)z^{-4} + ...], \quad (2.2.1.2),
\]

\[ = X_0(z^2) + z^{-1}X_1(z^2), \]

where \( X_0(z^2) \) and \( X_1(z^2) \) are the z-transforms of \( x(2k) \) and \( x(2k+1) \), respectively. In (2.2.1.2), \( X(z) \) is decomposed into two polyphases. The length-N filter coefficients \( H(z) \), can be similarly decomposed as

\[ H(z) = H_0(z^2) + z^{-1}H_1(z^2), \quad (2.2.1.3), \]
where $H_0(z^2)$ and $H_1(z^2)$ are of the length $\frac{N}{2}$ and are referred to as the even sub-filter and the odd sub-filter, respectively. The even-numbered output sequence $y(2k)$ and the odd numbered output sequence $y(2k+1)$ for $0 \leq k \leq \infty$, can be computed as,

$$Y(z) = Y_0(z^2) + z^{-1}Y_1(z^2)$$

$$= (X_0(z^2) + z^{-1}X_1(z^2))(H_0(z^2) + z^{-1}H_1(z^2))$$

$$= X_0(z^2)H_0(z^2) + z^{-1}(X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2)) + z^{-2}X_1(z^2)H_1(z^2)$$

that is,

$$Y_0(z^2) = X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2), Y_1(z^2)$$

$$= X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2)$$

Where $Y_0(z^2)$ and $Y_1(z^2)$ correspond to the $y(2k)$ and $y(2k+1)$ in the time domain, respectively. The filtering operations in (2.2.1.5) process the two inputs $x(2k)$ and $x(2k+1)$ and generate the two outputs $y(2k)$ and $y(2k+1)$ during every iteration, it is referred to as the 2-parallel FIR filter. In matrix form, it is written as,

$$\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} H_0 & z^{-2}H_1 \\ H_1 & H_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix},$$

(2.2.1.6).

The 2-parallel FIR filter implementation is shown in Figure 1. The resulting 2-parallel FIR filtering structure, requires $2N$ multiplication operations and $2(N-1)$ addition operations.
2.2.2 Fast FIR Algorithms

The fast FIR algorithms (FFAs) are based on Winograd's work to produce, reduced-complexity parallel filtering structures. Based on Winograd's work, two polynomials of degree $L-1$ can be multiplied using only $2L-1$ product terms. The reduction in the number of multiplication operations results in the increase of the number of additions required for implementation. Because the product terms in the polynomial formulation of the FIR filter are equivalent to filtering operations in the block formulation, this implies that the parallel FIR filters can be realized using approximately $2L-1$ FIR filters of length $\frac{N}{L}$.

Using this approach, the $L$-parallel FIR filter can be implemented using approximately $(2L-1)$ FIR filtering operations of length $\frac{N}{L}$. Thus deriving an efficient VLSI implementation for parallel FIR filters based on FFAs.
2.2.2.1 Low Complexity 2-Parallel Fast FIR Filter

The polyphase decomposed 2-parallel FIR filter from (2.2.1.5) can be rewritten as:

\[ Y_0(z^2) = X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2), \]
\[ Y_1(z^2) = (H_0(z^2) + H_1(z^2))(X_0(z^2) + X_1(z^2)) - X_0(z^2)H_0(z^2) - X_1(z^2)H_1(z^2) \] (2.2.2.1.1).

This 2-parallel fast FIR filter contains 5 sub-filters. The two terms \( H_0(z^2)X_0(z^2) \) and \( H_1(z^2)X_1(z^2) \) are common and can be shared for the computation of \( Y_0(z^2) \) and \( Y_1(z^2) \).

The reduced-complexity 2-parallel FIR filter implementation is depicted in Figure 2.

The low-complexity 2-parallel filter structure computes a block of two outputs using three distinct sub-filters of length \( \frac{N}{2} \) and four pre/post-processing addition operations.

Requiring \( \frac{3N}{2} \) multiplication operations and \( 3 \left( \frac{N}{2} - 1 \right) + 4 \) addition operations, as opposed to \( 2N \) multiplication operations and \( 2(N-1) \) addition operations in the traditional parallel filter derived from polyphase decomposition.

The 2-parallel filter in (2.2.2.1.1) can be written in matrix form as

\[ Y_2 = Q_2H_2P_2X_2, \]
\[
\begin{bmatrix}
Y_0 \\
Y_1
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & z^{-2} \\
-1 & 1 & -1
\end{bmatrix}
\text{diag}
\begin{bmatrix}
H_0 & 1 & 0 \\
H_0 + H_1 & 1 & 1 \\
H_1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_0 \\
X_1
\end{bmatrix},
\tag{2.2.2.1.2}
\]

\(Q_2\) is the post-processing matrix that determines the manner in which the filter outputs should be combined to correctly produce the parallel outputs. \(P_2\) is the pre-processing matrix that determines the manner in which the inputs should be combined. The entries on the diagonal of \(H_2\), are the sub-filters required in this parallel FIR filter. [1][8]

### 2.2.2.2 Parallel Filtering Algorithms from Linear Convolution

Any \(L \times L\) convolution algorithm can be used to derive an \(L\)-parallel fast filter structure. Starting with an optimal linear convolution, take the optimal linear convolution’s transpose to generate the parallel filtering algorithm. For example, the transpose of the matrix in a \(2\times2\) linear convolution algorithm is given by,

\[
\begin{bmatrix}
S_2 \\
S_1 \\
S_0
\end{bmatrix} =
\begin{bmatrix}
h_1 & 0 \\
h_0 & h_1 \\
0 & h_0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_0 \\
x_0
\end{bmatrix},
\tag{2.2.2.2.1}
\]

It can be used to obtain the 2-parallel FIR filter,

\[
\begin{bmatrix}
Y_0 \\
Y_1
\end{bmatrix} =
\begin{bmatrix}
H_1 & H_0 & 0 \\
0 & H_1 & H_0
\end{bmatrix}
\begin{bmatrix}
z^{-2}X_1 \\
X_0 \\
X_0
\end{bmatrix},
\tag{2.2.2.2.2}
\]

To generate the 2-parallel FIR filter using \(2\times2\) fast convolution, consider the following optimal \(2\times2\) linear convolution written in matrix form,

\[
s = \text{CH}x,
\]

\[
\begin{bmatrix}
S_2 \\
S_1 \\
S_0
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
-1 & 1 & -1 \\
0 & 0 & 1
\end{bmatrix}
\text{diag}
\begin{bmatrix}
h_1 \\
h_0 + h_1 \\
h_0
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
1 & 1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_0 \\
x_1
\end{bmatrix},
\tag{2.2.2.2.3}
\]
Taking the transpose of this algorithm and the proper substitution of the \( s_i \)'s, \( h_i \)'s and \( x_i \)'s we have:

\[
Y = (CHA)^T X = QHPX
\]

\[
\begin{bmatrix}
Y_0 \\
Y_1
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix}
\text{diag}
\begin{bmatrix}
H_1 \\
H_1 + H_0 \\
H_0
\end{bmatrix}
\begin{bmatrix}
1 & -1 & 0 \\
0 & 1 & 0 \\
0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
z^{-1}X_1 \\
X_0 \\
X_1
\end{bmatrix}, \quad (2.2.2.4).
\]

The resulting 2-parallel FIR filter architecture is depicted in Figure 3.

![Block diagram of the transposition of the linear convolution reduced-complexity 2-parallel FIR filter.](image)

Figure 3. Block diagram of the transposition of the linear convolution reduced-complexity 2-parallel FIR filter.

This method can be applied to any FFA to generate an equivalent parallel filtering realization. [1][9]

### 2.2.2.3 Fast Parallel FIR Algorithms for Large Block Sizes

Parallel FIR filters with large block sizes can be designed by cascading smaller length fast parallel FIR filters. When cascading the FFAs, it is important to apply the smallest parallel FIR filter first, thus leading to the lowest implementation cost. It is extremely important to keep track of the number of multiplication operations and the number of additions required for the filtering structure.

The number of required multiplication is given as
\[ M = \frac{N}{\prod_{i=1}^{r} L_i} \prod_{i=1}^{r} M_i, \quad (2.2.2.3.1). \]

In [1] the number of additions is given as

\[ A = A_1 \prod_{i=2}^{r} L_i + \sum_{i=2}^{r} A_i \left( \prod_{j=i+1}^{r} L_j \right) \left( \prod_{k=1}^{i-1} M_k \right) + \left[ \prod_{i=1}^{r} M_i \right] \left[ \frac{N}{\prod_{i=1}^{r} L_i} - 1 \right] \quad (2.2.2.3.2). \]

Given an L-parallel filter with \( L = L_1 L_2 ... L_r \), where \( r \) is the number of fast FIR algorithms (FFAs) used. \( L_i \) is the block size of the FFA at step-\( i \). \( M_i \) is the number of filters that result from the application of the \( i \)-th FFA. \( N \) is the length of the filter. \( A_i \) is the number of pre/post-processing adders required by the \( i \)-th FFA.

### 2.3 Numerical Strength Reduction Techniques

The numerical strength reduction transformations are based on sub-expression elimination to restructure the transform algorithm-architecture in such a manner that the performance in terms of speed, power and area are improved. This type of strength reduction reduces the total capacitance of the chip. Therefore, reducing the overall power consumption of the chip.

The presence of constant multiplication operations that operate on a common variable in a DSP algorithm presents the possibility of using dedicated shift-and-add multipliers. Sub-expression elimination is applied to a set of constant multiplication operations that have the same constant multiplicand. The method begins by examining the dedicated shift-and-add implementations of constant multiplication operations to find redundant operations that can be performed once and shared amongst the constant multiplication operations. Sub-expression elimination is a numerical transformation of the constant
multiplication operations that can lead to efficient hardware implementations in terms of area, power and speed [1][10].

2.4 Algorithm-Architecture Transform of the 1-D DCT

The DCT is a frequency transform used in the compression of images and video. The DCT is applied over fixed-size blocks, each eight-points by eight-points (i.e. 8 x 8) in size, to keep the correlation between pixels to a maximum. The spatial redundancy within the image is used to reduce the number of required bits needed to represent the image.

The forward N-point DCT of the data sequences \(x(n)\) for \(n=0,1,2, \ldots, N-1\) is \(X(k)\) for \(k=0,1,2, \ldots, N-1\), given by

\[
X(k) = e(k) \sum_{n=0}^{N-1} x(n) \cos \left(\frac{(2n + 1)k\pi}{2N}\right), \quad (2.4.1)
\]

where,

\[
e(k) = \begin{cases} 
1, & k = 0 \\
\sqrt{2}, & k \neq 0 
\end{cases}
\]

The inverse N-point DCT (IDCT) of the sequences \(X(k)\) for \(k=0,1,2, \ldots, N-1\) is \(x(n)\) for \(n=0,1,2, \ldots, N-1\), given by

\[
x(n) = \frac{2}{N} e(k) \sum_{k=0}^{N-1} X(k) \cos \left(\frac{(2n + 1)k\pi}{2N}\right) \quad (2.4.2).
\]

Note, that the inverse transformation matrix is a scaled version of the transposed forward transformation matrix.

Directly implementing (2.4.1) or (2.4.2) requires \(N(N-1)\) multiplication operations, which results in a large amount of silicon being used for VLSI fabrication. To reduce the
amount of silicon used, the algorithm-architecture transformation is used. If we consider

the 8-point DCT, where N=8, the forward 8-point DCT can be written in matrix form as

\[
\begin{bmatrix}
X(0) \\
X(1) \\
X(2) \\
X(3) \\
X(4) \\
X(5) \\
X(6) \\
X(7)
\end{bmatrix} =
\begin{bmatrix}
c4 & c4 & c4 & c4 & c4 & c4 & c4 & x(0) \\
c1 & c3 & c5 & c7 & c9 & c11 & c13 & c15 \\
c2 & c6 & c10 & c14 & c18 & c22 & c26 & c30 \\
c3 & c9 & c15 & c21 & c27 & c1 & c7 & c13 \\
c4 & c12 & c20 & c28 & c4 & c12 & c20 & c28 \\
c5 & c15 & c25 & c3 & c13 & c23 & c1 & c11 \\
c6 & c18 & c30 & c10 & c22 & c2 & c14 & c26 \\
c7 & c21 & c3 & c17 & c31 & c13 & c27 & c9
\end{bmatrix} \cdot \begin{bmatrix}
x(1) \\
x(2) \\
x(3) \\
x(4) \\
x(5) \\
x(6) \\
x(7)
\end{bmatrix},
\]

(2.4.3),

where \( c_i = \cos \frac{i\pi}{16} \) are the DCT transform coefficients. Step 1, using trigonometric

properties, the 8-point DCT can be reduced into a simpler form, such as

\[
\begin{bmatrix}
X(0) \\
X(1) \\
X(2) \\
X(3) \\
X(4) \\
X(5) \\
X(6) \\
X(7)
\end{bmatrix} =
\begin{bmatrix}
c4 & c4 & c4 & c4 & c4 & c4 & c4 & x(0) \\
c1 & c3 & c5 & c7 & -c7 & -c5 & -c3 & -c1 \\
c2 & c6 & -c6 & -c2 & -c2 & -c6 & c6 & c2 \\
c3 & -c7 & -c1 & -c5 & c5 & c1 & c7 & -c3 \\
c4 & -c4 & -c4 & c4 & c4 & -c4 & -c4 & c4 \\
c5 & -c4 & -c4 & -c4 & c4 & c4 & c4 & c4 \\
c6 & -c2 & c2 & -c6 & -c6 & c2 & -c2 & c6 \\
c7 & -c5 & c3 & -c1 & c1 & -c3 & c5 & -c7
\end{bmatrix} \cdot \begin{bmatrix}
x(1) \\
x(2) \\
x(3) \\
x(4) \\
x(5) \\
x(6) \\
x(7)
\end{bmatrix},
\]

(2.4.4).

The 8-point DCT can then be rewritten in parallel form as

\[
\begin{align*}
X(1) &= M_0c_1 + M_1c_7 + M_2c_3 + M_3c_5 \\
X(7) &= M_0c_7 - M_1c_1 - M_2c_5 + M_3c_3 \\
X(3) &= M_0c_5 - M_1c_3 - M_2c_7 - M_3c_1 \\
X(5) &= M_0c_2 + M_1c_3 - M_2c_1 + M_3c_7 \\
X(2) &= M_{10}c_2 + M_{11}c_6 \\
X(6) &= M_{10}c_6 - M_{11}c_2 \\
X(4) &= M_{100}c_4 \\
X(0) &= P_{100}c_4
\end{align*}
\]

(2.4.5),

where
\[ M_0 = x_0 - x_7, \quad P_0 = x_0 + x_7 \]
\[ M_1 = x_1 - x_6, \quad P_1 = x_1 + x_6 \]
\[ M_2 = x_2 - x_5, \quad P_2 = x_2 + x_5 \]
\[ M_3 = x_3 - x_4, \quad P_3 = x_3 + x_4 \]
\[ M_{10} = P_0 - P_1, \quad P_{10} = P_0 + P_1 \]
\[ M_{11} = P_2 - P_3, \quad P_{11} = P_2 + P_3 \]
\[ M_{100} = P_{10} - P_{11}, \quad P_{100} = P_{10} + P_{11} \]

Resulting in 22 multiplication operations according to (2.4.4) and (2.4.5). The resulting architecture is illustrated in Figure 4.

Figure 4. Step 1 of the 8-point DCT structure.

To generate a block diagram group of the structure depicted in Figure 4, butterfly pairs in Figure 4 are grouped into different functional units, using the functional units described
in Figure 5.

Figure 5. Functional units represented in the final block diagram.

The block depicted in Figure 5a was found to be realized using 3 multiplication operations and 3 addition operations, instead of using 4 multiplication operations and 2 addition operations. If we define the block in Figure 5a, with \( a = \sin(\theta) \) and \( b = \cos(\theta) \), reverse the outputs, the butterfly addition block can be converted into a rotator block that computes,

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  \cos \theta & -\sin \theta \\
  \sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}.
\]

The final step of the algorithm-architecture transformation exploits the reduced-complexity implementation of the butterfly block that contains the DCT coefficients as the multiplication factors \( b \) and \( a \) as in Figure 5a. The final implementation structure is
depicted in Figure 6, where only 13 multiplication operations are required.

Figure 6. Final 8-point DCT structure for implementation.

2.5 Summary

The long convolution algorithms are derived using several fast short-length convolutions. The short-length convolution algorithms can be based on the linear convolution algorithms, or the Winograd algorithms, or the cyclic convolution algorithms. The short-length convolution algorithms form the basis for the design of fast parallel FIR algorithms.

The algorithm-architecture transformations adapt a given transform to an algorithm that reduces the complexity of the architecture, or changes the algorithm's description in a systematic way, reducing the total number of multiplication operations in the final architecture. In the case of the 1-D DCT, the application of the algorithm-architecture transformations showed a large reduction in the number of multiplication operations from 56 to 13.
Chapter 3

One - Dimensional binDCT Algorithm

3.1 Introduction

A novel block transform is introduced in [11][13][14] that possesses integer-mapping capabilities. The coefficients have a dyadic rational form that leads to an architecture implementation utilizing only shift-and-add operations. The binDCT substitutes the rotation blocks in the architecture with cascading shear transformations. Borrowing from linear algebra[12], the rotation in a plane is a linear transformation, that is,

\[ T(1,0) = (\cos \theta, \sin \theta) \]

\[ T(0,1) = \left( \cos \left( \theta + \frac{\pi}{2} \right), \sin \left( \theta + \frac{\pi}{2} \right) \right) = (-\sin \theta, \cos \theta) \]

\[ T \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ 0 \end{bmatrix} + T \begin{bmatrix} 0 \\ y \end{bmatrix} \]

\[ = x T \begin{bmatrix} 1 \\ 0 \end{bmatrix} + y T \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]

\[ = x \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} + y \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix} \]

\[ = \begin{bmatrix} x \cos \theta - y \sin \theta \\ x \sin \theta + y \cos \theta \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]

, (3.1.1).

Linear algebra [12] states that a shear is a linear transformation that transforms a line segment in one direction. A shear in the x-direction is
\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
= x \begin{bmatrix}
    1 \\
    0
\end{bmatrix}
+ y \begin{bmatrix}
    0 \\
    1
\end{bmatrix}
= \begin{bmatrix}
    1 + yk \\
    y
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]
(3.1.2).

A shear in the y-direction is

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
= x \begin{bmatrix}
    1 \\
    0
\end{bmatrix}
+ y \begin{bmatrix}
    0 \\
    1
\end{bmatrix}
= \begin{bmatrix}
    x \\
    kx + y
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]
(3.1.3).

Resulting in a multiply free block transform that approximates the DCT closely for efficient VLSI implementation in terms of the amount of silicon used and power consumed.

3.2 Summary of the Algorithm

Starting with the filter-bank approach as in [17][15] of parallel FIR filters, the original signal is decomposed into eight subband signals to realize an 8-parallel FIR filter bank. The 8-parallel FIR filter analysis section of the filter bank represents the forward transform architecture of the floating-point DCT. The algorithmic strength reduction techniques are applied to the resulting 8-parallel FIR filter architecture to reduce the complexity of the architecture. The resulting architecture is shown in Figure 7, which
requires 13 multiplication operations and 29 addition operations for the floating-point DCT.

![Diagram](image)

Figure 7. The fast forward floating-point DCT implementation.

Where $C_i = i\pi/16$ and $S_i = i\pi/16$.

The next step removes the multiplication operations, that is the rotation angles. The rotation angles are performed using two cascaded shears (lifting steps). The final 8-point
The approximated DCT, binDCT version B is computed using only 14 bit shifts and 31 addition operations. Resulting in a significant reduction in the amount of silicon used for implementation.

3.3 Lifting Steps

The lifting step is chosen to be dyadic, that is, a rational that can be written in the form of

\[
\frac{k}{2^m}, k, m \in \mathbb{Z}.
\]

For perfect reconstruction of integers, that is integers mapping to integers.
a floor operator is included in each lifting step, as indicated below in Figure 9.

Figure 9. The lifting step incorporating the floor operator integer to integer mapping.

\[
\begin{align*}
x(0) & \quad \rightarrow \quad x(0) \\
3/8 \\
\lfloor x(1) \rfloor & \quad \rightarrow \quad x(1) - \text{floor}(3/8 \times x(0))
\end{align*}
\]

It is incorporated into the division by using simple binary bit shifting to the right by \( m \) places. The numerator of the dyadic rational lifting step, \( k \), is implemented by using simple bit shift-and-add operations, as illustrated in Figure 10.

Figure 10. Lifting step using binary bit shifting to perform division and flooring.

\[
\begin{align*}
x(0) & \quad \rightarrow \quad x(0) \\
x(0) \ll 1 + x(0) \\
\gg 3 \\
x(1) & \quad \rightarrow \quad x(1) - \text{floor}(3/8 \times x(0))
\end{align*}
\]

To construct the multiplierless DCT approximation, each rotation angle must be approximated using cascading dyadic lifting steps or dyadic shears. Starting with the fast one-dimensional 8-point DCT algorithm that requires as little as, nine multiplication
operations and 29 addition operations for the implementation of the five rotation angles
\[
\left\{ \frac{\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{4}, \frac{7\pi}{16}, \frac{3\pi}{16} \right\}
\] and eight butterfly pairs.

Each rotation angle (or block) can be viewed as a plane rotation, such that according to [12][16], the plane rotation can be performed by three shears,

\[
\begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix} = \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ p & 1 \end{bmatrix}, \quad (3.3.1),
\]

where

\[
u = \frac{\cos \theta - 1}{\sin \theta},
\]

and \(p = \sin \theta\). For VLSI implementation, \(u\) and \(p\) are approximated as dyadic rational, to exploit binary arithmetic.

For an efficient solution, Tran [11][12][13] tries to eliminate the dependency between lifting coefficients. By limiting the factorization of each rotation angle into no more than two dyadic lifting steps, the dependency between lifting coefficients can still be considered.

The following lifting choices were determined by Tran to yield a transform with high coding gain for the symmetric basis functions of the filter bank:

\[
\text{Rotation} \frac{\pi}{4} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad (3.3.2).
\]

\[
\text{Rotation} \frac{3\pi}{8} = \begin{bmatrix} 1 & 0 \\ \frac{3}{8} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{3}{8} \\ 0 & 1 \end{bmatrix}
\]

The anti-symmetric basis functions lifting step choices for the filter bank are:
\[
Rotation \frac{\pi}{4} = \begin{bmatrix}
-1 & 5/8 \\
0 & 3/8 \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix},
\]

\[
Rotation \frac{7\pi}{16} = \begin{bmatrix}
1 & -1/8 \\
0 & 1 \\
\end{bmatrix}
\]

\[
Rotation \frac{3\pi}{16} = \begin{bmatrix}
-1/8 & 1 \\
0 & 1/2 \\
\end{bmatrix}
\begin{bmatrix}
1 & -3/4 \\
0 & 1 \\
\end{bmatrix}
\]

Resulting in the binDCT version B transform architecture depicted in Figure 8 that computes eight subband signals in 14 shift operations and 31 addition operations.

### 3.4 Matlab Simulation Results

An m-function called the *lifstep* is created in Matlab that simulates the integer to integer mapping capabilities of the dyadic lifting step. The function utilizes the binary bit shift built-in function in Matlab to perform the binary division of \( 2^m \).

Function \([y]=\text{lifstep}(x,k,m)\)

Xproduct=x\( \ast k\); \% compute the numerator product.

Y=bitshift(xproduct,-m); \% the negative sign, indicates right binary shift by \( m \) positions.

The forward and inverse binDCT-B m-functions are located in Appendix A.

The one-dimensional DCT transform is applied to the clown picture found in MATLAB.

The reconstruction of the clown image is accomplished by using the one-dimensional inverse DCT transform. The one-dimensional approximate DCT transform, the binDCT is also applied to the clown picture found in MATLAB. The reconstruction of the clown image is accomplished by using the one-dimensional inverse binDCT transform. The forward binDCT transform and inverse binDCT transform maps the integer almost perfect, such that the human eye does not detect a difference while looking at the final
output matrix coefficients there is a slight difference numerically. The floating point forward and inverse DCT transform, fails to reconstruct the clown image with perfect visual reconstruction. The MATLAB results are shown in Figure 11.

3.5 Summary

The binDCT architecture is efficient for VLSI implementation because it removes the multiplication operations of the rotation angles, leaving a transform that is completely multiply free and easily implemented in binary. For the application of video processing and image processing the binDCT possesses the ability to perform an integer transformation on a digital image to obtain an integer output. The floating-point DCT
possesses the ability to perform a floating-point transformation of a digital image to obtain a floating-point output.
Chapter 4

VLSI Implementation of the One-Dimensional binDCT

4.1 System Level Design

The system level block diagram is first devised in order for the preliminary word-length calculations to be completed, such that input and output word size can be determined. The system-level design also gives the necessary information for the floor planning of the chip, where the power pads are to be placed.

The main assumption behind the choice for the length of input word was that the input data signal would be video or images. Thus, the input word is an 8-bit unsigned integer to represent 256 levels of color. The resulting output data is an 11-bit unsigned integer.

The system level design of the 1-D 8-point DCT is shown below in Figure 12.

Figure 12. System-Level design of the 1-D 8-point binDCT chip.

The overall chip design consists of three main parts. The first part consists of the serial to parallel transmission circuitry, to reduce the number of pins on the final chip design. The second part consists of the 1-D binDCT transform architecture, which can be furthered
sub-divided into four parts, the pre-processing stage, the first lifting step stage, the mid-
processing stage and finally the lifting processing stage. The third part of the overall 1-D
binDCT chip consists of the parallel to serial transmission circuitry. A more detailed
look inside the system-level design is depicted in Figure 13, shown below.

Figure 13. Inside the System-Level design of the 1-D binDCT chip

The fast forward binDCT block implementation in Figure 8 is shown again below.

Figure 8. Block Diagram of the Fast Forward binDCT-B transform
4.2 Addition

The adders use the assumption that the inputs are unsigned integers. Accordingly then, given two inputs, x and y of 8-bits, their summation will yield an 8-bit sum with a 1-bit carryout. Since many adders are used, the 1-bit carryout is attached to the MSB of the sum register, to avoid an incorrect result from occurring [19][20]. Thus, for an 8-bit adder a 9-bit output is generated.

To increase the speed of the adder, a dedicated adder can be designed by creating a technology specific logic library that considers the size of the p-mos and n-mos transistors for the general logic cells. The new logic cells would then require testing to determine the internal capacitance and delay of the new cells using HSPICE. For the portability of the final design across technologies, the adders are designed using a Register Transfer Level (RTL) description that can be synthesized to the gate-level using any technology in Synopsys' Design Analyzer.

The RTL description of the 8-bit adder using Verilog HDL is as follows:

```verilog
module add8 (x,y,reset,sum);
input [7:0] x, y;
input reset;
output [8:0] sum;
reg [8:0] sum;

always @ (x or y or reset ) begin
    if (!reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
endmodule
```
This 8-bit adder can be implemented in any technology using Synopsys’ Design Analyzer and Compiler to generate the gate-level netlist.

4.3 Subtraction

The subtraction operations are implemented using adders. The minuend is binary two’s complemented and added to the subtrahend. The input word size is equal to the output word size, because the output carry becomes insignificant, in unsigned binary subtraction. Overflow does not occur because the output is always positive, like the input [19][20].

4.4 Dedicated Lifting Steps

The lifting steps are implemented using individual dedicated logic units. The individual dedicated lifting steps are written in HDL using the RTL description language. The lifting step $\frac{5}{8}$ is implemented using two shift operations and one addition operation, as depicted below in Figure 14,

![Diagram of lifting step](image)

Figure 14. The 5/8 lifting step is implemented using two shifts and one addition.

Thus, depending on the order of implementing the multiplication and division, the lifting step $\frac{5}{8}$ can be performed on an 8-bit unsigned binary word utilizing two shift operations and one addition operation. To yield an 8-bit output, first divide the input word by eight,
and then multiply by five. However, if the multiplication precedes the division the output word becomes 11 bits long.

The division of the word $x$ by $2^m$ is accomplished in binary arithmetic by shifting the word, $x$ to the right $m$ places (e.g. $x >> m$). In binary arithmetic, multiplication is accomplished using a combination of shift and add operations. The shift operations are accomplished using shift registers. The RTL code for the lifting step $\frac{5}{8}$ is as follows,

```verilog
module liftstep_5_8 (x,reset,lift);
  input [7:0] x;
  input reset;
  output [7:0] lift;
  reg [7:0] lift;
  reg [7:0] y1, y2;
  always @(x or reset) begin
    if (!reset) begin
      lift=0;
      end
    else begin
      y1=x>>1;
      y2=x>>3;
      lift=y1+y2;
    end
  endmodule
```

The above RTL description inputs an 8-bit word and outputs an 8-bit word. An alternate RTL description is given below for the same lifting step.

```verilog
module liftstep_5_8 (x,reset,lift);
  input [7:0] x;
  input reset;
  output [10:0] lift;
  reg [10:0] lift;
  reg [9:0] y1;
  reg [10:0] sum;
  always @(x or reset) begin
    if (!reset) begin
      lift=0;
      end
    else begin
```
```cpp
y1=x<<2;
sum=x+y1;
result=sum>>>3;
end
endmodule
```

The alternate RTL description results in an 11-bit output for an 8-bit input word. The choice of implementation affects the output word size and the amount of silicon used.

### 4.5 Serial to Parallel Input Transmission

To obtain a core-bounded chip, the number of pads on the chip are reduced, resulting in a cost-effective chip design. The input data is serially read onto the chip, and then transmitted to the binDCT functional unit parallel-wise. The serial to parallel input module consists of an 8-bit register, 1:8 multiplexor, and an 8x8-bit memory array, shown in Figure 15.

![Serial-to-parallel Input Transmission](image)

The RTL description of the serial to parallel input module is described as:

```verilog
module s_p_in(data_in,s2,s1,s0, write_enable, parallel_clock, serial_clock, x0, x1, x2, x3, x4, x5, x6, x7)
input [7:0] data_in;
input s2, s1, s0, read_enable, parallel_clock, serial_clock;
output [7:0] x0, x1, x2, x3, x4, x5, x6, x7;

reg [7:0] memory2 [0:7];
```

37
reg [7:0] x0, x1, x2, x3, x4, x5, x6, x7;

always @ (data_in or read_enable or s2 or s1 or s0 or serial_clock) begin
  if (write_enable==1'b1) begin
    if (serial_clock==1'b1) memory2[ (s2, s1, s0) ] = data_in;
  end
end

always @ (posedge parallel_clock) begin
  if (parallel_clock==1'b1) begin
    memory1[3'd0]=x0;
    memory1[3'd1]=x1;
    memory1[3'd2]=x2;
    memory1[3'd3]=x3;
    memory1[3'd4]=x4;
    memory1[3'd5]=x5;
    memory1[3'd6]=x6;
    memory1[3'd7]=x7; end
end
endmodule

4.6 Parallel to Serial Output Transmission

To keep the silicon area to a minimum and the cost of fabricating the chip reasonable, the parallel output from the binDCT functional unit is transmitted serially off of the chip. The parallel to serial output module consists of an 8x11-bit memory array, an 8:1 demultiplexor and an 11-bit register, shown in Figure 16.

Figure 16. Parallel-to-serial output transmission.
The RTL description of the parallel-to-serial output module follows.

```verilog
text
module p_s_out(y0, y1, y2, y3, y4, y5, y6, y7, s2, s1, s0, read_enable, parallel_clock, serial_clock, data_out)
input [7:0] y0, y1, y2, y3, y4, y5, y6, y7;
input s2, s1, s0, read_enable, parallel_clock, serial_clock;
output [7:0] data_out;

reg [7:0] memory1 [0:7];
reg [7:0] data_out;
always @ (y0 or y1 or y2 or y3 or y4 or y5 or y6 or y7 or s2 or s1 or s0 or parallel_clock)
begin
if (parallel_clock == 1'b1) begin
memory1[3'd0]=y0;
memory1[3'd1]=y1;
memory1[3'd2]=y2;
memory1[3'd3]=y3;
memory1[3'd4]=y4;
memory1[3'd5]=y5;
memory1[3'd6]=y6;
memory1[3'd7]=y7; end
end
always @ (read_enable or s2 or s1 or s0 or serial_clock) begin
if (read_enable == 1'b1) begin
if (serial_clock==1'b1) data_out=memory1[{{s2,s1,s0}}];
end
endmodule
```

4.7 Constraints for Gate-Level Synthesis

Once the RTL description of the design is written in Verilog HDL, and verified to work correctly using Verilog-XL. The RTL description is synthesized to gate-level using Synopsys' Design Analyzer and Compiler. An initial estimate of the size of the design is determined using a RTL Verilog HDL description that only contains the binDCT chip architecture without the input and output pads being initialized. The RTL description is synthesized without constraints using Synopsys' Design Compiler, to generate a starting
point for establishing design constraints for the final gate-level synthesis. The area, timing and power values generated along with the critical path's time are reported. (See Appendix H).

The individual bindct8 module is compiled using a multiplexed flip-flop scan style and scan circuitry is inserted, the final fault coverage of the compiled bindct8 module is calculated by Synopsys Design Analyzer to be 100%. But when the overall chip is compiled using the multiplexed flip-flop scan style and a scan test check is done to test the designs' ability to insert a scan. The test fails due to the presence of inferred memory devices and latches in the serial-to-parallel and parallel-to-serial modules. The presence of two inferred memories arrays in the RTL description utilizing a latch, to pass parallel data onto and off the binDCT module of the chip. The serial-to-parallel and parallel-to-serial modules memory devices cannot be substituted with scan based equivalent flip-flops. Therefore, no Design for Testability (DFT) technique is implemented in the synthesis of the final gate-level description of the binDCT chip.

The initial and final constraints used for gate-level synthesis are found in Appendix H. The net slew violation that occurs in the serial-to-parallel module is assumed to be resolved during the placement and routing of the design.

The main clock of the binDCT chip had to be increased significantly to avoid any racing conditions from occurring between the serial-to-parallel module and the binDCT module data transmission during gate-level synthesis. Due to the use of the black-box cell library's internal gate delays, affect the timing of the data arrival for the serial-to-parallel module's timing analysis. To avoid this problem, the binDCT clock was increased significantly to accommodate the input clock for serial data entering the chip and the
parallel data being latched to the binDCT chip using the binDCT clock even as it is triggered. If there were no pin limitations on the bonding pad, parallel loading of the binDCT chip could occur, reducing the clock period constraint for gate-level synthesis from 200ns to 40ns.

4.8 Floor-planning, Routing, and LVS

The forward binDCT chip has an estimated size of 2.154 mm x 1.314 mm, while the inverse binDCT chip has an estimated size of 1.818 mm x 1.818 mm from initial calculations based on the 0.35μ CMOS technology specifications for estimated pad sizes. For the power constraints to be met, four pairs of core pads and two pairs of ring pads are used for both the forward and inverse one-dimensional binDCT chip designs. To avoid any open spaces around the pad ring when floor-planning, an extra VSS-ring pad is included.
Figure 17a. The forward binDCT chip after floor planning is complete. The power pads have been added. The I/O pads and standard cells have been placed. The power stripes and rings have also been placed. The binDCT clock tree has been generated at this point and the design re-routed.
The binDCT clock is the only clock generated using the clock tree generator in the Design Planner. The binDCT clock tree is generated to synchronize the clocks and minimize the skew effect that is propagated throughout the chip. The final clock tree results are located in Appendix F. The input and output clocks are dependent on the rate the user inputs the data onto the chip and reads the data off of the chip. Once the power is routed on the design chip, the chip undergoes its final routing in Silicon Ensemble. A timing analysis is performed using Pearl, the full results are located
in Appendix D. The Pearl analysis of the forward binDCT chip determined that the
insertion delay between I/O pad to leaf-cell was in the range between 1.136\(ns\) to 1.163\(ns\)
for a maximum clock skew of 0.027\(ns\). The inverse binDCT chip insertion delay range
from 1.231\(ns\) to 1.308\(ns\) for a maximum clock skew of 0.077\(ns\). The worst timing paths
were determined to be located in the serial-to-parallel module between the memory
register array and the latch for both the forward and inverse chip. There is only one net
in the forward binDCT chip that violates the black box cell library’s maximum net
transition rate of 1.5\(ns\). The net is calculated to have a maximum slew rate of 1.59\(ns\),
which can be assumed to be negligible.
The Layout versus Schematic (LVS) check in the DFII indicates a netlist match for both the forward and inverse binDCT chips. The LVS results can be found in Appendix E.

The Design Rule Check (DRC) in Cadence's Design Framework resulted in zero errors or violations for both the forward and inverse binDCT chips. The CMC DRC check results stated that poly1 was under used in both designs and needed to satisfy a coverage of 14% of the chip's core. The inverse binDCT chip had two antenna rule violations revealed by the CMC DRC check. All of the violations were easily corrected.
Figure 20. The final forward binDCT chip in DFI prior to filling.
4.9 Summary

The forward and inverse binDCT chips RTL descriptions are implemented using the 0.35\(\mu\) CMOS standard digital cell libraries. Resulting in area-efficient chips that consume little power. According to Appendix H, where the gate-level power analysis calculations were completed, the final gate-level designs consume power in the low to medium mW (e.g. milliwatts) range. The final forward binDCT chip has a layout dimension of 1794\(\times\)3434\(\mu\)m. The final inverse binDCT chip has a layout dimension of 2778\(\times\)2778\(\mu\)m.
Chapter 5

Conclusions and Future Work

5.1 Conclusions

The results from the MATLAB simulations indicate that the binDCT transform is better than the floating-point DCT transform for the compression and reconstruction of images. Thus, the binDCT transform can replace the floating-point DCT transform in the JPEG and MPEG standards for video processing and image processing applications.

The binary-friendly architecture of the binDCT transform enables its implementation in VLSI. The final VLSI design is cost-effective because the chip size is relatively small due to its multiply free architecture. The power consumption of the chip is estimated to be relatively low at the gate-level. The 0.35μ CMOS binDCT chip speed is slow as compared to the speed of floating-point DCT chips. The speed can be increased by implementing a full custom VLSI version of the binDCT architecture.

5.2 Future Work

Future work can be done in the implementation of the two-dimensional binDCT transform in VLSI. As stated in the introduction of the thesis, the two-dimensional DCT transform is separable. Since the 2-D DCT can be implemented using two 1-D DCT transforms, one to perform the row-wise transform, and the second to perform the column-wise transform. The 2-D binDCT transform can also be implemented utilizing row-column decomposition. Figure 22 depicts the row-column decomposition of the 2-D binDCT.
Figure 22. Block diagram of the row-column approach for 2-D binDCT.

The timing control circuitry is designed using the folding transformation [1].

5.2.1 The use of the folding transformation in 2-D orthogonal transform algorithms

For VLSI implementation, a 2-D transform must keep the chip area to a minimum by reducing the number of multipliers, adders, registers, multiplexes and interconnecting wires. The folding transformation provides a systematic technique for designing the timing control and other control circuits for hardware, where several algorithmic operations (e.g. addition operations) are time-multiplexed to a single functional unit (e.g. a pipeline adder). A reduction in chip area occurs during VLSI implementation when multiple algorithm operations are executed on a single functional unit, thus reducing the number of functional units used. The one side effect of the folding technique is that the new architecture uses a large number of registers that consume excessively much chip area. Register minimization techniques can be used to compute a minimum number of registers required to implement the folded 2-D transform architecture. [1]
Appendix A

MATLAB Code

A.1 Lifting Steps

The lifting step is generated by first multiplying the numerator, \( m \) and then dividing by the denominator, \( 2^k \). For perfect reconstruction of integers a floor operator is placed on the lifting step. The \( m \)-function \( \text{liftstep} \) can calculate the general dyadic lifting step of a signal, given the numerator, \( m \) and the denominator, \( k \) and binary bit-wise shifting.

There are several dedicated lifting functions also defined utilizing binary bit-wise shifting and addition.

\[
\begin{align*}
\% & \text{ This function generates the result of a lifting step,} \\
& \text{ using bit-wise shift and add operators.} \\
\% & \text{ To perform a dyadic lifting step of } m/2^k \text{ on } x, \\
& \text{ where } m \text{ is basically a simple multiplication that} \\
& \text{ can be accomplished by bit-wise shifting} \\
& \text{ to the right and adding.} \\
\% & \text{ The division by } 2^k \text{ is accomplished by a simple left} \\
& \text{ bit-wise shift by } k \text{ positions.} \\
\% & y = \text{floor}(x*m/2^k). \\
\text{function } y=\text{liftstep}(x,m,k); \\
x\text{product}=x*m; \\
y=\text{bitshift}(x,-k);
\end{align*}
\]

% Lifting step 1/2
function \( y=\text{liftstep1_2}(x) \) 
\( y=\text{bitshift}(x,-1); \)

% Lifting step 1/8
function \( y=\text{liftstep1_8}(x) \) 
\( y=\text{bitshift}(x,-3); \)

% Lifting step 3/4
function \( y=\text{liftstep3_4}(x) \) 
\( y=\text{bitshift}(x,-2) + \text{bitshift}(x,-1); \)

% Lifting step 3/8
function \( y=\text{liftstep3_8}(x) \) 
\( y=\text{bitshift}(x,-3) + \text{bitshift}(x,-2); \)

% Lifting step 5/8
function \( y=\text{liftstep5_8}(x) \) 
\( y=\text{bitshift}(x,-3) + \text{bitshift}(x,-1); \)
A.2 The forward and inverse binDCT m-functions using dedicated lifting steps

The dedicated lifting steps are included in the modified m-functions for the fast forward and inverse binDCT version B architectures.

```matlab
% Created by: Tracy Franklin
% Fast Implementation of 8-point binDCT version B, Integer DCT % using dyadic lifting steps. Modified version.
% based Trac D. Tran algorithm % See also plus, minus, liftstep.
function [y]=bindct(zx)
[m,n]=size(zx);
y=zeros(size(zx));
x=double(zx);
for i=1:m
    for j=1:8:n
        % Perform data stream assignment into subband signals
        x0=x(i,j);
        x1=x(i,j+1);
        x2=x(i,j+2);
        x3=x(i,j+3);
        x4=x(i,j+4);
        x5=x(i,j+5);
        x6=x(i,j+6);
        x7=x(i,j+7);

        % Pre-addition stage
        a0=plus(x0,x7);
        a1=plus(x1,x6);
        a2=plus(x2,x5);
        a3=plus(x3,x4);
        a4=minus(x4,x3);
        a5=minus(x5,x2);
        a6=minus(x6,x1);
        a7=minus(x0,x7);

        % First rotation stage is implemented using dyadic lifting steps
        % Rotation of pi/4 follows
        b6=plus(liftstep3_8(a5),a6);
        b5=minus(liftstep5_8(b6),a5);

        % Secondary addition stage
        c0=plus(a0,a3);
        c1=plus(a1,a2);
        c2=minus(a1,a2);
        c3=minus(a0,a3);
        c4=plus(a4,b5);
        c5=minus(a4,b5);
        c6=minus(a7,b6);
        c7=plus(a7,b6);

        % Second rotation stage is implemented using dyadic lifting steps
        d0=plus(c0,c1);
```

52
\[ d1 = \text{minus}(\text{liftstep1}_2(d0)), c1); \]

\% Third rotation stage is implemented using dyadic lifting steps
\[ d2 = \text{minus}(c2, \text{liftstep3}_8(c3)); \]
\[ d3 = \text{plus}(\text{liftstep3}_8(d2)), c3); \]

\% Fourth rotation stage is implemented using dyadic lifting steps
\[ d4 = \text{minus}(c4, \text{liftstep1}_8(c7)); \]
\[ d7 = c7; \]

\% Fifth rotation stage is implemented using dyadic lifting steps
\[ d5 = \text{plus}(c5, \text{liftstep1}_2(\text{minus}(c6, \text{liftstep3}_4(c5))))); \]
\[ d6 = \text{minus}(\text{minus}(c6, \text{liftstep3}_4(c5)), \text{liftstep1}_8(d5))); \]
\[ y(i,j) = d0; \]
\[ y(i,j+1) = d7; \]
\[ y(i,j+2) = d3; \]
\[ y(i,j+3) = d6; \]
\[ y(i,j+4) = d1; \]
\[ y(i,j+5) = d5; \]
\[ y(i,j+6) = d2; \]
\[ y(i,j+7) = d4; \]
end
end

\% Created by: Tracy Franklin
\% Fast Implementation of the Inverse binDCT version B
\% Modified version.
\% Inverse Integer DCT by on Trac D. Tran algorithm

\text{function } [z] = \text{imbindct}(x) \text{ end}

[n,m] = \text{size}(x); \text{ end}
\[ z = \text{zeros}(\text{size}(x)); \]
\[ y = \text{double}(x); \]
for \[ i = 1:n \]
for \[ j = 1:8:m \]
\[ a0 = y(i,j); \]
\[ a1 = y(i,j+1); \]
\[ a2 = y(i,j+2); \]
\[ a3 = y(i,j+3); \]
\[ a4 = y(i,j+4); \]
\[ a5 = y(i,j+5); \]
\[ a6 = y(i,j+6); \]
\[ a7 = y(i,j+7); \]
% First rotation stage is implemented using dyadic lifting steps
\[ b4 = \text{liftstep1}_2(a0) - a4; \]
\[ b0 = a0 - b4; \]
% Second rotation stage is implemented using dyadic lifting steps
\[ b2 = a2 - \text{liftstep3}_8(a6); \]
\[ b6 = a6 + \text{liftstep3}_8(b2); \]
% Third rotation stage is implemented using dyadic lifting steps
b1 = a1;
b7 = a7 + liftstep1_8(a1);

% Fourth rotation stage is implemented using dyadic lifting steps
b5 = a5 - (liftstep1_2(a3 + (liftstep1_8(a5))));
b3 = a3 + liftstep1_8(a5) + liftstep3_4(b5);

% First addition/subtraction stage
c0 = b0 + b2;
c4 = b4 + b6;
c6 = b6 - b6;
c2 = b0 - b2;
c7 = b7 + b5;
c5 = b7 - b5;
c3 = b1 - b3;
c1 = b1 + b3;

% Fifth rotation stage is implemented using dyadic lifting steps
d5 = liftstep5_8(c3) - c5;
d3 = c3 - liftstep3_8(d5);

% Second addition/subtraction stage
e0 = c0 + c1;
e4 = c4 + d3;
e6 = c6 + d5;
e2 = c2 + c7;
e7 = c2 - c7;
e5 = c6 - d5;
e3 = c4 - d3;
e1 = c0 - c1;

% Reposition output in the inverse transform matrix
z(i,j) = bitshift(e0, -2);
z(i,j+1) = bitshift(e4, -2);
z(i,j+2) = bitshift(e6, -2);
z(i,j+3) = bitshift(e2, -2);
z(i,j+4) = bitshift(e7, -2);
z(i,j+5) = bitshift(e5, -2);
z(i,j+6) = bitshift(e3, -2);
z(i,j+7) = bitshift(e1, -2);
end
end

A.3 The floating point DCT

The original DCT architecture that is approximated by the binDCT architecture is given below.

% Original DCT algorithm that is alter for the bindct
function [y] = fdct(xz);
[m,n] = size(xz);
y = zeros(m,n);
x = double(xz);
for i = 1:m
for j=1:8:n
    x0=x(i,j);
    x1=x(i,j+1);
    x2=x(i,j+2);
    x3=x(i,j+3);
    x4=x(i,j+4);
    x5=x(i,j+5);
    x6=x(i,j+6);
    x7=x(i,j+7);

    a0=x0+x7;
    a1=x1+x6;
    a2=x2+x5;
    a3=x3+x4;
    a4=x3-x4;
    a5=x2-x5;
    a6=x1-x6;
    a7=x0-x7;

    b5=(x6*sin(pi/4))-(x5*cos(pi/4));
    b6=(x5*sin(pi/4))+(x6*cos(pi/4));

    c0=a0+a3;
    c1=a1+a2;
    c2=a1-a2;
    c3=a0-a3;
    c4=a4+b5;
    c5=a4-b5;
    c6=a7-b6;
    c7=a7+b6;

    d0=(c0*cos(pi/4))+(c1*sin(pi/4));
    d1=(c0*sin(pi/4))-(c1*cos(pi/4));
    d2=(c2*cos(3*pi/8))+(c3*sin(3*pi/8));
    d3=(c3*cos(3*pi/8))-(c2*sin(3*pi/8));
    d4=(c4*cos(7*pi/16))+(c7*sin(7*pi/16));
    d5=(c5*cos(3*pi/16))+(c6*sin(3*pi/16));
    d6=(c6*cos(3*pi/16))-(c5*sin(3*pi/16));
    d7=(c7*cos(7*pi/16))-(c4*sin(7*pi/16));

    y(i,j)=d0;
    y(i,j+4)=d1;
    y(i,j+2)=d2;
    y(i,j+6)=d3;
    y(i,j+1)=d4;
    y(i,j+5)=d5;
    y(i,j+3)=d6;
    y(i,j+7)=d7;
end
end

55
A.4 The forward and inverse binDCT m-functions that use the general lifting step

The fast forward and inverse binDCT m-functions, that utilize the general lifting step m-
function.

% Created by: Tracy Franklin
% Fast Implementation of the Inverse binDCT version B
% Un-normalized version.
% Inverse Integer DCT by on Trac D. Tran algorithm

function [z]=ibindct(x)
[n,m]=size(x);
z=zeros(size(x));
y=double(x);
for i=1:n
    for j=1:8:m
        a0=y(i,j);
        a1=y(i,j+1);
        a2=y(i,j+2);
        a3=y(i,j+3);
        a4=y(i,j+4);
        a5=y(i,j+5);
        a6=y(i,j+6);
        a7=y(i,j+7);
        % First rotation stage is implemented using dyadic lifting steps
        b4=liftstep(a0,1,1)- a4;
        b0=a0 - b4;
        % Second rotation stage is implemented using dyadic lifting steps
        b2=a2 - liftstep(a6,3,3);
        b6=a6 + liftstep(b2,3,3);
        % Third rotation stage is implemented using dyadic lifting steps
        b1=a1;
        b7=a7 + liftstep(a1,1,3);
        % Fourth rotation stage is implemented using dyadic lifting steps
        b5 = a5 - (liftstep((a3+(liftstep(a5,1,3))),1,1));
        b3 = a3 + liftstep(a5,1,3) + liftstep(b5,3,3);
        % First addition/subtraction stage
        c0=b0+b2;
        c4=b4+b6;
        c6=b4-b6;
        c2=b0-b2;
        c7=b7+b5;
        c5=b7-b5;
        c3=b1-b3;
        c1=b1+b3;
        % Fifth rotation stage is implemented using dyadic lifting steps
        d5=liftstep(c3,5,3) - c5;
d3=c3 - liftstep(d5,3,3);

% Second addition/subtraction stage
e0 = c0 + c1;
e4 = c4 + d3;
e6 = c6 + d5;
e2 = c2 + c7;
e7 = c2 - c7;
e5 = c6 - d5;
e3 = c4 - d3;
e1 = c0 - c1;

% Reposition output in the inverse transform matrix
z(i,j)=bitshift(e0,-2);
z(i,j+1)=bitshift(e4,-2);
z(i,j+2)=bitshift(e6,-2);
z(i,j+3)=bitshift(e2,-2);
z(i,j+4)=bitshift(e7,-2);
z(i,j+5)=bitshift(e5,-2);
z(i,j+6)=bitshift(e3,-2);
z(i,j+7)=bitshift(e1,-2);
end
end

% Created by: Tracy Franklin
% Fast Implementation of 8-point binDCT version B, Integer DCT
% using dyadic lifting steps. Un-normalized version.
% based Trac D. Tran algorithm
% See also plus, minus, liftstep.

function [y]=bindct(zx)
[m,n]=size(zx);
y=zeros(size(zx));
x=double(zx);
for i=1:m
    for j=1:8:n
        % Perform data stream assignment into subband signals
        x0=x(i,j);
        x1=x(i,j+1);
        x2=x(i,j+2);
        x3=x(i,j+3);
        x4=x(i,j+4);
        x5=x(i,j+5);
        x6=x(i,j+6);
        x7=x(i,j+7);
        % Pre-addition stage
        a0=plus(x0,x7);
        a1=plus(x1,x6);
        a2=plus(x2,x5);
        a3=plus(x3,x4);
        a4=minus(x3,x4);
        a5=minus(x2,x5);
        a6=minus(x1,x6);
        a7=minus(x0,x7);
    end
end
% First rotation stage is implemented using dyadic lifting steps
% rotation of π/4 follows
b6=plus(lifstep3_8(a5),a6);
b5=minus(lifstep5_8(b6),a5);

% Secondary addition stage
c0=plus(a0,a3);
c1=plus(a1,a2);
c2=minus(a1,a2);
c3=minus(a0,a3);
c4=plus(a4,b5);
c5=minus(a4,b5);
c6=minus(a7,b6);
c7=plus(a7,b6);

% Second rotation stage is implemented using dyadic lifting steps
d0=plus(c0,c1);
d1=minus((lifstep1_2(d0)),c1);

% Third rotation stage is implemented using dyadic lifting steps
d2=minus(c2,(lifstep3_8(c3))); 
d3=plus((lifstep3_8(d2)),c3);

% Fourth rotation stage is implemented using dyadic lifting steps
d4=minus(c4,(lifstep1_8(c7))); 
d7=c7;

% Fifth rotation stage is implemented using dyadic lifting steps
d5=plus(c5,(lifstep1_3(minus(c6,(lifstep3_4(c5))))));
d6=minus(minus(c6,(lifstep3_4(c5))),(lifstep1_8(d5)));

\[ y(i,j)=d0; \]
\[ y(i,j+1)=d7; \]
\[ y(i,j+2)=d3; \]
\[ y(i,j+3)=d6; \]
\[ y(i,j+4)=d1; \]
\[ y(i,j+5)=d5; \]
\[ y(i,j+6)=d2; \]
\[ y(i,j+7)=d4; \]
end
end

A.5 The forward and inverse binDCT m-functions derived from Real number arithmetic.

The fast forward and inverse binDCT m-functions utilize the floating-point arithmetic to implement the lifting steps, then round the final value down toward zero.

% Created by: Tracy Franklin
% Fast Implementation of binDCT version B, Integer DCT
% based Trac D. Tran algorithm
function [y]=bindctb(w)
[m,n]=size(w);
y=zeros(size(w));
x=double(w);
for i=1:m
  for j=1:8:n
    % Perform data stream assignment into subband signals
    x0=x(i,j);
    x1=x(i,j+1);
    x2=x(i,j+2);
    x3=x(i,j+3);
    x4=x(i,j+4);
    x5=x(i,j+5);
    x6=x(i,j+6);
    x7=x(i,j+7);
    % Pre-addition stage
    a0=x0+x7;
    a1=x1+x6;
    a2=x2+x5;
    a3=x3+x4;
    a4=x3-x4;
    a5=x2-x5;
    a6=x1-x6;
    a7=x0-x7;
    % First rotation stage is implemented using dyadic lifting steps
    % Rotation of pi/4 follows
    b6=(round((a5*3)/8)) + a6;
    b5=(round((b6*5)/8)) - a5;
    % Secondary addition stage
    c0=a0+a3;
    c1=a1+a2;
    c2=a1-a2;
    c3=a0-a3;
    c4=a4+b5;
    c5=a4-b5;
    c6=a7-b6;
    c7=a7+b6;
    % Second rotation stage is implemented using dyadic lifting steps
    d0=c0+c1;
    d1=(round(d0/2)) - c1;
    % Third rotation stage is implemented using dyadic lifting steps
    d2=c2 - (round((c3*3)/8));
    d3=(round((d2*3)/8)) + c3;
    % Fourth rotation stage is implemented using dyadic lifting steps
    d4=c4 - (round(c7/8));
    d7=c7;
    % Fifth rotation stage is implemented using dyadic lifting steps
temp = c6 - (round((c5*3)/4));
d5 = c5 + (round(temp/2));
d6 = temp - (round(d5/8));

y(i,j) = d0;
y(i,j+1) = d7;
y(i,j+2) = d3;
y(i,j+3) = d6;
y(i,j+4) = d1;
y(i,j+5) = d5;
y(i,j+6) = d2;
y(i,j+7) = d4;
end
end

% Created by: Tracy Franklin
% Fast Implementation of the Inverse binDCT version B
% Inverse Integer DCT by on Trac D. Tran algorithm

function [z]=ibindctb(y)
[n,m]=size(y);
z=zeros(size(y));

for i=1:n
    for j=1:8:m
        a0=y(i,j);
        a1=y(i,j+1);
        a2=y(i,j+2);
        a3=y(i,j+3);
        a4=y(i,j+4);
        a5=y(i,j+5);
        a6=y(i,j+6);
        a7=y(i,j+7);

        % First rotation stage is implemented using dyadic lifting steps
        b4=(round(a0/2)) - a4;
        b0= a0 - b4;

        % Second rotation stage is implemented using dyadic lifting steps
        b2=a2 - (round((a6*3)/8));
        b6=a6 + (round((b2*3)/8));

        % Third rotation stage is implemented using dyadic lifting steps
        b1=a1;
        b7=a7 + (round(a1/8));

        % Fourth rotation stage is implemented using dyadic lifting steps
        b5 = a5 - (round((a3 + (round(a5/8)))/2));
        b3 = a3 + (round(a5/8)) + (round((b5*3)/8));

        % First addition/subtraction stage
        c0=b0+b2;
        c4=b4+b6;
        c6=b4-b6;
        c2=b0-b2;
c7 = b7 + b5;
c5 = b7 - b5;
c3 = b1 - b3;
c1 = b1 + b3;

% Fifth rotation stage is implemented using dyadic lifting steps
d5 = round((c3 * 5) / 8) - c5;
d3 = c3 - round((d5 * 3) / 8);

% Second addition/subtraction stage
e0 = c0 + c1;
e4 = c4 + d3;
e6 = c6 + d5;
e2 = c2 + c7;
e7 = c2 - c7;
e5 = c6 - d5;
e3 = c4 - d3;
e1 = c0 - c1;

% Reposition output in the inverse transform matrix
z(i,j) = round(e0 / 4);
z(i,j+1) = round(e4 / 4);
z(i,j+2) = round(e6 / 4);
z(i,j+3) = round(e2 / 4);
z(i,j+4) = round(e7 / 4);
z(i,j+5) = round(e5 / 4);
z(i,j+6) = round(e3 / 4);
z(i,j+7) = round(e1 / 4);
end
end
Appendix B

RTL design description in Verilog HDL

B.1 One-Dimensional 8-point binDCT

// One-dimensional Forward Integer binDCT
// Uses dyadic lifting steps to accomplish the rotation function
// In the fast forward implementation of the Discrete
// Cosine transforms
// Created by: Tracy Franklin

timescale 1ns/10ps

module forwarddct (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outre, out_clock, y);

input [7:0] x;
input [2:0] in_select, out_select;
input inwe, outre;
input in_clock, out_clock;
input bindct_clock, reset;
// output [10:0] z0, z1, z2, z3, z4, z5, z6, z7; // for submodule testing
output [10:0] y;

wire [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
wire [10:0] z0, z1, z2, z3, z4, z5, z6, z7;

s_p_in8 s_p_in8 (x, in_select, inwe, in_clock, bindct_clock, x0, x1, x2, x3, x4, x5, x6, x7);
bindct8 bindct8 (x0, x1, x2, x3, x4, x5, x6, x7, bindct_clock, reset, z0, z1, z2, z3, z4, z5, z6, z7);
p_s_out11 p_s_out11 (z0, z1, z2, z3, z4, z5, z6, z7, out_select, outre, bindct_clock, out_clock, y);

endmodule

// The output circuitry to resolve the eight parallel outputs
// into one serial output.
module p_s_out11 (z0, z1, z2, z3, z4, z5, z6, z7, select_out, read_enable, parallel_clock, serial_clock, zout);

input [10:0] z0, z1, z2, z3, z4, z5, z6, z7;
input [2:0] select_out;
input serial_clock, parallel_clock, read_enable;
output [10:0] zout;

reg [10:0] zout;
reg [10:0] outputmem [0:7];

always @(parallel_clock or z0 or z1 or z2 or z3 or z4 or z5 or z6 or z7) begin
if (parallel_clock == 1'b1) begin
outputmem[3'd0]=z0;
outputmem[3'd1]=z1;
outputmem[3'd2]=z2;

62
outputmem[3'd3]=z3;
outputmem[3'd4]=z4;
outputmem[3'd5]=z5;
outputmem[3'd6]=z6;
outputmem[3'd7]=z7;
end
end

always @ (read_enable or select_out or serial_clock) begin
    if (read_enable == 1'b1) begin
        if (serial_clock == 1'b1)
            zout== outputmem[select_out];
        end
    end
endmodule

// Note inpclk has a period eight times the size of the insclk.
// Input circuitry to the 1-D DCT chip to reduce the number of pads
module s_p_in8 (x, select, write_enable, serial_clock, parallel_clock, x0, x1, x2, x3, x4, x5, x6, x7);

input [7:0] x;
input [2:0] select;
input write_enable;
input serial_clock, parallel_clock;
output [7:0] x0, x1, x2, x3, x4, x5, x6, x7;

reg [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [7:0] inputmem [0:7];

always @ (serial_clock or select or x or write_enable) begin
    if (write_enable == 1'b1) begin
        if (serial_clock == 1'b1)
            inputmem[select]= x;
        end
    end
end

always @ (posedge parallel_clock) begin
    x0=inputmem[3'd0];
    x1=inputmem[3'd1];
    x2=inputmem[3'd2];
    x3=inputmem[3'd3];
    x4=inputmem[3'd4];
    x5=inputmem[3'd5];
    x6=inputmem[3'd6];
    x7=inputmem[3'd7];
end
endmodule

// HDL description of the 8x8 1-D binDCT-B
// normalized version
// Created by: Tracy Franklin

module bindct8 (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, z0, z1, z2, z3, z4, z5, z6, z7);
input [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input clk, reset;

63
output [10:0] z0,z1,z2,z3,z4,z5,z6,z7;

wire [8:0] a0,a1,a2,a3;
wire [7:0] a4,a5,a6,a7;
wire [8:0] b5,b6;
wire [9:0] c0,c1,c4,c7;
wire [8:0] c2,c3,c5,c6;
wire [10:0] d0;
wire [9:0] d4,d1,d7,d5,d2;
wire [8:0] d3,d6;
wire [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [10:0] z0,z1,z2,z3,z4,z5,z6,z7;

preadd f1 (x0, x1, x2, x3, x4, x5, x6, x7, reset, a0, a1, a2, a3, a4, a5, a6, a7);
prelift f2 (a5, a6, reset, b5, b6);
midadd f3 (a0, a1, a2, a3, a4, b5, b6, a7, reset, c0, c1, c2, c3, c4, c5, c6, c7);
postlift f4 (c0, c1, c2, c3, c4, c5, c6, c7, reset, d0, d1, d2, d3, d4, d5, d6, d7);
rearrange f5 (d0, d1, d2, d3, d4, d5, d6, d7, reset, e0, e1, e2, e3, e4, e5, e6, e7);

always @(posedge clk or negedge reset) begin
    if (reset) begin
        z0=0;
        z1=0;
        z2=0;
        z3=0;
        z4=0;
        z5=0;
        z6=0;
        z7=0;
    end
    else begin
        z0=e0;
        z1=e1;
        z2=e2;
        z3=e3;
        z4=e4;
        z5=e5;
        z6=e6;
        z7=e7;
    end
end
endmodule

module preadd (x0, x1, x2, x3, x4, x5, x6, x7, reset, a0, a1, a2, a3, a4, a5, a6, a7);
input [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input reset;
output [8:0] a0,a1,a2,a3;
output [7:0] a4,a5,a6,a7;
add8 ra1 (x0,x7,reset,a0);
add8 ra2 (x1,x6,reset,a1);
add8 ra3 (x2,x5,reset,a2);
add8 ra4 (x3,x4,reset,a3);
sub8 ra5 (x3,x4,reset,a4);
sub8 ra6 (x2,x5,reset,a5);
sub8 ra7 (x1,x6,reset,a6);
sub8 ra8 (x0,x7,reset,a7);
endmodule

module prelift (a5, a6, reset, b5, b6);
input [7:0] a5,a6;
input reset;
output [8:0] b5,b6;
wire [6:0] temp1;
wire [8:0] temp2;
wire [8:0] b6;
ls388 r11 (a5,reset,temp1);
add8 r12 ((1'b0,temp1[6:0]), a6,reset,b6);
ls589 r13 (b6,reset,temp2);
sub9 r14 (temp2, {1'b0,a5[8:0]}, reset,b5);
endmodule

module midadd (a0, a1, a2, a3, a4, b5, b6, a7, reset, c0, c1, c2, c3, c4, c5, c6, c7);
input [8:0] a0,a1,a2,a3,b5,b6:
input [7:0] a4,a7;
input reset;
output [9:0] c0,c1,c4,c7;
output [8:0] c2,c3,c5,c6;
add9 ma1 (a0,a3,reset,c0);
add9 ma2 (a1,a2,reset,c1);
sub9 ma3 (a1,a2,reset,c2);
sub9 ma4 (a0,a3,reset,c3);
add9 ma5 ((1'b0,a4[7:0]),b5,reset,c4);
sub9 ma6 ((1'b0,a4[7:0]),b5,reset,c5);
sub9 ma7 ((1'b0,a7[7:0]),b6,reset,c6);
add9 ma8 ((1'b0,a7[7:0]),b6,reset,c7);
endmodule

module postlift (c0, c1, c2, c3, c4, c5, c6, c7,reset, d0, d1, d2, d3, d4, d5, d6, d7);
input [9:0] c0,c1,c4,c7;
input [8:0] c2,c3,c5,c6;
input reset;
output [10:0] d0;
output [9:0] d4,d1,d7,d5,d2;
output [8:0] d3,d6;
wire [9:0] temp3;
wire [8:0] temp5,temp6;
wire [7:0] temp7,temp9,temp10;
wire [6:0] temp4,temp8;
add10 o1 (c0,c1,reset,d0);
ls1211 o2 (d0,reset,temp3);
sub10 o3 (temp3,c1,reset,d4);
ls1810 o4 (c7,reset,temp4);
sub10 o5 (c4, {3'b000,temp4[6:0]},reset,d7);
ls349 o6 (c5,reset,temp5);
sub9 o7 (c6,temp5,reset,temp6);
ls129 o8 (temp6.reset,temp7);
add9 o9 (c5,(1'b0,temp7[7:0]),reset,d5);
ls1810 o10 (d5.reset,temp8);
sub9 o11 (temp6,(2'b00,temp8[6:0]),reset,d3);
ls389 o12 (c3.reset,temp9);
sub9 o13 (c2,(1'b0,temp9[7:0]),reset,d6);
ls389 o14 (d6.reset,temp10);
add9 o15 (c3,(1'b0,temp10[7:0]),reset,d2);
assign d1 = c7;
endmodule

module rearrange (d0, d1, d2, d3, d4, d5, d6, d7, reset, e0, e1, e2, e3, e4, e5, e6, e7);
input [10:0] d0;
input [9:0] d1,d2,d4,d5,d7;
input [8:0] d3,d6;
input reset;
output [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
always @ (d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
  if (!reset) begin
    e0=0;
    e1=0;
    e2=0;
    e3=0;
    e4=0;
    e5=0;
    e6=0;
    e7=0;
  end
  else begin
    e0=d0;
    e1=(1'b0,d1[9:0]);
    e2=(1'b0,d2[9:0]);
    e3=(2'b00,d3[8:0]);
    e4=(1'b0,d4[9:0]);
    e5=(1'b0,d5[9:0]);
    e6=(2'b00,d6[8:0]);
    e7=(1'b0,d7[9:0]);
  end
end
endmodule

module add8 (x,y,reset,sum);
input [7:0] x,y;
input reset;
output [8:0] sum;
reg [8:0] sum;
always @ (x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    ...
  end
endmodule
sum=x+y;
end
end
endmodule

module sub8 (x,y,reset,diff);
input [7:0] x,y;
input reset;
output [7:0] diff;
reg [7:0] diff;
always @(x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
endmodule

module ls388 (x,reset,lift);
input [7:0] x;
input reset;
output [6:0] lift;
wire [8:0] shiftx;
wire [9:0] sumx;
reg [6:0] lift;
assign shiftx = ( {1'b0,x} )<<1;
add9 u1 (shiftx, {1'b0,x},reset,sumx);
always @(sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
endmodule

module ls589 (x,reset,lift);
input [8:0] x;
input reset;
output [8:0] lift;
wire [10:0] shiftx;
wire [11:0] sumx;
reg [8:0] lift;
assign shiftx = ( {2'b00,x} )<<2;
add11 u2 (shiftx,(2b00,x),reset,sumx);

always @(sumx or reset) begin
    if (reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
endmodule

module add9 (x,y,reset,sum);
    input [8:0] x,y;
    input reset;
    output [9:0] sum;

    reg [9:0] sum;

    always @(x or y or reset) begin
        if (!reset) begin
            sum=0;
        end
        else begin
            sum=x+y;
        end
    end
endmodule

module sub9 (x,y,reset,diff);
    input [8:0] x,y;
    input reset;
    output [8:0] diff;

    reg [8:0] diff;

    always @(x or y or reset) begin
        if (!reset) begin
            diff=0;
        end
        else begin
            diff=x-y;
        end
    end
endmodule

module add10 (x,y,reset,sum);
    input [9:0] x,y;
    input reset;
    output [10:0] sum;

    reg [10:0] sum;

    always @(x or y or reset) begin
        if (!reset) begin

sum=0;
end
else begin
    sum=x+y;
end
end
endmodule

module sub10 (x,y,reset,diff);
input [9:0] x,y;
input reset;
output [9:0] diff;
reg [9:0] diff;
always @ (x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
endmodule

module add11 (x,y,reset,sum);
input [10:0] x,y;
input reset;
output [11:0] sum;
reg [11:0] sum;
always @ (x or y or reset) begin
    if (!reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
end
endmodule

module ls1211 (x,reset,lift);
input [10:0] x;
input reset;
output [9:0] lift;
reg [9:0] lift;
always @ (x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift = (x >> 1);
    end
end
end
endmodule

module ls129 (x,reset,lift);
input [8:0] x;
input reset;
output [7:0] lift;
reg [7:0] lift;
always @(x or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift = (x >> 1);
  end
end
endmodule

module ls1810 (x,reset,lift);
input [9:0] x;
input reset;
output [6:0] lift;
reg [6:0] lift;
always @(x or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(x>>3);
  end
end
endmodule

module ls349 (x,reset,lift);
input [8:0] x;
input reset;
output [8:0] lift;
wire [9:0] shiftx;
wire [10:0] sumx;
reg [8:0] lift;
assign shiftx={1'b0,x}<<1;
add10 u3 (shiftx,{1'b0,x},reset,sumx);
always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx << 2);
end
module ls389 (x, reset, lift);
input [8:0] x;
input reset;
output [7:0] lift;

wire [9:0] shiftx;
wire [10:0] sumx;
reg [7:0] lift;

assign shiftx = ((1'b0,x)<<1);
add10 u4 (shiftx, (1'b0,x), reset, sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
    end
  else begin
    lift = (sumx >> 3);
  end
end
endmodule

B.2 One-Dimensional 8-point binDCT RTL description including I/O pads for 0.35μ CMOS technology

// HDL description of the 8-point forward binDCT-B chip
// Created by: Tracy A. Franklin

timescale 1ns/10 ps
module forwardbindctchip (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outr, out_clock, y);
input [7:0] x;
input [2:0] in_select;
ininput inwe;
ininput in_clock;
ininput bindct_clock;
ininput reset;
ininput [2:0] out_select;
ininput outr;
ininput out_clock;
output [10:0] y;

wire [7:0] x_top;
wire [2:0] in_select_top;
wire inwe_top;
wire in_clock_top;
wire bindct_clock_top;
wire reset_top;
wire [2:0] out_select_top;
wire outre_top;
wire out_clock_top;
wire [10:0] y_top;

forwarddct forwarddct (x_top, in_select_top, inwe_top, in_clock_top, bindct_clock_top, reset_top, out_select_top, outre_top, out_clock_top, y_top);

PDI px0 (.C(x_top[0]), .PAD(x[0]));
PDI px1 (.C(x_top[1]), .PAD(x[1]));
PDI px2 (.C(x_top[2]), .PAD(x[2]));
PDI px3 (.C(x_top[3]), .PAD(x[3]));
PDI px4 (.C(x_top[4]), .PAD(x[4]));
PDI px5 (.C(x_top[5]), .PAD(x[5]));
PDI px6 (.C(x_top[6]), .PAD(x[6]));
PDI px7 (.C(x_top[7]), .PAD(x[7]));
PDI pin_select0 (.C(in_select_top[0]), .PAD(in_select[0]));
PDI pin_select1 (.C(in_select_top[1]), .PAD(in_select[1]));
PDI pin_select2 (.C(in_select_top[2]), .PAD(in_select[2]));
PDI pinwe (.C(inwe_top), .PAD(inwe));
PDI pin_clock (.C(in_clock_top), .PAD(in_clock));
PDI pbindct_clock (.C(bindct_clock_top), .PAD(bindct_clock));
PDI preset (.C(reset_top), .PAD(reset));
PDI pout_select0 (.C(out_select_top[0]), .PAD(out_select[0]));
PDI pout_select1 (.C(out_select_top[1]), .PAD(out_select[1]));
PDI pout_select2 (.C(out_select_top[2]), .PAD(out_select[2]));
PDI poutre (.C(outre_top), .PAD(outre));
PDI pout_clock (.C(out_clock_top), .PAD(out_clock));
PDO08C py0 (.PAD(y[0]), .I(y_top[0]));
PDO08C py1 (.PAD(y[1]), .I(y_top[1]));
PDO08C py2 (.PAD(y[2]), .I(y_top[2]));
PDO08C py3 (.PAD(y[3]), .I(y_top[3]));
PDO08C py4 (.PAD(y[4]), .I(y_top[4]));
PDO08C py5 (.PAD(y[5]), .I(y_top[5]));
PDO08C py6 (.PAD(y[6]), .I(y_top[6]));
PDO08C py7 (.PAD(y[7]), .I(y_top[7]));
PDO08C py8 (.PAD(y[8]), .I(y_top[8]));
PDO08C py9 (.PAD(y[9]), .I(y_top[9]));
PDO08C py10 (.PAD(y[10]), .I(y_top[10]));
endmodule

// One-dimensional Forward Integer binDCT
// Uses dyadic lifting steps to accomplish the rotation function
// in the fast forward implementation of the Discrete
// Cosine transform
// Created by: Tracy Franklin

module forwarddct (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outre, out_clock, y);

input [7:0] x;
input [2:0] in_select, out_select;
input inwe, outre;
input in_clock, out_clock;
input bindct_clock, reset;
output [10:0] y;
wire [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
wire [10:0] z0, z1, z2, z3, z4, z5, z6, z7;

s_p_in8 s_p_in8 (x, in_select, inwe, in_clock, bindct_clock, x0, x1, x2, x3, x4, x5, x6, x7);
bindct8 bindct8 (x0, x1, x2, x3, x4, x5, x6, x7, bindct_clock, reset, z0, z1, z2, z3, z4, z5, z6, z7);
p_s_out11 p_s_out11 (z0, z1, z2, z3, z4, z5, z6, z7, out_select, outre, bindct_clock, out_clock, y);
endmodule

// The output circuitry to resolve the eight parallel outputs
// into one serial output.
module p_s_out11 (z0, z1, z2, z3, z4, z5, z6, z7, select_out, read_enable, parallel_clock, serial_clock, zout);

input [10:0] z0, z1, z2, z3, z4, z5, z6, z7;
input [2:0] select_out;
input serial_clock, parallel_clock, read_enable;
output [10:0] zout;

reg [10:0] zout;
reg [10:0] outputmem [0:7];

always @(parallel_clock or z0 or z1 or z2 or z3 or z4 or z5 or z6 or z7) begin
  if (parallel_clock == 1'b1) begin
    outputmem[3’d0]=z0;
    outputmem[3’d1]=z1;
    outputmem[3’d2]=z2;
    outputmem[3’d3]=z3;
    outputmem[3’d4]=z4;
    outputmem[3’d5]=z5;
    outputmem[3’d6]=z6;
    outputmem[3’d7]=z7;
  end
end

always @(read_enable or select_out or serial_clock) begin
  if (read_enable == 1'b1) begin
    if (serial_clock == 1'b1)
      zout= outputmem[select_out];
  end
end
endmodule

// Note inpclock has a period eight times the size of the insclock.
// Input circuitry to the 1-D DCT chip to reduce the number of pads
module s_p_in8 (x, select, write_enable, serial_clock, parallel_clock, x0, x1, x2, x3, x4, x5, x6, x7);

input [7:0] x;
input [2:0] select;
input write_enable;
input serial_clock, parallel_clock;
output [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [7:0] inputmem [0:7];

always @ (serial_clock or select or x or write_enable) begin
    if (write_enable == 1'b1) begin
        if (serial_clock == 1'b1)
            inputmem[select]= x;
    end
end

always @ (posedge parallel_clock) begin
    x0= inputmem[3'd0];
    x1= inputmem[3'd1];
    x2= inputmem[3'd2];
    x3= inputmem[3'd3];
    x4= inputmem[3'd4];
    x5= inputmem[3'd5];
    x6= inputmem[3'd6];
    x7= inputmem[3'd7];
end
endmodule

// HDL description of the 8x8 1-D binDCT-B
// normalized version
// Created by: Tracy Franklin

module bindct8 (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, z0, z1, z2, z3, z4, z5, z6, z7);
inout [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input clk, reset;
output [10:0] z0,z1,z2,z3,z4,z5,z6,z7;

wire [8:0] a0,a1,a2,a3;
wire [7:0] a4,a5,a6,a7;
wire [8:0] b5,b6;
wire [9:0] c0,c1,c4,c7;
wire [8:0] c2,c3,c5,c6;
wire [10:0] d0;
wire [9:0] d4,d1,d7,d5,d2;
wire [8:0] d3,d6;
wire [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [10:0] z0,z1,z2,z3,z4,z5,z6,z7;

preadd f1 (x0, x1, x2, x3, x4, x5, x6, x7, reset, a0, a1, a2, a3, a4, a5, a6, a7);
prefit f2 (a5, a6, reset, b5, b6);
midadd f3 (a0, a1, a2, a3, a4, b5, b6, a7, reset, c0, c1, c2, c3, c4, c5, c6, c7);
postlift f4 (c0, c1, c2, c3, c4, c5, c6, c7, reset, d0, d1, d2, d3, d4, d5, d6, d7);
rearrangement f5 (d0, d1, d2, d3, d4, d5, d6, d7, reset, e0, e1, e2, e3, e4, e5, e6, e7);

always @ (posedge clk or negedge reset) begin
    if (!reset) begin
        z0<=0;
        z1<=0;
        z2<=0;
        z3<=0;
        z4<=0;
        z5<=0;
    end
end
z6=0;
z7=0;
end
else begin
  z0=e0;
  z1=e1;
  z2=e2;
  z3=e3;
  z4=e4;
  z5=e5;
  z6=e6;
  z7=e7;
end
end
endmodule

module preadd (x0, x1, x2, x3, x4, x5, x6, x7, reset, a0, a1, a2, a3, a4, a5, a6, a7);
input [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input reset;
output [8:0] a0,a1,a2,a3;
output [7:0] a4,a5,a6,a7;
add8 ra1 (x0,x7,reset,a0);
add8 ra2 (x1,x6,reset,a1);
add8 ra3 (x2,x5,reset,a2);
add8 ra4 (x3,x4,reset,a3);
sub8 ra5 (x3,x4,reset,a4);
sub8 ra6 (x2,x5,reset,a5);
sub8 ra7 (x1,x6,reset,a6);
sub8 ra8 (x0,x7,reset,a7);
endmodule

module prelift (a5, a6, reset, b5, b6);
input [7:0] a5,a6;
input reset;
output [8:0] b5,b6;
wire [6:0] temp1;
wire [8:0] temp2;
wire [8:0] b6;
ls388 rl1 (a5,reset,temp1);
add8 rl2 ( {1 b0,temp1[6:0]}, a6,reset,b6);
ls589 rl3 (b6,reset,temp2);
sub9 rl4 (temp2, {1 b0,a5[7:0]}, reset,b5);
endmodule

module midadd (a0, a1, a2, a3, a4, b5, b6, a7, reset, c0, c1, c2, c3, c4, c5, c6, c7);
input [8:0] a0,a1,a2,a3,b5,b6;
input [7:0] a4,a7;
input reset;
output [9:0] c0,c1,c4,c7;
output [8:0] c2,c3,c5,c6;
add9 ma1 (a0,a3,reset,c0);
add9 ma2 (a1,a2,reset,c1);
sub9 ma3 (a1,a2,reset,c2);
sub9 ma4 (a0,a3,reset,c3);
add9 ma5 ( [1 b0,a4[7:0]],b5,reset,c4);
sub9 ma6 ( [1 b0,a4[7:0]],b5,reset,c5);
sub9 ma7 ( [1 b0,a7[7:0]],b6,reset,c6);
add9 ma8 ( [1 b0,a7[7:0]],b6,reset,c7);
endmodule

module postlift (c0, c1, c2, c3, c4, c5, c6, c7,reset, d0, d1, d2, d3, d4, d5, d6, d7);
input [9:0] c0,c1,c4,c7;
input [8:0] c2,c3,c5,c6;
input reset;
output [10:0] d0;
output [9:0] d4,d1,d7,d5,d2;
output [8:0] d3,d6;
wire [9:0] temp3;
wire [8:0] temp5,temp6;
wire [7:0] temp7,temp9,temp10;
wire [6:0] temp4,temp8;

add10 o1 (c0,c1,reset,d0);
ls1211 o2 (d0,reset,temp3);
sub10 o3 (temp3,c1,reset,d4);
ls1810 o4 (c7,reset,temp4);
sub10 o5 (c4,[3 b000,temp4[6:0]],reset,d7);
ls349 o6 (c5,reset,temp5);
sub9 o7 (c6,temp5,reset,temp6);
ls129 o8 (temp6,reset,temp7);
add9 o9 (c5,[1 b0,temp7[7:0]],reset,d5);
ls1810 o10 (d5,reset,temp8);
sub9 o11 (temp6,[2 b00,temp8[6:0]],reset,d3);
ls389 o12 (c3,reset,temp9);
sub9 o13 (c2,[1 b0,temp9[7:0]],reset,d6);
ls389 o14 (d6,reset,temp10);
add9 o15 (c3,[1 b0,temp10[7:0]],reset,d2);
assign d1= c7;
endmodule

module rearrange (d0, d1, d2, d3, d4, d5, d6, d7, reset, e0, e1, e2, e3, e4, e5, e6, e7);
input [10:0] d0;
input [9:0] d1,d2,d4,d5,d7;
input [8:0] d3,d6;
input reset;
output [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [10:0] e0,e1,e2,e3,e4,e5,e6,e7;
always @ (d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
    if (reset) begin
        e0=0;
        e1=0;
        e2=0;
        e3=0;
        e4=0;
    end
e5=0;
e6=0;
e7=0;
end
else begin
e0=d0;
e1={1'b0,d1[9:0]};
e2={1'b0,d2[9:0]};
e3={2'b00,d3[8:0]};
e4={1'b0,d4[9:0]};
e5={1'b0,d5[9:0]};
e6={2'b00,d6[8:0]};
e7={1'b0,d7[9:0]};
end
end
endmodule

module add8 (x,y,reset,sum);
input [7:0] x,y;
input reset;
output [8:0] sum;
reg [8:0] sum;

always @ (x or y or reset) begin
  if (!reset) begin
    sum=0;
    end
  else begin
    sum=x+y;
  end
end
endmodule

module sub8 (x,y,reset,diff);
input [7:0] x,y;
input reset;
output [7:0] diff;
reg [7:0] diff;

always @ (x or y or reset) begin
  if (!reset) begin
    diff=0;
    end
  else begin
    diff=x-y;
  end
end
endmodule

module ls388 (x,reset,lift);
input [7:0] x;
input reset;
output [6:0] lift;
wire [8:0] shiftx;
wire [9:0] sumx;

reg [6:0] lift;

assign shiftx = ({1'b0,x})<<1;
add9 u1 (shiftx,{1'b0,x},reset,sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx)>>3;
  end
end
endmodule

module ls589 (x,reset,lift);
  input [8:0] x;
  input reset;
  output [8:0] lift;
wire [10:0] shiftx;
wire [11:0] sumx;

reg [8:0] lift;

assign shiftx = ({2'b00,x})<<2;
add11 u2 (shiftx,{2'b00,x},reset,sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx)>>3;
  end
end
endmodule

module add9 (x,y,reset,sum);
  input [8:0] x,y;
  input reset;
  output [9:0] sum;
reg [9:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end

end
endmodule

module sub9 (x,y,reset,diff);
input [8:0] x,y;
input reset;
output [8:0] diff;

reg [8:0] diff;

always @ (x or y or reset) begin
if (!reset) begin
  diff=0;
end
else begin
  diff=x-y;
end
endmodule

module add10 (x,y,reset,sum);
input [9:0] x,y;
input reset;
output [10:0] sum;

reg [10:0] sum;

always @ (x or y or reset) begin
if (!reset) begin
  sum=0;
end
else begin
  sum=x+y;
end
endmodule

module sub10 (x,y,reset,diff);
input [9:0] x,y;
input reset;
output [9:0] diff;

reg [9:0] diff;

always @ (x or y or reset) begin
if (!reset) begin
  diff=0;
end
else begin
  diff=x-y;
end
endmodule

module add11 (x,y,reset,sum);
input [10:0] x,y;


input reset;
output [11:0] sum;

reg [11:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum = 0;
  end
  else begin
    sum = x + y;
  end
end
endmodule

module ls1211 (x, reset, lift);
input [10:0] x;
input reset;
output [9:0] lift;

reg [9:0] lift;

always @(x or reset) begin
  if (!reset) begin
    lift = 0;
  end
  else begin
    lift = (x >> 1);
  end
end
endmodule

module ls129 (x, reset, lift);
input [8:0] x;
input reset;
output [7:0] lift;

reg [7:0] lift;

always @(x or reset) begin
  if (!reset) begin
    lift = 0;
  end
  else begin
    lift = (x >> 1);
  end
end
endmodule

module ls1810 (x, reset, lift);
input [9:0] x;
input reset;
output [6:0] lift;

reg [6:0] lift;
always @ (x or reset) begin
    if (!reset) begin
        lift=0;
        end
    else begin
        lift=(x>>3);
    end
end
endmodule

module ls349 (x,reset,lift);
    input [8:0] x;
    input reset;
    output [8:0] lift;

    wire [9:0] shiftx;
    wire [10:0] sumx;

    reg [8:0] lift;

    assign shiftx=((1'b0,x)<<1);
    add10 u3 (shiftx,(1'b0,x),reset,sumx);

always @ (sumx or reset) begin
    if (!reset) begin
        lift=0;
        end
    else begin
        lift=(sumx << 2);
    end
end
endmodule

module ls389 (x,reset,lift);
    input [8:0] x;
    input reset;
    output [7:0] lift;

    wire [9:0] shiftx;
    wire [10:0] sumx;

    reg [7:0] lift;

    assign shiftx=((1'b0,x)<<1);
    add10 u4 (shiftx,(1'b0,x),reset,sumx);

always @ (sumx or reset) begin
    if (!reset) begin
        lift=0;
        end
    else begin
        lift=(sumx >> 3);
    end
end
endmodule
B.3 Stimulus for the one-dimensional forward binDCT RTL description

```vhdl
// stimulus_forwarddct_rtl.v
// Testbench for forward dct module
// Created by: Tracy Franklin
'time scale 1ns/10ps
module stimulus_forwarddct;

reg in_write_enable, out_read_enable, reset;
reg in_clock, bindct_clock, out_clock;
reg [2:0] in_select, out_select;
reg [7:0] x;
wire [10:0] y0, y1, y2, y3, y4, y5, y6, y7;
//wire [10:0] y;

// rtl-level
forwarddct rtl_test (x, in_select, in_write_enable, in_clock, bindct_clock, reset, out_select,
out_read_enable, out_clock, y0, y1, y2, y3, y4, y5, y6, y7);

// gate-level
// forwarddct chip_test (x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0], in_select[2], in_select[1], in_select[0],
in_write_enable, in_clock, bindct_clock, reset, out_select[2], out_select[1], out_select[0],
out_read_enable, out_clock, y[10], y[9], y[8], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);

initial begin
// Initial conditions
  in_clock=0;
  bindct_clock=0;
  out_clock=0;
  out_read_enable='b1;
  in_write_enable='b1;
  #400 $finish;
end
always #2 in_clock= ~in_clock;
always #40 bindct_clock = ~bindct_clock;
always #2 out_clock= ~out_clock;

initial begin
  reset='b0;
  #3 reset='b1;
end

initial begin

  #4 in_select='b000; x=8'd1;
  #4 in_select='b001; x=8'd0;
  #4 in_select='b010; x=8'd0;
  #4 in_select='b011; x=8'd0;
  #4 in_select='b100; x=8'd0;
  #4 in_select='b101; x=8'd0;
  #4 in_select='b110; x=8'd0;
  #4 in_select='b111; x=8'd0;

end
```

82
// monitor the outputs

initial
begin
    $display("Time,", "x", "y0", y1, y2, y3, y4, y5, y6, y7");
end

always @ (posedge in_clock) begin
    $monitor("%0d\t %b\t %b\t %b\t %b\t %b\t %b\t %b\t \n", $time, x, 0, y1, y2, y3, y4, y5, y6, y7);
end

/*
always @ (posedge bindct_clock) begin
    wait (out_read_enable) out_select=3'd7;
    $display("%0d\t select=\%d, \%b\", $time, out_select, y);
end
*/
endmodule

B.4 Stimulus for one-dimensional forward binDCT chip RTL description

// stimulus_forwardbindctchip_rtl.v
// Testbench for forward dct module
// Created by: Tracy Franklin
'timescale 1ns/10ps
module stimulus_forwardbindctchip;

reg in_write_enable;
reg out_read_enable;
reg reset;
reg in_clock;
reg bindct_clock;
reg out_clock;
reg [2:0] in_select;
reg [2:0] out_select;
reg [7:0] x;
wire [10:0] y;

// rtl-level
forwardbindctchip rtl_test (x, in_select, in_write_enable, in_clock, bindct_clock, reset, out_select, out_read_enable, out_clock, y);

// gate-level
// forwardbindctchip chip_test (x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0], in_select[2], in_select[1], in_select[0], in_write_enable, in_clock, bindct_clock, reset, out_select[2], out_select[1], out_select[0], out_read_enable, out_clock. y[10], y[9], y[8], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);

initial
begin
// Initial conditions
  in_clock=0;
  bindct_clock=0;
  out_clock=0;
  out_read_enable=1'b1;
  in_write_enable=1'b1;
  #400 $finish;
end

always #2 in_clock= ~in_clock;
always #40 bindct_clock = ~bindct_clock;
always #2 out_clock= ~out_clock;

initial begin
  reset=1'b0;
  #3 reset=1'b1;
end

initial begin
#4 in_select=3'b000; x=8'd1;
#4 in_select=3'b001; x=8'd0;
#4 in_select=3'b010; x=8'd0;
#4 in_select=3'b011; x=8'd0;
#4 in_select=3'b100; x=8'd0;
#4 in_select=3'b101; x=8'd0;
#4 in_select=3'b110; x=8'd0;
#4 in_select=3'b111; x=8'd0;
#40 in_select=3'b000; x=8'd1;
#4 in_select=3'b001; x=8'd1;
#4 in_select=3'b010; x=8'd1;
#4 in_select=3'b011; x=8'd1;
#4 in_select=3'b100; x=8'd1;
#4 in_select=3'b101; x=8'd1;
end
#4 in_select=3'b110; x=8'd1;
#4 in_select=3'b111; x=8'd1;

#52 in_select=3'b000; x=8'd1;
#4 in_select=3'b001; x=8'd1;
#4 in_select=3'b010; x=8'd1;
#4 in_select=3'b011; x=8'd1;
#4 in_select=3'b100; x=8'd0;
#4 in_select=3'b101; x=8'd0;
#4 in_select=3'b110; x=8'd0;
#4 in_select=3'b111; x=8'd0;

end

// monitor the outputs
always @ (posedge bindct_clock) begin
  wait (out_read_enable) out_select=3'd0; $display("%0d: select=%d, y=%d", time, out_select, y);
end

endmodule

B.5 One-dimensional 8-point inverse binDCT RTL description

// One-dimensional Inverse Integer binDCT
// Uses dyadic lifting steps to accomplish the rotation function
// in the fast inverse implementation of the Discrete
// Cosine transform
// Created by: Tracy Franklin

`timescale 100 ns/ 1 ns

module inverse_dct (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outra, out_clock, z0, z1, z2, z3, z4, z5, z6, z7);

input [10:0] x;
input [2:0] in_select, out_select;
inout inwe, outra;
inout in_clock, out_clock;
input bindct_clock, reset;
output [12:0] z0, z1, z2, z3, z4, z5, z6, z7;
// output [12:0] y;

wire [10:0] x0, x1, x2, x3, x4, x5, x6, x7;
wire [12:0] z0, z1, z2, z3, z4, z5, z6, z7;
s_p_in11 s_p_in11 (x, in_select, inwe, in_clock, bindct_clock, x0, x1, x2, x3, x4, x5, x6, x7);
ibindct11 ibindct11 (x0, x1, x2, x3, x4, x5, x6, x7, bindct_clock, reset, z0, z1, z2, z3, z4, z5, z6, z7);
p_s_out13 p_s_out13 (z0, z1, z2, z3, z4, z5, z6, z7, out_select, outra, bindct_clock, out_clock, y);
endmodule
// The output circuitry to resolve the eight parallel outputs
// into one serial output.
module p_s_out13 (z0, z1, z2, z3, z4, z5, z6, z7, select_out, read_enable, parallel_clock, serial_clock, 
    zout);

input [12:0] z0, z1, z2, z3, z4, z5, z6, z7;
input [2:0] select_out;
input serial_clock, read_enable, parallel_clock;
output [12:0] zout;

reg [12:0] zout;
reg [12:0] outputmem [0:7];

always @(parallel_clock or z0 or z1 or z2 or z3 or z4 or z5 or z6 or z7 or select_out) begin
    if (parallel_clock == 1'b1) begin
        outputmem[3'd0]= z0;
        outputmem[3'd1]= z1;
        outputmem[3'd2]= z2;
        outputmem[3'd3]= z3;
        outputmem[3'd4]= z4;
        outputmem[3'd5]= z5;
        outputmem[3'd6]= z6;
        outputmem[3'd7]= z7;
    end
end

always @(read_enable or select_out or serial_clock) begin
    if (read_enable == 1'b1) begin
        if (serial_clock == 1'b1) zout=outputmem[select_out];
    end
end
endmodule

// Note inpclk has a period eight times the size of the insclk.
// Input circuitry to the 1-D DCT chip to reduce the number of pads
module s_p_in11 (x, select, write_enable, serial_clock, parallel_clock, x0, x1, x2, x3, x4, x5, x6, x7);

input [10:0] x;
input [2:0] select;
input write_enable;
input serial_clock, parallel_clock;
output [10:0] x0, x1, x2, x3, x4, x5, x6, x7;

reg [10:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [10:0] inputmem [0:7];

always @ (serial_clock or select or x or write_enable) begin
    if (write_enable == 1'b1) begin
        if (serial_clock== 1'b1) 
            inputmem[select]= x;
    end
end

always @ (posedge parallel_clock) begin
    x0=inputmem[3'd0];
end
x1=inputmem[3'd1];
x2=inputmem[3'd2];
x3=inputmem[3'd3];
x4=inputmem[3'd4];
x5=inputmem[3'd5];
x6=inputmem[3'd6];
x7=inputmem[3'd7];
end
endmodule

// ibindct_rtl.v
// A Configurable vector size of the inputs
// HDL description of the 8-point 1-D Inverse binDCT-B
// A normalized version, using a scale of sqrt(2/N), where N=8.
// Created by: Tracy Franklin

module ibindct11 (z0,z1,z2,z3,z4,z5,z6,z7,clk,reset,x0,x1,x2,x3,x4,x5,x6,x7);
input [10:0] z0,z1,z2,z3,z4,z5,z6,z7;
input clk,reset;
output [12:0] x0,x1,x2,x3,x4,x5,x6,x7;

wire [10:0] a0,a1,a2,a3,a4,a5;
wire [11:0] a6,a7;
wire [12:0] a3;
wire [10:0] b2;
wire [11:0] b0,b5,b6;
wire [12:0] b4,b7,b3;
wire [13:0] b1;
wire [12:0] c5,c3;
wire [10:0] d6,d5,d4;
wire [11:0] d7,d1,d2,d3;
wire [12:0] d0;
wire [12:0] e0,e1,e2,e3,e4,e5,e6,e7;

reg [12:0] x0,x1,x2,x3,x4,x5,x6,x7;

prelftadd i1 (z0,z1,z2,z3,z4,z5,z6,z7,reset,a0,a1,a2,a3,a4,a5,a6,a7);
midaddsub i2 (a0,a1,a2,a3,a4,a5,a6,a7,reset,b0,b1,b2,b3,b4,b5,b6,b7);
midlift i3 (b3,b5,reset,c3,c5);
postadd i4 (b0,b1,b2,c3,b4,c5,b6,b7,reset,d0,d1,d2,d3,d4,d5,d6,d7);
realign i5 (d0,d1,d2,d3,d4,d5,d6,d7,reset,e0,e1,e2,e3,e4,e5,e6,e7);

always @(posedge clk or negedge reset) begin
if (~reset) begin
   x0=0;
x1=0;
x2=0;
x3=0;
x4=0;
x5=0;
x6=0;
x7=0;
end
else begin
   x0=e0;
end
end
x1=e1;
x2=e2;
x3=e3;
x4=e4;
x5=e5;
x6=e6;
x7=e7;
end
end

doubleadd 
module prelifadd (z0,z1,z2,z3,z4,z5,z6,z7,reset,a0,a1,a2,a3,a4,a5,a6,a7);
input [10:0] z0,z1,z2,z3,z4,z5,z6,z7;
input reset;
output [10:0] a0,a1,a2,a4,a5;
output [11:0] a6,a7;
output [12:0] a3;
wire [7:0] temp4,temp5;
wire [9:0] temp1,temp2,temp3;
wire [10:0] a4,a2,temp7,temp8;
wire [11:0] temp6;
ls1211 11 (z0,reset,temp1);
sub11 12 (1 b0,temp1,z4,reset,a4);
sub11 13 (z0,a4,reset,a0);
ls3811 14 (z6,reset,temp2);
sub11 15 (z2,1 b0,temp2,reset,a2);
ls3811 16 (a2,reset,temp3);
add11 17 (1 b0,temp3,z6,reset,a6);
ls1811 18 (z1,reset,temp4);
ls1811 19 (z5,reset,temp5);
add11 110 (1 b000,temp4,z7,reset,a7);
add11 111 (1 b000,temp5,z3,reset,temp6);
ls1212 112 (temp6,reset,temp7);
sub11 113 (z5,temp7,reset,a5);
ls3411 114 (a5,reset,temp8);
add12 115 (1 b0,temp8,reset,a3);
assign a1=z1;
endmodule

module midaddsub (a0,a1,a2,a3,a4,a5,a6,a7,reset,b0,b1,b2,b3,b4,b5,b6,b7);
input [10:0] a0,a1,a2,a4,a5;
input [11:0] a6,a7;
input [12:0] a3;
input reset;
output [10:0] b2;
output [11:0] b0,b5,b6;
output [12:0] b4,b7,b3;
output [13:0] b1;
add11 m1 (a0,a2,reset,b0);
sub11 m2 (a0,a2,reset,b2);
add12 m3 (1 b0,a4),a6,reset,b4);
sub12 m4 (1 b0,a4),a6,reset,b6);

88
add12 m5 (a7, {1 b0, a5}, reset, b7);
sub12 m6 (a7, {1 b0, a5}, reset, b5);
add13 m7 ({2 b00, a1}, a3, reset, b1);
sub13 m8 ({2 b00, a1}, a3, reset, b3);
endmodule

module midlift (b3, b5, reset, c3, c5);
input [12:0] b3;
input [11:0] b5;
input reset;
output [12:0] c5, c3;
wire [12:0] temp9, c5;
wire [11:0] temp10;
ls5813 l1 (b3, reset, temp9);
sub13 l2 (temp9, {1 b0, b5}, reset, c5);
ls3813 l3 (c5, reset, temp10);
sub13 l4 (b3, {1 b0, temp10}, reset, c3);
endmodule

module postadd (b0, b1, b2, c3, b4, c5, b6, b7, reset, d0, d1, d2, d3, d4, d5, d6, d7);
input [10:0] b2;
input [11:0] b0, b6;
input [12:0] c3, b4, c5, b7;
input [13:0] b1;
input reset;
output [10:0] d6, d5, d4;
output [11:0] d7, d3, d2, d1;
output [12:0] d0;
wire [12:0] temp16, temp15, temp14;
wire [13:0] temp17, temp11, temp12, temp13;
wire [14:0] temp10;
add14 a1 ({2 b00, b0}, b1, reset, temp10);
sub14 a2 ({2 b00, b0}, b1, reset, temp17);
add13 a3 (b4, c3, reset, temp11);
sub13 a4 (b4, c3, reset, temp16);
add13 a5 ({1 b0, b6}, c5, reset, temp12);
sub13 a6 ({1 b0, b6}, c5, reset, temp15);
add13 a7 ({2 b00, b2}, b7, reset, temp13);
sub13 a8 ({2 b00, b2}, b7, reset, temp14);
scale_4_15 s1 (temp10, reset, d0);
scale_4_14 s2 (temp17, reset, d7);
scale_4_14 s3 (temp11, reset, d1);
scale_4_13 s4 (temp16, reset, d6);
scale_4_14 s5 (temp12, reset, d2);
scale_4_13 s6 (temp15, reset, d5);
scale_4_14 s7 (temp13, reset, d3);
scale_4_13 s8 (temp14, reset, d4);
endmodule

module realign (d0, d1, d2, d3, d4, d5, d6, d7, reset, e0, e1, e2, e3, e4, e5, e6, e7);
input [10:0] d4,d5,d6;
input [11:0] d1,d2,d3,d7;
input [12:0] d0;
inout reset;
output [12:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [12:0] e0,e1,e2,e3,e4,e5,e6,e7;

always @(d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
if (!reset) begin
  e0=0;
e1=0;
e2=0;
e3=0;
e4=0;
e5=0;
e6=0;
e7=0;
end
else begin
  e0=(d0);
e1=(1'b0,d1);
e2=(1'b0,d2);
e3=(1'b0,d3);
e4=(2'b00,d4);
e5=(2'b00,d5);
e6=(2'b00,d6);
e7=(1'b0,d7);
end
end
endmodule

module scale_4_l3(x,reset,y);
input [12:0] x;
inout reset;
output [10:0] y;
reg [10:0] y;
always @ (x or reset) begin
  if (!reset) begin
    y=0;
  end
  else begin
    y=(x>>2);
  end
end
endmodule

module scale_4_l4(x,reset,y);
input [13:0] x;
inout reset;
output [11:0] y;
reg [11:0] y;
always @ (x or reset) begin
  if (!reset) begin

y=0;
end
else begin
  y=(x>>2);
end
dendmodule

module scale_4_15(x,reset,y);
  input [14:0] x;
  input reset;
  output [12:0] y;
  reg [12:0] y;

  always @(x or reset) begin
    if (reset) begin
      y=0;
    end
    else begin
      y=(x>>2);
    end
  end
dendmodule

module ls1211 (x,reset,lift);
  input [10:0] x;
  input reset;
  output [9:0] lift;
  reg [9:0] lift;

  always @(x or reset) begin
    if (reset) begin
      lift=0;
    end
    else begin
      lift=(x>>1);
    end
  end
dendmodule

module sub14 (x,y,reset,diff);
  input [13:0] x,y;
  input reset;
  output [13:0] diff;
  reg [13:0] diff;

  always @(x or y or reset) begin
    if (reset) begin
      diff=0;
    end
    else begin
      diff=x-y;
    end
  end
end
endmodule

module ls3811 (x,reset,lift);
input [10:0] x;
input reset;
output [9:0] lift;

reg [11:0] shiftx;
wire [12:0] sumx;
reg [9:0] lift;

always @(x or reset) begin
  shiftx=(1'b0,x)<<1;
end

add12 u1 (shiftx,(1'b0,x),reset,sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx>>3);
  end
end
endmodule

module add12 (x,y,reset,sum);
input [11:0] x,y;
input reset;
output [12:0] sum;

reg [12:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module add15 (x,y,reset,sum);
input [14:0] x,y;
input reset;
output [15:0] sum;

reg [15:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
end

92
else begin
    sum=x+y;
end
endmodule

module ls1811 (x,reset,lift);
input [10:0] x;
input reset;
output [7:0] lift;
reg [7:0] lift;
always @(x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(x>>3);
    end
end
endmodule

module ls1212 (x,reset,lift);
input [11:0] x;
input reset;
output [10:0] lift;
reg [10:0] lift;
always @(x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(x>>1);
    end
end
endmodule

module ls3411 (x,reset,lift);
input [10:0] x;
input reset;
output [10:0] lift;
reg [11:0] shiftx;
wire [12:0] sumx;
reg [10:0] lift;
always @(x or reset) begin
    shiftx=({1'b0,x}<<1);
end
add12 u2 (shiftx,{1'b0,x},reset,sumx);
always @(sumx or reset) begin
if (!reset) begin
    lift=0;
end
else begin
    lift=(sumx>>2);
end
endmodule

module add11 (x,y,reset,sum);
    input [10:0] x,y;
    input reset;
    output [11:0] sum;
reg [11:0] sum;
always @(x or y or reset) begin
    if (!reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
end
endmodule

module sub12 (x,y,reset,diff);
    input [11:0] x,y;
    input reset;
    output [11:0] diff;
reg [11:0] diff;
always @(x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
endmodule

module sub11 (x,y,reset,diff);
    input [10:0] x,y;
    input reset;
    output [10:0] diff;
reg [10:0] diff;
always @(x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
module is5813 (x, reset, lift);
input [12:0] x;
input reset;
output [12:0] lift;

reg [14:0] shiftx;
wire [15:0] sumx;
reg [12:0] lift;

always @(x or reset) begin
    shiftx = ((2'b00,0) << 2);
end

add15 u3 (shiftx, {2'b00,0}, reset, sumx);

always @(sumx or reset) begin
    if (!reset) begin
        lift = 0;
    end
    else begin
        lift = (sumx >> 3);
    end
end
endmodule

module add14 (x, y, reset, sum);
input [13:0] x, y;
input reset;
output [14:0] sum;

reg [14:0] sum;

always @(x or y or reset) begin
    if (!reset) begin
        sum = 0;
    end
    else begin
        sum = x + y;
    end
end
endmodule

module is3813 (x, reset, lift);
input [12:0] x;
input reset;
output [11:0] lift;

reg [13:0] shiftx;
wire [14:0] sumx;
reg [11:0] lift;

always @(x or reset) begin

shiftx=(1'b0,x)<<1;
end

add14 #4 (shift,(1'b0,x),reset,sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx>>3);
  end
end
endmodule

module add13 (x,y,reset,sum);
input [12:0] x,y;
input reset;
output [13:0] sum;

reg [13:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module sub13 (x,y,reset,diff);
input [12:0] x,y;
input reset;
output [12:0] diff;

reg [12:0] diff;

always @(x or y or reset) begin
  if (!reset) begin
    diff=0;
  end
  else begin
    diff=x-y;
  end
end
endmodule
B.6 One-dimensional 8-point inverse binDCT RTL description with the

0.35μ CMOS technology pads included

// HDL description of the 8-point inverse binDCT-B chip
// Created by: Tracy A. Franklin
```
timescale 1ns/10 ps
module inversebindctchip (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outre, out_clock, y);

input [10:0] x;
input [2:0] in_select;
input inwe;
input in_clock;
input bindct_clock;
input reset;
input [2:0] out_select;
input outre;
inout out_clock;
output [12:0] y;

wire [10:0] x_top;
wire [2:0] in_select_top;
wire inwe_top;
wire in_clock_top;
wire bindct_clock_top;
wire reset_top;
wire [2:0] out_select_top;
wire outre_top;
wire out_clock_top;
wire [12:0] y_top;

inversebindct inversedct (x_top, in_select_top, inwe_top, in_clock_top, bindct_clock_top, reset_top,
 out_select_top, outre_top, out_clock_top, y_top);

PDI px0 (.C(x_top[0]), .PAD(x[0]));
PDI px1 (.C(x_top[1]), .PAD(x[1]));
PDI px2 (.C(x_top[2]), .PAD(x[2]));
PDI px3 (.C(x_top[3]), .PAD(x[3]));
PDI px4 (.C(x_top[4]), .PAD(x[4]));
PDI px5 (.C(x_top[5]), .PAD(x[5]));
PDI px6 (.C(x_top[6]), .PAD(x[6]));
PDI px7 (.C(x_top[7]), .PAD(x[7]));
PDI px8 (.C(x_top[8]), .PAD(x[8]));
PDI px9 (.C(x_top[9]), .PAD(x[9]));
PDI px10 (.C(x_top[10]), .PAD(x[10]));
PDI pin_select0 (.C(in_select_top[0]), .PAD(in_select[0]));
PDI pin_select1 (.C(in_select_top[1]), .PAD(in_select[1]));
PDI pin_select2 (.C(in_select_top[2]), .PAD(in_select[2]));
PDI pin_inwe (.C(inwe_top), .PAD(inwe));
PDI pin_in_clock (.C(in_clock_top), .PAD(in_clock));
PDI pin_bindct_clock (.C(bindct_clock_top), .PAD(bindct_clock));
PDI pin_reset (.C(reset_top), .PAD(reset));
PDI pin_out_select0 (.C(out_select_top[0]), .PAD(out_select[0]));
PDI pin_out_select1 (.C(out_select_top[1]), .PAD(out_select[1]));
```

97
module inversedct (x, in_select, inwe, in_clock, bindct_clock, reset, out_select, outre, out_clock, y);

input [10:0] x;
input [2:0] in_select, out_select;
input inwe, outre;
input in_clock, out_clock;
input bindct_clock, reset;
output [12:0] y;
wire [10:0] x0, x1, x2, x3, x4, x5, x6, x7;
wire [12:0] z0, z1, z2, z3, z4, z5, z6, z7;
s_p_in11 $p_p_in11 (x, in_select, inwe, in_clock, bindct_clock, x0, x1, x2, x3, x4, x5, x6, x7);
ibindct11 $ibindct11 (x0, x1, x2, x3, x4, x5, x6, x7, bindct_clock, reset, z0, z1, z2, z3, z4, z5, z6, z7);
p_s_out13 $p_s_out13 (z0, z1, z2, z3, z4, z5, z6, z7, out_select, outre, bindct_clock, out_clock, y);
endmodule

module p_s_out13 (z0, z1, z2, z3, z4, z5, z6, z7, select_out, read_enable, parallel_clock, serial_clock, zout);

input [12:0] z0, z1, z2, z3, z4, z5, z6, z7;
input [2:0] select_out;
input serial_clock, read_enable, parallel_clock;
output [12:0] zout;
reg [12:0] zout;
reg [12:0] outputmem [0:7];
always @ (parallel_clock or z0 or z1 or z2 or z3 or z4 or z5 or z6 or z7 or select_out) begin
  if (parallel_clock == 1'b1) begin
    outputmem[3'd0]= z0;
    outputmem[3'd1]= z1;
    outputmem[3'd2]= z2;
    outputmem[3'd3]= z3;
    outputmem[3'd4]= z4;
    outputmem[3'd5]= z5;
    outputmem[3'd6]= z6;
    outputmem[3'd7]= z7;
  end
end

always @ (read_enable or select_out or serial_clock) begin
  if (read_enable == 1'b1) begin
    if (serial_clock == 1'b1) zout=outputmem[select_out];
  end
end
endmodule

// Note inpclock has a period eight times the size of the insclock.
// Input circuitry to the 1-D DCT chip to reduce the number of pads
module s_p_in11 (x, select, write_enable, serial_clock, parallel_clock, x0, x1, x2, x3, x4, x5, x6, x7);

input [10:0] x;
input [2:0] select;
input write_enable;
input serial_clock, parallel_clock;
output [10:0] x0, x1, x2, x3, x4, x5, x6, x7;

reg [10:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [10:0] inputmem [0:7];

always @ (serial_clock or select or x or write_enable) begin
  if (write_enable == 1'b1) begin
    if (serial_clock == 1'b1)
      inputmem[select]= x;
  end
end

always @ (posedge parallel_clock) begin
  x0= inputmem[3'd0];
  x1= inputmem[3'd1];
  x2= inputmem[3'd2];
  x3= inputmem[3'd3];
  x4= inputmem[3'd4];
  x5= inputmem[3'd5];
  x6= inputmem[3'd6];
  x7= inputmem[3'd7];
end
endmodule

// ibindct_rtl.v
module ibindct11 (z0,z1,z2,z3,z4,z5,z6,z7,clk,reset,x0,x1,x2,x3,x4,x5,x6,x7);
input [10:0] z0,z1,z2,z3,z4,z5,z6,z7;
input clk,reset;
output [12:0] x0,x1,x2,x3,x4,x5,x6,x7;
wire [10:0] a0,a1,a2,a4,a5;
wire [11:0] a6,a7;
wire [12:0] a3;
wire [10:0] b2;
wire [11:0] b0,b5,b6;
wire [12:0] b4,b7,b3;
wire [13:0] b1;
wire [12:0] c5,c3;
wire [10:0] d6,d5,d4;
wire [11:0] d7,d1,d2,d3;
wire [12:0] d0;
wire [12:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [12:0] x0,x1,x2,x3,x4,x5,x6,x7;
preliftadd i1 (z0,z1,z2,z3,z4,z5,z6,z7,reset,a0,a1,a2,a3,a4,a5,a6,a7);
midaddsub i2 (a0,a1,a2,a3,a4,a5,a6,a7,reset,b0,b1,b2,b3,b4,b5,b6,b7);
midiadd i3 (b3,b5,reset,c3,c5);
postadd i4 (b0,b1,b2,c3,b4,c5,b6,b7,reset,d0,d1,d2,d3,d4,d5,d6,d7);
realign i5 (d0,d1,d2,d3,d4,d5,d6,d7,reset,e0,e1,e2,e3,e4,e5,e6,e7);
always @ (posedge clk or negedge reset) begin
  if (reset) begin
    x0=0;
    x1=0;
    x2=0;
    x3=0;
    x4=0;
    x5=0;
    x6=0;
    x7=0;
  end else begin
    x0=e0;
    x1=e1;
    x2=e2;
    x3=e3;
    x4=e4;
    x5=e5;
    x6=e6;
    x7=e7;
  end
end
endmodule
module preliftadd (z0,z1,z2,z3,z4,z5,z6,z7.reset,a0,a1,a2,a3,a4,a5,a6,a7);
input [10:0] z0,z1,z2,z3,z4,z5,z6,z7;
input reset;
output [10:0] a0,a1,a2,a4,a5;
output [11:0] a6,a7;
output [12:0] a3;

wire [7:0] temp4,temp5;
wire [9:0] temp1,temp2,temp3;
wire [10:0] a4,a2, temp7,temp8;
wire [11:0] temp6;

ls12 11 1 (z0.reset,temp1);
sub11 12 ((1 b0,temp1),z4.reset,a4);
sub11 13 (z0,a4,reset,a0);
ls3811 14 (z6.reset,temp2);
sub11 15 (z2,(1 b0,temp2),reset,a2);
ls3811 16 (a2.reset,temp3);
add11 17 ((1 b0,temp3),z6.reset,a6);
ls1811 18 (z1.reset,temp4);
ls1811 19 (z5.reset,temp5);
add11 110 ((3 b000,temp4),z7.reset,a7);
add11 111 ((3 b000,temp5),z3.reset,temp6);
ls1212 112 (temp6.reset,temp7);
sub11 113 (z5,temp7.reset,a5);
ls3411 114 (a5.reset,temp8);
add12 115 ((1 b0,temp8),temp6.reset,a3);
assign a1 = z1;
endmodule

module midaddsub (a0,a1,a2,a3,a4,a5,a6,a7.reset,b0,b1,b2,b3,b4,b5,b6,b7);
input [10:0] a0,a1,a2,a4,a5;
input [11:0] a6,a7;
input [12:0] a3;
input reset;
output [10:0] b2;
output [11:0] b0,b5,b6;
output [12:0] b4,b7,b3;
output [13:0] b1;

add11 1 m1 (a0,a2,reset,b0);
sub11 2 m2 (a0,a2,reset,b2);
add12 3 ((1 b0,a4),a6,reset,b4);
sub12 4 ((1 b0,a4),a6,reset,b6);
add12 5 (a7,(1 b0,a5),reset,b7);
sub12 6 (a7,(1 b0,a5),reset,b5);
add13 7 ((2 b00,a1),a3,reset,b1);
sub13 8 ((2 b00,a1),a3,reset,b3);
endmodule

module midlift (b3,b5,reset,c3,c5);
input [12:0] b3;
input [11:0] b5;
input reset;
output [12:0] c5,c3;

101
wire [12:0] temp9,c5;
wire [11:0] temp10;

ls513 ml1 (b3,reset,temp9);
sub13 ml2 (temp9,[1'b0,b5],reset,c5);
ls513 ml3 (c5,reset,temp10);
sub13 ml4 (b3,[1'b0,temp10],reset,c3);
endmodule

module postadd (b0,b1,b2,c3,b4,c5,b6,b7,reset,d0,d1,d2,d3,d4,d5,d6,d7);
input [10:0] b2;
input [11:0] b0,b6;
input [12:0] c3,b4,c5,b7;
input [13:0] b1;
input reset;
output [10:0] d6,d5,d4;
output [11:0] d7,d3,d2,d1;
output [12:0] d0;
wire [12:0] temp16,temp15,temp14;
wire [13:0] temp17,temp11,temp12,temp13;
wire [14:0] temp10;

add14 a1 ((2'b00,b0),b1,reset,temp10);
sub14 a2 ((2'b00,b0),b1,reset,temp17);
add13 a3 (b4,c3,reset,temp11);
sub13 a4 (b4,c3,reset,temp16);
add13 a5 ([1'b0,b6),c5,reset,temp12);
sub13 a6 ([1'b0,b6),c5,reset,temp15);
add13 a7 ((2'b00,b2),b7,reset,temp13);
sub13 a8 ((2'b00,b2),b7,reset,temp14);

scale_4_15 s1 (temp10,reset,d0);
scale_4_14 s2 (temp17,reset,d7);
scale_4_14 s3 (temp11,reset,d1);
scale_4_13 s4 (temp16,reset,d6);
scale_4_14 s5 (temp12,reset,d2);
scale_4_13 s6 (temp15,reset,d5);
scale_4_14 s7 (temp13,reset,d3);
scale_4_13 s8 (temp14,reset,d4);
endmodule

module realign (d0,d1,d2,d3,d4,d5,d6,d7,reset,e0,e1,e2,e3,e4,e5,e6,e7);
input [10:0] d4,d5,d6;
input [11:0] d1,d2,d3,d7;
input [12:0] d0;
input reset;
output [12:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [12:0] e0,e1,e2,e3,e4,e5,e6,e7;

always @(d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
if ((reset) begin
  e0=0;
  e1=0;
endmodule
e2=0;
e3=0;
e4=0;
e5=0;
e6=0;
e7=0;
end
else begin
  e0=(d0);
  e1={1'b0,d1};
  e2={1'b0,d2};
  e3={1'b0,d3};
  e4={2'b00,d4};
  e5={2'b00,d5};
  e6={2'b00,d6};
  e7={1'b0,d7};
end
end
endmodule

module scale_4_13(x,reset,y);
  input [12:0] x;
  input reset;
  output [10:0] y;
  reg [10:0] y;
always @(x or reset) begin
  if (!reset) begin
    y=0;
  end
  else begin
    y=(x>>2);
  end
end
endmodule

module scale_4_14(x,reset,y);
  input [13:0] x;
  input reset;
  output [11:0] y;
  reg [11:0] y;
always @(x or reset) begin
  if (!reset) begin
    y=0;
  end
  else begin
    y=(x>>2);
  end
end
endmodule

module scale_4_15(x,reset,y);
  input [14:0] x;
  input reset;
  output [12:0] y;
reg [12:0] y;

always @(x or reset) begin
    if (!reset) begin
        y=0;
    end
    else begin
        y=(x>>2);
    end
end
endmodule

module ls1211 (x,reset,lift);
input [10:0] x;
input reset;
output [9:0] lift;

reg [9:0] lift;

always @(x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(x>>1);
    end
end
endmodule

module sub14 (x,y,reset,diff);
input [13:0] x,y;
input reset;
output [13:0] diff;

reg [13:0] diff;

always @(x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
endmodule

module ls3811 (x,reset,lift);
input [10:0] x;
input reset;
output [9:0] lift;

reg [11:0] shiftx;
wire [12:0] sumx;
reg [9:0] lift;
always @ (x or reset) begin
  shift=(1'b0,x)<<(1);
end

add12 u1 (shift,(1'b0,x),reset,sumx);

always @ (sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx>>3);
  end
end
endmodule

module add12 (x,y,reset,sum);
input [11:0] x,y;
input reset;
output [12:0] sum;
reg [12:0] sum;

always @ (x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module add15 (x,y,reset,sum);
input [14:0] x,y;
input reset;
output [15:0] sum;
reg [15:0] sum;

always @ (x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module ls1811 (x,reset,lift);
input [10:0] x;
input reset;
output [7:0] lift;
reg [7:0] lift;
always @(x or reset) begin
    if (reset) begin
        lift=0;
    end
    else begin
        lift=(x>>3);
    end
end
endmodule

module ls1212 (x,reset,lift);
    input [11:0] x;
    input reset;
    output [10:0] lift;
    reg [10:0] lift;
    always @(x or reset) begin
        if (!reset) begin
            lift=0;
        end
        else begin
            lift=(x>>1);
        end
    end
endmodule

module ls3411 (x,reset,lift);
    input [10:0] x;
    input reset;
    output [10:0] lift;
    reg [11:0] shiftx;
    wire [12:0] sumx;
    reg [10:0] lift;
    always @(x or reset) begin
        shiftx=(1'b0,x)<<1;
    end
add12 u2 (shiftx,1'b0,x,reset,sumx);

always @(sumx or reset) begin
    if (reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>2);
    end
end
endmodule

module add11 (x,y,reset,sum);
    input [10:0] x,y;
    input reset;
    always @
output [11:0] sum;

reg [11:0] sum;

always @(x or y or reset) begin
    if (reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
end
endmodule

module sub12 (x,y,reset,diff);
    input [11:0] x,y;
    input reset;
    output [11:0] diff;

    reg [11:0] diff;

    always @(x or y or reset) begin
        if (reset) begin
            diff=0;
        end
        else begin
            diff=x-y;
        end
    end
endmodule

module sub11 (x,y,reset,diff);
    input [10:0] x,y;
    input reset;
    output [10:0] diff;

    reg [10:0] diff;

    always @(x or y or reset) begin
        if (!reset) begin
            diff=0;
        end
        else begin
            diff=x-y;
        end
    end
endmodule

module ls5813 (x,reset,lift);
    input [12:0] x;
    input reset;
    output [12:0] lift;

    reg [14:0] shiftx;
    wire [15:0] sumx;
    reg [12:0] lift;
always @ (x or reset) begin
    shiftx=(2'b00.x)<<2;
end

add15 u3 (shiftx,2'b00.x,reset,sumx);

always @ (sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
endmodule

module add14 (x,y,reset,sum);
input [13:0] x,y;
input reset;
output [14:0] sum;

reg [14:0] sum;

always @ (x or y or reset) begin
    if (!reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
end
endmodule

module ls3813 (x,reset,shift);
input [12:0] x;
input reset;
output [11:0] shift;

reg [13:0] shiftx;
wire [14:0] sumx;
reg [11:0] lift;

always @ (x or reset) begin
    shiftx=(1'b0.x)<<1;
end

add14 u4 (shiftx,1'b0.x,reset,sumx);

always @ (sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
end
endmodule

module add13 (x,y,reset,sum);
input [12:0] x,y;
input reset;
output [13:0] sum;

reg [13:0] sum;

always @ (x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module sub13 (x,y,reset,diff);
input [12:0] x,y;
input reset;
output [12:0] diff;

reg [12:0] diff;

always @ (x or y or reset) begin
  if (!reset) begin
    diff=0;
  end
  else begin
    diff=x-y;
  end
end
endmodule

B.7 Stimulus for one-dimensional 8-point inverse binDCT RTL description

// stimulus_inversedct_rtl.v
// Testbench for inverse dct module
// Created by: Tracy Franklin
	imescale 1ns/10ps
module stimulus_inversedct;

reg in_write_enable, out_read_enable, reset;
reg in_clock, bindct_clock, out_clock;
reg [2:0] in_select, out_select;
reg [10:0] x;
wire [12:0] y0, y1, y2, y3, y4, y5, y6, y7;
wire [12:0] y;

// rtl-level
inversonedct rtl_test (x, in_select, in_write_enable, in_clock, bindct_clock, reset, out_select, out_read_enable, out_clock, y0, y1, y2, y3, y4, y5, y6, y7);
// gate-level
// inversedct chip_test (x[10],x[9],x[8],x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0], in_select[2], in_select[1],
in_select[0], in_write_enable, in_clock, bindct_clock, reset, out_select[2], out_select[1], out_select[0],
out_read_enable, out_clock, y[12], y[11], y[10], y[9], y[8], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);

initial
begin
  // Initial conditions
  in_clock=0;
  bindct_clock=0;
  out_clock=0;
  in_write_enable=1'b1;
  out_read_enable=1'b1;
  #400 $finish;
end

always #2 in_clock= !in_clock;
always #40 bindct_clock = !bindct_clock;
always #2 out_clock= !out_clock;

initial begin
  reset=1'b0;
  #3 reset=1'b1;
end

initial
begin

/*
 #4 in_select=3'b000; x=10'd1;
 #4 in_select=3'b001; x=10'd1;
 #4 in_select=3'b010; x=10'd1;
 #4 in_select=3'b011; x=10'd1;
 #4 in_select=3'b100; x=10'd0;
 #4 in_select=3'b101; x=10'd0;
 #4 in_select=3'b110; x=10'd0;
 #4 in_select=3'b111; x=10'd0;
 #40 in_select=3'b000; x=10'd8;
 #4 in_select=3'b001; x=10'd0;
 #4 in_select=3'b010; x=10'd0;
 #4 in_select=3'b011; x=10'd0;
 #4 in_select=3'b100; x=10'd0;
 #4 in_select=3'b101; x=10'd0;
 #4 in_select=3'b110; x=10'd0;
 #4 in_select=3'b111; x=10'd0;
*/
 #4 in_select=3'b000; x=10'd1020;
 #4 in_select=3'b001; x=10'd605;
 #4 in_select=3'b010; x=10'd0;
 #4 in_select=3'b011; x=10'd430;
 #4 in_select=3'b100; x=10'd0;
 #4 in_select=3'b101; x=10'd540;
 #4 in_select=3'b110; x=10'd0;
 #4 in_select=3'b111; x=10'd655;

110
// monitor the outputs

initial
begin
  $display("Time=", "x="', y0', y1', y2', y3', y4', y5', y6', y7', "\n");
end

always @(posedge in_clock) begin
  $monitor("%0d, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", ttime, x, y0, y1, y2, y3, y4, y5, y6, y7);
end

/*
always @(posedge bindct_clock) begin
  wait (out_read_enable) out_select=3'd0; $display("%0d, select=%d, y=%b", ttime, out_select, y);
end
*/
endmodule

B.8 Stimulus for one-dimensional 8-point inverse binDCT chip RTL description

// stimulus_inversebindctchip.v
// Testbench for inverse binDCT module
// Created by: Tracy Franklin
	timescale 1ns/10ps

module stimulus_inversebindctchip;

reg in_write_enable;
reg out_read_enable;
reg reset;
reg in_clock;
reg bindct_clock;
reg out_clock;
reg [2:0] in_select;
reg [2:0] out_select;
reg [10:0] x;
wire [12:0] y;

// rtl-level
inversebindctchip rtl_test (x, in_select, in_write_enable, in_clock, bindct_clock, reset, out_select, out_read_enable, out_clock, y);

// gate-level
// inversebindctchip chip_test (x[10], x[9], x[8], x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0], in_select[2], in_select[1], in_select[0], in_write_enable, in_clock, bindct_clock, reset, out_select[2], out_select[1], out_select[0], out_read_enable, out_clock, y[12], y[11], y[10], y[9], y[8], y[7], y[6], y[5], y[4], y[3], y[2], y[1], y[0]);

initial
begin
  // Initial conditions
  in_clock=0;
  bindct_clock=0;
  out_clock=0;
  in_write_enable=1'b1;
  out_read_enable=1'b1;
  #400 $finish;
end

always #2 in_clock= ~in_clock;
always #40 bindct_clock = ~bindct_clock;
always #2 out_clock= ~out_clock;

initial begin
  reset=1'b0;
  #3 reset=1'b1;
end

initial begin
  #4 in_select=3'b000; x=10'd1;
  #4 in_select=3'b001; x=10'd1;
  #4 in_select=3'b010; x=10'd1;
  #4 in_select=3'b011; x=10'd1;
  #4 in_select=3'b100; x=10'd0;
  #4 in_select=3'b101; x=10'd0;
  #4 in_select=3'b110; x=10'd0;
  #4 in_select=3'b111; x=10'd0;
  
  #40 in_select=3'b000; x=10'd8;
  #4 in_select=3'b001; x=10'd0;
  #4 in_select=3'b010; x=10'd0;
  #4 in_select=3'b011; x=10'd0;
  #4 in_select=3'b100; x=10'd0;
  #4 in_select=3'b101; x=10'd0;
  #4 in_select=3'b110; x=10'd0;
  #4 in_select=3'b111; x=10'd0;
  
  #52 in_select=3'b000; x=10'd4;
  #4 in_select=3'b001; x=10'd2;
  #4 in_select=3'b010; x=10'd0;
  #4 in_select=3'b011; x=10'd458;
  #4 in_select=3'b100; x=10'd0;
  #4 in_select=3'b101; x=10'd246;
  #4 in_select=3'b110; x=10'd0;
  #4 in_select=3'b111; x=10'd512;
end

// monitor the outputs
always @(posedge bindct_clock) begin
  wait (out_read_enable) out_select=3'd7; $display("%0d\t select=%d, \t y=%d",$time, out_select, y);
end

endmodule

112
Appendix C

Verilog-XL Simulation Results

C.1 The Verilog-XL simulation results for forwarddct_rtl.v design.

<table>
<thead>
<tr>
<th>Time</th>
<th>Y0</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>Y5</th>
<th>Y6</th>
<th>Y7</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>280</td>
<td>508</td>
<td>301</td>
<td>0</td>
<td>216</td>
<td>0</td>
<td>270</td>
<td>0</td>
<td>583</td>
</tr>
</tbody>
</table>

C.2 The Verilog-XL results of the inversedct_rtl.v design.

<table>
<thead>
<tr>
<th>Time</th>
<th>Y0</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>Y5</th>
<th>Y6</th>
<th>Y7</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>228</td>
<td>1665</td>
<td>1216</td>
<td>255</td>
<td>1920</td>
<td>959</td>
<td>509</td>
<td>3994</td>
</tr>
<tr>
<td>360</td>
<td>123</td>
<td>1523</td>
<td>1076</td>
<td>129</td>
<td>1920</td>
<td>972</td>
<td>525</td>
<td>3373</td>
</tr>
</tbody>
</table>

C.3 The Verilog-XL results of the forwardbindctchip_rtl.v design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>Time</td>
<td>Data Out (out selection = 0)</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>280</td>
<td>8</td>
</tr>
<tr>
<td>360</td>
<td>1020</td>
</tr>
<tr>
<td>440</td>
<td>4</td>
</tr>
</tbody>
</table>

C.4 The Verilog-XL results of the inversebindctchip_rtl.v design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>1</td>
</tr>
<tr>
<td>360</td>
<td>123</td>
</tr>
</tbody>
</table>

C.5 The Verilog-XL results of the forwardbindctchip_gates.v design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>8</td>
</tr>
<tr>
<td>360</td>
<td>4</td>
</tr>
<tr>
<td>440</td>
<td>4</td>
</tr>
</tbody>
</table>
### C.6 The Verilog-XL results of the `inversebindctchip_gates_1_3.v` design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>1</td>
</tr>
<tr>
<td>360</td>
<td>123</td>
</tr>
</tbody>
</table>

### C.7 The Verilog-XL results of the `forwardbindctchip_gold1.v` design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>280</td>
<td>8</td>
</tr>
<tr>
<td>360</td>
<td>1020</td>
</tr>
<tr>
<td>440</td>
<td>4</td>
</tr>
</tbody>
</table>

### C.8 The Verilog-XL results of the `inversebindctchip_gold1.v` design

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Out (out selection = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td>120</td>
<td>X</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>280</td>
<td>1</td>
</tr>
</tbody>
</table>
C.9 The Verilog-XL results of the bindct2d_rtl.v design

The time is 20; the output data is,

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The time is 60; the output data is,

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
The time is 100; the output data is,

<table>
<thead>
<tr>
<th>2040</th>
<th>454</th>
<th>0</th>
<th>415</th>
<th>0</th>
<th>591</th>
<th>0</th>
<th>619</th>
</tr>
</thead>
<tbody>
<tr>
<td>510</td>
<td>87</td>
<td>0</td>
<td>220</td>
<td>0</td>
<td>660</td>
<td>0</td>
<td>620</td>
</tr>
<tr>
<td>0</td>
<td>1042</td>
<td>0</td>
<td>1095</td>
<td>0</td>
<td>1081</td>
<td>0</td>
<td>1097</td>
</tr>
<tr>
<td>478</td>
<td>503</td>
<td>0</td>
<td>1001</td>
<td>0</td>
<td>391</td>
<td>0</td>
<td>631</td>
</tr>
<tr>
<td>0</td>
<td>51</td>
<td>0</td>
<td>2018</td>
<td>0</td>
<td>2019</td>
<td>0</td>
<td>1926</td>
</tr>
<tr>
<td>255</td>
<td>303</td>
<td>0</td>
<td>330</td>
<td>0</td>
<td>293</td>
<td>0</td>
<td>583</td>
</tr>
<tr>
<td>0</td>
<td>791</td>
<td>0</td>
<td>666</td>
<td>0</td>
<td>864</td>
<td>0</td>
<td>696</td>
</tr>
<tr>
<td>1984</td>
<td>1027</td>
<td>0</td>
<td>749</td>
<td>0</td>
<td>867</td>
<td>0</td>
<td>994</td>
</tr>
</tbody>
</table>

The time is 140; the output data is,

<table>
<thead>
<tr>
<th>4080</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Appendix D

Pearl Analysis Results

The Pearl timing results for the forward and inverse 1-D binDCT chip designs are given below.

D.1 Pearl Analysis Result from Design Planner for the 1-D Forward binDCT

```
forwardbindctchip_comp.cmd> FindSDFPathConstraints
1: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_2__1_ID v
2: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_4__0_ID v
3: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_6__4_ID v
4: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_0__6_ID v
5: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_3__5_ID v
6: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_5__7_ID v
7: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_0__5_ID v
8: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_1__3_ID v
9: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_4__2_ID v
10: SDF max path slack 0.17ns forwardct_s_p_in8_inputmem_reg_4__3_ID v
forwardbindctchip_comp.cmd> # end of DPPearl-created commands

cmd> findpathsfrom in_clock
1: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__4__U3IB1 v
2: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__4__U4IA2 v
3: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__6__U4IA2 v
4: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__6__U3IB1 v
5: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__7__U3IB1 v
6: 14.43ns Path to forwardct_s_p_in8_inputmem_reg_2__7__U4IA2 v
7: 13.89ns Path to forwardct_s_p_in8_x2_reg_4_ID v
8: 13.89ns Path to forwardct_s_p_in8_x2_reg_6_ID v
9: 13.89ns Path to forwardct_s_p_in8_x2_reg_7_ID v
10: 13.60ns Path to forwardct_s_p_in8_inputmem_reg_2__4__IR v

cmd> findpathsfrom out_clock
1: 12.83ns Path to forwardct_p_s_out11_zout_reg_0_IE v
2: 12.83ns Path to forwardct_p_s_out11_zout_reg_2_IE v
3: 12.83ns Path to forwardct_p_s_out11_zout_reg_1_IE v
4: 12.83ns Path to forwardct_p_s_out11_zout_reg_5_IE v
5: 12.83ns Path to forwardct_p_s_out11_zout_reg_3_IE v
6: 12.83ns Path to forwardct_p_s_out11_zout_reg_4_IE v
7: 12.82ns Path to forwardct_p_s_out11_zout_reg_6_IE v
8: 12.82ns Path to forwardct_p_s_out11_zout_reg_10_IE v
9: 12.82ns Path to forwardct_p_s_out11_zout_reg_8_IE v
10: 12.82ns Path to forwardct_p_s_out11_zout_reg_7_IE v

cmd> findpathsfrom bindct_clock
1: 100.74ns Path to forwardct_bindct8_z1_reg_1_ICP v
2: 100.74ns Path to forwardct_p_s_out11_outputmem_reg_1__1_IE v
3: 100.74ns Path to forwardct_p_s_out11_outputmem_reg_1__2_IE v
4: 100.74ns Path to forwardct_bindct8_z1_reg_2_ICP v
5: 100.74ns Path to forwardct_s_p_in8_x0_reg_4_ICP v
```
6: 100.74ns Path to forwardct_s_p_in8_x0_reg_6_ICP v
7: 100.74ns Path to forwardct_s_p_in8_x7_reg_4_ICP v
8: 100.74ns Path to forwardct_s_p_in8_x7_reg_7_ICP v
9: 100.74ns Path to forwardct_s_p_in8_x7_reg_6_ICP v
10: 100.74ns Path to forwardct_s_p_in8_x4_reg_5_ICP v

cmd> checklimits
No limit violations were found

cmd> timing verify
1: Setup constraint slack 8.93ns forwardct_s_p_out11_zout_reg_1_ (LH1N D v -> E v)
2: Setup constraint slack 8.96ns forwardct_s_p_out11_zout_reg_2_ (LH1N D v -> E v)
3: Setup constraint slack 9.11ns forwardct_s_p_out11_zout_reg_3_ (LH1N D v -> E v)
4: Setup constraint slack 9.12ns forwardct_s_p_out11_zout_reg_7_ (LH1N D v -> E v)
5: Setup constraint slack 9.13ns forwardct_s_p_out11_zout_reg_6_ (LH1N D v -> E v)
6: Setup constraint slack 9.13ns forwardct_s_p_out11_zout_reg_4_ (LH1N D v -> E v)
7: Setup constraint slack 9.14ns forwardct_s_p_out11_zout_reg_8_ (LH1N D v -> E v)
8: Setup constraint slack 9.14ns forwardct_s_p_out11_zout_reg_9_ (LH1N D v -> E v)
9: Setup constraint slack 9.15ns forwardct_s_p_out11_zout_reg_1_ (LH1N D^ -> E v)
10: Setup constraint slack 9.16ns forwardct_s_p_out11_zout_reg_10_ (LH1N D v -> E v)
11: Hold constraint violation 0.10ns forwardct_s_p_in8_inputmem_reg_2_6_ (NR1I S v -> R v)
12: Hold constraint violation 0.10ns forwardct_s_p_in8_inputmem_reg_2_4_ (NR1I S v -> R v)
13: Hold constraint violation 0.10ns forwardct_s_p_in8_inputmem_reg_2_7_ (NR1I S v -> R v)
14: Hold constraint slack 0.10ns forwardct_s_p_in8_inputmem_reg_2_7_ (NR1I R v -> S v)
15: Hold constraint slack 0.10ns forwardct_s_p_in8_inputmem_reg_2_4_ (NR1I R v -> S v)
16: Hold constraint slack 0.10ns forwardct_s_p_in8_inputmem_reg_2_6_ (NR1I R v -> S v)
17: Hold constraint slack 0.52ns forwardct_s_p_in8_x0_reg_4_ (DFF1N D v -> CP^)
18: Hold constraint slack 0.53ns forwardct_s_p_in8_x0_reg_6_ (DFF1N D v -> CP^)
19: Hold constraint slack 0.53ns forwardct_s_p_in8_x4_reg_3_ (DFF1N D v -> CP^)
20: Hold constraint slack 0.53ns forwardct_s_p_in8_x7_reg_7_ (DFF1N D v -> CP^)
21: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_6_ (LH2Q E high)
22: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_4_ (LH2Q E high)
23: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_3_ (LH2Q E high)
24: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_5_ (LH2Q E high)
25: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_7_ (LH2Q E high)
26: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_6_ (LH2Q E high)
27: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_4_ (LH2Q E high)
28: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_3_ (LH2Q E high)
29: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_5_ (LH2Q E high)
30: Width constraint slack 11.53ns forwardct_s_p_in8_inputmem_reg_0_7_ (LH2Q E high)
31: Gated clock constraint violation 1.63ns forwardct_s_p_in8_inputmem_reg_2_6_U4
32: Gated clock constraint violation 1.63ns forwardct_s_p_in8_inputmem_reg_2_6_U3
33: Gated clock constraint violation 1.63ns forwardct_s_p_in8_inputmem_reg_2_4_U3
34: Gated clock constraint violation 1.63ns forwardct_s_p_in8_inputmem_reg_2_4_U4
35: Gated clock constraint violation 1.62ns forwardct_s_p_in8_inputmem_reg_2_7_U3
36: Gated clock constraint violation 1.62ns forwardct_s_p_in8_inputmem_reg_2_7_U4
37: Gated clock constraint violation 1.22ns forwardct_s_p_in8_inputmem_reg_2_4_U3
38: Gated clock constraint violation 1.22ns forwardct_s_p_in8_inputmem_reg_2_4_U4
39: Gated clock constraint violation 1.22ns forwardct_s_p_in8_inputmem_reg_2_6_U4
40: Gated clock constraint violation 1.22ns forwardct_s_p_in8_inputmem_reg_2_6_U3
D.2 Pearl Analysis Result from Design Planner for the 1-D Inverse binDCT

inversebindctchip_comp.cmd> FindSDFPathConstraints
1: SDF max path slack 0.25ns inversedct_s_p_in11linputmem_reg_4__9_ID v
2: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_6__7_ID v
3: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_4__0_ID v
4: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_5__10_ID v
5: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_5__7_ID v
6: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_2__1_ID v
7: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_3__4_ID v
8: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_2__3_ID v
9: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_4__10_ID v
10: SDF max path slack 0.26ns inversedct_s_p_in11linputmem_reg_0__8_ID v
inversebindctchip_comp.cmd> # end of DPPearl-created commands

cmd> findpathsfrom out_clock
1: 5.13ns Path to inversedct_p_s_out13tizout_reg_12_ID v
2: 5.13ns Path to inversedct_p_s_out13tizout_reg_10_ID v
3: 5.13ns Path to inversedct_p_s_out13tizout_reg_11_ID v
4: 5.13ns Path to inversedct_p_s_out13tizout_reg_3_ID v
5: 5.13ns Path to inversedct_p_s_out13tizout_reg_2_ID v
6: 5.13ns Path to inversedct_p_s_out13tizout_reg_0_ID v
7: 5.13ns Path to inversedct_p_s_out13tizout_reg_9_ID v
8: 5.13ns Path to inversedct_p_s_out13tizout_reg_7_ID v
9: 5.13ns Path to inversedct_p_s_out13tizout_reg_8_ID v
10: 5.13ns Path to inversedct_p_s_out13tizout_reg_5_ID v

cmd> findpathsfrom in_clock
1: 6.43ns Path to inversedct_s_p_in11linputmem_reg_5__9_ID v
2: 6.43ns Path to inversedct_s_p_in11linputmem_reg_5__9_ID v
3: 6.43ns Path to inversedct_s_p_in11linputmem_reg_7__0_ID v
4: 6.43ns Path to inversedct_s_p_in11linputmem_reg_7__0_ID v
5: 6.43ns Path to inversedct_s_p_in11linputmem_reg_1__6_ID v
6: 6.43ns Path to inversedct_s_p_in11linputmem_reg_1__6_ID v
7: 6.43ns Path to inversedct_s_p_in11linputmem_reg_4__8_ID v
8: 6.43ns Path to inversedct_s_p_in11linputmem_reg_4__8_ID v
9: 6.43ns Path to inversedct_s_p_in11linputmem_reg_1__2_ID v
10: 6.43ns Path to inversedct_s_p_in11linputmem_reg_1__2_ID v

cmd> checklimits
No limit violations were found

D.3 Pearl Analysis Result from Silicon Ensemble for the 1-D Forward binDCT

pearl.tmpcmd> checktiming

pearl.tmpcmd> checkLimits
1: Slow violation 0.09ns (error) forwardct_U425/ZN

pearl.tmpcmd> TimingVerify -check
setup,hold,gatedclock,recovery,removal,nochange,setup,nochange,hold,period,width,loop -max_slack 0
1: Hold constraint violation 0.10ns forwardct_s_p_in8_inputmem_reg_2__6_ (NRL 1 s v -> R v)
2: Hold constraint violation 0.10ns forwardct_s_p_in8_inputmem_reg_2__4_ (NRL 1 s v -> R v)
3: Hold constraint violation 0.09ns forwardct_s_p_in8_inputmem_reg_2_7_ (NRL1 S v -> R v)
4: Gated clock constraint violation 1.68ns forwardct_s_p_in8_inputmem_reg_2_6_U3
5: Gated clock constraint violation 1.68ns forwardct_s_p_in8_inputmem_reg_2_6_U4
6: Gated clock constraint violation 1.68ns forwardct_s_p_in8_inputmem_reg_2_4_U4
7: Gated clock constraint violation 1.68ns forwardct_s_p_in8_inputmem_reg_2_4_U3
8: Gated clock constraint violation 1.67ns forwardct_s_p_in8_inputmem_reg_2_7_U3
9: Gated clock constraint violation 1.66ns forwardct_s_p_in8_inputmem_reg_2_7_U4
10: Gated clock constraint violation 1.27ns forwardct_s_p_in8_inputmem_reg_2_6_U4
11: Gated clock constraint violation 1.27ns forwardct_s_p_in8_inputmem_reg_2_6_U3
12: Gated clock constraint violation 1.27ns forwardct_s_p_in8_inputmem_reg_2_4_U3
13: Gated clock constraint violation 1.27ns forwardct_s_p_in8_inputmem_reg_2_4_U4
14: Gated clock constraint violation 1.26ns forwardct_s_p_in8_inputmem_reg_2_7_U3
15: Gated clock constraint violation 1.26ns forwardct_s_p_in8_inputmem_reg_2_7_U4

```sh
$ cd>
findpaths from bindct_clock
1: 26.68ns Path to forwardct_bindct8_z3_reg_8_/D v
2: 26.66ns Path to forwardct_bindct8_z3_reg_8_/D ^
3: 26.26ns Path to forwardct_bindct8_z3_reg_7_/D v
4: 26.04ns Path to forwardct_bindct8_z3_reg_7_/D ^
5: 25.60ns Path to forwardct_bindct8_z3_reg_6_/EN v
6: 25.58ns Path to forwardct_bindct8_z3_reg_6_/EN ^
7: 25.31ns Path to forwardct_bindct8_z3_reg_5_/D ^
8: 25.10ns Path to forwardct_bindct8_z3_reg_5_/D v
9: 23.91ns Path to forwardct_bindct8_z3_reg_4_/D ^
10: 23.71ns Path to forwardct_bindct8_z3_reg_4_/D v

$ cd>
findpaths from out_clock
1: 4.08ns Path to y_9_ ^
2: 3.96ns Path to y_10_ ^
3: 3.94ns Path to y_1_ ^
4: 3.94ns Path to y_0_ ^
5: 3.92ns Path to y_8_ ^
6: 3.92ns Path to y_7_ ^
7: 3.82ns Path to y_2_ ^
8: 3.70ns Path to y_3_ ^
9: 3.70ns Path to y_9_ v
10: 3.69ns Path to y_6_ ^

$ cd>
findpaths from in_clock
1: 14.59ns Path to forwardct_s_p_in8_inputmem_reg_2_4_/U3/B1 v
2: 14.59ns Path to forwardct_s_p_in8_inputmem_reg_2_4_/U4/A2 v
3: 14.59ns Path to forwardct_s_p_in8_inputmem_reg_2_6_/U3/B1 v
4: 14.59ns Path to forwardct_s_p_in8_inputmem_reg_2_6_/U4/A2 v
5: 14.57ns Path to forwardct_s_p_in8_inputmem_reg_2_7_/U3/B1 v
6: 14.57ns Path to forwardct_s_p_in8_inputmem_reg_2_7_/U4/A2 v
7: 14.04ns Path to forwardct_s_p_in8_x2_reg_4_/D v
8: 14.04ns Path to forwardct_s_p_in8_x2_reg_6_/D v
9: 14.03ns Path to forwardct_s_p_in8_x2_reg_7_/D v
10: 13.74ns Path to forwardct_s_p_in8_inputmem_reg_2_4_/R v

$ cd>
checklimit
1: Slew violation 0.09ns (error) forwardct_U425/ZN ^

$ cd>
ShowPossibility 1
Possibility 1:
Slew violation 0.09ns (error) forwardct_U425/ZN ^
```

121
D.4 Pearl Analysis Result from Silicon Ensemble for the 1-D Inverse binDCT

```
pearl.tmpcmd> checkLimits
No limit violations were found
pearl.tmpcmd> TimingVerify -check
setup,hold,gatedclock,recovery,removal,nochangesetup,nochangehold,period,width,loop -max_slack 0
1: Hold constraint violation 0.10ns inversesect_s_p_in11/inputmem_reg_1__2__ (NRL1 S v -> R v)
2: Hold constraint violation 0.10ns inversesect_s_p_in11/inputmem_reg_7__0__ (NRL1 S v -> R v)
3: Hold constraint violation 0.10ns inversesect_s_p_in11/inputmem_reg_5__9__ (NRL1 S v -> R v)
4: Hold constraint violation 0.10ns inversesect_s_p_in11/inputmem_reg_4__8__ (NRL1 S v -> R v)
5: Hold constraint violation 0.10ns inversesect_s_p_in11/inputmem_reg_1__6__ (NRL1 S v -> R v)
6: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_7__0__U3
7: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_7__0__U4
8: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_1__2__U3
9: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_5__9__U3
10: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_5__9__U4
11: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_4__8__U4
12: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_4__8__U3
13: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_1__2__U4
14: Gated clock constraint violation 1.69ns inversesect_s_p_in11/inputmem_reg_1__6__U4
15: Gated clock constraint violation 1.68ns inversesect_s_p_in11/inputmem_reg_1__6__U3
16: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_7__0__U3
17: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_7__0__U4
18: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_4__8__U4
19: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_4__8__U3
20: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_5__9__U3
21: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_5__9__U4
22: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_1__2__U3
23: Gated clock constraint violation 1.27ns inversesect_s_p_in11/inputmem_reg_1__6__U4
24: Gated clock constraint violation 1.26ns inversesect_s_p_in11/inputmem_reg_1__6__U3
25: Gated clock constraint violation 1.26ns inversesect_s_p_in11/inputmem_reg_1__2__U4
```

```
cmd> findpathsfrom in_clock
1: 7.15ns Path to inversesect_s_p_in11/inputmem_reg_7__0__U3/B1 v
2: 7.15ns Path to inversesect_s_p_in11/inputmem_reg_7__0__U4/A2 v
3: 7.15ns Path to inversesect_s_p_in11/inputmem_reg_5__9__U3/B1 v
4: 7.15ns Path to inversesect_s_p_in11/inputmem_reg_5__9__U4/A2 v
5: 7.13ns Path to inversesect_s_p_in11/inputmem_reg_1__6__U4/A2 v
6: 7.13ns Path to inversesect_s_p_in11/inputmem_reg_1__6__U3/B1 v
7: 7.12ns Path to inversesect_s_p_in11/inputmem_reg_4__8__U3/B1 v
8: 7.12ns Path to inversesect_s_p_in11/inputmem_reg_4__8__U4/A2 v
9: 7.11ns Path to inversesect_s_p_in11/inputmem_reg_1__2__U3/B1 v
10: 7.11ns Path to inversesect_s_p_in11/inputmem_reg_1__2__U4/A2 v
```

```
cmd> findpathsfrom bindect_clock
1: 101.64ns Path to inversesect_ibindectl1/x1_reg_8_/CP v
2: 101.64ns Path to inversesect_s_p_in11/x0_reg_9_/CP v
```
3: 101.64ns Path to inversedct_ibindct11/x2_reg_2_/CP
4: 101.64ns Path to inversedct_s_p_in11/x2_reg_9_/CP
5: 101.64ns Path to inversedct_s_p_in11/x4_reg_4_/CP
6: 101.64ns Path to inversedct_s_p_in11/x6_reg_7_/CP
7: 101.64ns Path to inversedct_s_p_in11/x0_reg_8_/CP
8: 101.64ns Path to inversedct_s_p_in11/x0_reg_4_/CP
9: 101.64ns Path to inversedct_ibindct11/x6_reg_12_/CP
10: 101.64ns Path to inversedct_s_p_in11/x6_reg_9_/CP

```
cmd> findpathsfrom out_clock
1: 5.25ns Path to inversedct_p_s_out13/zout_reg_12_/EN
2: 5.25ns Path to inversedct_p_s_out13/zout_reg_10_/EN
3: 5.25ns Path to inversedct_p_s_out13/zout_reg_11_/EN
4: 5.25ns Path to inversedct_p_s_out13/zout_reg_3_/EN
5: 5.25ns Path to inversedct_p_s_out13/zout_reg_2_/EN
6: 5.25ns Path to inversedct_p_s_out13/zout_reg_0_/EN
7: 5.25ns Path to inversedct_p_s_out13/zout_reg_9_/EN
8: 5.25ns Path to inversedct_p_s_out13/zout_reg_8_/EN
9: 5.25ns Path to inversedct_p_s_out13/zout_reg_5_/EN
10: 5.25ns Path to inversedct_p_s_out13/zout_reg_7_/EN
```

```
cmd> checklimit
No limit violations were found
```
Appendix E

LVS Results

E.1 LVS result for the forward one-dimensional binDCT chip

Begin netlist: May 24 20:41:02 2000
  view name list = ("auLvs" "macrovlvs" "extracted" "netlist" "schematic")
  stop name list = ("auLvs" "macrovlvs")
  library name = "fdesign"
  cell name = "forwardbindictchip"
  view name = "extracted3"
  globals lib = "basic"
Running Artist Flat Netlisting ...

Moving original netlist to extNetlist
Removing parasitic components from netlist
  presistors removed: 0
  pcapacitors removed: 0
  pinductors removed: 0
  pdiodes removed: 0
  trans lines removed: 0
  1801 nodes merged into 1801 nodes

  view name list = ("auLvs" "macrovlvs" "netlist" "schematic" "extracted")
  stop name list = ("auLvs" "macrovlvs")
  library name = "fdesign"
  cell name = "forwardbindictchip"
  view name = "schematic"
  globals lib = "basic"
Running Artist Flat Netlisting ...
End netlist: May 24 20:42:53 2000

Moving original netlist to extNetlist
Removing parasitic components from netlist
  presistors removed: 0
  pcapacitors removed: 0
  pinductors removed: 0
  pdiodes removed: 0
  trans lines removed: 0
  1795 nodes merged into 1795 nodes

Running netlist comparison program: LVS
Begin comparison: May 24 20:42:54 2000
@(#)CDS: LVS version 4.4.3 08/25/1999 22:10 (cds230) $

The net-lists match.
<table>
<thead>
<tr>
<th>Layout/Schematic</th>
<th>Instances</th>
<th>Nets</th>
<th>Terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Un-matched</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rewired</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Size errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pruned</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Active</td>
<td>1559</td>
<td>1559</td>
<td>1793</td>
</tr>
<tr>
<td>Total</td>
<td>1559</td>
<td>1559</td>
<td>1793</td>
</tr>
</tbody>
</table>

End comparison: May 24 20:43:00 2000

Comparison program completed successfully.
Figure 20. The extracted view of the forward 1-D binDCT chip.
E.2 LVS result for the inverse one-dimensional binDCT chip

Begin netlist:  May 24 20:14:38 2000
view name list  = ("auLvs" "macroLvs" "extracted" "netlist" "schematic")
stop name list = ("auLvs" "macroLvs")
library name   = "idesign"
cell name      = "inversebindctchip"
view name      = "extracted1"
globals lib    = "basic"
Running Artist Flat Netlisting ...
End netlist:    May 24 20:16:33 2000

Moving original netlist to extNetlist
Removing parasitic components from netlist
  presistors removed: 0
  pcapacitors removed: 0
  pinductors removed: 0
  pdiodes removed:  0
  trans lines removed: 0
  2874 nodes merged into 2874 nodes

Begin netlist:  May 24 20:16:41 2000
view name list  = ("auLvs" "macroLvs" "netlist" "schematic" "extracted")
stop name list = ("auLvs" "macroLvs")
library name   = "idesign"
cell name      = "inversebindctchip"
view name      = "schematic"
globals lib    = "basic"
Running Artist Flat Netlisting ...
End netlist:    May 24 20:17:14 2000

Moving original netlist to extNetlist
Removing parasitic components from netlist
  presistors removed: 0
  pcapacitors removed: 0
  pinductors removed: 0
  pdiodes removed:  0
  trans lines removed: 0
  2868 nodes merged into 2868 nodes

Running netlist comparison program: LVS
Begin comparison: May 24 20:17:16 2000
@(#)SCDS: LVS version 4.4.3 08/25/1999 22:10 (cds230) $

The net-lists match.
<table>
<thead>
<tr>
<th></th>
<th>instances</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>un-matched</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>rewired</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>size errors</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pruned</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>active</td>
<td>2507</td>
<td>2507</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>2507</td>
<td>2507</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>nets</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>un-matched</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>merged</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pruned</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>active</td>
<td>2866</td>
<td>2866</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>2866</td>
<td>2866</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>terminals</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>un-matched</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>matched but</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>different type</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>0</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>

End comparison:      May 24 20:17:22 2000

Comparison program completed successfully.
Figure 21. The inverse one-dimensional binDCT chip.
Appendix F

The binDCT Clock Tree Generation Results from Design Planner

F.1 Forward one-dimensional binDCT chip clock tree initial analysis

Report: /rpt/initial.analysis
Design: forwardbindctchip

Clock tree root: pbindct_clock C

Min path

Root level: PDI
Component input: pbindct_clock C (x = 364.550, y = 784.200)
Component output: pbindct_clock C (x = 364.550, y = 784.200)
Next input: forwardct_p_s_out11_outputmem_reg_7__4__ E (x = 686.700, y = 1291.250)

Delay from root to next input: 0.318 ns
Delay from component input to next input: 0.318 ns
Delay from component input to component output: 0.000 ns
Delay from component output to next input: 0.318 ns
Transition time at next input: 0.679 ns
Total load at component output: 3.037 pF

Max path

Root level: PDI
Component input: pbindct_clock C (x = 364.550, y = 784.200)
Component output: pbindct_clock C (x = 364.550, y = 784.200)
Next input: forwardct_bindct8_z1_reg_1__ CP (x = 774.900, y = 1773.750)

Delay from root to next input: 0.845 ns
Delay from component input to next input: 0.845 ns
Delay from component input to component output: 0.000 ns
Delay from component output to next input: 0.845 ns
Transition time at next input: 1.875 ns
Total load at component output: 3.037 pF

F.2 Forward one-dimensional binDCT chip clock tree initial timing

Report: /rpt/initial.timing
Design: forwardbindctchip

Clock tree root: pbindct_clock C
Timing start pin: pbindct_clock C

Max. transition time at leaf pins: 1.875 ns
Min. insertion delay to leaf pins: 0.318 ns
Max. insertion delay to leaf pins: 0.845 ns
Max. skew between leaf pins: 0.527 ns
F.3 Forward one-dimensional binDCT chip clock tree final analysis

Report: /rpt/final_analysis
Design: forwardbindctchip

Clock tree root: pbindct_clock C

Min path

Root level: PDI
Component input: pbindct_clock C (x = 364.550, y = 784.200)
Component output: pbindct_clock C (x = 364.550, y = 784.200)
Next input: pbindct_clock_L1_L1_I (x = 912.100, y = 1401.250)

Delay from root to next input: 0.058 ns
Delay from component input to next input: 0.058 ns
Delay from component input to component output: 0.000 ns
Delay from component output to next input: 0.058 ns
Transition time at next input: 0.232 ns
Total load at component output: 0.369 pF

Level 1: BUF6
Component input: pbindct_clock_L1_L1_I (x = 912.100, y = 1401.250)
Component output: pbindct_clock_L1_L1_Z (x = 917.700, y = 1402.500)
Next input: pbindct_clock_L2_L2_I (x = 797.300, y = 1401.250)

Delay from root to next input: 0.457 ns
Delay from component input to next input: 0.399 ns
Delay from component input to component output: 0.398 ns
Delay from component output to next input: 0.001 ns
Transition time at next input: 0.109 ns
Total load at component output: 0.121 pF

Level 2: BUF6
Component input: pbindct_clock_L2_L2_I (x = 797.300, y = 1401.250)
Component output: pbindct_clock_L2_L2_Z (x = 791.700, y = 1402.500)
Next input: pbindct_clock_L3_L1_I (x = 786.100, y = 1415.000)

Delay from root to next input: 0.847 ns
Delay from component input to next input: 0.390 ns
Delay from component input to component output: 0.389 ns
Delay from component output to next input: 0.001 ns
Transition time at next input: 0.133 ns
Total load at component output: 0.187 pF

Level 3: BUF6
Component input: pbindct_clock_L3_L1_I (x = 786.100, y = 1415.000)
Component output: pbindct_clock_L3_L1_Z (x = 791.700, y = 1416.250)
Next input: forwardct_bindct8_z6_reg_0_CP (x = 805.700, y = 1428.750)

Delay from root to next input: 1.176 ns
Delay from component input to next input: 0.330 ns
Delay from component input to component output: 0.330 ns
Delay from component output to next input:  0.000 ns  
Transition time at next input:  0.064 ns  
Total load at component output:  0.009 pF

Max path

Root level: PDI  
Component input: pbindct_clock C (x = 364.550, y = 784.200)  
Component output: pbindct_clock C (x = 364.550, y = 784.200)  
Next input:  pbindct_clock_L1_I5 C (x = 910.000, y = 1457.500)  

Delay from root to next input:  0.060 ns  
Delay from component input to next input:  0.060 ns  
Delay from component input to component output:  0.000 ns  
Delay from component output to next input:  0.060 ns  
Transition time at next input:  0.234 ns  
Total load at component output:  0.369 pF

Level 1: ITB1  
Component input: pbindct_clock_L1_I5 C (x = 910.000, y = 1457.500)  
Component output: pbindct_clock_L1_I5 CZ (x = 896.700, y = 1457.500)  
Next input:  pbindct_clock_L2_I17 I (x = 893.900, y = 1456.250)  

Delay from root to next input:  0.601 ns  
Delay from component input to next input:  0.541 ns  
Delay from component input to component output:  0.541 ns  
Delay from component output to next input:  0.000 ns  
Transition time at next input:  0.158 ns  
Total load at component output:  0.078 pF

Level 2: INV4  
Component input: pbindct_clock_L2_I17 I (x = 893.900, y = 1456.250)  
Component output: pbindct_clock_L2_I17 ZN (x = 890.400, y = 1456.875)  
Next input:  pbindct_clock_L3_I136 I (x = 910.700, y = 1703.750)  

Delay from root to next input:  1.044 ns  
Delay from component input to next input:  0.444 ns  
Delay from component input to component output:  0.431 ns  
Delay from component output to next input:  0.013 ns  
Transition time at next input:  0.461 ns  
Total load at component output:  0.687 pF

Level 3: INV4  
Component input: pbindct_clock_L3_I136 I (x = 910.700, y = 1703.750)  
Component output: pbindct_clock_L3_I136 ZN (x = 914.200, y = 1704.375)  
Next input:  forwardct_p_s_out11_outputmem_reg_4__6__E (x = 1019.900, y = 1648.750)  

Delay from root to next input:  1.198 ns  
Delay from component input to next input:  0.153 ns  
Delay from component input to component output:  0.149 ns  
Delay from component output to next input:  0.004 ns  
Transition time at next input:  0.143 ns  
Total load at component output:  0.145 pF

132
F.4 Forward one-dimensional binDCT chip clock tree final timing

Report: /rpt/final.timing
Design: forwardbindctchip

Clock tree root: pbindct_clock C
Timing start pin: pbindct_clock C
  Max. transition time at leaf pins: 0.146 ns
  Min. insertion delay to leaf pins: 1.176 ns
  Max. insertion delay to leaf pins: 1.198 ns
  Max. skew between leaf pins: 0.021 ns

F.5 Inverse one-dimensional binDCT chip clock tree initial analysis

Report: /rpt/initial.analysis
Design: inversebindctchip

Clock tree root: pbindct_clock C

Min path

Root level: PDI
  Component input: pbindct_clock C (x = 364.550, y = 2096.200)
  Component output: pbindct_clock C (x = 364.550, y = 2096.200)
  Next input: inversedct_ibindct113x11_reg_11_CP (x = 1040.900, y = 1733.750)

  Delay from root to next input: 0.530 ns
  Delay from component input to next input: 0.530 ns
  Delay from component input to component output: 0.000 ns
  Delay from component output to next input: 0.530 ns
  Transition time at next input: 1.144 ns
  Total load at component output: 4.173 pF

Max path

Root level: PDI
  Component input: pbindct_clock C (x = 364.550, y = 2096.200)
  Component output: pbindct_clock C (x = 364.550, y = 2096.200)
  Next input: inversedct_s_p_in11x0_reg_4_CP (x = 1766.100, y = 1376.250)

  Delay from root to next input: 1.252 ns
  Delay from component input to next input: 1.252 ns
  Delay from component input to component output: 0.000 ns
  Delay from component output to next input: 1.252 ns
  Transition time at next input: 2.824 ns
  Total load at component output: 4.173 pF
F.6 Inverse one-dimensional binDCT clock tree initial timing

Report: /rpt/initial.timing
Design: inversebindctchip

Clock tree root: pbindct_clock C
Timing start pin: pbindct_clock C
  Max. transition time at leaf pins: 2.824 ns
  Min. insertion delay to leaf pins: 0.530 ns
  Max. insertion delay to leaf pins: 1.252 ns
  Max. skew between leaf pins: 0.722 ns

F.7 Inverse one-dimensional binDCT chip final clock tree analysis

Report: /rpt/final.analysis
Design: inversebindctchip

Clock tree root: pbindct_clock C

Min path

Root level: PD1
  Component input: pbindct_clock C (x = 364.550, y = 2096.200)
  Component output: pbindct_clock C (x = 364.550, y = 2096.200)
  Next input: pbindct_clock_L1_I2 I (x = 1483.300, y = 1458.750)

  Delay from root to next input: 0.058 ns
  Delay from component input to next input: 0.058 ns
  Delay from component input to component output: 0.000 ns
  Delay from component output to next input: 0.058 ns
  Transition time at next input: 0.232 ns
  Total load at component output: 0.324 pF

Level 1: BUF4
  Component input: pbindct_clock_L1_I2 I (x = 1483.300, y = 1458.750)
  Component output: pbindct_clock_L1_I2 Z (x = 1487.500, y = 1460.000)
  Next input: pbindct_clock_L2_I4 I (x = 1409.100, y = 1458.750)

  Delay from root to next input: 0.419 ns
  Delay from component input to next input: 0.361 ns
  Delay from component input to component output: 0.360 ns
  Delay from component output to next input: 0.001 ns
  Transition time at next input: 0.136 ns
  Total load at component output: 0.149 pF

Level 2: BUF4
  Component input: pbindct_clock_L2_I4 I (x = 1409.100, y = 1458.750)
  Component output: pbindct_clock_L2_I4 Z (x = 1413.300, y = 1460.000)
  Next input: pbindct_clock_L3_I3 I (x = 1409.100, y = 1513.750)

  Delay from root to next input: 0.712 ns
  Delay from component input to next input: 0.293 ns
Delay from component input to component output: 0.292 ns
Delay from component output to next input: 0.001 ns
Transition time at next input: 0.088 ns
Total load at component output: 0.068 pF

Level 3: BUF4
Component input: pbindct_clock_L3_I3 I (x = 1409.100, y = 1513.750)
Component output: pbindct_clock_L3_I3 Z (x = 1413.300, y = 1515.000)
Next input: pbindct_clock_L4_I5 I (x = 1413.300, y = 1472.500)

Delay from root to next input: 0.984 ns
Delay from component input to next input: 0.271 ns
Delay from component input to component output: 0.271 ns
Delay from component output to next input: 0.000 ns
Transition time at next input: 0.078 ns
Total load at component output: 0.052 pF

Level 4: BUF4
Component input: pbindct_clock_L4_I5 I (x = 1413.300, y = 1472.500)
Component output: pbindct_clock_L4_I5 Z (x = 1417.500, y = 1473.750)
Next input: inversedct_iibindct11x7_reg_5_CP (x = 1427.300, y = 1472.500)

Delay from root to next input: 1.337 ns
Delay from component input to next input: 0.353 ns
Delay from component input to component output: 0.353 ns
Delay from component output to next input: 0.000 ns
Transition time at next input: 0.171 ns
Total load at component output: 0.211 pF

Max path

Root level: PDI
Component input: pbindct_clock C (x = 364.550, y = 2096.200)
Component output: pbindct_clock C (x = 364.550, y = 2096.200)
Next input: pbindct_clock_L1_I2 I (x = 1483.300, y = 1458.750)

Delay from root to next input: 0.058 ns
Delay from component input to next input: 0.058 ns
Delay from component input to component output: 0.000 ns
Delay from component output to next input: 0.058 ns
Transition time at next input: 0.232 ns
Total load at component output: 0.324 pF

Level 1: BUF4
Component input: pbindct_clock_L1_I2 I (x = 1483.300, y = 1458.750)
Component output: pbindct_clock_L1_I2 Z (x = 1487.500, y = 1460.000)
Next input: pbindct_clock_L2_I3 I (x = 1159.900, y = 1458.750)

Delay from root to next input: 0.421 ns
Delay from component input to next input: 0.362 ns
Delay from component input to component output: 0.360 ns
Delay from component output to next input: 0.002 ns
Transition time at next input: 0.136 ns
Total load at component output: 0.149 pF

135
Level 2: BUF4
Component input: pbisdict_clock_L2, L3 I (x = 1159.900, y = 1458.750)
Component output: pbisdict_clock_L2, L3 Z (x = 1164.100, y = 1460.000)
Next input: pbisdict_clock_L3 I (x = 1234.100, y = 1637.500)

Delay from root to next input: 0.725 ns
Delay from component input to next input: 0.305 ns
Delay from component input to component output: 0.302 ns
Delay from component output to next input: 0.003 ns
Transition time at next input: 0.099 ns
Total load at component output: 0.086 pF

Level 3: BUF4
Component input: pbisdict_clock_L3, L4 I (x = 1234.100, y = 1637.500)
Component output: pbisdict_clock_L3, L4 Z (x = 1238.300, y = 1638.750)
Next input: pbisdict_clock_L4 I (x = 1234.100, y = 1582.500)

Delay from root to next input: 1.003 ns
Delay from component input to next input: 0.277 ns
Delay from component input to component output: 0.277 ns
Delay from component output to next input: 0.000 ns
Transition time at next input: 0.082 ns
Total load at component output: 0.057 pF

Level 4: BUF4
Component input: pbisdict_clock_L4, L5 I (x = 1234.100, y = 1582.500)
Component output: pbisdict_clock_L4, L5 Z (x = 1238.300, y = 1583.750)
Next input: inversedict_ibisdict11x0_reg_L2, CP (x = 1221.500, y = 1500.000)

Delay from root to next input: 1.350 ns
Delay from component input to next input: 0.348 ns
Delay from component input to component output: 0.345 ns
Delay from component output to next input: 0.003 ns
Transition time at next input: 0.161 ns
Total load at component output: 0.193 pF

F.8 Inverse one-dimensional binDCT chip clock tree final timing

Report: /rpt/final.timing
Design: inversebindctchips

Clock tree root: pbisdict_clock C
Timing start pin: pbisdict_clock C
Max. transition time at leaf pins: 0.830 ns
Min. insertion delay to leaf pins: 1.337 ns
Max. insertion delay to leaf pins: 1.350 ns
Max. skew between leaf pins: 0.013 ns
Appendix G

The two-dimensional binDCT RTL description

G.1 The 8x8 two-dimensional binDCT chip RTL description

This RTL description is a starting point for the realization of a 2-D binDCT chip prior to folding transformation being implemented.

// HDL description of the 8x8 2-D binDCT-B
// A normalized version, using a scale of sqrt(2/N), where N=8.
// Created by: Tracy Franklin

`timescale 1ns/10ps
module dct2dchip (data_in, row_address, column_address, data_input_enable, serial_clock, parallel_clock, bindct_clock, transpose_clock, out_clock, data_out);

input [7:0] data_in;
input [2:0] row_address, column_address;
input data_input_enable;
input serial_clock, parallel_clock, bindct_clock, transpose_clock, out_clock; output [11:0] data_out;

wire [7:0] data_in_top;
wire [2:0] row_address_top, column_address_top;
wire data_input_enable_top;
wire serial_clock_top, parallel_clock_top, bindct_clock_top, transpose_clock_top, out_clock_top;
wire [11:0] data_out_top;

bindct_2d dct2 (data_in_top, row_address_top, column_address_top, data_input_enable_top, serial_clock_top, parallel_clock_top, bindct_clock_top, transpose_clock_top, out_clock_top, data_out_top);

PDI pdin0 (.C(data_in_top[0]), .PAD(data_in[0]));
PDI pdin1 (.C(data_in_top[1]), .PAD(data_in[1]));
PDI pdin2 (.C(data_in_top[2]), .PAD(data_in[2]));
PDI pdin3 (.C(data_in_top[3]), .PAD(data_in[3]));
PDI pdin4 (.C(data_in_top[4]), .PAD(data_in[4]));
PDI pdin5 (.C(data_in_top[5]), .PAD(data_in[5]));
PDI pdin6 (.C(data_in_top[6]), .PAD(data_in[6]));
PDI pdin7 (.C(data_in_top[7]), .PAD(data_in[7]));
PDI pra0 (.C(row_address_top[0]), .PAD(row_address[0]));
PDI pra1 (.C(row_address_top[1]), .PAD(row_address[1]));
PDI pra2 (.C(row_address_top[2]), .PAD(row_address[2]));
PDI pca0 (.C(column_address_top[0]), .PAD(column_address[0]));
PDI pca1 (.C(column_address_top[1]), .PAD(column_address[1]));
PDI pca2 (.C(column_address_top[2]), .PAD(column_address[2]));
PDI pwe (.C(data_input_enable_top), .PAD(data_input_enable));
PDI psc (.C(serial_clock_top), .PAD(serial_clock));
PDI ppc (.C(parallel_clock_top), .PAD(parallel_clock));
PDI pbc (.C(bindct_clock_top), .PAD(bindct_clock));
PDI ptc (.C(transpose_clock_top), .PAD(transpose_clock));
PDI poc (.C(out_clock_top), .PAD(out_clock));
module bindc_2d (data_in, row_address, column_address, data_input_enable, serial_clock, parallel_clock, bindc_clock, transpose_clock, out_clock, data_out);

input [7:0] data_in;
input [2:0] row_address, column_address;
input data_input_enable;
input serial_clock, parallel_clock, bindc_clock, transpose_clock, out_clock; output [11:0] data_out;

wire bindc_enable = 1'b1;
wire transpose_enable = 1'b1;
wire output_enable = 1'b1;

// 64 x 8 input block
wire [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
wire [7:0] x8, x9, x10, x11, x12, x13, x14, x15;
wire [7:0] x16, x17, x18, x19, x20, x21, x22, x23;
wire [7:0] x24, x25, x26, x27, x28, x29, x30, x31;
wire [7:0] x32, x33, x34, x35, x36, x37, x38, x39;
wire [7:0] x40, x41, x42, x43, x44, x45, x46, x47;
wire [7:0] x48, x49, x50, x51, x52, x53, x54, x55;
wire [7:0] x56, x57, x58, x59, x60, x61, x62, x63;

// 64 x 10 row-wise transformed block
wire [9:0] y0, y1, y2, y3, y4, y5, y6, y7;
wire [9:0] y8, y9, y10, y11, y12, y13, y14, y15;
wire [9:0] y16, y17, y18, y19, y20, y21, y22, y23;
wire [9:0] y24, y25, y26, y27, y28, y29, y30, y31;
wire [9:0] y32, y33, y34, y35, y36, y37, y38, y39;
wire [9:0] y40, y41, y42, y43, y44, y45, y46, y47;
wire [9:0] y48, y49, y50, y51, y52, y53, y54, y55;
wire [9:0] y56, y57, y58, y59, y60, y61, y62, y63;

wire [9:0] w0, w1, w2, w3, w4, w5, w6, w7;
wire [9:0] w8, w9, w10, w11, w12, w13, w14, w15;
wire [9:0] w16, w17, w18, w19, w20, w21, w22, w23;
wire [9:0] w24, w25, w26, w27, w28, w29, w30, w31;
wire [9:0] w32, w33, w34, w35, w36, w37, w38, w39;
wire [9:0] w40, w41, w42, w43, w44, w45, w46, w47;
wire [9:0] w48, w49, w50, w51, w52, w53, w54, w55;
wire [9:0] w56, w57, w58, w59, w60, w61, w62, w63;

// 64 x 12 column-wise transformed block
wire [11:0] z0, z1, z2, z3, z4, z5, z6, z7;
wire [11:0] z8, z9, z10, z11, z12, z13, z14, z15;
wire [11:0] z16, z17, z18, z19, z20, z21, z22, z23;
wire [11:0] z24, z25, z26, z27, z28, z29, z30, z31;
wire [11:0] z32, z33, z34, z35, z36, z37, z38, z39;
wire [11:0] z40, z41, z42, z43, z44, z45, z46, z47;
wire [11:0] z48, z49, z50, z51, z52, z53, z54, z55;
wire [11:0] z56, z57, z58, z59, z60, z61, z62, z63;

// 64 x 12 row-wise transformed block
wire [11:0] v0, v1, v2, v3, v4, v5, v6, v7;

wire [2:0] rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7;
wire [2:0] coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7;
wire [2:0] tempadd0, tempadd1, tempadd2, tempadd3, tempadd4, tempadd5, tempadd6, tempadd7;

// Input the 64 8-bits data serially.
// then transmit in parallel form to 8 bindct blocks
s_p_input spin0 (data_in, row_address, column_address, data_input_enable, serial_clock, parallel_clock, rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7, x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, bindct_clock, bindct_enable, y0, y1, y2, y3, y4, y5, y6, y7); bindct8 bin0 (x0, x1, x2, x3, x4, x5, x6, x7.bindct_clock, bindct_enable, y0, y1, y2, y3, y4, y5, y6, y7);
bindct8 bin1 (x8, x9, x10, x11, x12, x13, x14, x15.bindct_clock, bindct_enable, y8, y9, y10, y11, y12, y13, y14, y15);
bindct8 bin2 (x16, x17, x18, x19, x20, x21, x22, x23.bindct_clock, bindct_enable, y16, y17, y18, y19, y20, y21, y22, y23);
bindct8 bin3 (x24, x25, x26, x27, x28, x29, x30, x31.bindct_clock, bindct_enable, y24, y25, y26, y27, y28, y29, y30, y31);
bindct8 bin4 (x32, x33, x34, x35, x36, x37, x38, x39.bindct_clock, bindct_enable, y32, y33, y34, y35, y36, y37, y38, y39);
bindct8 bin5 (x40, x41, x42, x43, x44, x45, x46, x47.bindct_clock, bindct_enable, y40, y41, y42, y43, y44, y45, y46, y47);
bindct8 bin6 (x48, x49, x50, x51, x52, x53, x54, x55.bindct_clock, bindct_enable, y48, y49, y50, y51, y52, y53, y54, y55);
bindct8 bin7 (x56, x57, x58, x59, x60, x61, x62, x63.bindct_clock, bindct_enable, y56, y57, y58, y59, y60, y61, y62, y63);

// Transpose the 64 10-bits of row-wise transformed data
transpose9 tr9 (y0, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19, y20, y21, y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33, y34, y35, y36, y37, y38, y39, y40, y41, y42, y43, y44, y45, y46, y47, y48, y49, y50, y51, y52, y53, y54, y55, y56, y57, y58, y59, y60, y61,
y62, y63, transpose_enable, bindct_clock, transpose_clock, rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7, coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, w0, w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16, w17, w18, w19, w20, w21, w22, w23, w24, w25, w26, w27, w28, w29, w30, w31, w32, w33, w34, w35, w36, w37, w38, w39, w40, w41, w42, w43, w44, w45, w46, w47, w48, w49, w50, w51, w52, w53, w54, w55, w56, w57, w58, w59, w60, w61, w62, w63);

// column-wise transform of 8 x 8 block
bindct10 bin8 (w0, w1, w2, w3, w4, w5, w6, w7, bindct_clock, , bindct_enable, z0, z1, z2, z3, z4, z5, z6, z7);
bindct10 bin9 (w8, w9, w10, w11, w12, w13, w14, w15, bindct_clock, , bindct_enable, z8, z9, z10, z11, z12, z13, z14, z15);
bindct10 bin10 (w16, w17, w18, w19, w20, w21, w22, w23, bindct_clock, , bindct_enable, z16, z17, z18, z19, z20, z21, z22, z23);
bindct10 bin11 (w24, w25, w26, w27, w28, w29, w30, w31, bindct_clock, , bindct_enable, z24, z25, z26, z27, z28, z29, z30, z31);
bindct10 bin12 (w32, w33, w34, w35, w36, w37, w38, w39, bindct_clock, , bindct_enable, z32, z33, z34, z35, z36, z37, z38, z39);
bindct10 bin13 (w40, w41, w42, w43, w44, w45, w46, w47, bindct_clock, , bindct_enable, z40, z41, z42, z43, z44, z45, z46, z47);
bindct10 bin14 (w48, w49, w50, w51, w52, w53, w54, w55, bindct_clock, , bindct_enable, z48, z49, z50, z51, z52, z53, z54, z55);
bindct10 bin15 (w56, w57, w58, w59, w60, w61, w62, bindct_clock, , bindct_enable, z56, z57, z58, z59, z60, z61, z62, z63);

transpose11 tr11 (z0, z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12, z13, z14, z15, z16, z17, z18, z19, z20, z21, z22, z23, z24, z25, z26, z27, z28, z29, z30, z31, z32, z33, z34, z35, z36, z37, z38, z39, z40, z41, z42, z43, z44, z45, z46, z47, z48, z49, z50, z51, z52, z53, z54, z55, z56, z57, z58, z59, z60, z61, z62, z63, transpose_enable, bindct_clock, transpose_clock, coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7, tempadd0, tempadd1, tempadd2, tempadd3, tempadd4, tempadd5, tempadd6, tempadd7, v0, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31, v32, v33, v34, v35, v36, v37, v38, v39, v40, v41, v42, v43, v44, v45, v46, v47, v48, v49, v50, v51, v52, v53, v54, v55, v56, v57, v58, v59, v60, v61, v62, v63, row_address, column_address, data_input_enable, serial_clock, out_clock, data_out);

endmodule

// serial-to-parallel input the data
module s_p_input (data, rowadd_clock, column_address, write_enable, enable, serial_clk, parallel_clk, rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7, x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29, x30, x31, x32, x33, x34, x35, x36, x37, x38, x39, x40, x41, x42, x43, x44, x45, x46, x47, x48, x49, x50, x51, x52, x53, x54, x55, x56, x57, x58, x59, x60, x61, x62, x63);

input [7:0] data;
input [2:0] row_address, column_address;
input write_enable, serial_clk, parallel_clk;
output [2:0] rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7;
output [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
output [7:0] x8, x9, x10, x11, x12, x13, x14, x15;
output [7:0] x16, x17, x18, x19, x20, x21, x22, x23;
output [7:0] x24, x25, x26, x27, x28, x29, x30, x31;
output [7:0] x32, x33, x34, x35, x36, x37, x38, x39;
output [7:0] x40, x41, x42, x43, x44, x45, x46, x47;
output [7:0] x48, x49, x50, x51, x52, x53, x54, x55;
output [7:0] x56, x57, x58, x59, x60, x61, x62, x63;

reg [5:0] mem_address;
reg [7:0] memblk1 [0:63];
reg [2:0] rowadd0, rowadd1, rowadd2, rowadd4, rowadd5, rowadd6, rowadd7;
reg [7:0] x0, x1, x2, x3, x4, x5, x6, x7;
reg [7:0] x8, x9, x10, x11, x12, x13, x14, x15;
reg [7:0] x16, x17, x18, x19, x20, x21, x22, x23;
reg [7:0] x24, x25, x26, x27, x28, x29, x30, x31;
reg [7:0] x32, x33, x34, x35, x36, x37, x38, x39;
reg [7:0] x40, x41, x42, x43, x44, x45, x46, x47;
reg [7:0] x48, x49, x50, x51, x52, x53, x54, x55;
reg [7:0] x56, x57, x58, x59, x60, x61, x62, x63;
reg [7:0] x0_reg, x1_reg, x2_reg, x3_reg, x4_reg, x5_reg, x6_reg, x7_reg;
reg [7:0] x8_reg, x9_reg, x10_reg, x11_reg, x12_reg, x13_reg, x14_reg, x15_reg;
reg [7:0] x16_reg, x17_reg, x18_reg, x19_reg, x20_reg, x21_reg, x22_reg, x23_reg;
reg [7:0] x24_reg, x25_reg, x26_reg, x27_reg, x28_reg, x29_reg, x30_reg, x31_reg;
reg [7:0] x32_reg, x33_reg, x34_reg, x35_reg, x36_reg, x37_reg, x38_reg, x39_reg;
reg [7:0] x40_reg, x41_reg, x42_reg, x43_reg, x44_reg, x45_reg, x46_reg, x47_reg;
reg [7:0] x48_reg, x49_reg, x50_reg, x51_reg, x52_reg, x53_reg, x54_reg, x55_reg;
reg [7:0] x56_reg, x57_reg, x58_reg, x59_reg, x60_reg, x61_reg, x62_reg, x63_reg;
reg data_out_enable_1, wcontrol;

always @ (data or write_enable or row_address or column_address or serial_clk) begin
if (write_enable == 1'b1) begin
  mem_address = {row_address, column_address};
  memblk1[mem_address]=data;
  wcontrol= ~write_enable;
end
end

always @ (posedge serial_clk) begin
if (wcontrol == 1'b0) begin
  data_out_enable_1=1'b1;
  x0_reg=memblk1[6'd0];
  x1_reg=memblk1[6'd1];
  x2_reg=memblk1[6'd2];
  x3_reg=memblk1[6'd3];
  x4_reg=memblk1[6'd4];
  x5_reg=memblk1[6'd5];
  x6_reg=memblk1[6'd6];
  x7_reg=memblk1[6'd7];
  x8_reg=memblk1[6'd8];
  x9_reg=memblk1[6'd9];
  x10_reg=memblk1[6'd10];
  x11_reg=memblk1[6'd11];
  x12_reg=memblk1[6'd12];
  x13_reg=memblk1[6'd13];
  x14_reg=memblk1[6'd14];
  x15_reg=memblk1[6'd15];
  x16_reg=memblk1[6'd16];
  x17_reg=memblk1[6'd17];
end

end
x18_reg=memblk1[6'd18];
x19_reg=memblk1[6'd19];
x20_reg=memblk1[6'd20];
x21_reg=memblk1[6'd21];
x22_reg=memblk1[6'd22];
x23_reg=memblk1[6'd23];
x24_reg=memblk1[6'd24];
x25_reg=memblk1[6'd25];
x26_reg=memblk1[6'd26];
x27_reg=memblk1[6'd27];
x28_reg=memblk1[6'd28];
x29_reg=memblk1[6'd29];
x30_reg=memblk1[6'd30];
x31_reg=memblk1[6'd31];
x32_reg=memblk1[6'd32];
x33_reg=memblk1[6'd33];
x34_reg=memblk1[6'd34];
x35_reg=memblk1[6'd35];
x36_reg=memblk1[6'd36];
x37_reg=memblk1[6'd37];
x38_reg=memblk1[6'd38];
x39_reg=memblk1[6'd39];
x40_reg=memblk1[6'd40];
x41_reg=memblk1[6'd41];
x42_reg=memblk1[6'd42];
x43_reg=memblk1[6'd43];
x44_reg=memblk1[6'd44];
x45_reg=memblk1[6'd45];
x46_reg=memblk1[6'd46];
x47_reg=memblk1[6'd47];
x48_reg=memblk1[6'd48];
x49_reg=memblk1[6'd49];
x50_reg=memblk1[6'd50];
x51_reg=memblk1[6'd51];
x52_reg=memblk1[6'd52];
x53_reg=memblk1[6'd53];
x54_reg=memblk1[6'd54];
x55_reg=memblk1[6'd55];
x56_reg=memblk1[6'd56];
x57_reg=memblk1[6'd57];
x58_reg=memblk1[6'd58];
x59_reg=memblk1[6'd59];
x60_reg=memblk1[6'd60];
x61_reg=memblk1[6'd61];
x62_reg=memblk1[6'd62];
x63_reg=memblk1[6'd63];

end
end

always @(posedge parallel_clk) begin
if (data_out_enable_1 == 1'b1) begin
  rowadd0 = 3'b000;
  rowadd1 = 3'b001;
  rowadd2 = 3'b010;
  rowadd3 = 3'b011;
end
rowadd4 = 3'b100;
rowadd5 = 3'b101;
rowadd6 = 3'b110;
rowadd7 = 3'b111;
x0=x0_reg;
x1=x1_reg;
x2=x2_reg;
x3=x3_reg;
x4=x4_reg;
x5=x5_reg;
x6=x6_reg;
x7=x7_reg;
x8=x8_reg;
x9=x9_reg;
x10=x10_reg;
x11=x11_reg;
x12=x12_reg;
x13=x13_reg;
x14=x14_reg;
x15=x15_reg;
x16=x16_reg;
x17=x17_reg;
x18=x18_reg;
x19=x19_reg;
x20=x20_reg;
x21=x21_reg;
x22=x22_reg;
x23=x23_reg;
x24=x24_reg;
x25=x25_reg;
x26=x26_reg;
x27=x27_reg;
x28=x28_reg;
x29=x29_reg;
x30=x30_reg;
x31=x31_reg;
x32=x32_reg;
x33=x33_reg;
x34=x34_reg;
x35=x35_reg;
x36=x36_reg;
x37=x37_reg;
x38=x38_reg;
x39=x39_reg;
x40=x40_reg;
x41=x41_reg;
x42=x42_reg;
x43=x43_reg;
x44=x44_reg;
x45=x45_reg;
x46=x46_reg;
x47=x47_reg;
x48=x48_reg;
x49=x49_reg;
x50=x50_reg;
x51=x51_reg;
x52=x52_reg;
x53=x53_reg;
x54=x54_reg;
x55=x55_reg;
x56=x56_reg;
x57=x57_reg;
x58=x58_reg;
x59=x59_reg;
x60=x60_reg;
x61=x61_reg;
x62=x62_reg;
x63=x63_reg;
end
endmodule

// Transpose module
// Write in 64 data points, assigning to a 64x10-bit memory array row-wise
// Read out 64 data points, column-wise from the 64x10-bit memory array.
module transpose9 ( y0, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19, y20, y21, y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33, y34, y35, y36, y37, y38, y39, y40, y41, y42, y43, y44, y45, y46, y47, y48, y49, y50, y51, y52, y53, y54, y55, y56, y57, y58, y59, y60, y61, y62, y63, data_in_enable, wtranspose_clk, rtranspose_clk, rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7, coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7, w0, w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16, w17, w18, w19, w20, w21, w22, w23, w24, w25, w26, w27, w28, w29, w30, w31, w32, w33, w34, w35, w36, w37, w38, w39, w40, w41, w42, w43, w44, w45, w46, w47, w48, w49, w50, w51, w52, w53, w54, w55, w56, w57, w58, w59, w60, w61, w62, w63);

input [9:0] y0, y1, y2, y3, y4, y5, y6, y7;
input [9:0] y8, y9, y10, y11, y12, y13, y14, y15;
input [9:0] y16, y17, y18, y19, y20, y21, y22, y23;
input [9:0] y24, y25, y26, y27, y28, y29, y30, y31;
input [9:0] y32, y33, y34, y35, y36, y37, y38, y39;
input [9:0] y40, y41, y42, y43, y44, y45, y46, y47;
input [9:0] y48, y49, y50, y51, y52, y53, y54, y55;
input [9:0] y56, y57, y58, y59, y60, y61, y62, y63;
input data_in_enable, wtranspose_clk, rtranspose_clk;
input [2:0] rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7;
output [2:0] coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7;
output [9:0] w0, w1, w2, w3, w4, w5, w6, w7;
output [9:0] w8, w9, w10, w11, w12, w13, w14, w15;
output [9:0] w16, w17, w18, w19, w20, w21, w22, w23;
output [9:0] w24, w25, w26, w27, w28, w29, w30, w31;
output [9:0] w32, w33, w34, w35, w36, w37, w38, w39;
output [9:0] w40, w41, w42, w43, w44, w45, w46, w47;
output [9:0] w48, w49, w50, w51, w52, w53, w54, w55;
output [9:0] w56, w57, w58, w59, w60, w61, w62, w63;

reg data_out_enable_2;
reg [2:0] coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7;
reg [9:0] w0, w1, w2, w3, w4, w5, w6, w7;
reg [9:0] w8, w9, w10, w11, w12, w13, w14, w15;
reg [9:0] w16, w17, w18, w19, w20, w21, w22, w23;
reg [9:0] w24, w25, w26, w27, w28, w29, w30, w31;
reg [9:0] w32, w33, w34, w35, w36, w37, w38, w39;
always @(data_in_enable or wtranspose_clk or rowadd0 or rowadd1 or rowadd2 or rowadd3 or rowadd4 or rowadd5 or rowadd6 or rowadd7 or y0 or y1 or y2 or y3 or y4 or y5 or y6 or y7 or y8 or y9 or y10 or y11 or y12 or y13 or y14 or y15 or y16 or y17 or y18 or y19 or y20 or y21 or y22 or y23 or y24 or y25 or y26 or y27 or y28 or y29 or y30 or y31 or y32 or y33 or y34 or y35 or y36 or y37 or y38 or y39 or y40 or y41 or y42 or y43 or y44 or y45 or y46 or y47 or y48 or y49 or y50 or y51 or y52 or y53 or y54 or y55 or y56 or y57 or y58 or y59 or y60 or y61 or y62 or y63)
begin
if (data_in_enable==1'b1) begin
  memblk2[[rowadd0.3'b000]]=y0;
  memblk2[[rowadd0.3'b001]]=y1;
  memblk2[[rowadd0.3'b010]]=y2;
  memblk2[[rowadd0.3'b011]]=y3;
  memblk2[[rowadd0.3'b100]]=y4;
  memblk2[[rowadd0.3'b101]]=y5;
  memblk2[[rowadd0.3'b110]]=y6;
  memblk2[[rowadd0.3'b111]]=y7;
  memblk2[[rowadd1.3'b000]]=y8;
  memblk2[[rowadd1.3'b001]]=y9;
  memblk2[[rowadd1.3'b010]]=y10;
  memblk2[[rowadd1.3'b011]]=y11;
  memblk2[[rowadd1.3'b100]]=y12;
  memblk2[[rowadd1.3'b101]]=y13;
  memblk2[[rowadd1.3'b110]]=y14;
  memblk2[[rowadd1.3'b111]]=y15;
  memblk2[[rowadd2.3'b000]]=y16;
  memblk2[[rowadd2.3'b001]]=y17;
  memblk2[[rowadd2.3'b010]]=y18;
  memblk2[[rowadd2.3'b011]]=y19;
  memblk2[[rowadd2.3'b100]]=y20;
  memblk2[[rowadd2.3'b101]]=y21;
  memblk2[[rowadd2.3'b110]]=y22;
  memblk2[[rowadd2.3'b111]]=y23;
  memblk2[[rowadd3.3'b000]]=y24;
  memblk2[[rowadd3.3'b001]]=y25;
  memblk2[[rowadd3.3'b010]]=y26;
  memblk2[[rowadd3.3'b011]]=y27;
  memblk2[[rowadd3.3'b100]]=y28;
  memblk2[[rowadd3.3'b101]]=y29;
  memblk2[[rowadd3.3'b110]]=y30;
  memblk2[[rowadd3.3'b111]]=y31;
  memblk2[[rowadd4.3'b000]]=y32;
  memblk2[[rowadd4.3'b001]]=y33;
end
```verilog
memblk2[\{rowadd4,3'b010\}]=y34;
memblk2[\{rowadd4,3'b011\}]=y35;
memblk2[\{rowadd4,3'b100\}]=y36;
memblk2[\{rowadd4,3'b101\}]=y37;
memblk2[\{rowadd4,3'b110\}]=y38;
memblk2[\{rowadd4,3'b111\}]=y39;
memblk2[\{rowadd5,3'b000\}]=y40;
memblk2[\{rowadd5,3'b001\}]=y41;
memblk2[\{rowadd5,3'b010\}]=y42;
memblk2[\{rowadd5,3'b011\}]=y43;
memblk2[\{rowadd5,3'b100\}]=y44;
memblk2[\{rowadd5,3'b101\}]=y45;
memblk2[\{rowadd5,3'b110\}]=y46;
memblk2[\{rowadd5,3'b111\}]=y47;
memblk2[\{rowadd6,3'b000\}]=y48;
memblk2[\{rowadd6,3'b001\}]=y49;
memblk2[\{rowadd6,3'b010\}]=y50;
memblk2[\{rowadd6,3'b011\}]=y51;
memblk2[\{rowadd6,3'b100\}]=y52;
memblk2[\{rowadd6,3'b101\}]=y53;
memblk2[\{rowadd6,3'b110\}]=y54;
memblk2[\{rowadd6,3'b111\}]=y55;
memblk2[\{rowadd7,3'b000\}]=y56;
memblk2[\{rowadd7,3'b001\}]=y57;
memblk2[\{rowadd7,3'b010\}]=y58;
memblk2[\{rowadd7,3'b011\}]=y59;
memblk2[\{rowadd7,3'b100\}]=y60;
memblk2[\{rowadd7,3'b101\}]=y61;
memblk2[\{rowadd7,3'b110\}]=y62;
memblk2[\{rowadd7,3'b111\}]=y63;
data_out_enable_2=1'b0;
wcontrol=!data_in_enable;
end
end

always @ (posedge wtranspose_clk) begin
if (wcontrol == 1'b0) begin
data_out_enable_2=1'b1;
    w0_reg = memblk2[\{rowadd0,3'b000\}];
    w1_reg = memblk2[\{rowadd1,3'b000\}];
    w2_reg = memblk2[\{rowadd2,3'b000\}];
    w3_reg = memblk2[\{rowadd3,3'b000\}];
    w4_reg = memblk2[\{rowadd4,3'b000\}];
    w5_reg = memblk2[\{rowadd5,3'b000\}];
    w6_reg = memblk2[\{rowadd6,3'b000\}];
    w7_reg = memblk2[\{rowadd7,3'b000\}];
    w8_reg = memblk2[\{rowadd0,3'b001\}];
    w9_reg = memblk2[\{rowadd1,3'b001\}];
    w10_reg= memblk2[\{rowadd2,3'b001\}];
    w11_reg= memblk2[\{rowadd3,3'b001\}];
    w12_reg= memblk2[\{rowadd4,3'b001\}];
    w13_reg= memblk2[\{rowadd5,3'b001\}];
    w14_reg= memblk2[\{rowadd6,3'b001\}];
    w15_reg= memblk2[\{rowadd7,3'b001\}];
    w16_reg = memblk2[\{rowadd0,3'b010\}];
    w17_reg = memblk2[\{rowadd1,3'b010\}];
end
```

146
end

always @(posedge rtranspose_clk) begin

if (data_out_enable_2 == 1'b1) begin

coladd0 = 3'b000;
coladd1 = 3'b001;
coladd2 = 3'b010;
coladd3 = 3'b011;
coladd4 = 3'b100;

end

end
coladd5 = 3'b101;
coladd6 = 3'b110;
coladd7 = 3'b111;
w0 = w0_reg;
w1 = w1_reg;
w2 = w2_reg;
w3 = w3_reg;
w4 = w4_reg;
w5 = w5_reg;
w6 = w6_reg;
w7 = w7_reg;
w8 = w8_reg;
w9 = w9_reg;
w10 = w10_reg;
w11 = w11_reg;
w12 = w12_reg;
w13 = w13_reg;
w14 = w14_reg;
w15 = w15_reg;
w16 = w16_reg;
w17 = w17_reg;
w18 = w18_reg;
w19 = w19_reg;
w20 = w20_reg;
w21 = w21_reg;
w22 = w22_reg;
w23 = w23_reg;
w24 = w24_reg;
w25 = w25_reg;
w26 = w26_reg;
w27 = w27_reg;
w28 = w28_reg;
w29 = w29_reg;
w30 = w30_reg;
w31 = w31_reg;
w32 = w32_reg;
w33 = w33_reg;
w34 = w34_reg;
w35 = w35_reg;
w36 = w36_reg;
w37 = w37_reg;
w38 = w38_reg;
w39 = w39_reg;
w40 = w40_reg;
w41 = w41_reg;
w42 = w42_reg;
w43 = w43_reg;
w44 = w44_reg;
w45 = w45_reg;
w46 = w46_reg;
w47 = w47_reg;
w48 = w48_reg;
w49 = w49_reg;
w50 = w50_reg;
w51 = w51_reg;
w52 = w52_reg;
w53 = w53_reg;
w54 = w54_reg;
w55 = w55_reg;
w56 = w56_reg;
w57 = w57_reg;
w58 = w58_reg;
w59 = w59_reg;
w60 = w60_reg;
w61 = w61_reg;
w62 = w62_reg;
w63 = w63_reg;
end
endmodule

// Transpose module to convert the final 64 data points
// back to row-wise direction.
// Write in 64 data points, assigning to a 64x12-bit memory array row-wise
// Read out 64 data points, column-wise from the 64x12-bit memory array.
module transpose11 (y0, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19, y20, y21, y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, y32, y33, y34, y35, y36, y37, y38, y39, y40, y41, y42, y43, y44, y45, y46, y47, y48, y49, y50, y51, y52, y53, y54, y55, y56, y57, y58, y59, y60, y61, y62, y63, data_in_enable, wtranspose_clk, rtranspose_clk, rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7, coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7, w0, w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16, w17, w18, w19, w20, w21, w22, w23, w24, w25, w26, w27, w28, w29, w30, w31, w32, w33, w34, w35, w36, w37, w38, w39, w40, w41, w42, w43, w44, w45, w46, w47, w48, w49, w50, w51, w52, w53, w54, w55, w56, w57, w58, w59, w60, w61, w62, w63);

input [11:0] y0, y1, y2, y3, y4, y5, y6, y7;
input [11:0] y8, y9, y10, y11, y12, y13, y14, y15;
input [11:0] y16, y17, y18, y19, y20, y21, y22, y23;
input [11:0] y24, y25, y26, y27, y28, y29, y30, y31;
input [11:0] y32, y33, y34, y35, y36, y37, y38, y39;
input [11:0] y40, y41, y42, y43, y44, y45, y46, y47;
input [11:0] y48, y49, y50, y51, y52, y53, y54, y55;
input [11:0] y56, y57, y58, y59, y60, y61, y62, y63;
input data_in_enable, wtranspose_clk, rtranspose_clk;
input [2:0] rowadd0, rowadd1, rowadd2, rowadd3, rowadd4, rowadd5, rowadd6, rowadd7;
output [2:0] coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7;
output [11:0] w0, w1, w2, w3, w4, w5, w6, w7;
output [11:0] w8, w9, w10, w11, w12, w13, w14, w15;
output [11:0] w16, w17, w18, w19, w20, w21, w22, w23;
output [11:0] w24, w25, w26, w27, w28, w29, w30, w31;
output [11:0] w32, w33, w34, w35, w36, w37, w38, w39;
output [11:0] w40, w41, w42, w43, w44, w45, w46, w47;
output [11:0] w48, w49, w50, w51, w52, w53, w54, w55;
output [11:0] w56, w57, w58, w59, w60, w61, w62, w63;

reg data_out_enable_5;
reg [2:0] coladd0, coladd1, coladd2, coladd3, coladd4, coladd5, coladd6, coladd7;
reg [11:0] w0, w1, w2, w3, w4, w5, w6, w7;
reg [11:0] w8, w9, w10, w11, w12, w13, w14, w15;
reg [11:0] w16, w17, w18, w19, w20, w21, w22, w23;
reg [11:0] w24, w25, w26, w27, w28, w29, w30, w31;
always @ (data_in_enable or wtranspose_clk or rowadd0 or rowadd1 or rowadd2 or rowadd3 or rowadd4 or rowadd5 or rowadd6 or rowadd7 or y0 or y1 or y2 or y3 or y4 or y5 or y6 or y7 or y8 or y9 or y10 or y11 or y12 or y13 or y14 or y15 or y16 or y17 or y18 or y19 or y20 or y21 or y22 or y23 or y24 or y25 or y26 or y27 or y28 or y29 or y30 or y31 or y32 or y33 or y34 or y35 or y36 or y37 or y38 or y39 or y40 or y41 or y42 or y43 or y44 or y45 or y46 or y47 or y48 or y49 or y50 or y51 or y52 or y53 or y54 or y55 or y56 or y57 or y58 or y59 or y60 or y61 or y62 or y63) begin
memblk4[[rowadd0,3'b000]]=y0;
memblk4[[rowadd0,3'b010]]=y1;
memblk4[[rowadd0,3'b001]]=y2;
memblk4[[rowadd0,3'b100]]=y3;
memblk4[[rowadd0,3'b101]]=y4;
memblk4[[rowadd0,3'b110]]=y5;
memblk4[[rowadd0,3'b111]]=y6;
memblk4[[rowadd1,3'b000]]=y7;
memblk4[[rowadd1,3'b010]]=y8;
memblk4[[rowadd1,3'b001]]=y9;
memblk4[[rowadd1,3'b100]]=y10;
memblk4[[rowadd1,3'b101]]=y11;
memblk4[[rowadd1,3'b110]]=y12;
memblk4[[rowadd1,3'b111]]=y13;
memblk4[[rowadd1,3'b100]]=y14;
memblk4[[rowadd1,3'b111]]=y15;
memblk4[[rowadd2,3'b000]]=y16;
memblk4[[rowadd2,3'b001]]=y17;
memblk4[[rowadd2,3'b100]]=y18;
memblk4[[rowadd2,3'b101]]=y19;
memblk4[[rowadd2,3'b110]]=y20;
memblk4[[rowadd2,3'b111]]=y21;
memblk4[[rowadd2,3'b110]]=y22;
memblk4[[rowadd2,3'b111]]=y23;
memblk4[[rowadd3,3'b000]]=y24;
memblk4[[rowadd3,3'b001]]=y25;
memblk4[[rowadd3,3'b010]]=y26;
memblk4[[rowadd3,3'b011]]=y27;
memblk4[[rowadd3,3'b100]]=y28;
memblk4[[rowadd3,3'b101]]=y29;
memblk4[[rowadd3,3'b110]]=y30;
memblk4[[rowadd3,3'b111]]=y31;
memblk4[[rowadd4,3'b000]]=y32;
memblk4[[rowadd4,3'b001]]=y33;
memblk4[[rowadd4,3'b010]]=y34;
membil4[[rowadd4,3'b011]]==y35;
membil4[[rowadd4,3'b100]]==y36;
membil4[[rowadd4,3'b101]]==y37;
membil4[[rowadd4,3'b110]]==y38;
membil4[[rowadd4,3'b111]]==y39;
membil4[[rowadd5,3'b000]]==y40;
membil4[[rowadd5,3'b001]]==y41;
membil4[[rowadd5,3'b010]]==y42;
membil4[[rowadd5,3'b011]]==y43;
membil4[[rowadd5,3'b100]]==y44;
membil4[[rowadd5,3'b101]]==y45;
membil4[[rowadd5,3'b110]]==y46;
membil4[[rowadd5,3'b111]]==y47;
membil4[[rowadd6,3'b000]]==y48;
membil4[[rowadd6,3'b001]]==y49;
membil4[[rowadd6,3'b010]]==y50;
membil4[[rowadd6,3'b011]]==y51;
membil4[[rowadd6,3'b100]]==y52;
membil4[[rowadd6,3'b101]]==y53;
membil4[[rowadd6,3'b110]]==y54;
membil4[[rowadd6,3'b111]]==y55;
membil4[[rowadd7,3'b000]]==y56;
membil4[[rowadd7,3'b001]]==y57;
membil4[[rowadd7,3'b010]]==y58;
membil4[[rowadd7,3'b011]]==y59;
membil4[[rowadd7,3'b100]]==y60;
membil4[[rowadd7,3'b101]]==y61;
membil4[[rowadd7,3'b110]]==y62;
membil4[[rowadd7,3'b111]]==y63;
data_out_enable_5==1'b0;
wcontrol==data_in_enable;
end

always @(posedge wtranspose_clk) begin
if (wcontrol == 1'b0) begin
data_out_enable_5==1'b1;

    w0_reg = membil4[[rowadd0,3'b000]];
    w1_reg = membil4[[rowadd1,3'b000]];
    w2_reg = membil4[[rowadd2,3'b000]];
    w3_reg = membil4[[rowadd3,3'b000]];
    w4_reg = membil4[[rowadd4,3'b000]];
    w5_reg = membil4[[rowadd5,3'b000]];
    w6_reg = membil4[[rowadd6,3'b000]];
    w7_reg = membil4[[rowadd7,3'b000]];
    w8_reg = membil4[[rowadd0,3'b001]];
    w9_reg = membil4[[rowadd1,3'b001]];
    w10_reg = membil4[[rowadd2,3'b001]];
    w11_reg = membil4[[rowadd3,3'b001]];
    w12_reg = membil4[[rowadd4,3'b001]];
    w13_reg = membil4[[rowadd5,3'b001]];
    w14_reg = membil4[[rowadd6,3'b001]];
    w15_reg = membil4[[rowadd7,3'b001]];
    w16_reg = membil4[[rowadd0,3'b010]];
    w17_reg = membil4[[rowadd1,3'b010]];
    w18_reg = membil4[[rowadd2,3'b010]];
    w19_reg = membil4[[rowadd3,3'b010]];

end
w20_reg = memblk4[{rowadd4,3'b010}];
w21_reg = memblk4[{rowadd5,3'b010}];
w22_reg = memblk4[{rowadd6,3'b010}];
w23_reg = memblk4[{rowadd7,3'b010}];
w24_reg = memblk4[{rowadd0,3'b011}];
w25_reg = memblk4[{rowadd1,3'b011}];
w26_reg = memblk4[{rowadd2,3'b011}];
w27_reg = memblk4[{rowadd3,3'b011}];
w28_reg = memblk4[{rowadd4,3'b011}];
w29_reg = memblk4[{rowadd5,3'b011}];
w30_reg = memblk4[{rowadd6,3'b011}];
w31_reg = memblk4[{rowadd7,3'b011}];
w32_reg = memblk4[{rowadd0,3'b100}];
w33_reg = memblk4[{rowadd1,3'b100}];
w34_reg = memblk4[{rowadd2,3'b100}];
w35_reg = memblk4[{rowadd3,3'b100}];
w36_reg = memblk4[{rowadd4,3'b100}];
w37_reg = memblk4[{rowadd5,3'b100}];
w38_reg = memblk4[{rowadd6,3'b100}];
w39_reg = memblk4[{rowadd7,3'b100}];
w40_reg = memblk4[{rowadd0,3'b101}];
w41_reg = memblk4[{rowadd1,3'b101}];
w42_reg = memblk4[{rowadd2,3'b101}];
w43_reg = memblk4[{rowadd3,3'b101}];
w44_reg = memblk4[{rowadd4,3'b101}];
w45_reg = memblk4[{rowadd5,3'b101}];
w46_reg = memblk4[{rowadd6,3'b101}];
w47_reg = memblk4[{rowadd7,3'b101}];
w48_reg = memblk4[{rowadd0,3'b110}];
w49_reg = memblk4[{rowadd1,3'b110}];
w50_reg = memblk4[{rowadd2,3'b110}];
w51_reg = memblk4[{rowadd3,3'b110}];
w52_reg = memblk4[{rowadd4,3'b110}];
w53_reg = memblk4[{rowadd5,3'b110}];
w54_reg = memblk4[{rowadd6,3'b110}];
w55_reg = memblk4[{rowadd7,3'b110}];
w56_reg = memblk4[{rowadd0,3'b111}];
w57_reg = memblk4[{rowadd1,3'b111}];
w58_reg = memblk4[{rowadd2,3'b111}];
w59_reg = memblk4[{rowadd3,3'b111}];
w60_reg = memblk4[{rowadd4,3'b111}];
w61_reg = memblk4[{rowadd5,3'b111}];
w62_reg = memblk4[{rowadd6,3'b111}];
w63_reg = memblk4[{rowadd7,3'b111}];

end
end
always @ (posedge rtranspose_clk) begin
if (data_out_enable_5==1'b1) begin
  coladd0 = '3b000;
coladd1 = '3b001;
coladd2 = '3b010;
coladd3 = '3b011;
coladd4 = '3b100;
coladd5 = '3b101;
coladd6 = '3b110;
coladd7 = '3b111;
end
w0 = w0_reg;
w1 = w1_reg;
w2 = w2_reg;
w3 = w3_reg;
w4 = w4_reg;
w5 = w5_reg;
w6 = w6_reg;
w7 = w7_reg;
w8 = w8_reg;
w9 = w9_reg;
w10 = w10_reg;
w11 = w11_reg;
w12 = w12_reg;
w13 = w13_reg;
w14 = w14_reg;
w15 = w15_reg;
w16 = w16_reg;
w17 = w17_reg;
w18 = w18_reg;
w19 = w19_reg;
w20 = w20_reg;
w21 = w21_reg;
w22 = w22_reg;
w23 = w23_reg;
w24 = w24_reg;
w25 = w25_reg;
w26 = w26_reg;
w27 = w27_reg;
w28 = w28_reg;
w29 = w29_reg;
w30 = w30_reg;
w31 = w31_reg;
w32 = w32_reg;
w33 = w33_reg;
w34 = w34_reg;
w35 = w35_reg;
w36 = w36_reg;
w37 = w37_reg;
w38 = w38_reg;
w39 = w39_reg;
w40 = w40_reg;
w41 = w41_reg;
w42 = w42_reg;
w43 = w43_reg;
w44 = w44_reg;
w45 = w45_reg;
w46 = w46_reg;
w47 = w47_reg;
w48 = w48_reg;
w49 = w49_reg;
w50 = w50_reg;
w51 = w51_reg;
w52 = w52_reg;
w53 = w53_reg;
w54 = w54_reg;
w55 = w55_reg;
w56 = w56_reg;
w57 = w57_reg;
w58 = w58_reg;
w59 = w59_reg;
w60 = w60_reg;
w61 = w61_reg;
w62 = w62_reg;
w63 = w63_reg;
end
endmodule

// Parallel to serial output module
module p_s_output (z0, z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12, z13, z14, z15, z16, z17, z18, z19, z20, z21, z22, z23, z24, z25, z26, z27, z28, z29, z30, z31, z32, z33, z34, z35, z36, z37, z38, z39, z40, z41, z42, z43, z44, z45, z46, z47, z48, z49, z50, z51, z52, z53, z54, z55, z56, z57, z58, z59, z60, z61, z62, z63, row_address, column_address, data_in_enable, parallel_clk, serial_clk, data_out);

input [11:0] z0, z1, z2, z3, z4, z5, z6, z7;
input [11:0] z8, z9, z10, z11, z12, z13, z14, z15;
input [11:0] z16, z17, z18, z19, z20, z21, z22, z23;
input [11:0] z24, z25, z26, z27, z28, z29, z30, z31;
input [11:0] z32, z33, z34, z35, z36, z37, z38, z39;
input [11:0] z40, z41, z42, z43, z44, z45, z46, z47;
input [11:0] z48, z49, z50, z51, z52, z53, z54, z55;
input [11:0] z56, z57, z58, z59, z60, z61, z62, z63;
input [2:0] row_address, column_address;
input data_in_enable;
input serial_clk, parallel_clk;
output [11:0] data_out;
output [1:0] select_out;

reg [11:0] data_out, z;
integer select_out;

// always @( z0 or z1 or z2 or z3 or z4 or z5 or z6 or z7 or z8 or z9 or z10 or z11 or z12 or z13 or z14 or z15 or z16 or z17 or z18 or z19 or z20 or z21 or z22 or z23 or z24 or z25 or z26 or z27 or z28 or z29 or z30 or z31 or z32 or z33 or z34 or z35 or z36 or z37 or z38 or z39 or z40 or z41 or z42 or z43 or z44 or z45 or z46 or z47 or z48 or z49 or z50 or z51 or z52 or z53 or z54 or z55 or z56 or z57 or z58 or z59 or z60 or z61 or z62 or z63 or serial_clk) begin
always begin
select_out = 7'd0;
while (select_out < 7'd64) begin

case (select_out)
  7'd0 : begin @(posedge serial_clk); data_out = z0; end
  7'd1 : begin @(posedge serial_clk); data_out = z1; end
  7'd2 : begin @(posedge serial_clk); data_out = z2; end
  7'd3 : begin @(posedge serial_clk); data_out = z3; end
  7'd4 : begin @(posedge serial_clk); data_out = z4; end
  7'd5 : begin @(posedge serial_clk); data_out = z5; end
  7'd6 : begin @(posedge serial_clk); data_out = z6; end
  7'd7 : begin @(posedge serial_clk); data_out = z7; end
  7'd8 : begin @(posedge serial_clk); data_out = z8; end
  7'd9 : begin @(posedge serial_clk); data_out = z9; end

endmodule

154
begin @(posedge serial_clk); data_out= z10; end
begin @(posedge serial_clk); data_out= z11; end
begin @(posedge serial_clk); data_out= z12; end
begin @(posedge serial_clk); data_out= z13; end
begin @(posedge serial_clk); data_out= z14; end
begin @(posedge serial_clk); data_out= z15; end
begin @(posedge serial_clk); data_out= z16; end
begin @(posedge serial_clk); data_out= z17; end
begin @(posedge serial_clk); data_out= z18; end
begin @(posedge serial_clk); data_out= z19; end
begin @(posedge serial_clk); data_out= z20; end
begin @(posedge serial_clk); data_out= z21; end
begin @(posedge serial_clk); data_out= z22; end
begin @(posedge serial_clk); data_out= z23; end
begin @(posedge serial_clk); data_out= z24; end
begin @(posedge serial_clk); data_out= z25; end
begin @(posedge serial_clk); data_out= z26; end
begin @(posedge serial_clk); data_out= z27; end
begin @(posedge serial_clk); data_out= z28; end
begin @(posedge serial_clk); data_out= z29; end
begin @(posedge serial_clk); data_out= z30; end
begin @(posedge serial_clk); data_out= z31; end
begin @(posedge serial_clk); data_out= z32; end
begin @(posedge serial_clk); data_out= z33; end
begin @(posedge serial_clk); data_out= z34; end
begin @(posedge serial_clk); data_out= z35; end
begin @(posedge serial_clk); data_out= z36; end
begin @(posedge serial_clk); data_out= z37; end
begin @(posedge serial_clk); data_out= z38; end
begin @(posedge serial_clk); data_out= z39; end
begin @(posedge serial_clk); data_out= z40; end
begin @(posedge serial_clk); data_out= z41; end
begin @(posedge serial_clk); data_out= z42; end
begin @(posedge serial_clk); data_out= z43; end
begin @(posedge serial_clk); data_out= z44; end
begin @(posedge serial_clk); data_out= z45; end
begin @(posedge serial_clk); data_out= z46; end
begin @(posedge serial_clk); data_out= z47; end
begin @(posedge serial_clk); data_out= z48; end
begin @(posedge serial_clk); data_out= z49; end
begin @(posedge serial_clk); data_out= z50; end
begin @(posedge serial_clk); data_out= z51; end
begin @(posedge serial_clk); data_out= z52; end
begin @(posedge serial_clk); data_out= z53; end
begin @(posedge serial_clk); data_out= z54; end
begin @(posedge serial_clk); data_out= z55; end
begin @(posedge serial_clk); data_out= z56; end
begin @(posedge serial_clk); data_out= z57; end
begin @(posedge serial_clk); data_out= z58; end
begin @(posedge serial_clk); data_out= z59; end
begin @(posedge serial_clk); data_out= z60; end
7'd61 : begin @(posedge serial_clk); data_out= z61; end
7'd62 : begin @(posedge serial_clk); data_out= z62; end
7'd63 : begin @(posedge serial_clk); data_out= z63; end
endcase
    select_out=select_out + 7'd1;
end
end

endmodule

// HDL description of the 8x8 1-D binDCT-B
// normalized version
// Created by: Tracy Franklin
module bindct8 (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, data_in_enable, z0, z1, z2, z3, z4, z5, z6, z7);
input [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input clk,reset;
input data_in_enable;
output [9:0] z0,z1,z2,z3,z4,z5,z6,z7;

wire [8:0] a0,a1,a2,a3;
wire [7:0] a4,a5,a6,a7;
wire [8:0] b5,b6;
wire [9:0] c0,c1,c4,c7;
wire [8:0] c2,c3,c5,c6;
wire [9:0] d0;
wire [8:0] d4,d1,d7,d5,d2;
wire [7:0] d3,d6;
wire [9:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [9:0] z0,z1,z2,z3,z4,z5,z6,z7;
reg [7:0] x0_in, x1_in, x2_in, x3_in, x4_in, x5_in, x6_in, x7_in;

always @(data_in_enable or x0 or x1 or x2 or x3 or x4 or x5 or x6 or x7 or clk) begin
    if (data_in_enable == 1'b1) begin
        x0_in=x0;
        x1_in=x1;
        x2_in=x2;
        x3_in=x3;
        x4_in=x4;
        x5_in=x5;
        x6_in=x6;
        x7_in=x7;
    end
end

preadd f1 (x0_in, x1_in, x2_in, x3_in, x4_in, x5_in, x6_in, x7_in, clk, reset, a0, a1, a2, a3, a4, a5, a6, a7);
prelift f2 (a5, a6, clk, reset, b5, b6);
midadd f3 (a0, a1, a2, a3, a4, b5, b6, a7, clk, reset, c0, c1, c2, c3, c4, c5, c6, c7);
postlift f4 (c0, c1, c2, c3, c4, c5, c6, c7, clk, reset, d0, d1, d2, d3, d4, d5, d6, d7);
rearrange f5 (d0, d1, d2, d3, d4, d5, d6, d7, clk, reset, e0, e1, e2, e3, e4, e5, e6, e7);

always @(posedge clk or negedge reset) begin
    if (!reset) begin
        z0=0;
    end
end

156
module preadd (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, a0, a1, a2, a3, a4, a5, a6, a7);
input [7:0] x0,x1,x2,x3,x4,x5,x6,x7;
input clk,reset;
output [8:0] a0,a1,a2,a3;
output [7:0] a4,a5,a6,a7;

add8 r1 (x0,x7,clk,reset,a0);
add8 r2 (x1,x6,clk,reset,a1);
add8 r3 (x2,x5,clk,reset,a2);
add8 r4 (x3,x4,clk,reset,a3);
sub8 r5 (x3,x4,clk,reset,a4);
sub8 r6 (x2,x5,clk,reset,a5);
sub8 r7 (x1,x6,clk,reset,a6);
sub8 r8 (x0,x7,clk,reset,a7);
endmodule

module prelift (a5, a6, clk, reset, b5, b6);
input [7:0] a5,a6;
input clk,reset;
output [8:0] b5,b6;

wire [6:0] temp1;
wire [8:0] temp2;
wire [8:0] b6;

ls388 r1 (a5,clk,reset,temp1);
add8 r2 ((1 b0,temp1),a6,clk,reset,b6);
ls589 r3 (b6,clk,reset,temp2);
sub9 r4 (temp2,(1 b0,a5),clk,reset,b5);
endmodule

module midadd (a0, a1, a2, a3, a4, b5, b6, a7, clk, reset, c0, c1, c2, c3, c4, c5, c6, c7);
input [8:0] a0,a1,a2,a3,b5,b6;
input [7:0] a4, a7;
input clk, reset;
output [9:0] c0, c1, c4, c7;
output [8:0] c2, c3, c5, c6;

add9 m1 (a0, a3, clk, reset, c0);
add9 m2 (a1, a2, clk, reset, c1);
sub9 m3 (a1, a2, clk, reset, c2);
sub9 m4 (a0, a3, clk, reset, c3);
add9 m5 ((1'b0, a4), b5, clk, reset, c4);
sub9 m6 ((1'b0, a4), b5, clk, reset, c5);
sub9 m7 ((1'b0, a7), b6, clk, reset, c6);
add9 m8 ((1'b0, a7), b6, clk, reset, c7);
endmodule

module postlift (c0, c1, c2, c3, c4, c5, c6, c7, clk, reset, d0, d1, d2, d3, d4, d5, d6, d7);
input [9:0] c0, c1, c4, c7;
input [8:0] c2, c3, c5, c6;
input clk, reset;
output [9:0] d0;
output [8:0] d4, d1, d7, d5, d2;
output [7:0] d3, d6;

wire [9:0] temp3;
wire [8:0] temp5, temp6;
wire [7:0] temp7, temp9, temp10;
wire [6:0] temp4, temp8;

wire [10:0] temp20;
wire [9:0] temp21, temp22, temp24, temp25, temp27;
wire [8:0] temp23, temp26;

add10 o1 (c0, c1, clk, reset, temp20);
ls121 o2 (temp20, clk, reset, temp3);
sub10 o3 (temp3, c1, clk, reset, temp24);
ls1810 o4 (c7, clk, reset, temp4);
sub10 o5 (c4, {3'b000}, temp4, clk, reset, temp27);
ls349 o6 (c5, clk, reset, temp5);
sub9 o7 (c6, temp5, clk, reset, temp6);
ls129 o8 (temp6, clk, reset, temp7);
add9 o9 (c5, {1'b0, temp7}, clk, reset, temp25);
ls1810 o10 (temp25, clk, reset, temp8);
sub9 o11 (temp6, {2'b00}, temp8, clk, reset, temp23);
ls389 o12 (c3, clk, reset, temp9);
sub9 o13 (c2, {1'b0, temp9}, clk, reset, temp26);
ls389 o14 (temp26, clk, reset, temp10);
add9 o15 (c3, {1'b0, temp10}, clk, reset, temp22);
assign temp21 = c7;

scale_by_2_10 k1 (temp20, reset, d0);
scale_by_2_9 k2 (temp21, reset, d1);
scale_by_2_9 k3 (temp22, reset, d2);
scale_by_2_8 k4 (temp23, reset, d3);
scale_by_2_9 k5 (temp24, reset, d4);
scale_by_2_9 k6 (temp25, reset, d5);
scale_by_2_8 k7 (temp26, reset, d6);
scale_by_2_9 k8 (temp27,reset,d7);
endmodule

module rearrange (d0, d1, d2, d3, d4, d5, d6, d7, clk, reset, e0, e1, e2, e3, e4, e5, e6, e7);
input [9:0] d0;
input [8:0] d1,d2,d4,d5,d7;
input [7:0] d3,d6;
input clk,reset;
output [9:0] e0,e1,e2,e3,e4,e5,e6,e7;

reg [9:0] e0,e1,e2,e3,e4,e5,e6,e7;

always @(d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
if (!reset) begin
  e0=0;
  e1=0;
  e2=0;
  e3=0;
  e4=0;
  e5=0;
  e6=0;
  e7=0;
end
else begin
  e0=d0;
  e1=(1'b0,d1);
  e2=(1'b0,d2);
  e3=(2'b00,d3);
  e4=(1'b0,d4);
  e5=(1'b0,d5);
  e6=(2'b00,d6);
  e7=(1'b0,d7);
end
end
endmodule

module scale_by_2_8 (x,reset,y);
input [8:0] x;
input reset;
output [7:0] y;

reg [7:0] y;

always @(x or reset) begin
if (!reset) begin
  y=0;
end
else begin
  y=(x>>1);
end
end
endmodule

module scale_by_2_9 (x,reset,y);
input [9:0] x;

endmodule
input reset;
output [8:0] y;

reg [8:0] y;

always @(x or reset) begin
    if (!reset) begin
        y=0;
    end
    else begin
        y=(x>>1);
    end
end
endmodule

module scale_by_2_10 (x,reset,y);
input [10:0] x;
input reset;
output [9:0] y;

reg [9:0] y;

always @(x or reset) begin
    if (!reset) begin
        y=0;
    end
    else begin
        y=(x>>1);
    end
end
endmodule

module add8 (x,y,clk,reset,sum);
input [7:0] x,y;
input clk,reset;
output [8:0] sum;

reg [8:0] sum;

always @(x or y or reset) begin
    if (!reset) begin
        sum=0;
    end
    else begin
        sum=x+y;
    end
end
endmodule

module sub8 (x,y,clk,reset,diff);
input [7:0] x,y;
input clk,reset;
output [7:0] diff;

reg [7:0] diff;
always @(x or y or reset) begin
    if (!reset) begin
        diff=0;
    end
    else begin
        diff=x-y;
    end
end
endmodule

module ls388 (x,clk,reset,lift);
input [7:0] x;
input clk,reset;
output [6:0] lift;

wire [8:0] shiftx;
wire [9:0] sumx;

reg [6:0] lift;

assign shiftx = (1'b0,x)<<1;
add9 u1 (shiftx,[1'b0,x],clk,reset,sumx);

always @(sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
endmodule

module ls589 (x,clk,reset,lift);
input [8:0] x;
input clk,reset;
output [8:0] lift;

wire [10:0] shiftx;
wire [11:0] sumx;

reg [8:0] lift;

assign shiftx = (2'b00,x)<<2;
add11 u2 (shiftx,[2'b00,x],clk,reset,sumx);

always @(sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift=(sumx>>3);
    end
end
endmodule
endmodule

module add9 (x,y,clk,reset,sum);
  input [8:0] x,y;
  input clk,reset;
  output [9:0] sum;

  reg [9:0] sum;

  always @(x or y or reset) begin
    if (!reset) begin
      sum=0;
      end
    else begin
      sum=x+y;
      end
    end
  end

endmodule

module sub9 (x,y,clk,reset,diff);
  input [8:0] x,y;
  input clk,reset;
  output [8:0] diff;

  reg [8:0] diff;

  always @(x or y or reset) begin
    if (!reset) begin
      diff=0;
      end
    else begin
      diff=x-y;
      end
    end
  end

endmodule

module add10 (x,y,clk,reset,sum);
  input [9:0] x,y;
  input clk,reset;
  output [10:0] sum;

  reg [10:0] sum;

  always @(x or y or reset) begin
    if (!reset) begin
      sum=0;
      end
    else begin
      sum=x+y;
      end
    end
  end

endmodule

module sub10 (x,y,clk,reset,diff);
  input [9:0] x,y;
input clk, reset;
output [9:0] diff;

reg [9:0] diff;

always @ (x or y or reset) begin
  if (!reset) begin
    diff = 0;
  end
  else begin
    diff = x - y;
  end
end
endmodule

module add11 (x, y, clk, reset, sum);
input [10:0] x, y;
input clk, reset;
output [11:0] sum;

reg [11:0] sum;

always @ (x or y or reset) begin
  if (!reset) begin
    sum = 0;
  end
  else begin
    sum = x + y;
  end
end
endmodule

module lsl11 (x, clk, reset, lift);
input [10:0] x;
input clk, reset;
output [9:0] lift;

reg [9:0] lift;

always @ (x or reset) begin
  if (!reset) begin
    lift = 0;
  end
  else begin
    lift = (x >> 1);
  end
end
endmodule

module lsl29 (x, clk, reset, lift);
input [8:0] x;
input clk, reset;
output [7:0] lift;

reg [7:0] lift;
always @ (x or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift = (x >> 1);
  end
end
endmodule

module ls1810 (x, clk, reset, lift);
input [9:0] x;
input clk, reset;
output [6:0] lift;

reg [6:0] lift;

always @ (x or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(x>>3);
  end
end
endmodule

module ls349 (x, clk, reset, lift);
input [8:0] x;
input clk, reset;
output [8:0] lift;

wire [9:0] shiftx;
wire [10:0] sumx;

reg [8:0] lift;

assign shiftx=(l'b0,x)<<1;
add10 u3 (shiftx, l'b0,x, clk, reset, sumx);

always @ (sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    lift=(sumx << 2);
  end
end
endmodule

module ls389 (x, clk, reset, lift);
input [8:0] x;
input clk, reset;
output [7:0] lift;

wire [9:0] shiftx;
wire [10:0] sumx:

reg [7:0] lift;

assign shiftx= ((1'b0,x)<<1;
add10 u4 (shiftx.1'b0,x,clk.reset,sumx);

always @(sumx or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        lift= (sumx >> 3);
    end
end
endmodule

// HDL description of the 8x8 1-D binDCT-B
// normalized version
// Created by: Tracy Franklin

timescale 1ns/10ps

module binDCT10 (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, data_in_enable, z0, z1, z2, z3, z4, z5, z6, z7);
input [9:0] x0,x1,x2,x3,x4,x5,x6,x7;
input clk, reset;
input data_in_enable;
output [11:0] z0,z1,z2,z3,z4,z5,z6,z7;

wire [10:0] a0,a1,a2,a3;
wire [9:0] a4,a5,a6,a7;
wire [10:0] b5,b6;
wire [11:0] c0,c1,c4,c7;
wire [10:0] c2,c3,c5,c6;
wire [11:0] d0;
wire [10:0] d4,d1,d7,d5,d2;
wire [9:0] d3,d6;
wire [11:0] e0,e1,e2,e3,e4,e5,e6,e7;
reg [11:0] x0,z1,z2,z3,z4,z5,z6,z7;
reg [9:0] x0_in, x1_in, x2_in, x3_in, x4_in, x5_in, x6_in, x7_in;
reg data_out_enable;

always @(data_in_enable or clk or x0 or x1 or x2 or x3 or x4 or x5 or x6 or x7) begin
    if (data_in_enable==1'b1) begin
        x0_in=x0;
        x1_in=x1;
        x2_in=x2;
        x3_in=x3;
        x4_in=x4;
        x5_in=x5;
        x6_in=x6;
        x7_in=x7;
    end
end
always @ (posedge clk or negedge reset) begin
    if (reset) begin
        z0 = 0;
        z1 = 0;
        z2 = 0;
        z3 = 0;
        z4 = 0;
        z5 = 0;
        z6 = 0;
        z7 = 0;
    end
    else begin
        z0 = e0;
        z1 = e1;
        z2 = e2;
        z3 = e3;
        z4 = e4;
        z5 = e5;
        z6 = e6;
        z7 = e7;
    end
end
endmodule

module preaddl (x0, x1, x2, x3, x4, x5, x6, x7, clk, reset, a0, a1, a2, a3, a4, a5, a6, a7);
input [9:0] x0, x1, x2, x3, x4, x5, x6, x7;
input clk, reset;
output [10:0] a0, a1, a2, a3;
output [9:0] a4, a5, a6, a7;
add81 re1 (x0, x7, clk, reset, a0);
add81 re2 (x1, x6, clk, reset, a1);
add81 re3 (x2, x5, clk, reset, a2);
add81 re4 (x3, x4, clk, reset, a3);
sub81 re5 (x3, x4, clk, reset, a4);
sub81 re6 (x2, x5, clk, reset, a5);
sub81 re7 (x1, x6, clk, reset, a6);
sub81 re8 (x0, x7, clk, reset, a7);
endmodule

module prelift1 (a5, a6, clk, reset, b5, b6);
input [9:0] a5, a6;
input clk, reset;
output [10:0] b5, b6;
wire [8:0] temp1;
wire [10:0] temp2;
wire [10:0] b6;
ls3881 r1 l (a5, clk, reset, temp1);
add81 r2 ([1:0], temp1, a6, clk, reset, b6);
ls5891 r3 (b6, clk, reset, temp2);
sub91 r4 (temp2, [1:0], a5, clk, reset, b5);
endmodule

module midadd1 (a0, a1, a2, a3, a4, b5, b6, a7, clk, reset, c0, c1, c2, c3, c4, c5, c6, c7);
input [10:0] a0, a1, a2, a3, b5, b6;
input [9:0] a4, a7;
input clk, reset;
output [11:0] c0, c1, c2, c3, c4, c7;
output [10:0] c2, c3, c5, c6;
add91 ma1 (a0, a3, clk, reset, c0);
add91 ma2 (a1, a2, clk, reset, c1);
sub91 ma3 (a1, a2, clk, reset, c2);
sub91 ma4 (a0, a3, clk, reset, c3);
add91 ma5 ([1:0], a4, b5, clk, reset, c4);
sub91 ma6 ([1:0], a4, b5, clk, reset, c5);
sub91 ma7 ([1:0], a7, b6, clk, reset, c6);
add91 ma8 ([1:0], a7, b6, clk, reset, c7);
endmodule

module postlift1 (c0, c1, c2, c3, c4, c5, c6, c7, clk, reset, d0, d1, d2, d3, d4, d5, d6, d7);
input [11:0] c0, c1, c4, c7;
input [10:0] c2, c3, c5, c6;
input clk, reset;
output [11:0] d0;
output [10:0] d4, d1, d7, d5, d2;
output [9:0] d3, d6;
wire [11:0] temp3;
wire [10:0] temp5, temp6;
wire [9:0] temp7, temp9, temp10;
wire [8:0] temp4, temp8;
wire [12:0] temp20;
wire [11:0] temp21, temp22, temp24, temp25, temp27;
wire [10:0] temp23, temp26;
add101 ol1 (c0, c1, clk, reset, temp20);
ls12111 ol2 (temp20, clk, reset, temp3);
sub101 ol3 (temp3, c1, clk, reset, temp24);
ls18101 ol4 (c7, clk, reset, temp4);
sub101 ol5 (c4, [3:000], temp4, clk, reset, temp27);
ls3491 ol6 (c5, clk, reset, temp5);
sub91 ol7 (c6, temp5, clk, reset, temp6);
ls1291 ol8 (temp6, clk, reset, temp7);
add91 ol9 (c5, [1:0], clk, reset, temp25);
ls18101 ol10 (temp25, clk, reset, temp8);
sub91 ol11 (temp6, [2:000], clk, reset, temp23);
ls3891 ol12 (c3, clk, reset, temp9);
sub91 ol13 (c2, [1:0], clk, reset, temp26);
ls3891 ol14 (temp26, clk, reset, temp10);
add91 ol15 (c3, [1:0], clk, reset, temp22);
assign temp21 = c7;

scale_by_2_101 ks1 (temp20, reset, d0);
scale_by_2_91 ks2 (temp21, reset, d1);
scale_by_2_91 ks3 (temp22, reset, d2);
scale_by_2_81 ks4 (temp23, reset, d3);
scale_by_2_91 ks5 (temp24, reset, d4);
scale_by_2_91 ks6 (temp25, reset, d5);
scale_by_2_81 ks7 (temp26, reset, d6);
scale_by_2_91 ks8 (temp27, reset, d7);
endmodule

module rearrange1 (d0, d1, d2, d3, d4, d5, d6, d7, clk, reset, e0, e1, e2, e3, e4, e5, e6, e7);
input [11:0] d0;
input [10:0] d1, d2, d4, d5, d7;
input [9:0] d3, d6;
input clk, reset;
output [11:0] e0, e1, e2, e3, e4, e5, e6, e7;
reg [11:0] e0, e1, e2, e3, e4, e5, e6, e7;

always @(d0 or d1 or d2 or d3 or d4 or d5 or d6 or d7 or reset) begin
if (reset) begin
    e0 = 0;
    e1 = 0;
    e2 = 0;
    e3 = 0;
    e4 = 0;
    e5 = 0;
    e6 = 0;
    e7 = 0;
end
else begin
    e0 = d0;
    e1 = {1'b0, d1};
    e2 = {1'b0, d2};
    e3 = {2'b00, d3};
    e4 = {1'b0, d4};
    e5 = {1'b0, d5};
    e6 = {2'b00, d6};
    e7 = {1'b0, d7};
end
endmodule

module scale_by_2_81 (x, reset, y);
input [10:0] x;
input reset;
output [9:0] y;
reg [9:0] y;

always @(x or reset) begin
    if (!reset) begin
        y = 0;
    end
else begin
  y=(x>>1);
end
end
endmodule

module scale_by_2_91 (x,reset,y);
  input [11:0] x;
  input reset;
  output [10:0] y;

  reg [10:0] y;
  always @(x or reset) begin
    if (!reset) begin
      y=0;
      end
    else begin
      y=(x>>1);
    end
  end
endmodule

module scale_by_2_101 (x,reset,y);
  input [12:0] x;
  input reset;
  output [11:0] y;

  reg [11:0] y;
  always @(x or reset) begin
    if (!reset) begin
      y=0;
    end
    else begin
      y=(x>>1);
    end
  end
endmodule

module add81 (x,y,clk,reset,sum);
  input [9:0] x,y;
  input clk,reset;
  output [10:0] sum;

  reg [10:0] sum;
  always @(x or y or reset) begin
    if (!reset) begin
      sum=0;
    end
    else begin
      sum=x+y;
    end
  end
end
dendmodule

module sub81 (x,y,clk,reset,diff);
input [9:0] x,y;
input clk,reset;
output [9:0] diff;
reg [9:0] diff;
always @(x or y or reset) begin
  if (reset) begin
    diff=0;
  end
  else begin
    diff=x-y;
  end
end
dendmodule

module ls3881 (x,clk,reset,lift);
input [9:0] x;
input clk,reset;
output [8:0] lift;
wire [10:0] shiftx;
wire [11:0] sumx;
reg [8:0] lift;
reg [11:0] temp;
assign shiftx = (1'b0, x)<<1;
add91 u1 (shiftx, (1'b0, x), clk, reset, sumx);
always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    temp=(sumx<<3);
    lift=temp[8:0];
  end
end
dendmodule

module ls5891 (x,clk,reset,lift);
input [10:0] x;
input clk,reset;
output [10:0] lift;
wire [12:0] shiftx;
wire [13:0] sumx;
reg [10:0] lift;
reg [13:0] temp;
assign shiftx = ((2'b00,x1)<<2;
add111 u2 (shiftx,[2b00,x1].clk,reset,sumx);

always @(sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    temp=(sumx>>3);
    lift= temp[10:0];
  end
end
endmodule

module add91 (x,y,clk,reset,sum);
input [10:0] x,y;
input clk,reset;
output [11:0] sum;

reg [11:0] sum;

always @(x or y or reset) begin
  if (!reset) begin
    sum=0;
  end
  else begin
    sum=x+y;
  end
end
endmodule

module sub91 (x,y,clk,reset,diff);
input [10:0] x,y;
input clk,reset;
output [10:0] diff;

reg [10:0] diff;

always @(x or y or reset) begin
  if (!reset) begin
    diff=0;
  end
  else begin
    diff=x-y;
  end
end
endmodule

module add101 (x,y,clk,reset,sum);
input [11:0] x,y;
input clk,reset;
output [12:0] sum;

reg [12:0] sum;
always @(x or y or reset) begin
    if (reset) begin
        sum = 0;
    end
    else begin
        sum = x + y;
    end
end
endmodule

module sub101(x,y,clk,reset,diff);
    input [11:0] x, y;
    input clk, reset;
    output [11:0] diff;
    reg [11:0] diff;
always @(x or y or reset) begin
    if (reset) begin
        diff = 0;
    end
    else begin
        diff = x - y;
    end
end
endmodule

module add111(x,y,clk,reset,sum);
    input [12:0] x, y;
    input clk, reset;
    output [13:0] sum;
    reg [13:0] sum;
always @(x or y or reset) begin
    if (reset) begin
        sum = 0;
    end
    else begin
        sum = x + y;
    end
end
endmodule

module ls12111(x,clk,reset,lift);
    input [12:0] x;
    input clk, reset;
    output [11:0] lift;
    reg [11:0] lift;
    reg [12:0] temp;
always @(x or reset) begin
    if (reset) begin
        lift = 0;
    end
end
else begin
    temp = (x >> 1);
    lift = temp[11:0];
end
endmodule

module ls1291 (x,clk,reset,lift);
input [10:0] x;
input clk,reset;
output [9:0] lift;
reg [10:0] temp;
reg [9:0] lift;

always @(x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        temp = (x >> 1);
        lift = temp[9:0];
    end
end
endmodule

module ls18101 (x,clk,reset,lift);
input [11:0] x;
input clk,reset;
output [8:0] lift;
reg [11:0] temp;
reg [8:0] lift;

always @(x or reset) begin
    if (!reset) begin
        lift=0;
    end
    else begin
        temp=(x>>3);
        lift=temp[8:0];
    end
end
endmodule

module ls3491 (x,clk,reset,lift);
input [10:0] x;
input clk,reset;
output [10:0] lift;

wire [11:0] shiftx;
wire [12:0] sumx;

reg [12:0] temp;
reg [10:0] lift;

assign shiftx=(1'b0,x)<<1;
add101 u3 (shiftx,(1'b0,x),clk,reset,sumx);
always @ (sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    temp=(sumx << 2);
    lift=temp[10:0];
  end
end
endmodule

module ls3891 (x,clk,reset,lift);
input [10:0] x;
input clk,reset;
output [9:0] lift;

wire [11:0] shiftx;
wire [12:0] sumx;

reg [12:0] temp;
reg [9:0] lift;

assign shiftx= ((1'b0,x)<<1;
add101 u4 (shiftx,(1'b0,x).clk,reset,sumx);

always @ (sumx or reset) begin
  if (!reset) begin
    lift=0;
  end
  else begin
    temp= (sumx >> 3);
    lift=temp[9:0];
  end
end
endmodule

G.2 The test bench for the 2-D binDCT chip

`timescale 1ns/10ps
module stimulus_bindct_2d ();
reg [7:0] data_in;
reg [2:0] row_address;
reg [2:0] column_address;
reg data_input_enable;
reg serial_clock;
reg bindct_clock;
reg parallel_clock;
reg transpose_clock;
reg out_clock;
wire [11:0] data_out;

// RTL - level
dct2dchip tester1 (data_in, row_address, column_address, data_input_enable, serial_clock, parallel_clock,
bindct_clock, transpose_clock, out_clock, data_out);
// Gate -level
// dct2dchip tester2 (data_in[0], data_in[1], data_in[2], data_in[3], data_in[4], data_in[5], data_in[6],
data_in[7], row_address[0], row_address[1], row_address[2], column_address[0], column_address[1],
column_address[2], data_input_enable, serial_clock, parallel_clock, bindct_clock, transpose_clock,
out_clock, data_out[0], data_out[1], data_out[2], data_out[3], data_out[4], data_out[5], data_out[6],
data_out[7], data_out[8], data_out[9], data_out[10], data_out[11]);

initial
begin
    serial_clock = 0;
    bindct_clock = 0;
    parallel_clock = 0;
    transpose_clock = 0;
    out_clock = 0;
    #16500 $finish;
end

always #10 serial_clock = ~serial_clock;
always #1500 bindct_clock = ~bindct_clock;
always #1500 parallel_clock = ~parallel_clock;
always #1500 transpose_clock = ~transpose_clock;
always #1 out_clock = ~out_clock;

initial begin
    #20 data_input_enable=1'b1;
    #20 row_address=3'd0; column_address=3'd0; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd1; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd2; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd3; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd4; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd5; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd6; data_in=8'd255;
    #20 row_address=3'd0; column_address=3'd7; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd0; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd1; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd2; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd3; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd4; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd5; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd6; data_in=8'd255;
    #20 row_address=3'd1; column_address=3'd7; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd0; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd1; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd2; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd3; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd4; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd5; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd6; data_in=8'd255;
    #20 row_address=3'd2; column_address=3'd7; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd0; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd1; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd2; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd3; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd4; data_in=8'd255;
    #20 row_address=3'd3; column_address=3'd5; data_in=8'd255;

175
always @(posedge out_clock) begin
$monitor("Time is %d, the output data is %d\n",Stime, data_out);
//$monitor("Time is %d, the output data is %d\n",Stime, data_out);

endmodule
Appendix H

The results for the gate-level constraints for

The design of the binDCT chips

H.1 Initial no gate-level constraint results for the forward binDCT

***********************************************************
Report: area
Design: forwarddct
Version: 1998.02
Date: Wed May 24 14:49:46 2000
***********************************************************

Library(s) Used:

    tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Number of ports: 31
Number of nets: 183
Number of cells: 3
Number of references: 3

Combinational area: 162155.00000
Noncombinational area: 74585.00000
Net Interconnect area: undefined (Wire load has zero net area)

Total cell area: 236740.00000
Total area: undefined

Performing power analysis through design. (low effort)
Warning: There is no defined clock in the design. (PWR-80)
Warning: There are sequential cells with no output activity annotation. (PWR-96)

***********************************************************
Report: power
    -analysis_effort low
Design: forwarddct
Version: 1998.02
Date: Wed May 24 14:51:51 2000
***********************************************************

Library(s) Used:

    tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Operating Conditions: WCCOM  Library: tcb773pwc
Wire Loading Model Mode: segmented

Design Wire Loading Model Library
forwarddt  TSMC8K_Conservative
tcb773pwc
s_p_in8   TSMC8K_Conservative
tcb773pwc
bindct8   TSMC8K_Conservative
tcb773pwc
preadd    TSMC8K_Conservative
tcb773pwc
add8_4    TSMC8K_Conservative
tcb773pwc
add8_4 DW01_add_9 0  TSMC8K_Conservative
tcb773pwc
sub8_3    TSMC8K_Conservative
tcb773pwc
sub8_3 DW01_sub_8 0  TSMC8K_Conservative
tcb773pwc
sub8_2    TSMC8K_Conservative
tcb773pwc
sub8_2 DW01_sub_8 0  TSMC8K_Conservative
tcb773pwc
add8_3    TSMC8K_Conservative
tcb773pwc
add8_3 DW01_add_9 0  TSMC8K_Conservative
tcb773pwc
add8_2    TSMC8K_Conservative
tcb773pwc
add8_2 DW01_add_9 0  TSMC8K_Conservative
tcb773pwc
sub8_1    TSMC8K_Conservative
tcb773pwc
sub8_1 DW01_sub_8 0  TSMC8K_Conservative
tcb773pwc
add8_1    TSMC8K_Conservative
tcb773pwc
add8_1 DW01_add_9 0  TSMC8K_Conservative
tcb773pwc
sub8_0    TSMC8K_Conservative
tcb773pwc
sub8_0 DW01_sub_8 0  TSMC8K_Conservative
tcb773pwc
prelift   TSMC8K_Conservative
tcb773pwc
ls388     TSMC8K_Conservative
tcb773pwc
add9_6    TSMC8K_Conservative
tcb773pwc
add9_6 DW01_add_10 0  TSMC8K_Conservative
tcb773pwc
add8_0    TSMC8K_Conservative
tcb773pwc
add8_0 DW01_add_9 0  TSMC8K_Conservative
tcb773pwc
ls589     TSMC8K_Conservative
tcb773pwc
add11     TSMC8K_Conservative
Global Operating Voltage = 3
Power-specific unit information:
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW  (derived from V,C,T units)
Leakage Power Units = 1pW

Cell Internal Power = 83.8404 mW  (49%)
Net Switching Power = 87.8458 mW  (51%)
---------
Total Dynamic Power = 171.6862 mW  (100%)
Cell Leakage Power = 1.5630 nW

1
design analyzer>
*******************************************************************************
Report: timing
-path full
-delay max
-max_paths 1
Design: forwarddct
Version: 1998.02
Date: Wed May 24 14:51:52 2000
*******************************************************************************

Operating Conditions: WCCOM  Library: tcb773pwc
Wire Loading Model Mode: segmented

<table>
<thead>
<tr>
<th>Design</th>
<th>Wire Loading Model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarddct</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>s_p_in8</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>bindct8</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>preadd</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_4</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_4_DW01_add_9_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub8_3</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub8_3_DW01_sub_8_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub8_2</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub8_2_DW01_sub_8_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_3</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_3_DW01_add_9_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_2</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add8_2_DW01_add_9_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
</tbody>
</table>

182
H.2 Initial no gate-level constraint results for the inverse binDCT chip

******************************************************************************
Report : area
Design : inversedct
Version: 1998.02
Date : Wed May 24 15:37:21 2000
******************************************************************************

Library(s) Used:

tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Number of ports: 36
Number of nets: 228
Number of cells: 3

185
Number of references: 3

Combinational area: 228147.500000
Noncombinational area: 93555.000000
Net Interconnect area: undefined (Wire load has zero net area)

Total cell area: 321702.500000
Total area: undefined

Performing power analysis through design. (low effort)
Warning: There is no defined clock in the design. (PWR-80)
Warning: There are sequential cells with no output activity annotation. (PWR-96)

*************************
Report : power
- analysis effort low
Design : inversedct
Version: 1998.02
Date : Wed May 24 15:43:53 2000
*************************

Library(s) Used:

tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Operating Conditions: WCCOM Library: tcb773pwc
Wire Loading Model Mode: segmented

<table>
<thead>
<tr>
<th>Design</th>
<th>Wire Loading Model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>inversedct</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>s_p_in11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ibindct11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>preliftadd</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1211</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_3</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_3_DW01_add_12_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_2</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_2_DW01_add_12_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1811_1</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1811_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub11_4</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
</tbody>
</table>
sub11_4_DW01_sub_11_0  TSMC8K_Conservative
tcb773pwc
sub11_3  TSMC8K_Conservative
tcb773pwc
sub11_3_DW01_sub_11_0  TSMC8K_Conservative
tcb773pwc
ls3811_1  TSMC8K_Conservative
tcb773pwc
add12_5  TSMC8K_Conservative
tcb773pwc
add12_5_DW01_add_13_0  TSMC8K_Conservative
tcb773pwc
ls3811_0  TSMC8K_Conservative
tcb773pwc
add12_4  TSMC8K_Conservative
tcb773pwc
add12_4_DW01_add_13_0  TSMC8K_Conservative
tcb773pwc
add11_1  TSMC8K_Conservative
tcb773pwc
add11_1_DW01_add_12_0  TSMC8K_Conservative
tcb773pwc
sub11_2  TSMC8K_Conservative
tcb773pwc
sub11_2_DW01_sub_11_0  TSMC8K_Conservative
tcb773pwc
ls3411  TSMC8K_Conservative
tcb773pwc
add12_3  TSMC8K_Conservative
tcb773pwc
add12_3_DW01_add_13_0  TSMC8K_Conservative
tcb773pwc
add12_2  TSMC8K_Conservative
tcb773pwc
add12_2_DW01_add_13_0  TSMC8K_Conservative
tcb773pwc
sub11_1  TSMC8K_Conservative
tcb773pwc
sub11_1_DW01_sub_11_0  TSMC8K_Conservative
tcb773pwc
ls1212  TSMC8K_Conservative
tcb773pwc
midaddsub  TSMC8K_Conservative
tcb773pwc
add11_0  TSMC8K_Conservative
tcb773pwc
add11_0_DW01_add_12_0  TSMC8K_Conservative
tcb773pwc
sub11_0  TSMC8K_Conservative
tcb773pwc
sub11_0_DW01_sub_11_0  TSMC8K_Conservative
tcb773pwc
add12_1  TSMC8K_Conservative
tcb773pwc
add12_1_DW01_add_13_0  TSMC8K_Conservative
tcb773pwc
Global Operating Voltage = 3
Power-specific unit information:
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1pW

Cell Internal Power = 209.5900 mW (57%)
Net Switching Power = 156.3230 mW (43%)
---------
Total Dynamic Power = 365.9130 mW (100%)
Cell Leakage Power = 2.1300 nW

1
design_analyzer>
************************************************
Report : timing
-path full
-delay max
-nworst 3
-max_paths 8
Design : inveredct
Version: 1998.02
Date : Wed May 24 15:43:57 2000
************************************************

Operating Conditions: WCCOM  Library: tcb773pwc
Wire Loading Model Mode: segmented

<table>
<thead>
<tr>
<th>Design</th>
<th>Wire Loading Model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>inveredct</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>s_p_in11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ibindct11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>preliftadd</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1211</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_3</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_3_DW01_add_12_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_2</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add11_2_DW01_add_12_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1811_1</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls1811_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub11_4</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub11_4_DW01_sub_11_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub11_3</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>sub11_3_DW01_sub_11_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ls3811_1</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add12_5</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>add12_5_DW01_add_13_0</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
</tbody>
</table>
ls3811_0 TSMC8K_Conservative
tcb773pwc
add12_4 TSMC8K_Conservative
tcb773pwc
add12_4_DW01_add_13_0 TSMC8K_Conservative
tcb773pwc
add11_1 TSMC8K_Conservative
tcb773pwc
add11_1_DW01_add_12_0 TSMC8K_Conservative
tcb773pwc
sub11_2 TSMC8K_Conservative
tcb773pwc
sub11_2_DW01_sub_11_0 TSMC8K_Conservative
tcb773pwc
ls3411 TSMC8K_Conservative
tcb773pwc
add12_3 TSMC8K_Conservative
tcb773pwc
add12_3_DW01_add_13_0 TSMC8K_Conservative
tcb773pwc
add12_2 TSMC8K_Conservative
tcb773pwc
add12_2_DW01_add_13_0 TSMC8K_Conservative
tcb773pwc
sub11_1 TSMC8K_Conservative
tcb773pwc
sub11_1_DW01_sub_11_0 TSMC8K_Conservative
tcb773pwc
ls1212 TSMC8K_Conservative
tcb773pwc
midaddsub TSMC8K_Conservative
tcb773pwc
add11_0 TSMC8K_Conservative
tcb773pwc
add11_0_DW01_add_12_0 TSMC8K_Conservative
tcb773pwc
sub11_0 TSMC8K_Conservative
tcb773pwc
sub11_0_DW01_sub_11_0 TSMC8K_Conservative
tcb773pwc
add12_1 TSMC8K_Conservative
tcb773pwc
add12_1_DW01_add_13_0 TSMC8K_Conservative
tcb773pwc
add12_0 TSMC8K_Conservative
tcb773pwc
add12_0_DW01_add_13_0 TSMC8K_Conservative
tcb773pwc
sub12_1 TSMC8K_Conservative
tcb773pwc
sub12_1_DW01_sub_12_0 TSMC8K_Conservative
tcb773pwc
sub12_0 TSMC8K_Conservative
tcb773pwc
sub12_0_DW01_sub_12_0 TSMC8K_Conservative
tcb773pwc
add13_3  TSMC8K_Conservative
tcb773pwc
add13_3_DW01_add_14_0  TSMC8K_Conservative
tcb773pwc
sub13_5  TSMC8K_Conservative
tcb773pwc
sub13_5_DW01_sub_13_0  TSMC8K_Conservative
tcb773pwc
realign  TSMC8K_Conservative
tcb773pwc
midlift  TSMC8K_Conservative
tcb773pwc
lsS813  TSMC8K_Conservative
tcb773pwc
add15  TSMC8K_Conservative
tcb773pwc
add15_DW01_add_16_0  TSMC8K_Conservative
tcb773pwc
sub13_4  TSMC8K_Conservative
tcb773pwc
sub13_4_DW01_sub_13_0  TSMC8K_Conservative
tcb773pwc
ls3813  TSMC8K_Conservative
tcb773pwc
add14_1  TSMC8K_Conservative
tcb773pwc
add14_1_DW01_add_15_0  TSMC8K_Conservative
tcb773pwc
sub13_3  TSMC8K_Conservative
tcb773pwc
sub13_3_DW01_sub_13_0  TSMC8K_Conservative
tcb773pwc
postadd  TSMC8K_Conservative
tcb773pwc
scale_4_13_2  TSMC8K_Conservative
tcb773pwc
add14_0  TSMC8K_Conservative
tcb773pwc
add14_0_DW01_add_15_0  TSMC8K_Conservative
tcb773pwc
add13_2  TSMC8K_Conservative
tcb773pwc
add13_2_DW01_add_14_0  TSMC8K_Conservative
tcb773pwc
scale_4_15  TSMC8K_Conservative
tcb773pwc
scale_4_13_1  TSMC8K_Conservative
tcb773pwc
sub14  TSMC8K_Conservative
tcb773pwc
sub14_DW01_sub_14_0  TSMC8K_Conservative
tcb773pwc
add13_1  TSMC8K_Conservative
tcb773pwc
add13_1_DW01_add_14_0  TSMC8K_Conservative
tcb773pwc

192
sub13_2  TSMC8K_Conservative
tcb773pwc
sub13_2_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_1  TSMC8K_Conservative
tcb773pwc
sub13_1_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_0_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
scale_4_14_3  TSMC8K_Conservative
tcb773pwc
scale_4_14_2  TSMC8K_Conservative
tcb773pwc
scale_4_14_1  TSMC8K_Conservative
tcb773pwc
add13_0  TSMC8K_Conservative
tcb773pwc
add13_0_DW01_add14_0  TSMC8K_Conservative
tcb773pwc
scale_4_14_0  TSMC8K_Conservative
tcb773pwc
scale_4_13_0  TSMC8K_Conservative
tcb773pwc
p_s_out13  TSMC8K_Conservative
tcb773pwc

Startpoint: p_s_out13/zout_reg_5_
(positive level-sensitive latch)
Endpoint: y_5_ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_5_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_5_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U59/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_5_ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_5_ (out)</td>
<td></td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p_s_out13/zout_reg_6_
(positive level-sensitive latch)
Endpoint: y_6_ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>Incr</td>
<td>Path</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_7_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_7_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U60/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_8_/ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_7_/ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p_s_out13/zout_reg_7_/ (positive level-sensitive latch)
Endpoint: y_7_/ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_7_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_7_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U61/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_8_/ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_7_/ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p_s_out13/zout_reg_8_/ (positive level-sensitive latch)
Endpoint: y_8_/ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_8_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_8_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U62/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_9_/ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_8_/ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p_s_out13/zout_reg_9_/ (positive level-sensitive latch)
Endpoint: y_9_/ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_9_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_9_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>Point</td>
<td>Incr</td>
<td>Path</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>p_s_out13/U63/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_9_ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_9_ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p\_s\_out13/zout\_reg\_10\_
(positive level-sensitive latch)

Endpoint: y\_10\_ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_10_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_10_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U64/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_10_ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_10_ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p\_s\_out13/zout\_reg\_11\_
(positive level-sensitive latch)

Endpoint: y\_11\_ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_11_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_11_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U65/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_11_ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>y_11_ (out)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>1.17</td>
</tr>
</tbody>
</table>

(Path is unconstrained)

Startpoint: p\_s\_out13/zout\_reg\_12\_
(positive level-sensitive latch)

Endpoint: y\_12\_ (output port)
Path Group: (none)
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_s_out13/zout_reg_12_/E (LH1N)</td>
<td>0.00</td>
<td>0.00 r</td>
</tr>
<tr>
<td>p_s_out13/zout_reg_12_/QN (LH1N)</td>
<td>0.93</td>
<td>0.93 f</td>
</tr>
<tr>
<td>p_s_out13/U66/ZN (INV0)</td>
<td>0.23</td>
<td>1.17 r</td>
</tr>
<tr>
<td>p_s_out13/zout_12_ (p_s_out13)</td>
<td>0.00</td>
<td>1.17 r</td>
</tr>
</tbody>
</table>
H.3 Final results for the gate-level constrained design of the forward binDCT chip

******************************************************************************
Report : area
Design : forwardbindctchip
Version: 1998.02
Date : Wed May 24 20:42:17 2000
******************************************************************************

Library(s) Used:

  tpd773pnw (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tpd773pnwc.db)
  tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Number of ports: 31
Number of nets: 1704
Number of cells: 1393
Number of references: 45

Combinational area: 1098650.000000
Noncombinational area: 81970.000000
Net Interconnect area: undefined (Wire load has zero net area)

Total cell area: 1180620.000000
Total area: undefined
1

design_analyzer>
******************************************************************************
Report : constraint
Design : forwardbindctchip
Version: 1998.02
Date : Wed May 24 20:42:18 2000
******************************************************************************

<table>
<thead>
<tr>
<th>Group (max_delay/setup)</th>
<th>Weighted Cost</th>
<th>Weight</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>in_clock</td>
<td>1.15</td>
<td>1.00</td>
<td>1.15</td>
</tr>
<tr>
<td>out_clock</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>default</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>max_delay/setup</td>
<td></td>
<td></td>
<td>1.15</td>
</tr>
</tbody>
</table>

Total Neg Critical

<table>
<thead>
<tr>
<th>Group (critical_range)</th>
<th>Slack</th>
<th>Endpoints</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>in_clock</td>
<td>69.27</td>
<td>53</td>
<td>60.70</td>
</tr>
</tbody>
</table>

196
design_analyzer> Warning: In design 'forwardbindctchip', there are 3 nets with connection class violations. (LINT-30)

Information: Use the 'check_design' command for more information about warnings. (LINT-99)

Performing power analysis through design. (low effort)
Warning: There are sequential cells with no output activity annotation. (PWR-96)

********************************************************************************
Report : power
          -analysis_effort low
Design : forwardbindctchip
Version: 1998.02
Date : Wed May 24 20:43:56 2000
********************************************************************************

Library(s) Used:

  tpd773pwn (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tpd773pwc.db)
  tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Operating Conditions: WCCOM  Library: tcb773pwc
Wire Loading Model Mode: segmented
Design Wire Loading Model Library
--------------------------------------
forwardbindctchip TSMC32K_Conservative tcb773pwc

Global Operating Voltage = 3
Power-specific unit information:
  Voltage Units = 1V
  Capacitance Units = 1.000000pf
  Time Units = 1ns
  Dynamic Power Units = 1mW (derived from V,C,T units)
  Leakage Power Units = 1pW

  Cell Internal Power = 7.3758 mW (22%)
  Net Switching Power = 25.7522 mW (78%)
  Total Dynamic Power = 33.1280 mW (100%)
  Cell Leakage Power = 1.3930 nW

1
design_analyzer>
******************************
Report: timing
  -path full
  -delay max
  -nworst 3
  -max_paths 8
Design: forwardbindctchip
Version: 1998.02
Date: Wed May 24 20:43:57 2000
******************************

Operating Conditions: WCCOM Library: tcb773pwc
Wire Loading Model Mode: segmented

Design Wire Loading Model Library
--------------------------------------
forwardbindctchip TSMC32K_Conservative tcb773pwc

Startpoint: forwardct_bindct8_z0_reg_0_
  (rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: forwardct_p_s_out11_outputmem_reg_0__0_
  (positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
</tbody>
</table>

198
<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_0_/CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_0_/Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__0_/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__0_/E (LH1N)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-2.64</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

Time Borrowing Information

bindct_clock pulse width 100.00
library setup time -0.04
clock uncertainty -0.50

max time borrow 99.46
actual time borrow 1.39

Startpoint: forwardct_bindct8_z0_reg_1_
(rising edge-triggered flip-flop clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_outputmem_reg_0__1_
(positive level-sensitive latch clocked by bindct_clock)

Path Group: bindct_clock
Path Type: max
<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_2/CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_2/Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0/2/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0/2/E (LH1N)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Time Borrowing Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bindct_clock pulse width</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.04</td>
<td></td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
<td></td>
</tr>
<tr>
<td>max time borrow</td>
<td>99.46</td>
<td></td>
</tr>
<tr>
<td>actual time borrow</td>
<td>1.39</td>
<td></td>
</tr>
</tbody>
</table>

Startpoint: forwardct_bindct8_z0_reg_2
(rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: forwardct_p_s_out11_outputmem_reg_0/2
(positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max

Startpoint: forwardct_bindct8_z0_reg_3
(rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: forwardct_p_s_out11_outputmem_reg_0/3
(positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max
<table>
<thead>
<tr>
<th>Clock</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct Bindct8_z0_reg_3_CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct Bindct8_z0_reg_3_Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0_3_D (LH1N)</td>
<td>0.00</td>
<td>2.64</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0_3_E (LH1N)</td>
<td>0.00</td>
<td>1.25</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-2.64</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

bindct_clock pulse width 100.00
library setup time -0.04
clock uncertainty -0.50

max time borrow 99.46
actual time borrow 1.39

Startpoint: forwardct_bindct8_z0_reg_4_
(rising edge-triggered flip-flop clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_outputmem_reg_0_4_
(positive level-sensitive latch clocked by bindct_clock)

Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_4_CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_4_Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0_4_D (LH1N)</td>
<td>0.00</td>
<td>2.64</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0_4_E (LH1N)</td>
<td>0.00</td>
<td>1.25</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-2.64</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>
### Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock pulse width</td>
<td>100.00</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.04</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max time borrow</td>
<td>99.46</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>1.39</td>
</tr>
</tbody>
</table>

---

Startpoint: `forwardct_bindct8_z0_reg_5_`  
(rising edge-triggered flip-flop clocked by `bindct_clock`)  

Endpoint: `forwardct_p_s_out11_outputmem_reg_0__5_`  
(positive level-sensitive latch clocked by `bindct_clock`)  

Path Group: `bindct_clock`  
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock <code>bindct_clock</code> (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td><code>forwardct_bindct8_z0_reg_5_/CP (DFCNE1Q)</code></td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td><code>forwardct_bindct8_z0_reg_5_/Q (DFCNE1Q)</code></td>
<td>1.39</td>
<td>2.64 f</td>
</tr>
<tr>
<td><code>forwardct_p_s_out11_outputmem_reg_0__5_/D (LH1N)</code></td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.64</td>
<td></td>
</tr>
</tbody>
</table>

| clock `bindct_clock` (rise edge)                  | 0.00 | 0.00 |
| clock network delay (propagated)                  | 1.25 | 1.25 |
| `forwardct_p_s_out11_outputmem_reg_0__5_/E (LH1N)`| 0.00 | 1.25 r |
| time borrowed from endpoint                       | 1.39 | 2.64 |
| data required time                                 | 2.64 |      |

| data required time                                 | 2.64 |
| data arrival time                                   | -2.64|

 slack (MET) 0.00

### Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock pulse width</td>
<td>100.00</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.04</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max time borrow</td>
<td>99.46</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>1.39</td>
</tr>
</tbody>
</table>

---

Startpoint: `forwardct_bindct8_z0_reg_6_`  
(rising edge-triggered flip-flop clocked by `bindct_clock`)  

Endpoint: `forwardct_p_s_out11_outputmem_reg_0__6_`  
(positive level-sensitive latch clocked by `bindct_clock`)  

Path Group: `bindct_clock`  
Path Type: max

202
<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindc1_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_6_CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_6_Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__6__/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>clock bindc1_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__6__/E (LH1N)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.64</td>
<td></td>
</tr>
</tbody>
</table>

**slack (MET):** 0.00

**Time Borrowing Information**

| bindc1_clock pulse width | 100.00 |
| library setup time | -0.04 |
| clock uncertainty | -0.50 |

**max time borrow:** 99.46  
**actual time borrow:** 1.39

---

**Startpoint:** forwardct_bindct8_z0_reg_7  
(rising edge-triggered flip-flop clocked by bindc1_clock)  
**Endpoint:** forwardct_p_s_out11_outputmem_reg_0__7  
(positive level-sensitive latch clocked by bindc1_clock)  
**Path Group:** bindc1_clock  
**Path Type:** max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindc1_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_7__/CP (DFCNE1Q)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>forwardct_bindct8_z0_reg_7__/Q (DFCNE1Q)</td>
<td>1.39</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__7__/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>clock bindc1_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__7__/E (LH1N)</td>
<td>0.00</td>
<td>1.25 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.64</td>
<td></td>
</tr>
</tbody>
</table>
Time Borrowing Information

bindct_clock pulse width 100.00
library setup time -0.04
clock uncertainty -0.50

max time borrow 99.46
actual time borrow 1.39

Startpoint: forwardct_s_p_in8_inputmem_reg_0__0_
    (positive level-sensitive latch clocked by in_clock)
Endpoint: forwardct_s_p_in8_inputmem_reg_0__0_
    (positive level-sensitive latch clocked by in_clock)
Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__0_/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__0_/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwardct_U722/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__0_/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__0_/E (LH2Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.69</td>
<td></td>
</tr>
</tbody>
</table>

slack (VIOLATED) -1.15

Time Borrowing Information

in_clock pulse width 12.21
library setup time -0.50
clock uncertainty -0.50

max time borrow 11.21
actual time borrow 0.34

Startpoint: forwardct_s_p_in8_inputmem_reg_0__2_
    (positive level-sensitive latch clocked by in_clock)
Endpoint: forwardct_s_p_in8_inputmem_reg_0__2_

204
**Path Group: in_clock**  
**Path Type: max**

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__2__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__2__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwarded_U720/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__2__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__2__/E (LH2Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.69</td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-1.15</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<table>
<thead>
<tr>
<th>in_clock pulse width</th>
<th>12.21</th>
</tr>
</thead>
<tbody>
<tr>
<td>library setup time</td>
<td>-0.50</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.21</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

**Startpoint:** forwarded_s_p_in8_inputmem_reg_0__3__  
*(positive level-sensitive latch clocked by in_clock)*

**Endpoint:** forwarded_s_p_in8_inputmem_reg_0__3__  
*(positive level-sensitive latch clocked by in_clock)*

**Path Group: in_clock**  
**Path Type: max**

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__3__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__3__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwarded_U719/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwarded_s_p_in8_inputmem_reg_0__3__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

205
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__4_/E (LH2Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.69</td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-1.15</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
<td></td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
<td></td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.21</td>
<td></td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
<td></td>
</tr>
</tbody>
</table>

**Startpoint:** forwardct_s_p_in8_inputmem_reg_0__4__

(positive level-sensitive latch clocked by in_clock)

**Endpoint:** forwardct_s_p_in8_inputmem_reg_0__4__

(positive level-sensitive latch clocked by in_clock)

Path Group: in_clock

Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__4__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__4__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwardct_U718Z (MUX2DQ)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__4__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
</tbody>
</table>

| clock in_clock (rise edge) | 0.00 | 0.00 |
| clock network delay (propagated) | 1.21 | 1.21 |
| forwardct_s_p_in8_inputmem_reg_0__4__/E (LH2Q) | 0.00 | 1.21 r |
| time borrowed from endpoint | 0.34 | 1.55 |
| data required time | 1.55 |
| data required time | 1.55 |
| data arrival time | -2.69 |
| slack (VIOLATED) | -1.15 |

**Time Borrowing Information**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
<td></td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
<td></td>
</tr>
</tbody>
</table>
max time borrow 11.21  
actual time borrow 0.34

Startpoint: forwardct_s_p_in8_inputmem_reg_0__5_  
(positive level-sensitive latch clocked by in_clock)  
Endpoint: forwardct_s_p_in8_inputmem_reg_0__5_  
(positive level-sensitive latch clocked by in_clock)  
Path Group: in_clock  
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__5_/D (LH2Q)</td>
<td>0.00</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__5_/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12</td>
</tr>
<tr>
<td>forwardct_U717/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__5_/D (LH2Q)</td>
<td>0.00</td>
<td>2.69</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>2.69</td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__5_/E (LH2Q)</td>
<td>0.00</td>
<td>1.21</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>1.55</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-2.69</td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td></td>
<td>-1.15</td>
</tr>
</tbody>
</table>

Time Borrowing Information

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.21</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_s_p_in8_inputmem_reg_0__6_  
(positive level-sensitive latch clocked by in_clock)  
Endpoint: forwardct_s_p_in8_inputmem_reg_0__6_  
(positive level-sensitive latch clocked by in_clock)  
Path Group: in_clock  
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value 1</td>
<td>Value 2</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__6__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__6__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwardct_U716/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__6__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__6__/E (LH2Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data required time</td>
<td>1.55</td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.69</td>
</tr>
</tbody>
</table>

slack (VIOLATED)  -1.15

**Time Borrowing Information**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

max time borrow 11.21
actual time borrow 0.34

Startpoint: forwardct_s_p_in8_inputmem_reg_0__7__
(positive level-sensitive latch clocked by in_clock)

Endpoint: forwardct_s_p_in8_inputmem_reg_0__7__
(positive level-sensitive latch clocked by in_clock)

Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__7__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__7__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwardct_U716/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_0__7__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
</tbody>
</table>

| clock in_clock (rise edge)                 | 0.00 | 0.00 |
| clock network delay (propagated)           | 1.21 | 1.21 |
| forwardct_s_p_in8_inputmem_reg_0__7__/E (LH2Q)| 0.00 | 1.21 r |
| time borrowed from endpoint                | 0.34 | 1.55 |
| data required time                         | 1.55 |

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data required time</td>
<td>1.55</td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.69</td>
</tr>
</tbody>
</table>
slack (VIOLATED)  -1.15

Time Borrowing Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.21</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_s_p_in8_inputmem_reg_1__1__
(positive level-sensitive latch clocked by in_clock)

Endpoint: forwardct_s_p_in8_inputmem_reg_1__1__
(positive level-sensitive latch clocked by in_clock)

Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_1__1__/D (LH2Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_1__1__/Q (LH2Q)</td>
<td>0.57</td>
<td>2.12 f</td>
</tr>
<tr>
<td>forwardct_U713/Z (MUX2D2)</td>
<td>0.58</td>
<td>2.69 f</td>
</tr>
<tr>
<td>forwardct_s_p_in8_inputmem_reg_1__1__/D (LH2Q)</td>
<td>0.00</td>
<td>2.69 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.69</td>
<td></td>
</tr>
</tbody>
</table>

Startpoint: forwardct_p_s_out11_outputmem_reg_4__1__
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_1__

slack (VIOLATED)  -1.15

Time Borrowing Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>12.21</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.50</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.21</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_p_s_out11_outputmem_reg_4__1__
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_1__
(positive level-sensitive latch clocked by out_clock)

Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_4__1_/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_4__1_/QN (LH1N)</td>
<td>0.69</td>
<td>3.34 r</td>
</tr>
<tr>
<td>forwardct_U201/ZN (INV0)</td>
<td>0.37</td>
<td>3.70 f</td>
</tr>
<tr>
<td>forwardct_U729/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.03 r</td>
</tr>
<tr>
<td>forwardct_U619/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_1_/D (LH1N)</td>
<td>0.00</td>
<td>4.34 f</td>
</tr>
</tbody>
</table>

data arrival time 4.34

clock out_clock (rise edge) 0.00 0.00

clock network delay (propagated) 0.43 0.43

forwardct_p_s_out11_zout_reg_1_/E (LH1N) 0.00 0.43 r

time borrowed from endpoint 3.91 4.34

data required time 4.34

data required time 4.34

data arrival time -4.34

slack (MET) 0.00

Time Borrowing Information

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

max time borrow 11.53
actual time borrow 3.91

Startpoint: forwardct_p_s_out11_outputmem_reg_4__2_
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_2_
(positive level-sensitive latch clocked by out_clock)

Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_4__2_/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_4__2_/QN (LH1N)</td>
<td>0.69</td>
<td>3.34 r</td>
</tr>
<tr>
<td>forwardct_U204/ZN (INV0)</td>
<td>0.37</td>
<td>3.70 f</td>
</tr>
<tr>
<td>forwardct_U733/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.03 r</td>
</tr>
<tr>
<td>forwardct_U618/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34 f</td>
</tr>
</tbody>
</table>

210
<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.16</td>
<td>2.41</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_7__10_/D (LH1N)</td>
<td>0.00</td>
<td>2.41</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_7__10_/QN (LH1N)</td>
<td>0.65</td>
<td>3.06</td>
</tr>
<tr>
<td>forwardct_U84ZN (NR2D0)</td>
<td>0.65</td>
<td>3.71</td>
</tr>
<tr>
<td>forwardct_U763ZN (A0121D1H)</td>
<td>0.33</td>
<td>4.04</td>
</tr>
<tr>
<td>forwardct_U610ZN (ND4D1)</td>
<td>0.29</td>
<td>4.34</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_10_/D (LH1N)</td>
<td>0.00</td>
<td>4.34</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>4.34</td>
</tr>
<tr>
<td>clock out_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_10_/E (LH1N)</td>
<td>0.00</td>
<td>0.43</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>3.91</td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>4.34</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>
**Time Borrowing Information**

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>startpoint: forwardct_p_s_out11_outputmem_reg_0__0_</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__0_/D (LH1N)</td>
<td>0.69</td>
<td>3.34 r</td>
</tr>
<tr>
<td>forwardct_U197/ZN (INV0)</td>
<td>0.37</td>
<td>3.70 f</td>
</tr>
<tr>
<td>forwardct_U726/ZN (AO12ID1H)</td>
<td>0.32</td>
<td>4.02 r</td>
</tr>
<tr>
<td>forwardct_U620/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34 f</td>
</tr>
<tr>
<td>endpoint: forwardct_p_s_out11_zout_reg_0__0_/D (LH1N)</td>
<td>0.00</td>
<td>4.34 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>clock out_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>3.91</td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-4.34</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.53</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_p_s_out11_outputmem_reg_0__0__
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_0__
(positive level-sensitive latch clocked by out_clock)

Path Group: out_clock
Path Type: max

**Time Borrowing Information**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.53</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_p_s_out11_outputmem_reg_0__1__
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_1__
(positive level-sensitive latch clocked by out_clock)
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__1_/D (LH1N)</td>
<td>0.00</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__1_/Q (LH1N)</td>
<td>0.69</td>
<td>3.34</td>
</tr>
<tr>
<td>forwardct_U200/ZN (INV0)</td>
<td>0.37</td>
<td>3.70</td>
</tr>
<tr>
<td>forwardct_U730/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.02</td>
</tr>
<tr>
<td>forwardct_U619/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_1_/D (LH1N)</td>
<td>0.00</td>
<td>4.34</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>4.34</td>
</tr>
</tbody>
</table>

| clock out_clock (rise edge)                        | 0.00 | 0.00 |
| clock network delay (propagated)                   | 0.43 | 0.43 |
| forwardct_p_s_out11_zout_reg_1_/E (LH1N)           | 0.00 | 0.43 |
| time borrowed from endpoint                        | 3.91 | 4.34 |
| data required time                                  |      | 4.34 |

**Time Borrowing Information**

| out_clock pulse width                               | 12.46 |
| library setup time                                  | -0.43 |
| clock uncertainty                                   | -0.50 |

| max time borrow                                     | 11.53 |
| actual time borrow                                  | 3.91 |

Startpoint: forwardct_p_s_out11_outputmem_reg_0__2_  
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_2_            
(positive level-sensitive latch clocked by out_clock)
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__2_/D (LH1N)</td>
<td>0.00</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__2_/Q (LH1N)</td>
<td>0.69</td>
<td>3.34</td>
</tr>
<tr>
<td>forwardct_U203/ZN (INV0)</td>
<td>0.37</td>
<td>3.70</td>
</tr>
<tr>
<td>forwardct_U734/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.02</td>
</tr>
<tr>
<td>forwardct_U618/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34</td>
</tr>
<tr>
<td>Point</td>
<td>Incre</td>
<td>Path</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_2_/D (LH1N)</td>
<td>0.00</td>
<td>4.34 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>clock out_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_2_/E (LH1N)</td>
<td>0.00</td>
<td>0.43 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>3.91</td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.53</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Startpoint: forwardct_p_s_out11_outputmem_reg_0__3_
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: forwardct_p_s_out11_zout_reg_3_
(positive level-sensitive latch clocked by out_clock)

Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incre</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__3_/D (LH1N)</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_outputmem_reg_0__3_/QN (LH1N)</td>
<td>0.69</td>
<td>3.34 r</td>
</tr>
<tr>
<td>forwardct_U206/ZN (INV0)</td>
<td>0.37</td>
<td>3.70 f</td>
</tr>
<tr>
<td>forwardct_U738/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.02 r</td>
</tr>
<tr>
<td>forwardct_U617/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34 f</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_3_/D (LH1N)</td>
<td>0.00</td>
<td>4.34 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>clock out_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>forwardct_p_s_out11_zout_reg_3_/E (LH1N)</td>
<td>0.00</td>
<td>0.43 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>3.91</td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td>4.34</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>data required time</td>
<td>4.34</td>
</tr>
<tr>
<td>data arrival time</td>
<td>-4.34</td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
</tr>
</tbody>
</table>
### Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.53</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Startpoint: `forwardct_p_s_out1l_outputmem_reg_0_4_`  
(positive level-sensitive latch clocked by bindct_clock)

Endpoint: `forwardct_p_s_out1l_zout_reg_4_`  
(positive level-sensitive latch clocked by out_clock)

Path Group: `out_clock`
Path Type: `max`

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.39</td>
<td>2.64</td>
</tr>
<tr>
<td>forwardct_p_s_out1l_outputmem_reg_0_4_/D</td>
<td>0.00</td>
<td>2.64 f</td>
</tr>
<tr>
<td>forwardct_p_s_out1l_outputmem_reg_0_4_/QN</td>
<td>0.69</td>
<td>3.34 r</td>
</tr>
<tr>
<td>forwardct_U208/ZN (INV0)</td>
<td>0.37</td>
<td>3.70 f</td>
</tr>
<tr>
<td>forwardct_U742/ZN (AOI21D1H)</td>
<td>0.32</td>
<td>4.02 r</td>
</tr>
<tr>
<td>forwardct_U616/ZN (ND4D1)</td>
<td>0.31</td>
<td>4.34 f</td>
</tr>
<tr>
<td>forwardct_p_s_out1l_zout_reg_4_/D (LH1N)</td>
<td>0.00</td>
<td>4.34 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>4.34</td>
</tr>
<tr>
<td>clock out_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>forwardct_p_s_out1l_zout_reg_4_/E (LH1N)</td>
<td>0.00</td>
<td>0.43 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>3.91</td>
<td>4.34</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>4.34</td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock pulse width</td>
<td>12.46</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.43</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>11.53</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>3.91</td>
</tr>
</tbody>
</table>
H.4. Final results for the gate-level constrained design of the Inverse binDCT chip

******************************************************************************
Report : area
Design : inversesbindctchip
Version: 1998.02
******************************************************************************

Library(s) Used:

tpd773pwc (File: /CMC/tools/synopsys.1998.02/cm0sp35/syn/tpd773pwc.db)
tcb773pwc (File: /CMC/tools/synopsys.1998.02/cm0sp35/syn/tcb773pwc.db)

Number of ports: 36
Number of nets: 277
Number of cells: 52
Number of references: 6

Combinational area: 1370127.500000
Noncombinational area: 95952.500000
Net Interconnect area: undefined (Wire load has zero net area)

Total cell area: 1466080.000000
Total area: undefined

**
design_analyzer>
******************************************************************************
Report : constraint
Design : inversesbindctchip
Version: 1998.02
******************************************************************************

<table>
<thead>
<tr>
<th>Group (max_delay/setup)</th>
<th>Weighted Cost</th>
<th>Weight</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>in_clock</td>
<td>1.11</td>
<td>1.00</td>
<td>1.11</td>
</tr>
<tr>
<td>out_clock</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>default</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

max_delay/setup 1.11

<table>
<thead>
<tr>
<th>Total Neg Critical Group (critical_range)</th>
<th>Slack</th>
<th>Endpoints</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>in_clock</td>
<td>91.85</td>
<td>82</td>
<td>90.77</td>
</tr>
<tr>
<td>out_clock</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>default</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>
critical_range 90.77

<table>
<thead>
<tr>
<th>Group (min_delay/hold)</th>
<th>Weighted Cost</th>
<th>Weight</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindct_clock (no fix_hold)</td>
<td>0.00</td>
<td>1.00 0.00</td>
<td></td>
</tr>
<tr>
<td>in_clock (no fix_hold)</td>
<td>0.00</td>
<td>1.00 0.00</td>
<td></td>
</tr>
<tr>
<td>out_clock (no fix_hold)</td>
<td>0.00</td>
<td>1.00 0.00</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>0.00</td>
<td>1.00 0.00</td>
<td></td>
</tr>
<tr>
<td>min_delay/hold</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraint Cost

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_transition</td>
<td>11.30 (VIOLATED)</td>
</tr>
<tr>
<td>max_fanout</td>
<td>0.00 (MET)</td>
</tr>
<tr>
<td>max_delay/setup</td>
<td>1.11 (VIOLATED)</td>
</tr>
<tr>
<td>critical_range</td>
<td>90.77 (VIOLATED)</td>
</tr>
</tbody>
</table>

Test Constraint Cost

<table>
<thead>
<tr>
<th>Test Constraint</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_fault_coverage</td>
<td>None (UNKNOWN)</td>
</tr>
</tbody>
</table>

Performing power analysis through design. (low effort)
Warning: There are sequential cells with no output activity annotation. (PWR-96)

Report : power
-analysis_effort low
Design : inversebindctchip
Version: 1998.02
Date : Wed May 24 21:24:43 2000

Library(s) Used:

tpd773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tpd773pwc.db)
tcb773pwc (File: /CMC/tools/synopsys.1998.02/cmc/cmosp35/syn/tcb773pwc.db)

Operating Conditions: WCCOM Library: tcb773pwc
Wire Loading Model Mode: segmented

<table>
<thead>
<tr>
<th>Design</th>
<th>Wire Loading Model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>inversebindctchip</td>
<td>TSMC32K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>s_p_in11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>ibindct11</td>
<td>TSMC8K_Conservative</td>
<td>tcb773pwc</td>
</tr>
<tr>
<td>prelifadd</td>
<td>TSMC8K_Conservative</td>
<td></td>
</tr>
</tbody>
</table>
sub13_3  TSMC8K_Conservative
tcb773pwc
sub13_3_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
postadd  TSMC8K_Conservative
tcb773pwc
scale_4_13_2  TSMC8K_Conservative
tcb773pwc
add14_0  TSMC8K_Conservative
tcb773pwc
add14_0_DW01_add15_0  TSMC8K_Conservative
tcb773pwc
add13_2  TSMC8K_Conservative
tcb773pwc
add13_2_DW01_add14_0  TSMC8K_Conservative
tcb773pwc
scale_4_15  TSMC8K_Conservative
tcb773pwc
scale_4_13_1  TSMC8K_Conservative
tcb773pwc
sub14  TSMC8K_Conservative
tcb773pwc
sub14_DW01_sub14_0  TSMC8K_Conservative
tcb773pwc
add13_1  TSMC8K_Conservative
tcb773pwc
add13_1_DW01_add14_0  TSMC8K_Conservative
tcb773pwc
sub13_2  TSMC8K_Conservative
tcb773pwc
sub13_2_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_1  TSMC8K_Conservative
tcb773pwc
sub13_1_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_0  TSMC8K_Conservative
tcb773pwc
sub13_0_DW01_sub13_0  TSMC8K_Conservative
tcb773pwc
scale_4_14_3  TSMC8K_Conservative
tcb773pwc
scale_4_14_2  TSMC8K_Conservative
tcb773pwc
scale_4_14_1  TSMC8K_Conservative
tcb773pwc
add13_0  TSMC8K_Conservative
tcb773pwc
add13_0_DW01_add14_0  TSMC8K_Conservative
tcb773pwc
scale_4_14_0  TSMC8K_Conservative
tcb773pwc
scale_4_13_0  TSMC8K_Conservative
tcb773pwc
p_s_out13  TSMC8K_Conservative
Global Operating Voltage = 3
Power-specific unit information:
  Voltage Units = 1V
  Capacitance Units = 1.0000000pf
  Time Units = 1ns
  Dynamic Power Units = 1mW (derived from V,C,T units)
  Leakage Power Units = 1pW

Cell Internal Power = 44.7173 mW (35%)  
Net Switching Power = 84.8133 mW (65%)  
  
Total Dynamic Power = 129.5306 mW (100%)  
Cell Leakage Power = 2.4670 nW

1
design analyzer>
*******************************
Report : timing
  -path full
  -delay max
  -nworst 3
  -max_paths 8
Design : inversebindctchip
Version: 1998.02
Date : Wed May 24 21:24:45 2000
*******************************
Operating Conditions: WCCOM  Library: tcb773pwc
Wire Loading Model Mode: segmented

Design    Wire Loading Model   Library
--------------------------------------
inversebindctchip  TSMC32K_Conservative
                   tcb773pwc
s_p_in11    TSMC8K_Conservative
           tcb773pwc
ibindct11   TSMC8K_Conservative
           tcb773pwc
prelifadd   TSMC8K_Conservative
           tcb773pwc
ls1211      TSMC8K_Conservative
           tcb773pwc
add11_3     TSMC8K_Conservative
           tcb773pwc
add11_3_DW01_add_12_0   TSMC8K_Conservative
                              tcb773pwc
add11_2     TSMC8K_Conservative
           tcb773pwc
add11_2_DW01_add_12_0   TSMC8K_Conservative
                              tcb773pwc
ls1811_1    TSMC8K_Conservative
add12_1   TSMC8K_Brare
          tcb773pwc
add12_1_DW01_add_13_0 TSMC8K_Brare
          tcb773pwc
add12_0   TSMC8K_Brare
          tcb773pwc
add12_0_DW01_add_13_0 TSMC8K_Brare
          tcb773pwc
sub12_1   TSMC8K_Brare
          tcb773pwc
sub12_1_DW01_sub_12_0 TSMC8K_Brare
          tcb773pwc
sub12_0   TSMC8K_Brare
          tcb773pwc
sub12_0_DW01_sub_12_0 TSMC8K_Brare
          tcb773pwc
add13_3   TSMC8K_Brare
          tcb773pwc
add13_3_DW01_add_14_0 TSMC8K_Brare
          tcb773pwc
sub13_5   TSMC8K_Brare
          tcb773pwc
sub13_5_DW01_sub_13_0 TSMC8K_Brare
          tcb773pwc
realign   TSMC8K_Brare
          tcb773pwc
midlift   TSMC8K_Brare
          tcb773pwc
ls5813    TSMC8K_Brare
          tcb773pwc
add15    TSMC8K_Brare
          tcb773pwc
add15_DW01_add_16_0 TSMC8K_Brare
          tcb773pwc
sub13_4   TSMC8K_Brare
          tcb773pwc
sub13_4_DW01_sub_13_0 TSMC8K_Brare
          tcb773pwc
ls3813    TSMC8K_Brare
          tcb773pwc
add14_1   TSMC8K_Brare
          tcb773pwc
add14_1_DW01_add_15_0 TSMC8K_Brare
          tcb773pwc
sub13_3   TSMC8K_Brare
          tcb773pwc
sub13_3_DW01_sub_13_0 TSMC8K_Brare
          tcb773pwc
postadd   TSMC8K_Brare
          tcb773pwc
scale_4_13_2 TSMC8K_Brare
          tcb773pwc
add14_0   TSMC8K_Brare
          tcb773pwc
add14_0_DW01_add_15_0 TSMC8K_Brare
          tcb773pwc
Startpoint: inversedct_i bindct11/x0_reg_0
(rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/outputmem_reg_0__0
(positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
</table>

224
clock bindct_clock (rise edge) 0.00 0.00
clock network delay (propagated) 1.34 1.34
inversedct_ibindct11/x0_reg_0_/CP (DFCN2N) 0.00 1.34 r
inversedct_ibindct11/x0_reg_0_/QN (DFCN2N) 1.35 2.69 r
inversedct_ibindct11/U256/ZN (INV3) 0.12 2.81 f
inversedct_ibindct11/x0_0_/ (ibindct11) 0.00 2.81 f
inversedct_p_s_out13/x0_0_/ (p_s_out13) 0.00 2.81 f
inversedct_p_s_out13/outputmem_reg_0___0_/D (LH1N) 0.00 2.81 f
data arrival time 2.81

clock bindct_clock (rise edge) 0.00 0.00
clock network delay (propagated) 1.34 1.34
inversedct_p_s_out13/outputmem_reg_0___0_/E (LH1N) 0.00 1.34 r
time borrowed from endpoint 1.47 2.81
data required time 2.81

data required time 2.81
data arrival time -2.81

slack (MET) 0.00

Time Borrowing Information

bindct_clock pulse width 100.00
library setup time 0.02
clock uncertainty -0.50

max time borrow 99.53
actual time borrow 1.47

Startpoint: inversedct_ibindct11/x0_reg_1_
    (rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/outputmem_reg_0___1_
    (positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
clock network delay (propagated) 1.34 1.34
inversedct_ibindct11/x0_reg_1_/CP (DFCN2N) 0.00 1.34 r
inversedct_ibindct11/x0_reg_1_/QN (DFCN2N) 1.35 2.69 r
inversedct_ibindct11/U256/ZN (INV3) 0.12 2.81 f
inversedct_ibindct11/x0_1_/ (ibindct11) 0.00 2.81 f
inversedct_p_s_out13/x0_1_/ (p_s_out13) 0.00 2.81 f
inversedct_p_s_out13/outputmem_reg_0___1_/D (LH1N) 0.00 2.81 f
data arrival time 2.81

clock bindct_clock (rise edge) 0.00 0.00
clock network delay (propagated) 1.34 1.34
inversedct_p_s_out13/outputmem_reg_0___1_/E (LH1N) 0.00 1.34 r
time borrowed from endpoint 1.47 2.81
data required time 2.81

225
data required time | 2.81  
data arrival time   | -2.81

slack (MET) | 0.00

Time Borrowing Information

bindct_clock pulse width | 100.00  
library setup time      | 0.02    
clock uncertainty       | -0.50   

max time borrow | 99.53
actual time borrow | 1.47

Startpoint: inversedct_ibindct11/x0_reg_2_
(rising edge-triggered flip-flop clocked by bindct_clock)

Endpoint: inversedct_p_s_out13/outputmem_reg_0_2_
(positive level-sensitive latch clocked by bindct_clock)

Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_2_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_2_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/U255/ZN (INV3)</td>
<td>0.12</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_2_ (bindct11)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/x0_2_ (p_s_out13)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0_2_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
</tbody>
</table>

data arrival time | 2.81

clock bindct_clock (rise edge) | 0.00 | 0.00
clock network delay (propagated) | 1.34 | 1.34
inversedct_p_s_out13/outputmem_reg_0_2_/E (LH1N) | 0.00 | 1.34 r

time borrowed from endpoint | 1.47 | 2.81
data required time | 2.81

data required time | 2.81
data arrival time | -2.81

slack (MET) | 0.00

Time Borrowing Information

bindct_clock pulse width | 100.00  
library setup time      | 0.02    
clock uncertainty       | -0.50   

max time borrow | 99.53
actual time borrow | 1.47
Startpoint: `inversedct_ibindctl1/x0_reg_3`
(rising edge-triggered flip-flop clocked by `bindct_clock`)

Endpoint: `inversedct_p_s_out13/outputmem_reg_0__3`
(positive level-sensitive latch clocked by `bindct_clock`)

Path Group: `bindct_clock`
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindctl1/x0_reg_3_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindctl1/x0_reg_3_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>inversedct_ibindctl1/U254/ZN (INV3)</td>
<td>0.12</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_ibindctl1/x0_3_/ (ibindctl1)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/z0_3_/ (p_s_out13)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__3_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>2.81</td>
</tr>
</tbody>
</table>

| clock bindct_clock (rise edge) | 0.00 | 0.00 |
| clock network delay (propagated) | 1.34 | 1.34 |
| inversedct_p_s_out13/outputmem_reg_0__3_/E (LH1N) | 0.00 | 1.34 r |
| time borrowed from endpoint | 1.47 | 2.81 |
| data required time | | 2.81 |

| data required time | 2.81 |
| data arrival time | -2.81 |

| slack (MET) | 0.00 |

**Time Borrowing Information**

| bindct_clock pulse width | 100.00 |
| library setup time | 0.02 |
| clock uncertainty | -0.50 |

| max time borrow | 99.53 |
| actual time borrow | 1.47 |

Startpoint: `inversedct_ibindctl1/x0_reg_4`
(rising edge-triggered flip-flop clocked by `bindct_clock`)

Endpoint: `inversedct_p_s_out13/outputmem_reg_0__4`
(positive level-sensitive latch clocked by `bindct_clock`)

Path Group: `bindct_clock`
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindctl1/x0_reg_4_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindctl1/x0_reg_4_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>Point</td>
<td>Incr</td>
<td>Path</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_5_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_5_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/U252/ZN (INV3)</td>
<td>0.12</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_5_ (ibindct11)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/z0_5_ (p_s_out13)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__5_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.81</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__5_/E (LH1N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>data required time</td>
<td>2.81</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>data required time</td>
<td>2.81</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.81</td>
<td></td>
</tr>
</tbody>
</table>

Slack (MET): 0.00

Time Borrowing Information

- bindct_clock pulse width: 100.00
- library setup time: 0.02
- clock uncertainty: -0.50

- max time borrow: 99.53
- actual time borrow: 1.47
slack (MET) 0.00

Time Borrowing Information

<table>
<thead>
<tr>
<th>bindct_clock pulse width</th>
<th>100.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>library setup time</td>
<td>0.02</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

| max time borrow | 99.53 |
| actual time borrow | 1.47 |

Startpoint: inversedct_ibindct11/x0_reg_6_
(rising edge-triggered flip-flop clocked by bindct_clock)

Endpoint: inversedct_p_s_out13/outputmem_reg_0__6_
(positive level-sensitive latch clocked by bindct_clock)

Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_6_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_6_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/U251/ZN (INV3)</td>
<td>0.12</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_6_ (ibindct11)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/x0_6_ (p_s_out13)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__6_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.81</td>
<td></td>
</tr>
</tbody>
</table>

| clock bindct_clock (rise edge) | 0.00 | 0.00 |
| clock network delay (propagated) | 1.34 | 1.34 |
| inversedct_p_s_out13/outputmem_reg_0__6_/E (LH1N) | 0.00 | 1.34 r |
| time borrowed from endpoint | 1.47 | 2.81 |
| data required time | 2.81 |

| data required time | 2.81 |
| data arrival time | -2.81 |

| slack (MET) | 0.00 |

Time Borrowing Information

<table>
<thead>
<tr>
<th>bindct_clock pulse width</th>
<th>100.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>library setup time</td>
<td>0.02</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

| max time borrow | 99.53 |
| actual time borrow | 1.47 |

Startpoint: inversedct_ibindct11/x0_reg_7_
(rising edge-triggered flip-flop clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/outputmem_reg_0__7__
(positive level-sensitive latch clocked by bindct_clock)
Path Group: bindct_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_7_/CP (DFCN2N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_7_/QN (DFCN2N)</td>
<td>1.35</td>
<td>2.69 r</td>
</tr>
<tr>
<td>inversedct_ibindct11/U250/ZN (INV3)</td>
<td>0.12</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_ibindct11/x0_reg_7_ (bindct11)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/x0_7_ (p_s_out13)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__7_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.81</td>
<td></td>
</tr>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_0__7_/E (LH1N)</td>
<td>0.00</td>
<td>1.34 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>data required time</td>
<td>2.81</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Time Borrowing Information

| bindct_clock pulse width                   | 100.00 |
| library setup time                        | 0.02   |
| clock uncertainty                         | -0.50  |

max time borrow 99.53
actual time borrow 1.47

Startpoint: inversedct_s_p_in11/inputmem_reg_0__0__
(positive level-sensitive latch clocked by in_clock)
Endpoint: inversedct_s_p_in11/inputmem_reg_0__0__
(positive level-sensitive latch clocked by in_clock)
Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__0_/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__0_/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/U223/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__0_/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
</tbody>
</table>

230
<table>
<thead>
<tr>
<th>Description</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__2_/E (LH1Q)</td>
<td>0.00</td>
<td>1.21</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.66</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-1.11</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>3.77</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

**Startpoint:** inversedct_s_p_in11/inputmem_reg_0__2__  
(positive level-sensitive latch clocked by in_clock)  
**Endpoint:** inversedct_s_p_in11/inputmem_reg_0__2__  
(positive level-sensitive latch clocked by in_clock)  
**Path Group:** in_clock  
**Path Type:** max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__2__/D (LH1Q)</td>
<td>0.00</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__2__/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09</td>
</tr>
<tr>
<td>inversedct_s_p_in11/U220/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__2__/D (LH1Q)</td>
<td>0.00</td>
<td>2.66</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__2_/E (LH1Q)</td>
<td>0.00</td>
<td>1.21</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.66</td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-1.11</td>
<td></td>
</tr>
</tbody>
</table>

**Time Borrowing Information**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>Point</td>
<td>Incr</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__3_/D</td>
<td>0.00</td>
</tr>
<tr>
<td>(LH1Q)</td>
<td></td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__3_/Q</td>
<td>0.53</td>
</tr>
<tr>
<td>(LH1Q)</td>
<td></td>
</tr>
<tr>
<td>inversedct_s_p_in11/MUX2D2</td>
<td>0.57</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__3_/D</td>
<td>0.00</td>
</tr>
<tr>
<td>(LH1Q)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__3_/E</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>(LH1Q)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data availability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time Borrowing Information

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startpoint: inversedct_s_p_in11/inputmem_reg_0__4__ (positive level-sensitive latch clocked by in_clock)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max time borrow</td>
<td>3.77</td>
<td></td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value 1</td>
<td>Value 2</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/U217/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/E (LH1Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.66</td>
<td></td>
</tr>
</tbody>
</table>

**slack (VIOLATED):** -1.11

**Time Borrowing Information**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max time borrow</td>
<td>3.77</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: inversedct_s_p_in11/inputmem_reg_0__5__
(positive level-sensitive latch clocked by in_clock)

Endpoint: inversedct_s_p_in11/inputmem_reg_0__5__
(positive level-sensitive latch clocked by in_clock)

Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/U217/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__5_/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
</tbody>
</table>

| clock in_clock (rise edge)                | 0.00 | 0.00                                      |
| clock network delay (propagated)          | 1.21 | 1.21                                      |
| inversedct_s_p_in11/inputmem_reg_0__5_/E (LH1Q) | 0.00    | 1.21 r  |
| time borrowed from endpoint               | 0.34 | 1.55                                      |
| data required time                         | 1.55  |                                           |

| data required time                         | 1.55  |                                           |
data arrival time | -2.66
slack (VIOLATED)   | -1.11

Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max time borrow</td>
<td>3.77</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: inversedct_s_p_in11/inputmem_reg_0__6__
(positive level-sensitive latch clocked by in_clock)

Endpoint: inversedct_s_p_in11/inputmem_reg_0__6__
(positive level-sensitive latch clocked by in_clock)

Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__6__/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__6__/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/U216/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>inversedct_s_p_in11/inputmem_reg_0__6__/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
</tbody>
</table>

| clock in_clock (rise edge)                 | 0.00 | 0.00 |
| clock network delay (propagated)           | 1.21 | 1.21 |
| inversedct_s_p_in11/inputmem_reg_0__6__/E (LH1Q) | 0.00 | 1.21 r |
| time borrowed from endpoint                | 0.34 | 1.55 |
| data required time                         | 1.55 |

| data required time                         | 1.55 |
| data arrival time                          | -2.66 |

slack (VIOLATED) | -1.11

Time Borrowing Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_clock pulse width</td>
<td>4.71</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.44</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max time borrow</td>
<td>3.77</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Startpoint: inversedct_s_p_in11/inputmem_reg_0__7__
(positive level-sensitive latch clocked by in_clock)
Endpoint: invertedct_s_p_in11/inputmem_reg_0__7__
(positive level-sensitive latch clocked by in_clock)
Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__7__/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__7__/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/U215/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__7__/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
</tbody>
</table>

| clock in_clock (rise edge)                | 0.00 | 0.00 |
| clock network delay (propagated)          | 1.21 | 1.21 |
| invertedct_s_p_in11/inputmem_reg_0__7__/E (LH1Q) | 0.00 | 1.21 r |
| time borrowed from endpoint               | 0.34 | 1.55 |
| data required time                        | 1.55 |
| data arrival time                         | -2.66 |

slack (VIOLATED) -1.11

Time Borrowing Information

| in_clock pulse width         | 4.71  |
| library setup time           | -0.44 |
| clock uncertainty            | -0.50 |

| max time borrow              | 3.77  |
| actual time borrow           | 0.34  |

Startpoint: invertedct_s_p_in11/inputmem_reg_0__8__
(positive level-sensitive latch clocked by in_clock)
Endpoint: invertedct_s_p_in11/inputmem_reg_0__8__
(positive level-sensitive latch clocked by in_clock)
Path Group: in_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__8__/D (LH1Q)</td>
<td>0.00</td>
<td>1.55 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__8__/Q (LH1Q)</td>
<td>0.53</td>
<td>2.09 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/U214/Z (MUX2D2)</td>
<td>0.57</td>
<td>2.66 f</td>
</tr>
<tr>
<td>invertedct_s_p_in11/inputmem_reg_0__8__/D (LH1Q)</td>
<td>0.00</td>
<td>2.66 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.66</td>
<td></td>
</tr>
<tr>
<td>Point</td>
<td>Incr</td>
<td>Path</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>clock_in_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.21</td>
<td>1.21</td>
</tr>
<tr>
<td>inversedct_s_p_in11/outputmem_reg_0__8_/E (LH1Q)</td>
<td>0.00</td>
<td>1.21 r</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>0.34</td>
<td>1.55</td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.66</td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-1.11</td>
<td></td>
</tr>
</tbody>
</table>

Time Borrowing Information

| in_clock pulse width | 4.71 |
| library setup time | -0.44 |
| clock uncertainty | -0.50 |

max time borrow 3.77
actual time borrow 0.34

Startpoint: inversedct_p_s_out13/outputmem_reg_7__1__
   (positive level-sensitive latch clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/zout_reg_1__
   (negative level-sensitive latch clocked by out_clock)
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.45</td>
<td>2.79</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__1__/D (LH2N)</td>
<td>0.00</td>
<td>2.79 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__1__/QN (LH2N)</td>
<td>0.62</td>
<td>3.42 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U48/ZN (NR2D1)</td>
<td>0.35</td>
<td>3.77 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U164/ZN (AOI21D1H)</td>
<td>0.26</td>
<td>4.03 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U143/ZN (ND4D1)</td>
<td>0.36</td>
<td>4.39 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_1__/D (LN1N)</td>
<td>0.00</td>
<td>4.39 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.39</td>
<td></td>
</tr>
</tbody>
</table>

clock out_clock' (fall edge) 0.00 0.00
clock network delay (propagated) 0.32 0.32
inversedct_p_s_out13/zout_reg_1__/EN (LN1N) 0.00 0.32 f
time borrowed from endpoint 4.07 4.39
data required time 4.39

<table>
<thead>
<tr>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>data required time</td>
<td>4.39</td>
</tr>
<tr>
<td>data arrival time</td>
<td>-4.39</td>
</tr>
</tbody>
</table>

slack (MET) 0.00

Time Borrowing Information
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock' pulse width</td>
<td>4.97</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.38</td>
<td></td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
<td></td>
</tr>
</tbody>
</table>

|                                                                 |       |       |
| max time borrow                                                 | 4.09  |       |
| actual time borrow                                              | 4.07  |       |

Startpoint: inversedct_p_s_out13/outputmem_reg_7__2__
(positive level-sensitive latch clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/zout_reg_2__
(negative level-sensitive latch clocked by out_clock')
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.45</td>
<td>2.79</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__2__/D (LH2N)</td>
<td>0.00</td>
<td>2.79 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__2__/QN (LH2N)</td>
<td>0.62</td>
<td>3.42 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U45/ZN (NR2D1)</td>
<td>0.35</td>
<td>3.77 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U168/ZN (AOI21D1H)</td>
<td>0.26</td>
<td>4.03 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U142/ZN (ND4D1)</td>
<td>0.36</td>
<td>4.39 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_2__/D (LN1N)</td>
<td>0.00</td>
<td>4.39 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.39</td>
<td></td>
</tr>
</tbody>
</table>

| clock out_clock' (fall edge)                 | 0.00 | 0.00 |
| clock network delay (propagated)             | 0.32 | 0.32 |
| inversedct_p_s_out13/zout_reg_2__/EN (LN1N) | 0.00 | 0.32 f |
| time borrowed from endpoint                  | 4.07 | 4.39 |
| data required time                           | 4.39 |      |

| data required time                            | 4.39 |      |
| data arrival time                             | -4.39|      |

slack (MET)                                      | 0.00 |      |

Time Borrowing Information

|                                                                 |       |       |
| out_clock' pulse width                                          | 4.97  |       |
| library setup time                                              | -0.38 |       |
| clock uncertainty                                               | -0.50 |       |

| max time borrow                                                 | 4.09  |       |
| actual time borrow                                              | 4.07  |       |

Startpoint: inversedct_p_s_out13/outputmem_reg_7__3__
(positive level-sensitive latch clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/zout_reg_3__
(negative level-sensitive latch clocked by out_clock')
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.45</td>
<td>2.79</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__3__/D (LH2N)</td>
<td>0.00</td>
<td>2.79 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__3__/QN (LH2N)</td>
<td>0.62</td>
<td>3.42 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U42/ZN (NR2D1)</td>
<td>0.35</td>
<td>3.77 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U172/ZN (AOI21D1H)</td>
<td>0.26</td>
<td>4.03 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U141/ZN (ND4D1)</td>
<td>0.36</td>
<td>4.39 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_3_/D (LN1N)</td>
<td>0.00</td>
<td>4.39 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>clock out_clock' (fall edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_3_/EN (LN1N)</td>
<td>0.00</td>
<td>0.32 f</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>4.07</td>
<td>4.39</td>
</tr>
<tr>
<td>data required time</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-4.39</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Time Borrowing Information

| out_clock' pulse width | 4.97 |
| library setup time | -0.38 |
| clock uncertainty | -0.50 |

| max time borrow | 4.09 |
| actual time borrow | 4.07 |

Startpoint: inversedct_p_s_out13/outputmem_reg_7__4_
  (positive level-sensitive latch clocked by bindct_clock)

Endpoint: inversedct_p_s_out13/zout_reg_4_
  (negative level-sensitive latch clocked by out_clock')

Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.45</td>
<td>2.79</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__4__/D (LH2N)</td>
<td>0.00</td>
<td>2.79 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_7__4__/QN (LH2N)</td>
<td>0.62</td>
<td>3.42 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U39/ZN (NR2D1)</td>
<td>0.35</td>
<td>3.77 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U176/ZN (AOI21D1H)</td>
<td>0.26</td>
<td>4.03 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U140/ZN (ND4D1)</td>
<td>0.36</td>
<td>4.39 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_4_/D (LN1N)</td>
<td>0.00</td>
<td>4.39 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>Clock Name</td>
<td>Rise Time (ns)</td>
<td>Fall Time (ns)</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>clock</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>network</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverted</td>
<td>0.00</td>
<td>0.32 f</td>
</tr>
<tr>
<td>s_out13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zout_reg_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EN (LN1N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>4.07</td>
<td>4.39</td>
</tr>
<tr>
<td>borrowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>from</td>
<td></td>
<td></td>
</tr>
<tr>
<td>endpoint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.39</td>
<td></td>
</tr>
</tbody>
</table>

| Data Required Time | 4.39 |
| Data Arrival Time  | -4.39 |

slack (MET) 0.00

Time Borrowing Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock' pulse width</td>
<td>4.97</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.38</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
</tbody>
</table>

max time borrow 4.09
actual time borrow 4.07

Startpoint: invertedp_s_out13/outputmem_reg_1___1___
(positive level-sensitive latch clocked by bindctl_clock)

Endpoint: invertedp_s_out13/zout_reg_1___
(negative level-sensitive latch clocked by out_clock')

Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindctl_clock (rise</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>edge)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock network delay</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>(propagated)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>invertedp_s_out13/outputmem</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>reg_1___1___/D (LN1N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invertedp_s_out13/outputmem</td>
<td>0.63</td>
<td>3.44 r</td>
</tr>
<tr>
<td>reg_1___1___/QN (LN1N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invertedp_s_out13/U89/ZN</td>
<td>0.34</td>
<td>3.78 f</td>
</tr>
<tr>
<td>(INV0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invertedp_s_out13/U165/ZN</td>
<td>0.30</td>
<td>4.08 r</td>
</tr>
<tr>
<td>(AOI21D1H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invertedp_s_out13/U143/ZN</td>
<td>0.29</td>
<td>4.37 f</td>
</tr>
<tr>
<td>(ND4D1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invertedp_s_out13/zout_reg_1</td>
<td>0.00</td>
<td>4.37 f</td>
</tr>
<tr>
<td>_/D (LN1N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.37</td>
<td></td>
</tr>
</tbody>
</table>

clock out_clock' (fall edge) 0.00 0.00
clock network delay (propagated) 0.32 0.32
invertedp_s_out13/zout_reg_1___/EN (LN1N) 0.00 0.32 f
time borrowed from endpoint 4.04 4.37
data required time 4.37
data required time 4.37
data arrival time -4.37

slack (MET) 0.00

Time Borrowing Information
<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_1__2_/D(LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmereg_1__2_/QN(LH1N)</td>
<td>0.63</td>
<td>3.44 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U86/ZN (INV0)</td>
<td>0.34</td>
<td>3.78 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U169/ZN (AO121DH1)</td>
<td>0.30</td>
<td>4.08 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U142/ZN (ND4D1)</td>
<td>0.29</td>
<td>4.37 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_2_/D(LN1N)</td>
<td>0.00</td>
<td>4.37 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.37</td>
<td></td>
</tr>
<tr>
<td>clock out_clock’(fall edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_2_/EN(LN1N)</td>
<td>0.00</td>
<td>0.32 f</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>4.04</td>
<td>4.37</td>
</tr>
<tr>
<td>data required time</td>
<td>4.37</td>
<td></td>
</tr>
<tr>
<td>slack (MET)</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Time Borrowing Information

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock’ pulse width</td>
<td>4.97</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.25</td>
<td></td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
<td></td>
</tr>
<tr>
<td>max time borrow</td>
<td>4.22</td>
<td></td>
</tr>
<tr>
<td>actual time borrow</td>
<td>4.04</td>
<td></td>
</tr>
</tbody>
</table>

Startpoint: inversedct_p_s_out13/outputmem_reg_1__3__
(positive level-sensitive latch clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/zout_reg_3__
(negative level-sensitive latch clocked by out_clock')
Path Group: out_clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_1__3_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_1__3_/QN (LH1N)</td>
<td>0.63</td>
<td>3.44 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U83/ZN (INV0)</td>
<td>0.34</td>
<td>3.78 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U173/ZN (AOI21D1H)</td>
<td>0.30</td>
<td>4.08 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U141/ZN (ND4D1)</td>
<td>0.29</td>
<td>4.37 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_3_/D (LN1N)</td>
<td>0.00</td>
<td>4.37 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>4.37</td>
</tr>
<tr>
<td>clock out__clock (fall edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_3_/EN (LN1N)</td>
<td>0.00</td>
<td>0.32 f</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>4.04</td>
<td>4.37</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>4.37</td>
</tr>
</tbody>
</table>

| data required time                         |      | 4.37 |
| data arrival time                          |      | -4.37 |

| slack (MET)                                |      | 0.00 |

Time Borrowing Information

| out__clock pulse width                    |      | 4.97 |
| library setup time                        |      | -0.25 |
| clock uncertainty                         |      | -0.50 |

max time borrow                            | 4.22 |
actual time borrow                          | 4.04 |

Startpoint: inversedct_p_s_out13/outputmem_reg_1__4__
(positive level-sensitive latch clocked by bindct_clock)
Endpoint: inversedct_p_s_out13/zout_reg_4__
(negative level-sensitive latch clocked by out__clock)
Path Group: out__clock
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock bindct_clock (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>1.34</td>
<td>1.34</td>
</tr>
<tr>
<td>time given to startpoint</td>
<td>1.47</td>
<td>2.81</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_1__4_/D (LH1N)</td>
<td>0.00</td>
<td>2.81 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/outputmem_reg_1__4_/QN (LH1N)</td>
<td>0.63</td>
<td>3.44 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U83/ZN (INV0)</td>
<td>0.34</td>
<td>3.78 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U177/ZN (AOI21D1H)</td>
<td>0.30</td>
<td>4.08 r</td>
</tr>
<tr>
<td>inversedct_p_s_out13/U140/ZN (ND4D1)</td>
<td>0.29</td>
<td>4.37 f</td>
</tr>
<tr>
<td>inversedct_p_s_out13/zout_reg_4_/D (LN1N)</td>
<td>0.00</td>
<td>4.37 f</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>4.37</td>
</tr>
<tr>
<td>Description</td>
<td>Value1</td>
<td>Value2</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>clock out_clock' (fall edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (propagated)</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>inversedc1p_s_out13/zout_reg_4_/EN (LN1N)</td>
<td>0.00</td>
<td>0.32 f</td>
</tr>
<tr>
<td>time borrowed from endpoint</td>
<td>4.04</td>
<td>4.37</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>4.37</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-4.37</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Time Borrowing Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_clock' pulse width</td>
<td>4.97</td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.25</td>
</tr>
<tr>
<td>clock uncertainty</td>
<td>-0.50</td>
</tr>
<tr>
<td>max time borrow</td>
<td>4.22</td>
</tr>
<tr>
<td>actual time borrow</td>
<td>4.04</td>
</tr>
</tbody>
</table>
REFERENCES


VITA AUCTORIS

Tracy Ann Franklin was born in 1973 in Georgetown, Guyana. She graduated from Riverside Secondary School in 1992. From there she went on to the University of Windsor where she obtained a B. A. Sc. in Electrical Engineering in 1996. She is currently a candidate for the Master's degree in Signals and Systems at the University of Windsor and hopes to graduate in the summer 2000.