To the Graduate Council:

I am submitting herewith a thesis written by Sowmyan Rajagopalan entitled “Implementation of Wavelet Transform and Area, Power and Delay Design Space Exploration“. I have examined the final copy of this thesis for form and content and recommend that it can be accepted in partial fulfillment of the requirement for the degree of Master of Science, with a major in Electrical Engineering.

Paul.B.Crilly, Major Professor

We have read this thesis and

recommend its acceptance:

Donald.W.Bouldin

Marshall Pace

Accepted for the council:

Anne Mayhew

Interim Vice Provost and

Dean of the Graduate School

(Original signatures are on file in Graduate Student Services Office)

IMPLEMENTATION OF WAVELET TRANSFORM AND AREA, POWER AND DELAY DESIGN SPACE EXPLORATION

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Sowmyan Rajagopalan
August 2001

Acknowledgment

I wish to thank all those who helped me in completing Master of Science in Electrical and Computer Engineering. I thank Dr. Crilly for his guidance and his effort in making me familiar to the concept of Wavelet Transform. I thank Dr.Bouldin for introducing me to various CAD tools, also for his ideas and valuable guidance all through the research work. I thank Dr. Pace for serving in my committee.

I thank my family, friends whose suggestions and encouragement made this work possible.

Abstract

The need for designing chips that are smaller, faster and less power consuming has been increasing, as these chips are increasingly being used in communication and portable electronic devices. These design space vectors also play a big role in the profitability and productivity of the vendor and hence prove to be one of the most important steps in the Application Specific Integrated Circuit (ASIC) flow (design process). The research reported here consists of implementing the Wavelet Transform using a hardware description language (VHDL) and testing the Register Transfer Level (RTL) for its functionality. The RTL was synthesized and optimized using the Synopsys Design Analyzer for multiple values of area, power and delay. Design constraints were setup in the tool to do this before the layout stage. The synthesized netlist was then placed and routed using EPOCH, which optimizes the area, power and delay of the ASIC at the post layout stage. Thus, fifteen different ASICs over a range of area, power and delay values were produced.

Table of contents

Chapters *Page*

Chapters *Page*

List of Figures

Figures *Page*

Figures *Page*

List of Tables

Tables *Page*

CHAPTER 1

Introduction

1.1 Chip Optimization Problem Statement

The increasing use of sophisticated design tools and end product applications of the integrated circuits (ICs) since the early 90's has dramatically increased the need to optimize a chip with respect to area, power and delay. Yield, which determines the productivity and profitability, is greatly influenced by chip area [5]. For example, with a five-inch wafer with a defect density of 2/sq.cm, the yield increases by five times when the area of the die is reduced by four times [17]. The interest in chip optimization is necessitated by the large-scale use of low power electronic and communication systems as these products demand faster, smaller and low power devices. A chip could be optimized for these parameters at either the (1) System design level, (2) Behavioral synthesis level, (3) Logic synthesis level, (4) Physical design level and/or (5) Circuit design level. The following describes these optimization techniques:

- **System Design Level**

At system level, inactive hardware modules may be turned off to save power, or may be provided with the optimum supply voltage and thus control propagation delay and power dissipation. Also a given task may be partitioned between various hardware modules or programmable processors or both so as to reduce the system-level power consumption.

- **Behavioral Synthesis Level**

Behavioral synthesis is the process of generating a Register-Transfer Level (RTL) design from an algorithmic behavioral specification.

The behavioral synthesis process consists of three steps: (1) Allocation, (2) Assignment and (3) Scheduling. These steps determine how many instances of each resource are needed, on what resource each operation is performed and when each operation is executed. A multitude of transformations can be done at this level and most are typically aimed at either reducing the number of cycles in a computation or reducing the number of resources used in the computation. Behavioral level estimation is essential because lower level estimation tools are time consuming and their use precludes a complete exploration of the design space. Furthermore the information necessary to perform this type of analysis is generally not available at the specification in time [13].

- **Logic Synthesis Level**

Logic synthesis provides the automatic synthesis of netlists minimizing some objective function subject to various constraints. Depending on the input specification, the target implementation, the objective function (area, power, delay, and testability) and the delay models used (zero-delay, unit-delay or library delay models), different techniques are applied to transform and optimize the original RTL description. Logic decomposition or delay insertion which lead to path balanced circuit structures and power dissipation may be reduced by gate resizing, signal-to-pin assignment and I/O encoding.

- **Physical Design Level**

Physical design level is an optimization step between the gate level netlist specification and the geometric (mask) representation known as the layout. It provides the automatic layout of the circuits minimizing some objective function subject to the constraints. Depending on the design style, the packaging technology and the objective function, various optimization techniques are used to partition, place, resize and route the gates.

At the physical design, power may be reduced by using appropriate net weights during netlists partitioning, floor planning, placement and routing. Individual transistors may be sized down to reduce the power dissipation along non-critical paths in a circuit. Large capacitive loads can be buffered using optimally sized inverter chains, also wire and driver sizing may be combined to reduce interconnect delay with only a small increase in power dissipation.

- **Circuit Design Level**

In this level of design abstraction, the power estimation is done using tools like SPICE. Techniques based on combining self-timed circuits with a mechanism for selective adjustment of the supply voltages that minimizes the power while satisfying the performance constraints.

1.2 Thesis Goals and Approach

The goal of this research work is to synthesize the RTL that performs a Wavelet Transform for maximum and minimum values of the objective functions. The objective functions for this synthesis are area, power and delay. The research work concentrated on high-level synthesis (behavioral and logic) as this level in design abstraction provides a better estimation of the objective functions. High-level synthesis has initiated considerable interest in the recent years [13]. The high level synthesis was followed by physical level synthesis (place and route). The result would be ASICs that's were characterized by their speed, size and power consumption.

The first part of this ASIC design process is to describe and implement the wavelet transform algorithm using the hardware description language, VHDL. The RTL coding for this research work was obtained from Honeywell Inc. Then the RTL was tested for its functionality. It was done in two stages; in the first part wavelet transform, implemented using VHDL was performed on a set of image data.

The input was also processed using a software routine written in 'C' which does wavelet transform to quantify the difference between the techniques. The second approach was to compile and simulate the RTL using the Mentor Graphics design tool. The values obtained from the simulation were compared with the similar input used in a C routine, which performed wavelet transform. In both the cases the error between software and hardware routines were found, and the functionality was tested.

The next step was to derive the design constraints for synopsys synthesis and optimization. These constraints were setup in Design Analyzer and the synthesis was performed for multiple values of area, power and delay. The gate level netlist was then placed and routed using a design tool EPOCH. Constraints were also set up in this tool for compilations and place & route. The result of this research work was functionally tested ASICs, which were characterized by maximum and minimum values of area, power and delay. This work finds applications in libraries used by design tools. The user could specify the range of values for either area, power or delay of the devices when designing a circuit. The devices in the library would be optimized for area, power and delay and corresponding to the range of area, power and delay specified by the user, a device with those attributes could be used for the design.

1.3 Chapters Contents

Chapter two is a tutorial on the theory of signal representation of both stationary and non-stationary signals. It also reviews the wavelet theory. An introduction is then given about how area power and delay consideration affect these parameters. Chapter two ends with a brief explanation of the ASIC design process or the ASIC flow.

Chapter three gives a brief introduction to the IC design process followed by a detailed explanation of the ASIC flow adopted for the research. This chapter also deals with the synthesis and optimization flow explaining in detail the constraints used. Results and discussions are presented in chapter four. Chapter five deals with summary and conclusions of this research work.

CHAPTER 2

Background

The first part of this chapter will give an overview of the techniques for time-frequency representation, their applications and the relative merits of each method. The second part of the chapter describes various sources of power dissipation, delay, and area in CMOS circuits.

2.1 Limitations of Transform Techniques in Signal Processing

Most signals in their original form are time domain functions. When we plot time-domain signals, we obtain a **time-amplitude representation** of the signal. This does not always give a complete representation of the signal for most signal processing related applications [1]. An important component, frequency remains hidden. It is frequency, which holds most of distinguishing information. There are many ways of getting a frequency spectrum of a signal, they are (1) Fourier transform (FT), (2) Radon transform, (3) Wigner transform, (4) Short Term Fourier Transform (STFT) etc. The most popular being the Fourier transform. However, the FT does not contain any information for the time instant when a particular frequency occurs. The requirement for the information depends on the type of signal and its application. In the case of stationary signals this is of paramount importance [1].

Signals from time invariant systems and stationary sources do not have their frequency components changing with time. This means that all the frequencies are present at all times. An example for this kind of a signal would be

$$s(t) = \sin\left(2\times\pi\times10\times t\right) + \sin\left(2\times\pi\times25\times t\right) + \sin\left(2\times\pi\times50\times t\right) + \sin\left(2\times\pi\times100\times t\right)$$

In this case the signal has its frequency components present at all values of 't'. Hence the need to know the time instants at which a frequency component occurs is not felt. The time domain representation of this signal is given in Fig 2.1a. We can see that the signal has four different frequencies of 10, 25, 50 and 100 Hz at any given instance of time. The frequency domain representation of the signal is given in Fig 2.1b. There are four spikes to represent the four different frequencies in the signal 10,25,50 and 100 Hz.

Fig 2.2a is a chirp signal in which the frequencies vary for various time instants. It has the same frequency contents as the signal shown in Fig 2.1a and its frequency spectrum is given in Fig 2.2b. The striking aspect of the signal in Fig 2.2b is that the spikes seem to appear at the same frequency instants as in Fig 2.1b. However, there are subtle differences, note the small ripples in Fig 2.2b that are due the sudden change in frequency in the input signal.

These types of signals where the frequency components are present only at specific intervals are called non-stationary signals. So we can conclude that FT can be applied to non-stationary signals if we are just interested in the frequency components and not in the instants where they occur. If we are interested in knowing the instants when such frequency components occur then the FT is not the suitable technique to use.

2.2 Time Frequency Representation

We have seen that FT can be used to represent the frequency spectrum of a non-stationary signal, if we are interested only in the frequency components but not in the time instants at which they occur. Time-Frequency representation is a technique by which the frequency components along with the instants at which they occur are illustrated. Many of the signals that we come across in day-to-day life are non-stationary.

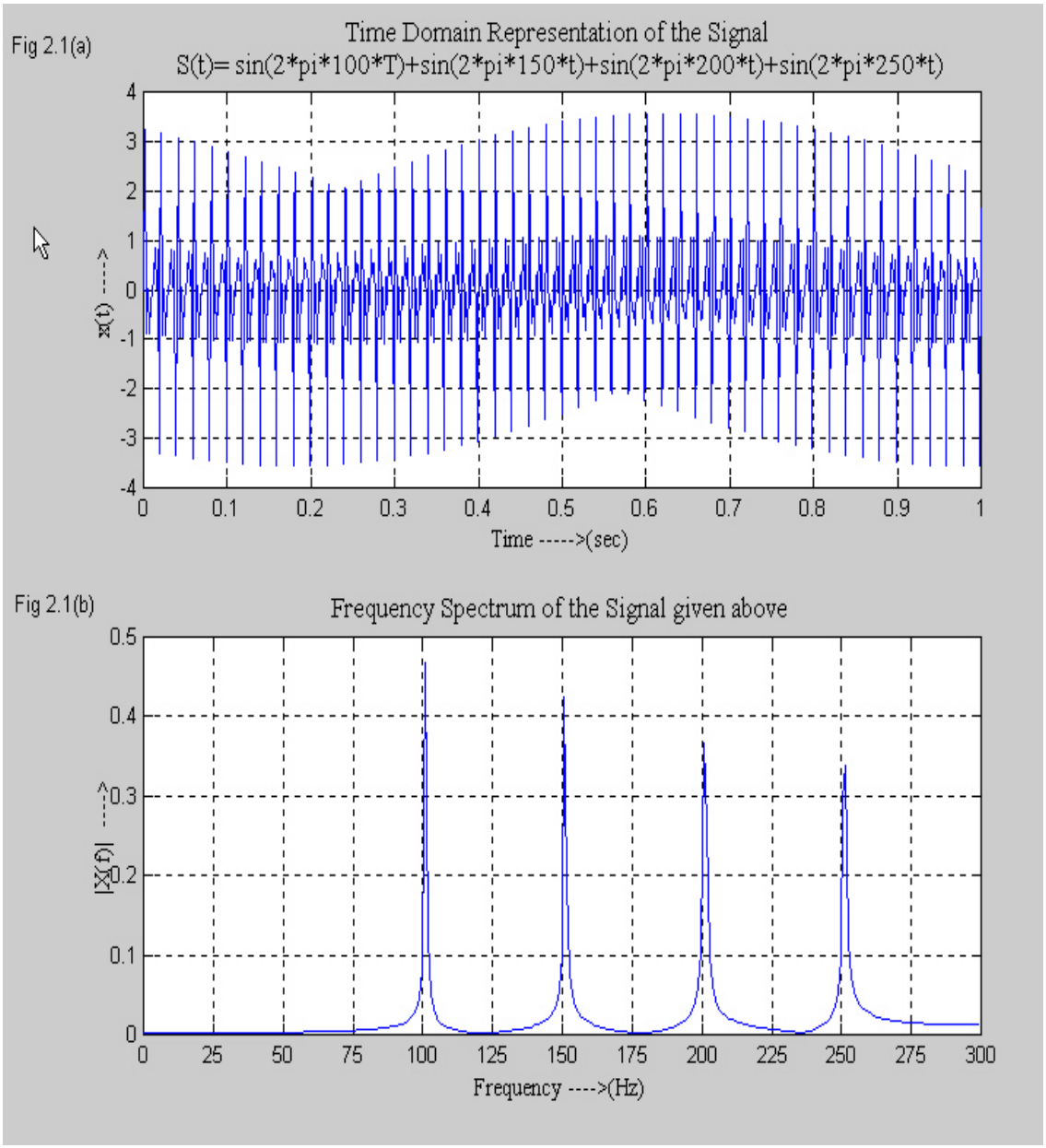


FIGURE 2.1 STATIONARY SIGNAL AND ITS FREQUENCY SPECTRUM

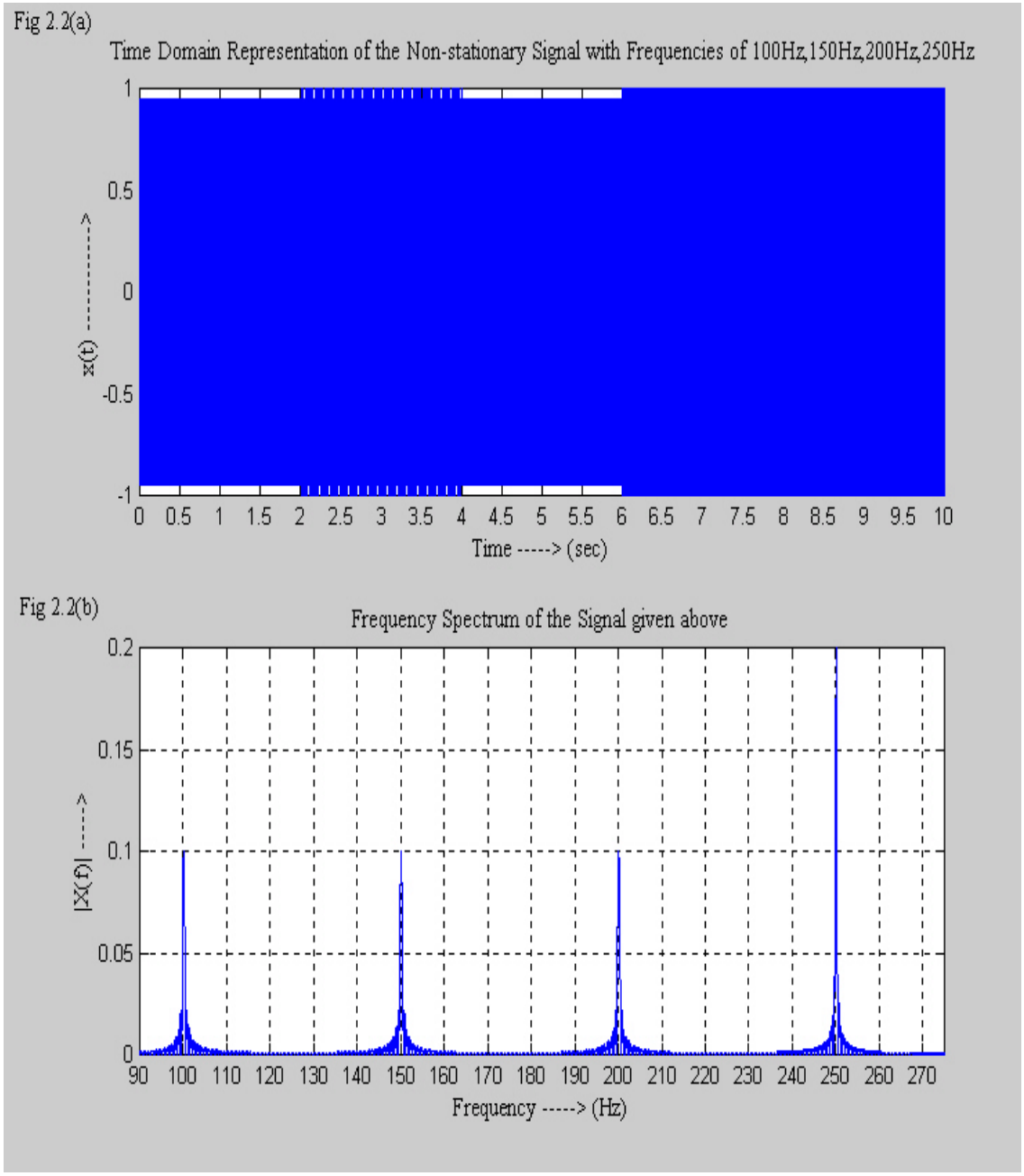


FIGURE 2.2 NON-STATIONARY SIGNAL AND ITS FREQUENCY SPECTRUM

The most frequent ones being Electroencephalogram (EEG) and Electrocardiogram readings. In these signals, the instances at which certain attributes occur play a big role in diagnosing the condition of a person. An example of this would be in the classification of different types of arrhythmia from the ECGs. There are 16 types of the disease and each level describes a certain level of severity. To diagnose a patient to one of the types of the disease, the values of ECG readings along with the instants at which they occur is required. Time-frequency representation using one of the techniques discussed below can be solution to this requirement. This technique would help in depicting the values of ECG readings for various instants, and prove the accuracy of diagnosing.

2.3 Time Frequency Representation Techniques

We need a transform method, which gives a representation of frequency components along with the time instants at which they occur. There are a number of ways of achieving this kind of representation. STFT is one way of achieving this. There is only a small difference between STFT and FT. In STFT, the signal is divided into small enough segments, where these segments of the signal can be assumed to be stationary. For this purpose, a window function **"w"** is chosen.

The width of this window must be equal to the segment of the signal where its stationarity is valid. The window is moved at equal time instants and multiplied with the signal. Then FT is taken on this product. The following gives the expression for STFT of a signal x (t).

$$\mathrm{STFT}_X^{(\omega)}(t_1, f) = \int_t [x(t) \times \omega'(t - t_1)] \times e^{-j2\pi ft} dt$$

The expression for the STFT gives a time frequency representation. This has given the result, which was not achieved through the FT. This technique has some shortcomings. The multiplication of the window function with the signal in time domain is a convolution in frequency domain. Also the window function used is of finite width, hence the frequency resolution is reduced. So we no longer know the exact frequency components present at various intervals of time, instead we only know band of frequencies present. In the case of FT, the kernel function is a window of infinite duration, and hence gave a good frequency resolution and poor time resolution in frequency domain. Smaller window width results in better time resolution and poor frequency resolution and wider window results in good frequency resolution and poor time resolution. If we increase the window width to infinity we get a standard FT.

2.4 Wavelet Transform and its Applications

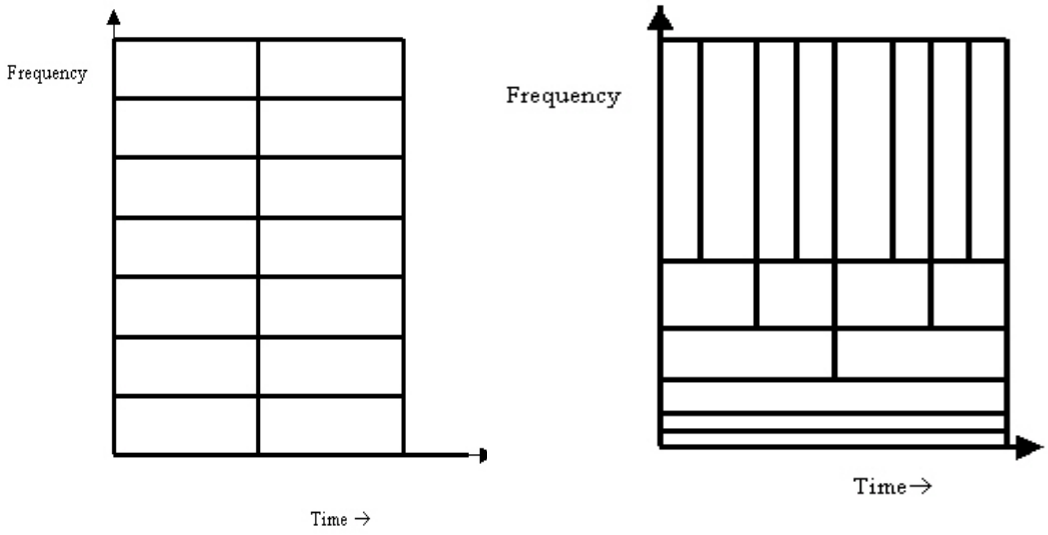
An alternative to the STFT technique that enables good time and frequency resolution is the Wavelet transform. A wavelet is a "small wave" which has its energy concentrated in time to give a tool for the analysis of transient, non-stationary or time varying phenomena [2]. This small wave is still oscillating in nature but also has the ability to allow simultaneous time frequency analysis. A wavelet representation of a signal is analogous to a music score where the notes intensity is shown with respect to time and frequency.

In the previous technique for time-frequency representation we came across a limitation i.e. the resolution of the resultant signal was constant for all frequencies. The advantage of Wavelet transform is in its multi-resolution property which allows it to be faster and more accurate because importance is given to the details while low frequency components are represented with less coefficients.

Multi-resolution analysis (MRA) as the name suggests analyzes the signal at different frequencies at different resolutions. Every spectral component is not resolved equally as it is in STFT. MRA is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies. This approach is practical since most of the signal we encounter in day-to-day life have high frequency content for a short duration and low frequency content for a longer duration. The multi resolution property can be seen by comparing the time frequency tiling offered in STFT with discrete wavelet transform as shown in Fig 2.3 [6].

In the case of the STFT, time and frequency resolutions are determined by the width of the analysis window, which is selected once for the entire analysis, i.e., both time and frequency resolutions are constant. So the time-frequency plane consists of equal sized boxes as shown in Fig 2.3a. Every box in Fig 2.3 b corresponds to a value of the wavelet transform in the time-frequency plane. We see that the boxes have a certain non-zero area, which implies that the value of a particular point in the time-frequency plane cannot be known. All the points in the time-frequency plane that falls into a box is represented by one value of the WT. In the Fig 2.3b, each box represents an equal portion of the time-frequency plane, but giving different proportions to time and frequency. At low frequencies, the height of the boxes are shorter (which corresponds to better frequency resolutions, since there is less ambiguity regarding the value of the exact frequency), but their widths are longer (which correspond to poor time resolution, since there is more ambiguity regarding the value of the exact time). At higher frequencies the width of the boxes decreases, i.e., the time resolution gets better, and the heights of the boxes increase, i.e., the frequency resolution gets poorer. The expression for continuous wavelet transform is given by

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s}\right) dt$$



a) STFT basis **b) Wavelet basis**

FIGURE 2.3 TIME FREQUENCY TILING FOR STFT AND DWT

The terms s and τ in the above expression refers to scaling and translation. Translation can be compared to time instant as in STFT and scaling can be compared to frequency, only that lower scaling means higher frequency and vice versa. The signal analysis method in the case of wavelet transform is shown in Fig 2.4.

One major application of Wavelet transform is data compression, because wavelets enable high compression ratio with good quality of reconstitution and low computation complexity. Fig 2.5 shows the result of wavelet transform on an image. Fig 2.5a is the image data that was used as input for the wavelet transform. Fig 2.5b is the result of WT on the input image. The resultant image obtained from the inverse transform shown in Fig 2.5c shows the effectiveness of wavelet transform in performing the synthesis of a signal. All transform techniques are lossy in nature, which means on transforming a signal from one domain to another and re-transforming it to the original domain may result in the some loss of information in the signal.

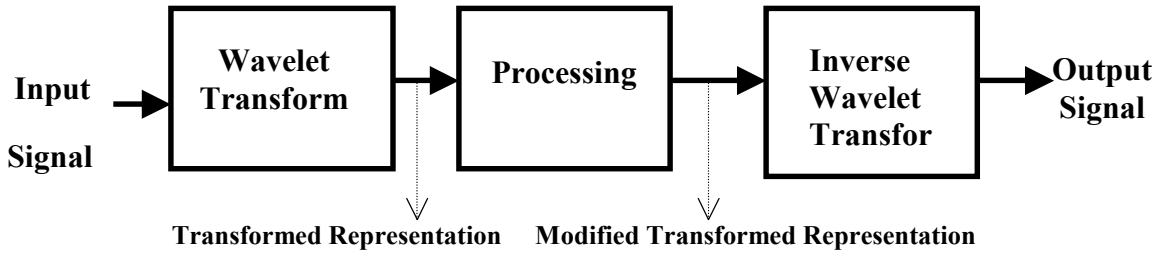


FIGURE 2.4 OVERVIEW OF THE WAVELET SIGNAL ANALYSIS METHOD

We see in Fig 2.5c that the resultant image is a close approximation of the input signal. We can clearly find that the number of pixels required by the original picture is far more than that of the resultant image. This shows its applicability in data compression.

Figure 2.6 gives an illustration of the quality of the reconstitution of wavelet transform. The input shown in Fig 2.6a are the gray level values of the left most column of an image file. Wavelet transform was performed on the data using commercially available software. The result of forward transform is shown in Fig 2.6b. Then inverse wavelet transform was performed to the result of forward transform and is given in Fig 2.6c. Comparing the original data set and the data got after performing the inverse transform we see that the magnitude of difference is not substantial. The values are pretty close to each other, and this shows the quality of reconstitution of wavelet transform. This quality of wavelet transform finds application in data compression, image processing, data transfer etc.

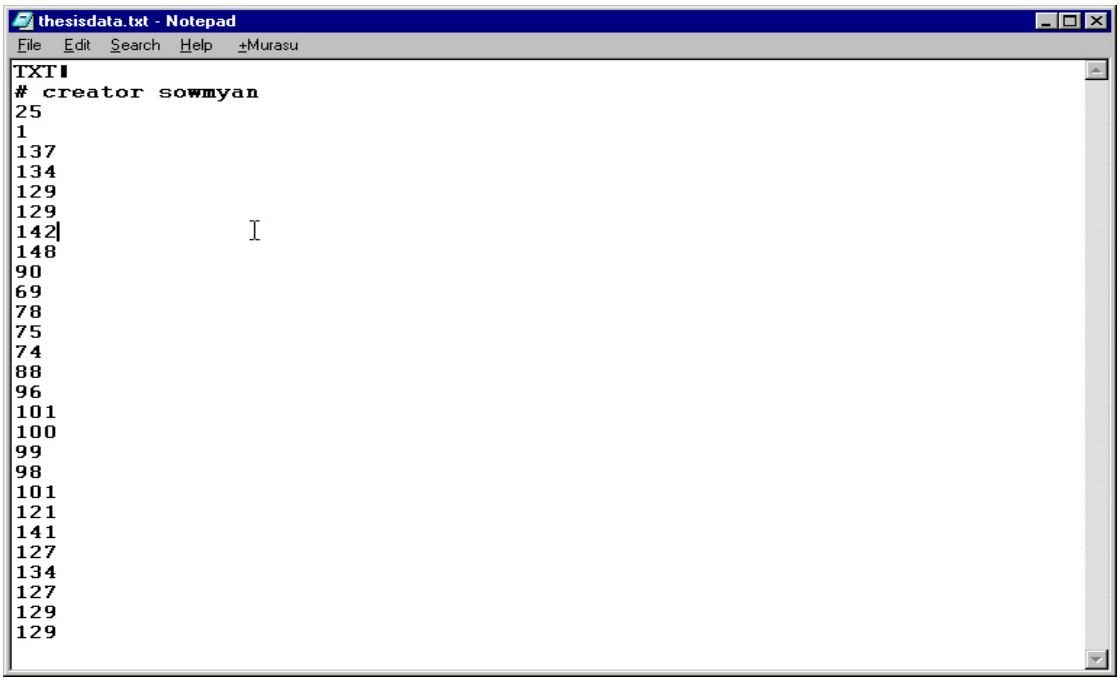
a) Image Data

b) Transformed representation

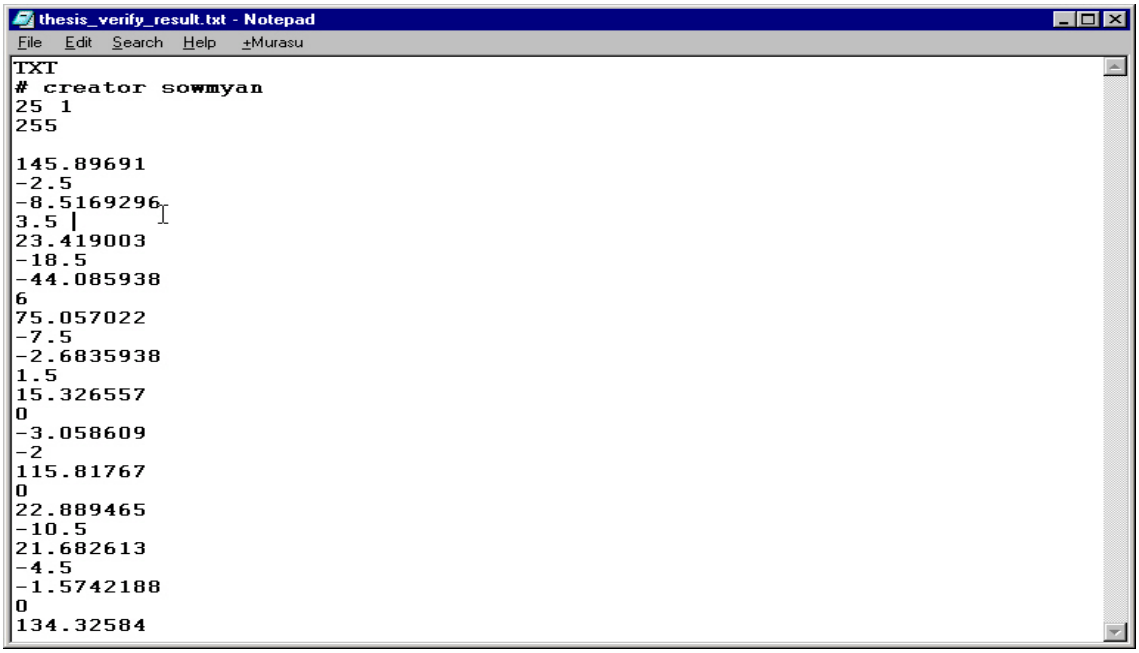
(After Wavelet transform)

c) Modified transformed representation
(After Inverse wavelet transform)

FIGURE 2.5 WAVELET TRANSFORM ON AN IMAGE

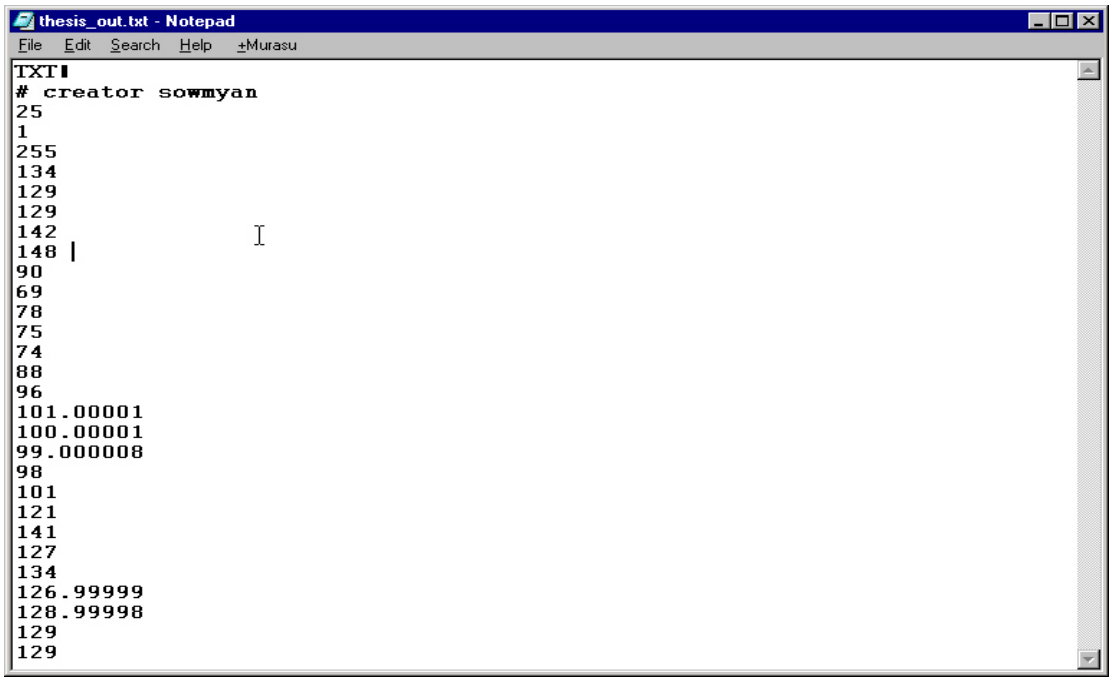


a) Input Data



b) After Wavelet Transform

FIGURE 2.6 QUALITATIVE COMPARISON OF THE RESULT OF WAVELET TRANSFORM ON A DATA SET



c) After Inverse Wavelet transform

Figure 2.6 (Continued)

2.5 Power, Delay & Area Consideration

This section will give an overview about the factors that affect chip area, power and delay.

2.5.1 Power

There are three components that establish the amount of power dissipated in a CMOS circuit. They are

- Static Dissipation
- Dynamic Dissipation
- Short Circuit Dissipation

2.5.1.1 Static Dissipation

This is due to leakage current or other current drawn continuously from power supply. The static power is ideally equal to zero. Consider the CMOS inverter in Fig 2.7, the input is '0' and the N-device is switched off while the P-device is conducting. Hence the output voltage is '1'. Fig 2.8 illustrates the working of a CMOS inverter as a switch. When the input is '1'as shown in Fig 2.9 and Fig 2.10, the N-device conducts while the P-device is switched off. The output voltage in this case is '0' or Gnd. We see that no current flows into the gate terminal and there is no DC current path from Vdd and ground, hence P_s (static dissipation power) is zero. However, there is some small static dissipation due to reverse bias leakage between diffusion regions and the substrate. Sub threshold conduction also contributes to the static dissipation. Total static power dissipation P_s is expressed as

$$P_s = \sum_{1}^{n} \text{leakage current} \times \text{supply voltage}$$

where n= number of devices

The contribution of the static power is generally ignored because of the low range (between .1 nA and .5 nA) of the leakage current, which creates a power dissipation equal to less then .05mW for hundred thousand devices operating with a Vdd of 5V[3].

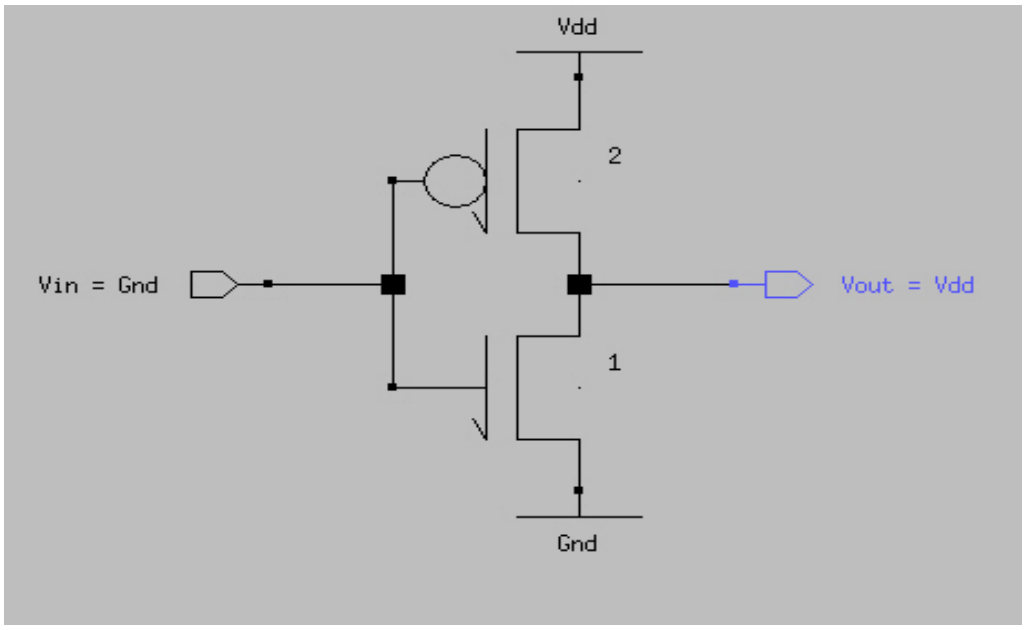


FIGURE 2.7 CMOS INVERTER MODEL WITH LOW INPUT

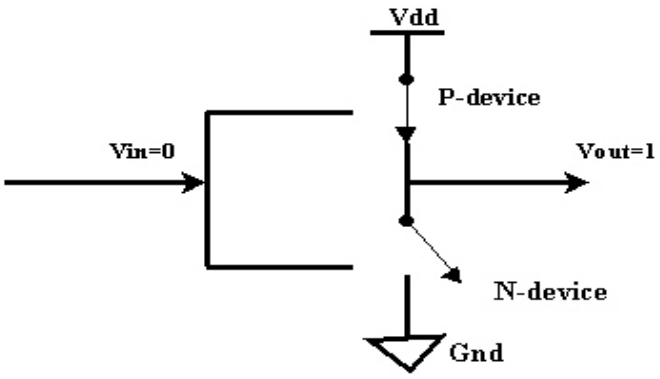


FIGURE 2.8 CMOS INVERTER REPRESENTED AS A SWITCH

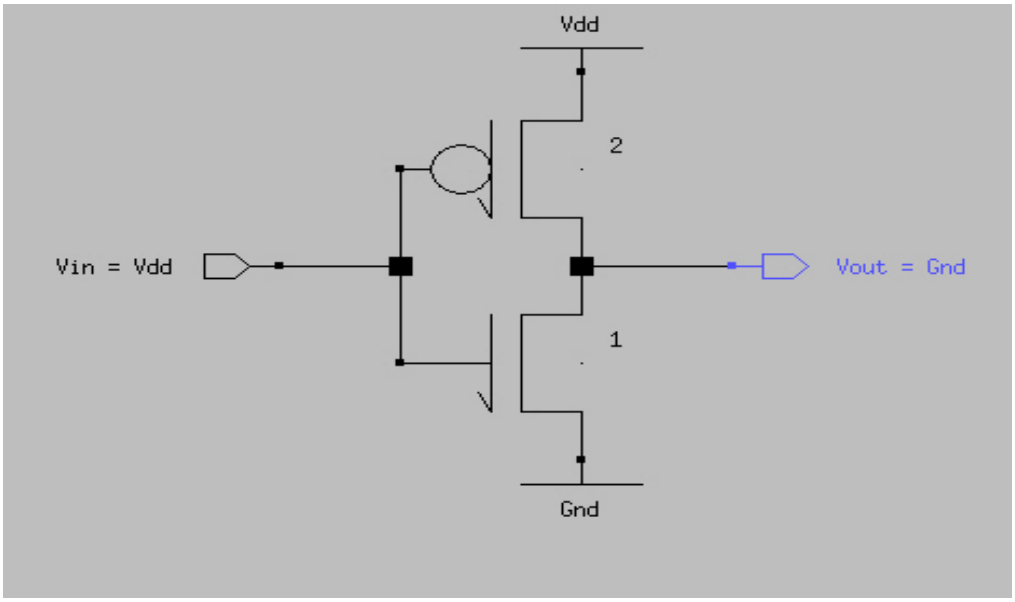


FIGURE 2.9 CMOS INVERTER MODEL WITH HIGH INPUT

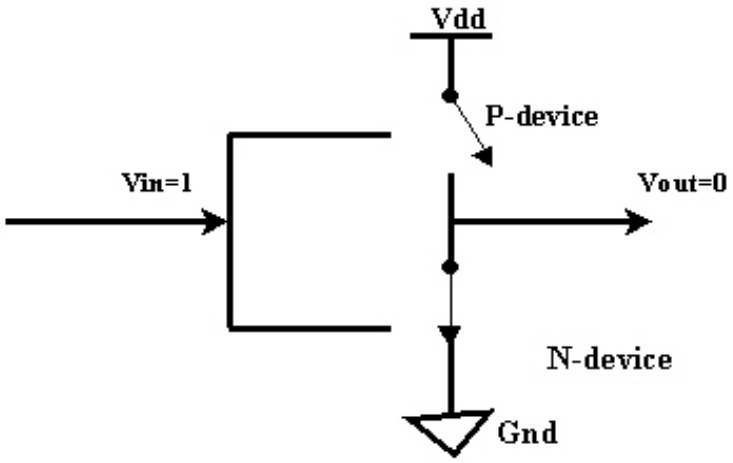


FIGURE 2.10 CMOS INVERTER REPRESENTED AS A SWITCH

2.5.1.2 Dynamic Dissipation

The dynamic power is the dominant component of the power consumption and is given by

$$P_d = \sum_{\text{i}} \alpha_i f_c C_{Li} Vdd^2$$

$\mathbf{C}_{Li}$ is the load capacitance on the gate output node i

α_i is the probability of the input node changing on each clock cycle

$\mathbf{f}_c$ is the clock frequency

Vdd is the operating voltage

Fig 2.11 illustrates a condition when the discharge and charge current due to capacitive loading dominates the current drawn from the power supply. Dynamic power consumption is proportional to the square of the supply voltage, and hence one way of reducing this dissipation is to reduce the supply voltage. However the reduction of supply voltage would also reduce the speed performance because the propagation delay is dependent on the ratio V_{th}/Vdd. The challenge is then to reduce both the factors to keep a low propagation delay while decreasing the dynamic power dissipation. The power dissipation due to these hazards is also called the toggle power and may be up to 67% of the global power dissipation [4].

2.5.1.3 Short-Circuit Dissipation

Another source of power dissipation is the short circuit power consumption, which is produced when CMOS gates switch. As seen in the Fig 2.7 and 2.9, both the CMOS transistors when switching between ‘0’ and ‘1’ would be ON for a short period of time.

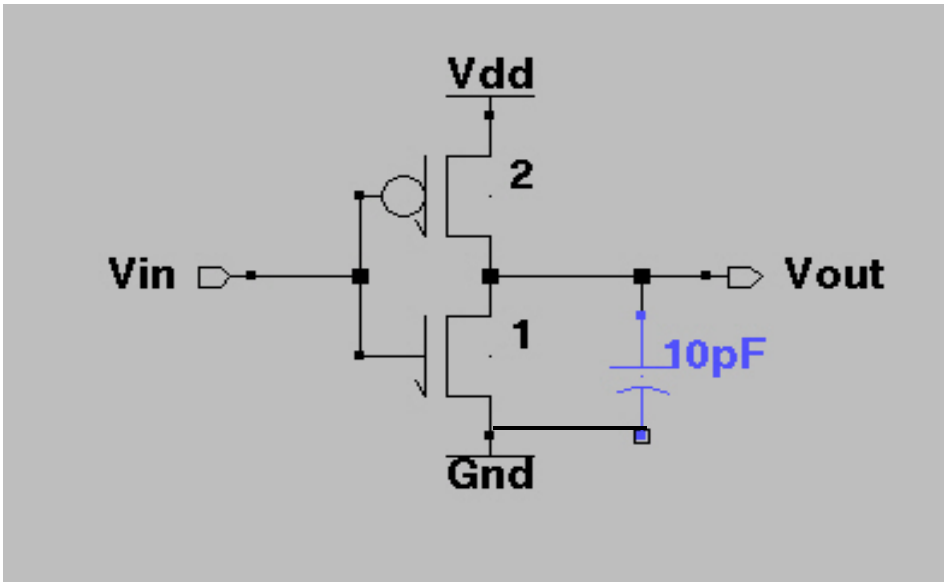


FIGURE 2.11 DYNAMIC POWER DISSIPATION MODEL

This results in a short current pulse between Vdd and Ground. This current pulse is dependent on the input rise/fall time, the load capacitance C_l and gate design. The short circuit power dissipation is given by

$$P_{sc} = I_{mean} \times Vdd$$

Average short circuit power dissipation can also be expressed by [19]

$$P_{sc} = (P_{scr} + P_{scf}) f$$

Where P_{scf} refers to fall short-circuit power dissipation and P_{scr} to rise short-circuit power dissipation. The total power dissipation can be obtained from the sum of the three dissipation components, given by

$$P_{total} = P_s + P_d + P_{sc}$$

2.5.1.4 Power Economy

In large designs the need for minimizing power is critical, so each module is assigned a power budget. This is a power dissipation that the module should not exceed. There are a number of ways by which power can be reduced. DC power dissipation may be reduced to leakage by using complimentary logic gates. The leakage in turn is proportional to the area of diffusion, so the use of minimum-sized devices is of advantage. Dynamic dissipation can be limited by reducing supply voltage, switched capacitance and the frequency at which logic is clocked [5].

2.5.2 Delay

The switching speed of a CMOS gate is limited by the time taken to charge and discharge the load capacitance. An input transition results in an output transition that either charges the load capacitor towards Vdd or discharges load capacitance to Gnd. The delay time or the propagation delay t_d is the time difference between the input transition (50%) and the 50% output level. This is the time taken for a logic transition to pass from input to output. The CMOS device shown in Fig 2.12 has a load capacitor C_l connected to it. The fall time can be given by

$$t_f \approx k \times \frac{C_1}{\beta_n Vdd}$$

Where k =3 to 4 for values of Vdd =3 to 5 volts and V_{tn} =. 5 to 1 volt. We see that the delay is directly proportional to the load capacitance and inversely proportional to Vdd. Thus on reducing the load capacitance and increasing the supply voltage one may achieve a faster circuit. Also on increasing or decreasing the width of the transistor we can alter the speed of the circuit.

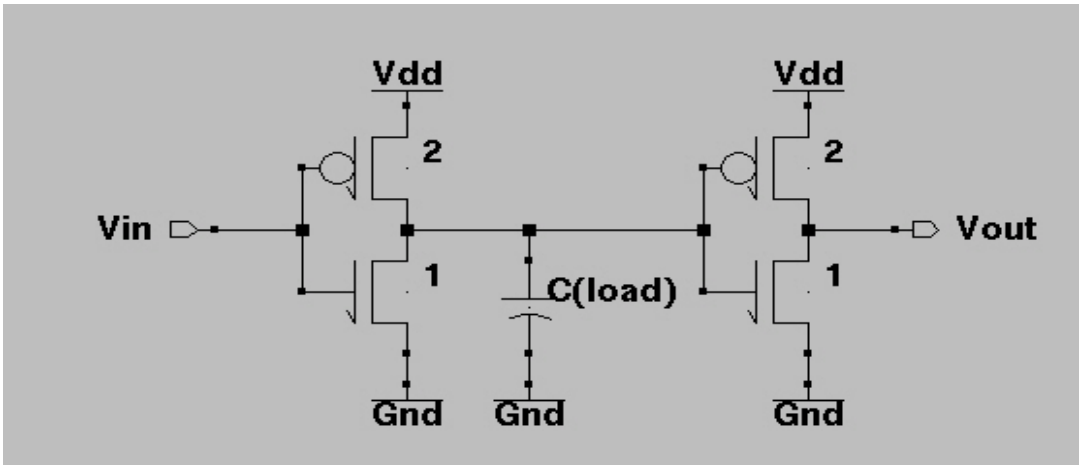


FIGURE 2.12 SWITCHING CHARACTERISTICS MODEL

When changing the supply voltage value care must be taken as the propagation delay is related to the ratio of V_{tn}/Vdd, so a proportional change of V_{tn} and Vdd must be done to ensure a controlled change in propagation delay.

The rise time can be given by

$$t_r \approx k \times \frac{C_l}{\beta_p Vdd}$$ So for an equal sized n and p transistors, $t_f = t_r / 2$

The delay of a single gate is dominated by the output rise and fall time and is given by

$$t_{dr} = t_r / 2$$

and

$$t_{df} = t_f / 2$$

and the average gate delay for rising and falling transitions is

$$t_{av} = \frac{t_{df} + t_{dr}}{2}$$

2.5.3 Area

Routing area, area of diffusion, transistor size and switching capacitance are some of the important factors that influence the area of a device. The routing area, for design rules with two metal layers, is about 50% of the chip area [7]. For design rules with three metal layers, it is possible to aggressively use over-the-cell routing [8] to reduce routing area. The leakage power dissipated by CMOS devices is dependent on the area of diffusion.

So reducing the area of diffusion reduces the cell area and the power dissipation. Also a cell with no diffusion gaps has minimal area and minimal parasitic capacitance [7]. Minimizing the switched capacitance results in minimum-sized devices and helps in allocation of resources like adders and registers. Manual layout techniques are also used to reduce routing capacitance.

An important issue in the manufacture of VLSI structures is the yield [9]. Yield is influenced by such factors like technology, chip area, and layout [5]. There are two models, which describe yield as a function of chip area.

The Seeds model [10], is given by

$$Y = e^{-\sqrt{AD}}$$ Where Y is the yield, A is the chip area, D is the defect density

This model is used for large chips and for yields less than 30 percent.

The Murphy's model [11], is given by

$$Y = \left[\frac{1 - e^{-AD}}{AD} \right]^2$$

This model is used for small chips and for yields greater than 30 percent.

The more recent generalized model is as follows [12]

$$Y = \prod_{i=1}^{N} \left(1 + \sum_{j} \frac{A_j D_i P_{ij}}{c_i} \right)^{-c_i}$$

Where i = the ith type of defect

j = the j th module

P_{ij} = Probability that an i defect will cause a fault in the j th area

c_i = the constant relating to the density of a ith type of defect.

These relations indicate that the yield decreases by a large magnitude when the chip area increases.

2.6 ASIC Flow

Figure 2.13 gives an illustration of ASIC flow that was used for this research work. Figure 2.8 has five blocks, the first one being VHDL implementation. In an ASIC design process, the first step is to have the ASIC specification implemented using a hardware definition language. VHDL was used for this description and the coding was obtained from Honeywell Inc.

Modifications include the addition of synopsys libraries, changes to some of the modules were made so that the RTL is compatible for synthesis and optimization. The next step was the functional testing of the RTL. The functional testing of the RTL was performed by comparing the results of wavelet transform on an image, implemented using VHDL and C. After testing the functionality of the RTL, Synopsys design Analyzer was used for synthesis and optimization. The design constraints were set up, and the tool was made to optimize the chip for maximum and minimum values of area, power and delay.

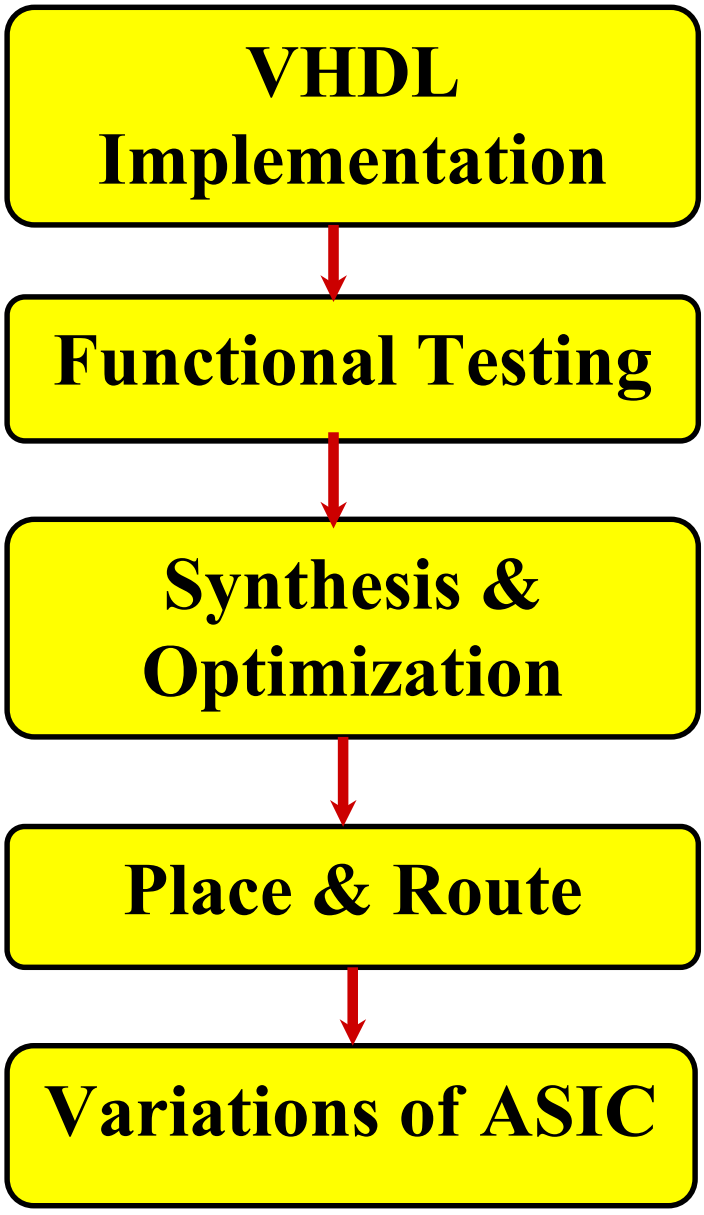


FIGURE 2.13 ASIC FLOW USED FOR IMPLEMENTATION

This gate level netlist was then placed and routed using a design tool EPOCH. The results of this flow were ASICs characterized by minimum and maximum values of area, power and delay.

CHAPTER 3

Implementation

This chapter describes the ASIC flow and the design methodology for the research reported herein. The design constraints used for synthesis and optimization are explained in detail.

3.1 Introduction to the Design Process

ASIC flow can be divided into four main subdivisions. They are (1) System architecture and planning, (2) Functional Implementation, (3) Layout and (4) Fabrication. The system architecture and planning stage consists of project planning and architecture specification. In this stage the designer decides on the methodology, the tools to be used, specification, models and matrices. The functional implementation stage consists of implementation, synthesis, timing analysis and testing. The designer implements the system architecture in a hardware description language, performs high-level synthesis and optimizes the RTL. Then timing analysis is done to verify the functionality. Scan circuits are then added to make the chip testable by the customer and the vendor after fabrication. The layout stage consists of floor planning, static timing analysis, routing, functional vectors generation, design rules check and physical verification. The designer works at the transistor level in this stage of hierarchy. The floor planning is performed on the layout to make it more efficient with respect to area and delay. Timing analysis and routing are performed. The design rules are checked and the layout is compared with the schematic (LVS) to check the functionality.

The next stage in the design process is fabrication. Once the layout is tested for its functionality, it is fabricated. This research work deals with the functional implementation and the layout stages in the ASIC hierarchy.

3.2 ASIC Flow Description

Fig 3.1 shows the ASIC flow used in this research. The hardware implementation of wavelet transform algorithm was done using VHDL. The RTL for this research work was obtained from Honeywell Inc. The next step in the ASIC flow was to test the RTL for its functionality, which was performed in two stages. The first part of the functional testing was to compare the results of wavelet transform on a set of image data implemented using VHDL and C. The methodology that was followed was to load the image data into the memory of the WILDFORCE board and then image was converted into 4 byte words. Wildforce board is a reconfigurable board developed by Annapolis micro systems. It has five FPGAs one control-processing element and four array processing elements. Wavelet transform was performed on the image data. The results were then copied from the memory of the wildforce board onto the host. Run-length encoding and entropy coding was performed on the host followed by a decompress routine. To compare the processed image with the input image a compare routine was executed. This was done to perform a qualitative analysis of the results. Software routines, which perform compression and decompression, were also executed. Fig 3.2 compares the result of compression on an image using VHDL and C routine. The images were decompressed using the same software routine. The compress routine performs wavelet transform, quantization, run length encoding and entropy coding.

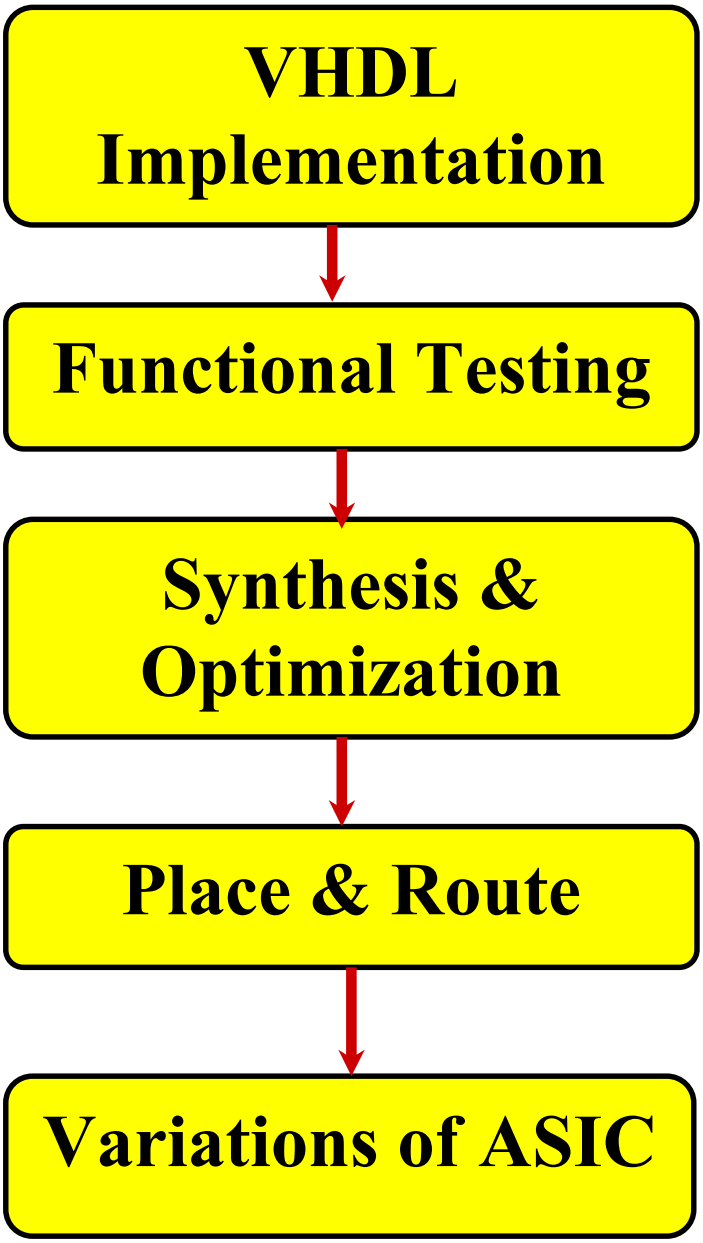


FIGURE 3.1 ASIC FLOW

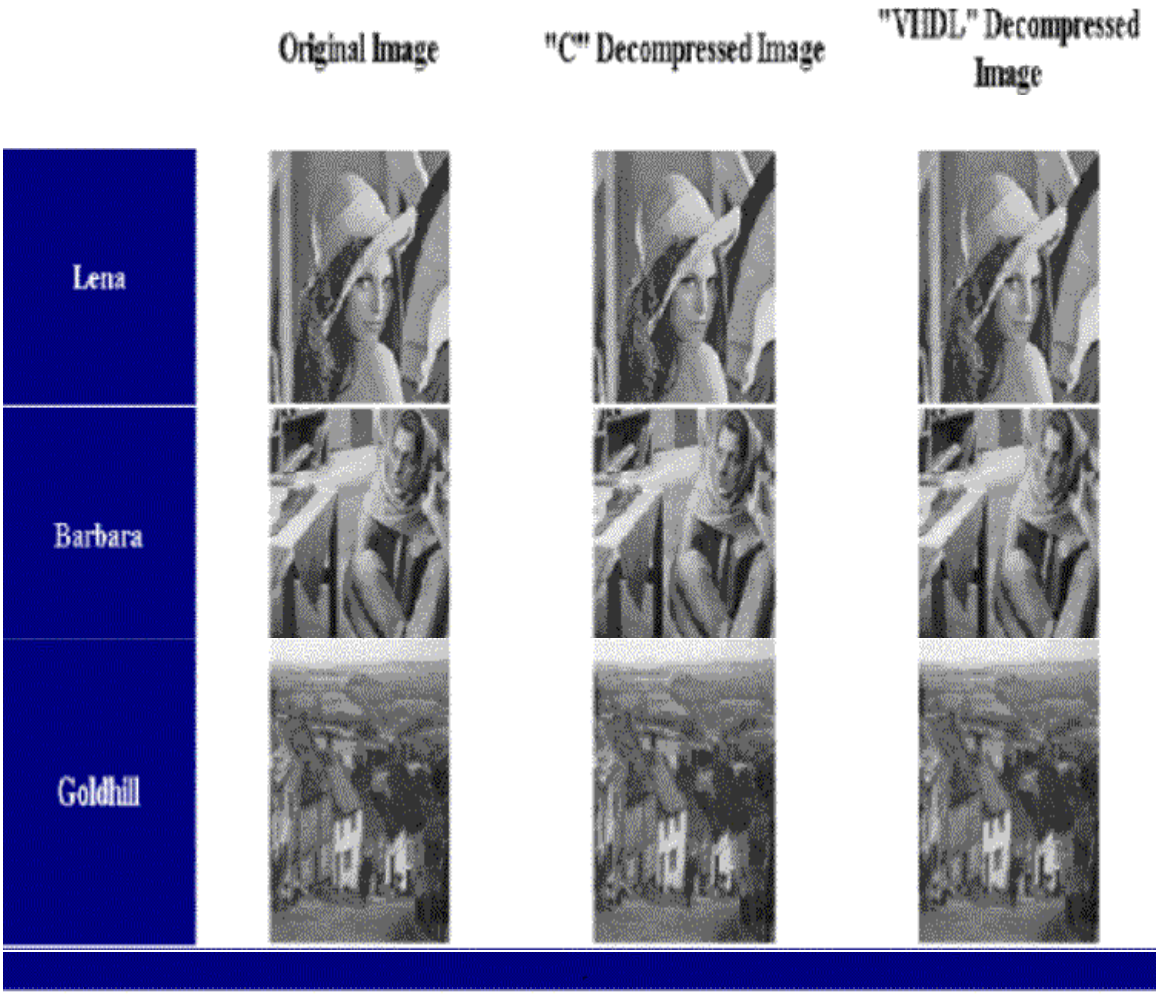


FIGURE 3.2 COMPARISON OF THE RESULTS OF WAVELET TRANSFORM ON IMAGES IMPLEMENTED USING VHDL AND C

The decompress routine performs inverse wavelet transform, dequantization, run length decoding, entropy decoding. Fig 3.3 gives a quantitative comparison of the result of compress and decompress routines on an image data. The second part of the functional testing was performed using the Mentor Graphics FPGA Advantage and a software routine that performs wavelet transform. The RTL was compiled and simulated using FPGA Advantage. The input in this case was the gray level values of the left most column of an image file. The result was then compared with the output of the software routine. The execution of the software routine is shown in Fig 3.4 with Fig 3.5 giving the comparison of the result from both of the techniques. The error was calculated by comparing the pixel values. The difference in pixel values after the transform can be attributed to the fact that VHDL uses integer data type and C uses floating point during arithmetic. The algorithm used for the implementation of wavelet transform could also play a role in the difference in pixel values.

The next stage in the ASIC flow hierarchy was the Synthesis and Optimization at the logic level. The optimization was done for library defaults, area optimization, power optimization and delay optimization. The constraints to optimize the objective functions were setup along with the environment variables. This was followed by place and route of the gate level netlist using a design tool, EPOCH. The result of this flow were ASICs that were distinguished by maximum and minimum values of the objective functions.

3.3 Synthesis and Optimization

This section describes the constraints used for optimization. The Synopsys design analyzer was used for the high level synthesis and EPOCH was used for the physical level synthesis.

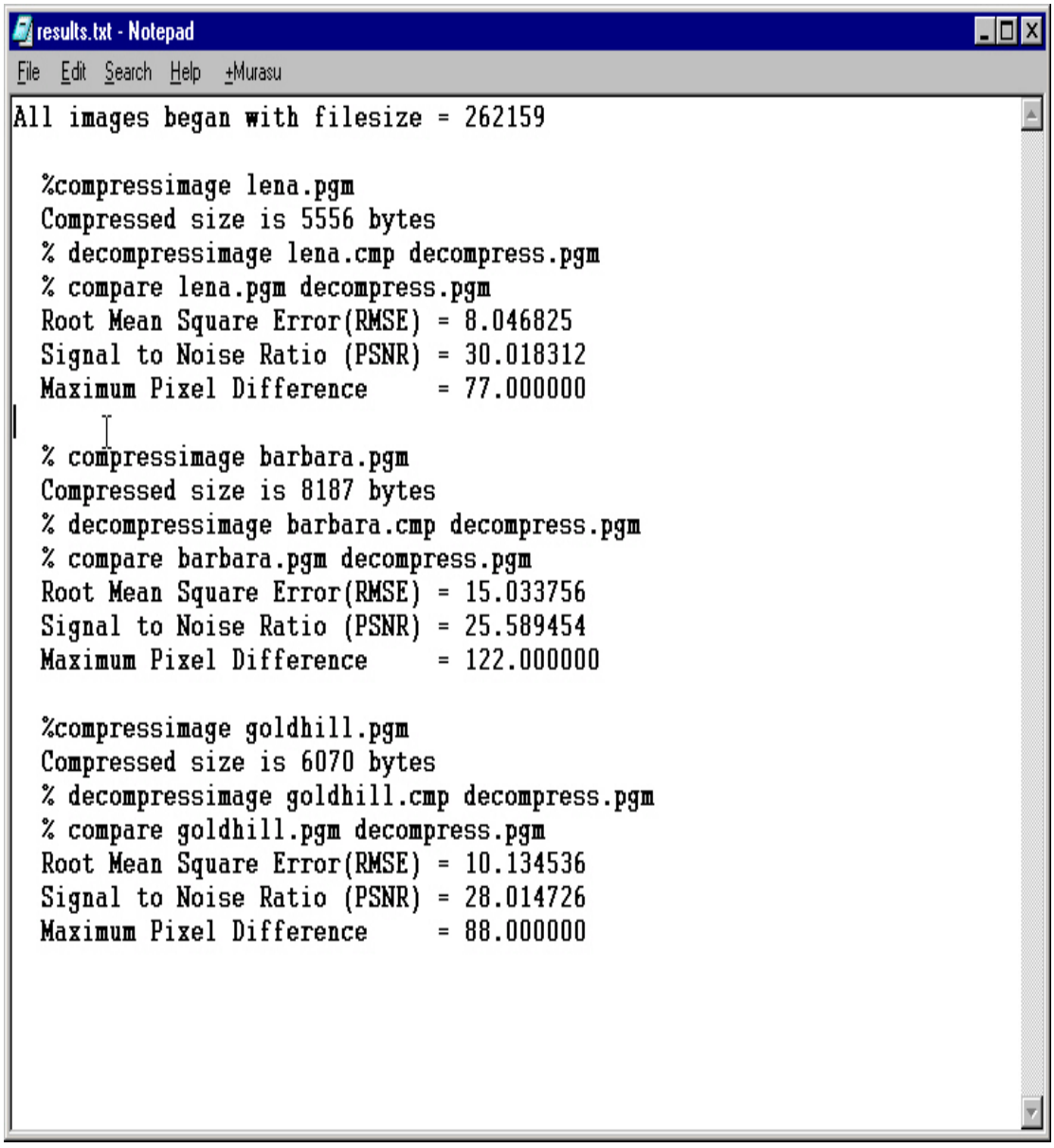


FIGURE 3.3 QUALITATIVE COMPARISON OF THE RESULTS OF WAVELET TRANSFORM ON AN IMAGE DATA

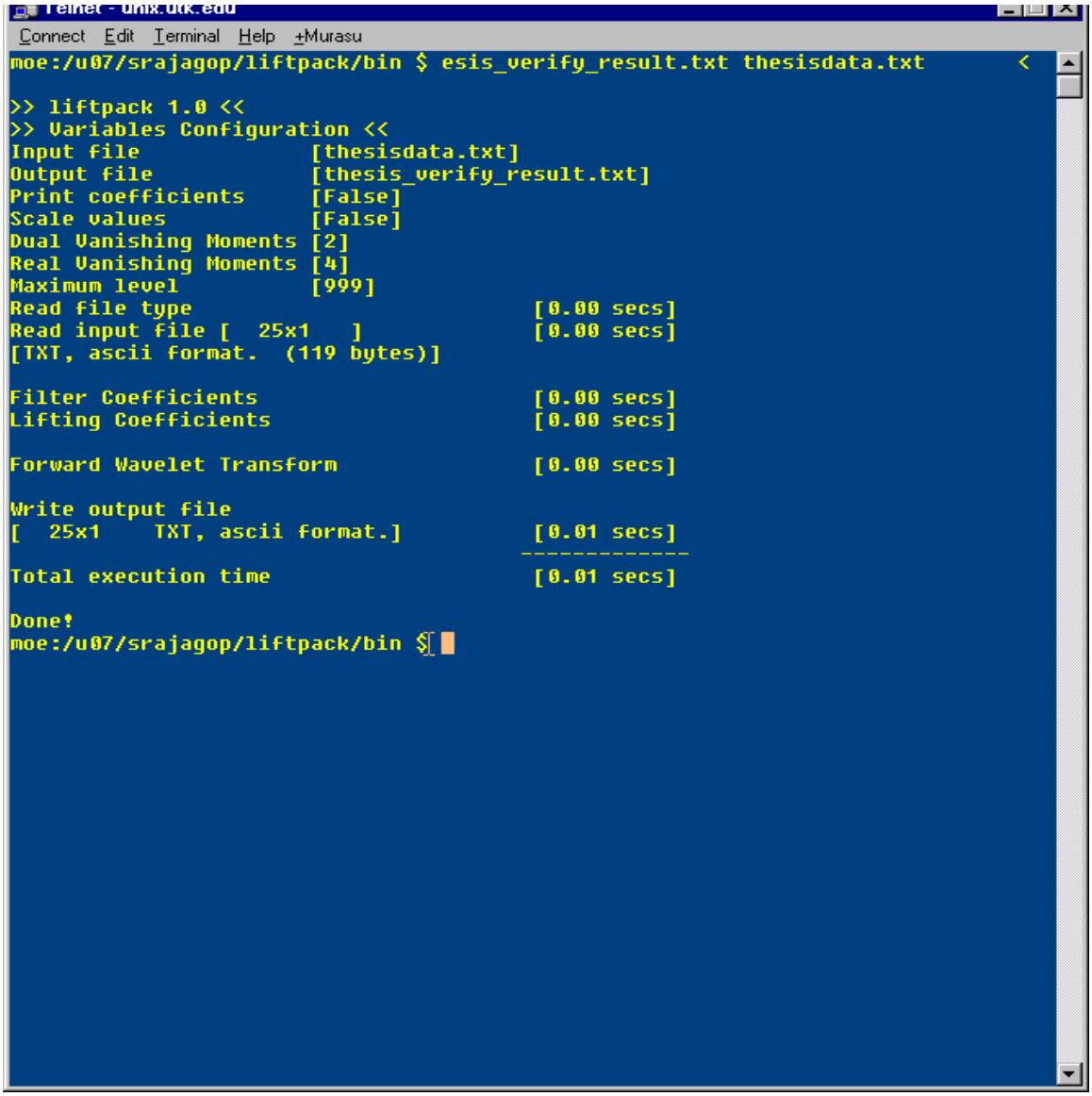


FIGURE 3.4 WAVELET TRANSFORM ON AN TEXT INPUT USING A SOFTWARE ROUTINE

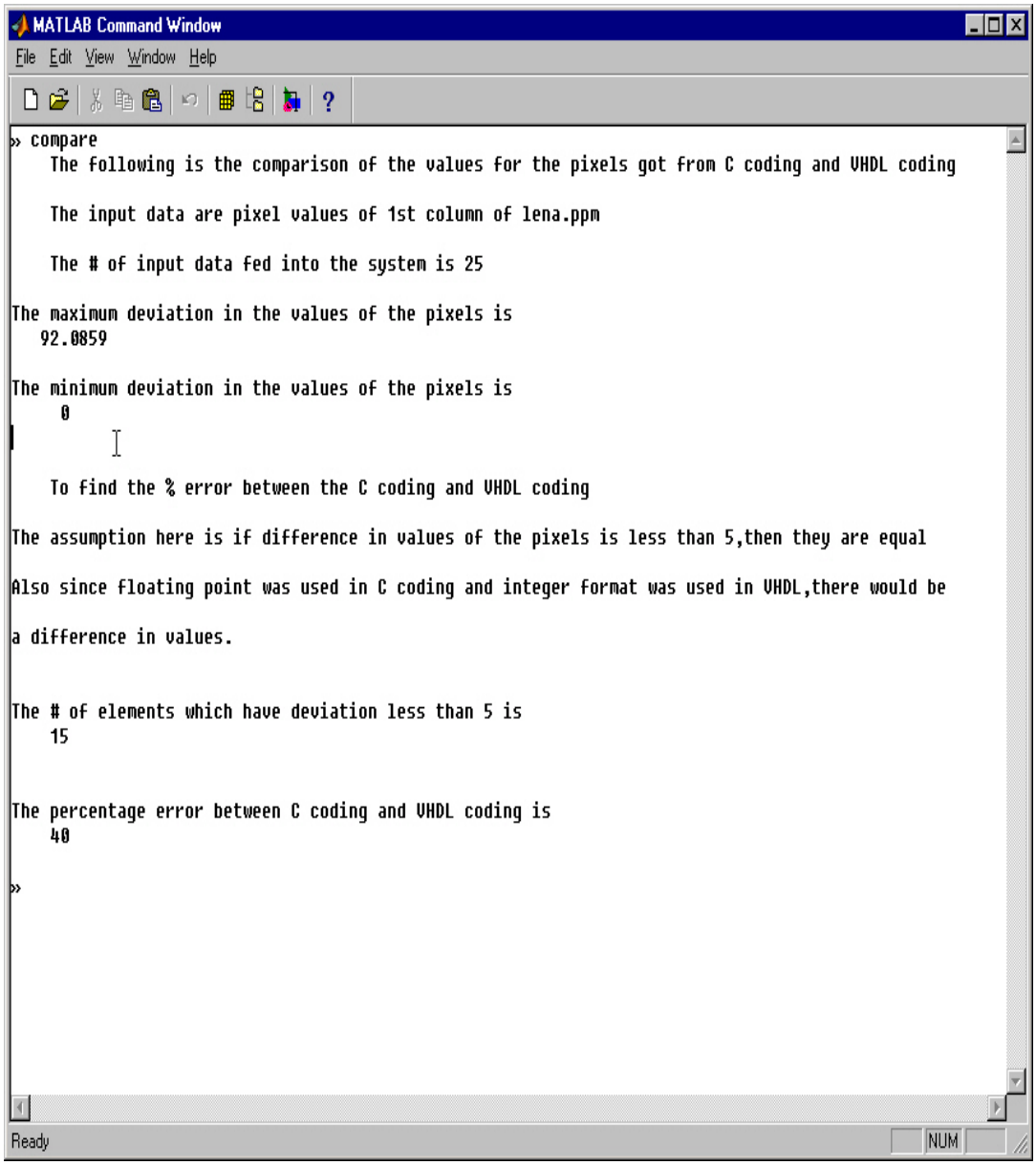


FIGURE 3.5 COMPARISON OF THE RESULTS OBTAINED USING VHDL AND C ROUTINE

Most of the constraints were set up at the logic synthesis level, since the synthesis provides a better estimation of area, power and delay. Fig 3.6 shows the flow adopted for synthesis and optimization in this research work. Also the magnitude of optimization of these objective functions is better at the high level rather than at the low level. The constraints that were used are (1) Input and output delay, (2) Timing range, (3) Maximum area, (4) Maximum power, (5) Minimum and maximum delay, (6) Deriving timing constraints, (7) Flattening and (8) Compile options.

- **Input and Output delay**

The input and output delays characterize the operating environment of the current design. This constraint sets the input or output path delays on the input and output ports relative to the clock edge. Input and output ports are assumed to have zero delay, unless specified. The delay values must be consistent with the technology library used during optimization. Input delay values represent the amount of time the signal is available after a clock edge. This usually represents a combinational path delay from the clock pin of a register. The clock is used a reference for the delay. Output delay values represent the amount of time that the signal is required before a clock edge. The maximum output delay represents a combinational path delay to the register plus the library setup. The minimum output delay represents the shortest path delay to a register minus the library hold time. These constraints were setup accordingly for maximum and minimum delay synthesis.

- **Timing range**

They are scaling factors that are used to scale timing path totals. This constraint is referenced as speed profile in the synthesis flow. The slowest factor is the largest value of any of the specified time ranges and the fastest factor is the smallest of the specified time ranges.

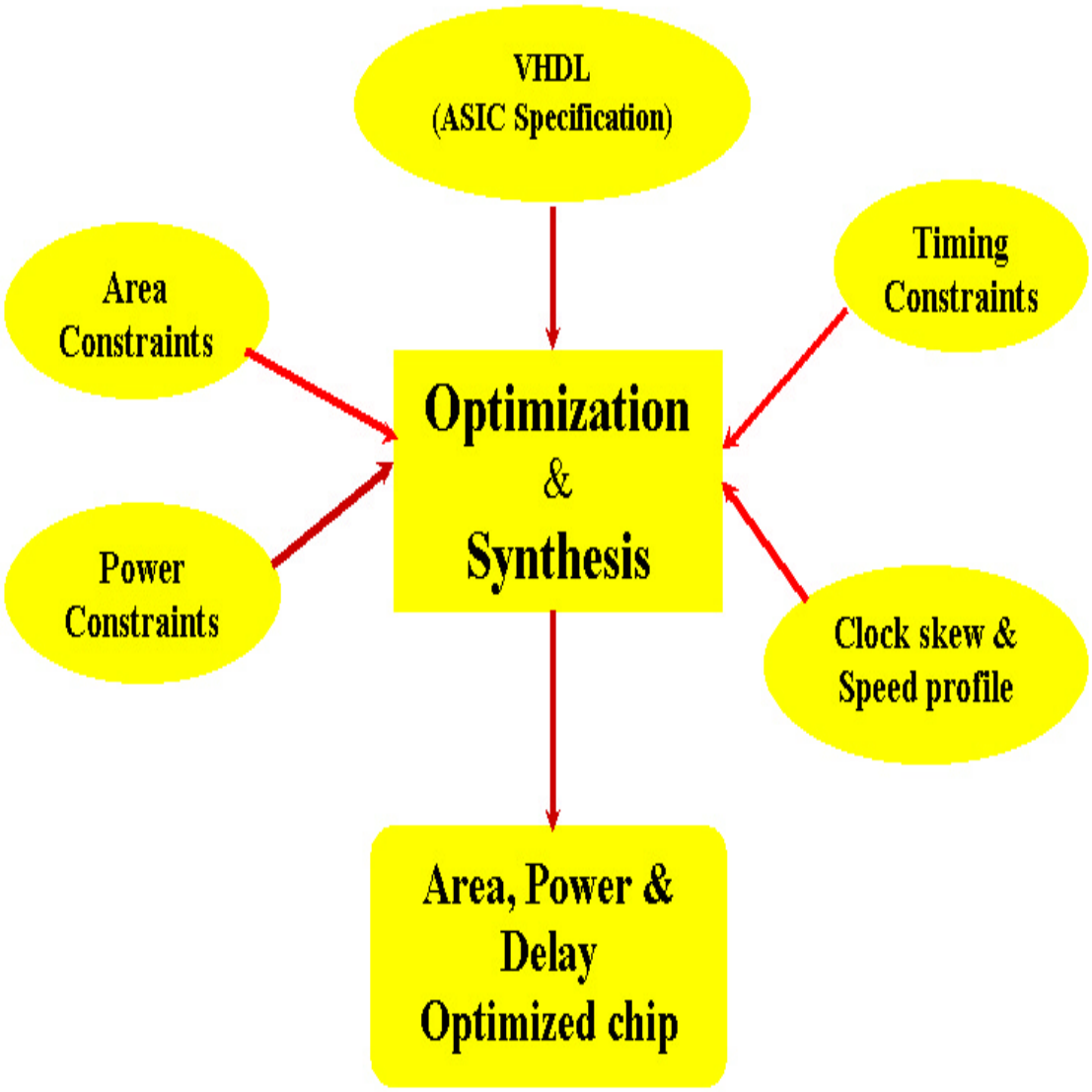


FIGURE 3.6 SYNTHESIS AND OPTIMIZATION FLOW

On choosing the faster factor, the arrival times are scaled by a floating number to model faster times. The slower factor scales the arrival time by a floating-point number to model slower times due to environmental variations.

- **Maximum Area**

This constraint sets the maximum area attributes to the current design. This attribute represents the target area of the design and is used by compile to calculate area cost of the design. When optimizing for maximum area, a value larger than the area found by using library defaults was setup for the synthesis and for minimum area optimization a value smaller was used as a constraint.

- **Maximum Power**

This constraint sets the maximum power attributes to the current design. This constraint represents the target power of the design. The constraints for maximum and minimum power were setup similar to the maximum area constraint.

- **Area and Timing critical**

This constraint makes area or timing constraints to take priority over meeting specified lower bound on fault grade. Fault grade means the extent of the circuit that could be tested, higher the fault grade better the chances of identifying a fault in the design. If the fault grade were to be increased, which means more scan circuits in the design, then there would be an increase in area and delay of the design. Note scan circuits are on board testing circuits. On the other hand, a lower fault grade means lesser scan circuits, hence a reduction in area and delay but a reduction in the testability of the design. In this research no scan circuits were incorporated into the circuit, since the main objective was to optimize area and delay. So a fault grade of 84% was achieved for all synthesis.

- **Minimum and Maximum delay**

Minimum delay constraint specifies the value of the desired minimum delay for paths between start and the end points and maximum delay specifies the desired maximum delay. The maximum and minimum delay could be specified between the ports, pins or the leaf cells. If a path start point is on a sequential device, clock skew is included in the computed delay. Clock skew reduces the allowable path delay due to combinational logic between the output of one flip-flop and the input of the next. Note skew is the extent of variation of the clock speed. If a path start point or the endpoint has a delay specified, then that delay is added to the path delay. If a path end point is on a sequential device, clock skew and library setup time are included to the computed delay.

- **Deriving Timing Constraints**

This option derives timing constraints from the existing timing on the current design. Both sequential and combinational timing constraints are derived from all previously unconstrained timing paths in the current design. Three options in deriving constraints were used in this research. They were minimum delay, maximum delay and maximum period. The minimum delay option derives the minimum delay timing constraints for unconstrained combinational paths. The derived constraint target value is set to the current timing of the path. The maximum delay option derives the maximum delay timing constraints for the combinational paths. The maximum period option derives the period attributes for the clock.

The constraint uses a scaling factor, which is multiplied to the existing timing on the design, and the new timing is placed on the design. In the case of maximum delay and period a scaling factor of less than one results in a more restrictive timing then the actual timing of the design. For minimum delay, a scaling factor of less than one results in a less restrictive timing.

- **Flattening**

This optimization constraint reduces a logic network to a two-level sum of products (AND/OR) representation. Flattening fuses multiple modules present in a design into one. For example in the design of a counter, the sub modules are a D-flip flop, XOR gate and an AND gate. In the layout, there would be three separate modules present and this would result in a larger delay and area. If the flattening option is enabled then the tool combines the three modules into one. Thereby resulting in a reduction of area and delay.

Full flattening eliminates all existing logic structure. Options like single/multiple output can be used along with flattening. With single output option, the tool tries to minimize the equations for an output at a time. So this results in the smallest implementation for a single output, but may not be most efficient for the design as a whole because the products are not well shared between the outputs. Using the multiple output option, the tool tries to minimize the equations for all the outputs at the same time. This results in an efficient design.

- **Compile Options**

There are three compilation options in design analyzer, and these determine the effort of synthesis and optimization. They are low, medium and high and the default is medium. In the case of EPOCH, the options used were automatic and timing driven. Timing driven compile was used in the case of delay optimization, while automatic was used for the rest of the synthesis.

CHAPTER 4

Results and Discussions

This chapter presents the results obtained from this research along with a detailed discussion. The synthesis and optimization were done for library defaults, area, power and delay optimization. The RTL after being tested for its functionality was synthesized and optimized. The synthesis and optimization was performed at logic design level and at physical design level with emphasis on logic level synthesis. At the logic level synthesis Synopsys design analyzer was used for optimizing the gate level netlist. It took between ten and twenty-five minutes to complete the entire synthesis flow. The logic level synthesis took longer to complete in comparison to physical level synthesis.

4.1 Results for Library Defaults

The first part of the synthesis was setting up the constraints for the optimization to be library defaults. With this option the design tool tries to minimize all the three objective functions simultaneously. So the result of this optimization is a chip that is smaller, faster and low power. The next constraint was to make the area critical along with the library defaults. Since the synthesis was already being performed for minimum area, and scan circuit were not being added to the design the result of this synthesis was a chip with area, power and delay similar to the previous case. The third option under library defaults was flattening. During this synthesis, single output option was enabled.

Since flattening fuses multiple modules into one, it was expected that there would be a reduction in area and delay but the result had an increase in area and a decrease in power of the chip. The reason could be attributed to the way the Synopsys tool performs flattening. Design analyzer reduces a logic network to a two level sum of products representation. With the single output option the tool minimizes the equations for every output separately. This results in an efficient design for an output but may not be for the complete design. Since products are not well shared between the outputs and hence an increase in area and reduction in power. The final option under library defaults was modifying the RTL so that it performs WT on a 256*256 image instead of 512*512. This was done to understand the effect of number of pins and input statistics. As expected there was a reduction of area, power and delay. The gate level netlist was then placed and routed for all the four cases and resulted in a small reduction in area, power and delay. Table 1 gives a summary of the results obtained by using library defaults during the synthesis.

Table 1: Optimization result – Library Defaults

Specification of the constraints	Area (Sq. μm)	Power (mw)	Delay (ns)	EPOCH Values	
				Area	Power
1) Library defaults	**222334.7**	**3.4297**	**9607**	**2.11e5**	**3.372**
2) Area critical	**222334.7**	**3.4297**	**9607**	**2.11e5**	**3.372**
3) Flattening	**232973.2**	**3.1857**	**9607**	**2.15e5**	**3.3274**
4) RTL modified for a 256*256 image	**221718.1**	**3.1961**	**8340**	**2.10e5**	**3.334**

4.2 Results for Area Optimization

The second stage of the synthesis flow was setting up constraints for area optimization. The constraints were setup for maximum area and power and the result was a chip that had larger area and delay. The increase in delay with increase in area can be attributed to the delays that the tool introduces when optimizing for maximum area. The next step in optimization was setting up the constraint for minimum area. This results in an ASIC with reduced area and increased delay. In larger chips there would be redundancy of logic blocks. This means faster operation, hence the smaller delays. The third option in the area optimization was making the area critical along with maximum area and power optimization. The result was similar to the optimization for maximum area and power. This is because in this case optimization is already being done for maximum area and scan circuits were not being introduced into the netlist. The option of making the area or timing critical influences the synthesis only if the scan circuits are to be added to the circuit for the purpose of testability. Hence the optimization resulted in a maximum area chip. In the fourth case the area was made critical along with minimum area constraint. As even in this case scan circuits were not added, so the result was a minimum area chip. The final option in area optimization was flattening along with minimum area constraints. It was expected a reduction in area but the result showed an increase of area. Sum of product representation of the logic network by design analyzer can be attributed to the increase in area and reduction in power. As expected the reduction in the values of the objective functions from the high level synthesis to the physical level synthesis was not substantial. Table 2 gives a summary of the values for area, power and delay obtained from logic level and physical level synthesis for area optimization.

Table 2: Optimization result – Area Optimization

Specification of the constraints	Area (sq. μm)	Power (mw)	Delay (ns)	EPOCH Values	
				Area	Power
5) Area=300000 Power = 10	231313.3	3.3675	10160	2.16e5	3.654
6) Area = 0 Power =10	226757.18	3.3482	10423	2.12e5	3.353
7) Area=300000 Power =10, Area Critical	231313.3	3.3675	10160	2.16e5	3.654
8) Area = 0 Power =10 Area, Timing critical	226757.18	3.3482	10423	2.12e5	3.353
9) Area = 0 Power =10 Area, Timing critical and Flattening	236397.25	3.1395	10179.03	2.18e5	3.278

4.3 Results for Power Optimization

The first constraint under power optimization was minimum power. The tool was setup for minimum power and maximum area. As expected, there was a reduction in power but there was a substantial reduction in area as well. The reason for the reduction can be attributed to the technology library used. Some of the technology libraries are not characterized for power. If the library doesn't have a low power equivalent of the device being optimized then this results in the removal of the device or module from the layout. Hence the substantial reduction in area and power. HP26G was the library used for this research work. The second constraint was making the area critical. As in the previous cases scan circuits were not introduced into the circuit. So the values of area, power and delay are same as the minimum power optimization. The next constraint was flattening along with minimum power. The result had an increase in area and a reduction in power compared to the other results in power optimization. The last constraint in power optimization was minimum area and power. The result of this synthesis was a chip that had the least area and power. Since some of the modules were removed from the layout by the tool during the logic level synthesis, the gate level netlist could not be placed and routed using EPOCH. Table 3 gives the values for area, power and delay obtained from power optimization.

4.4 Results for Delay Optimization

The optimization for speed was performed on area constrained and power constrained chip. The constraints for this optimization were for maximum delay. The constraints for area and speed were setup for the first case. The result was a comparatively smaller chip with a large delay. Timing optimization involved a large set of constraints and care had to be taken to avoid all timing violations.

Table 3: Optimization result – Power Optimization

Specification of the constraints	*Area* (sq. μm)	Power (mw)	Delay (ns)	EPOCH Values	
				Area	Power
10) Area = 300000 Power = 0	162294.5	1.9984	9094	-	-
11) Area = 300000 Power =0 Area & Timing critical	162294.5	1.9984	9094	-	-
12) Area = 300000 Power =0 Area & Timing critical With Flattening	172133.92	1.4905	9133.01	-	-
13) Area = 0 Power =0	161752.84	1.9983	9925	-	-

The second case was delay optimization on a power-constrained chip. The constraints were setup for power and delay optimization and the result was a low power chip with a large delay. This part of the research, which dealt with understanding and setting up the timing constraint, was the time consuming. The values for area, power and delay are tabulated in Table 4.

4.5 Design Space plots

Tables 5 and 6 give values of area and power obtained by logic level synthesis and physical level synthesis along with the ratios of the differences. The ratios found are almost the same, so if a designer wants to estimate the values for area and power of the chip layout after the logic design level, he could just divide those values by the ratio. The values for area, power and delay were plotted using commercial tool, Tecplot. The Fig 4.1 gives a plot of these values in three dimensions.

Table 4: Optimization result – Delay Optimization

Specification of the constraints	Area (sq. μm)	Power (mw)	Delay (ns)	EPOCH Values	
				Area	Power
14)Area = 0 Power = 10 , Delay Values Area Constrained chip	**229142.2**	**3.1786**	**10260**	**2.13e5**	**3.32**
15) Area = 300000 Power = 0 , Delay Values Power Constrained chip	**185832.8**	**1.892**	**10107**	**-**	**-**

Table 5: Values for area obtained through logic level and physical level synthesis, with the ratios of the difference

Specification	Values for area from Design analyzer	Values for area from EPOCH	Ratios
1) Library defaults	222334.7	2.11e5	1.0537
2) Area critical	222334.7	2.11e5	1.0537
3) Flattening	232973.2	2.15e5	1.0836
4) RTL modified for a 256*256 image	221718.1	2.10e5	1.0558
5) Area=300000 Power = 10	231313.3	2.16e5	1.0709
6) Area = 0 Power =10	226757.18	2.12e5	1.0696
7) Area=300000 Power =10 , Area Critical	231313.3	2.16e5	1.0709
8) Area = 0 Power =10 Area, Timing critical	226757.18	2.12e5	1.0696
9) Area = 0 Power =10 Area ,Timing critical and Flattening	236397.25	2.18e5	1.0844
10) Area = 300000 Power = 0	162294.5	-	-
11) Area = 300000 Power =0 Area & Timing critical	162294.5	-	-
12) Area = 300000 Power =0 Area & Timing critical With Flattening	172133.92	-	-
13) Area = 0 Power =0	161752.84	-	-
14) Area = 0,Power = 10, Delay Values Area Constrained chip	229142.2	2.13e5	1.0758
15) Area = 300000 Power = 0, Delay Values Power Constrained chip	185832.8	-	-

Table 6: Values for power obtained through logic level and physical level synthesis, with the ratios of the difference

Specification	Values for Power from Design analyzer	Values for Power from EPOCH	Ratios
1) Library defaults	3.4297	3.372	1.0171
2) Area critical	3.4297	3.372	1.0171
3) Flattening	3.1857	3.3274	0.9574
4) RTL modified for a 256*256 image	3.1961	3.334	0.9750
5) Area=300000 Power = 10	3.3675	3.654	0.9216
6) Area = 0 Power =10	3.3482	3.353	0.9466
7) Area=300000 Power =10 , Area Critical	3.3675	3.654	0.9216
8) Area = 0 Power =10 Area ,Timing critical	3.3482	3.353	0.9466
9) Area = 0 Power =10 Area ,Timing critical and Flattening	3.1395	3.278	0.9577
10) Area = 300000 Power = 0	1.9984	-	-
11) Area = 300000 Power =0 Area & Timing critical	1.9984	-	-
12) Area = 300000 Power =0 Area & Timing critical With Flattening	1.4905	-	-
13) Area = 0 Power =0	1.9983	-	-
14) Area = 0,Power = 10 , Delay Values Area Constrained chip	3.1786	3.32	0.9574
15) Area = 300000 Power = 0 ,Delay Values Power Constrained chip	1.892	-	-

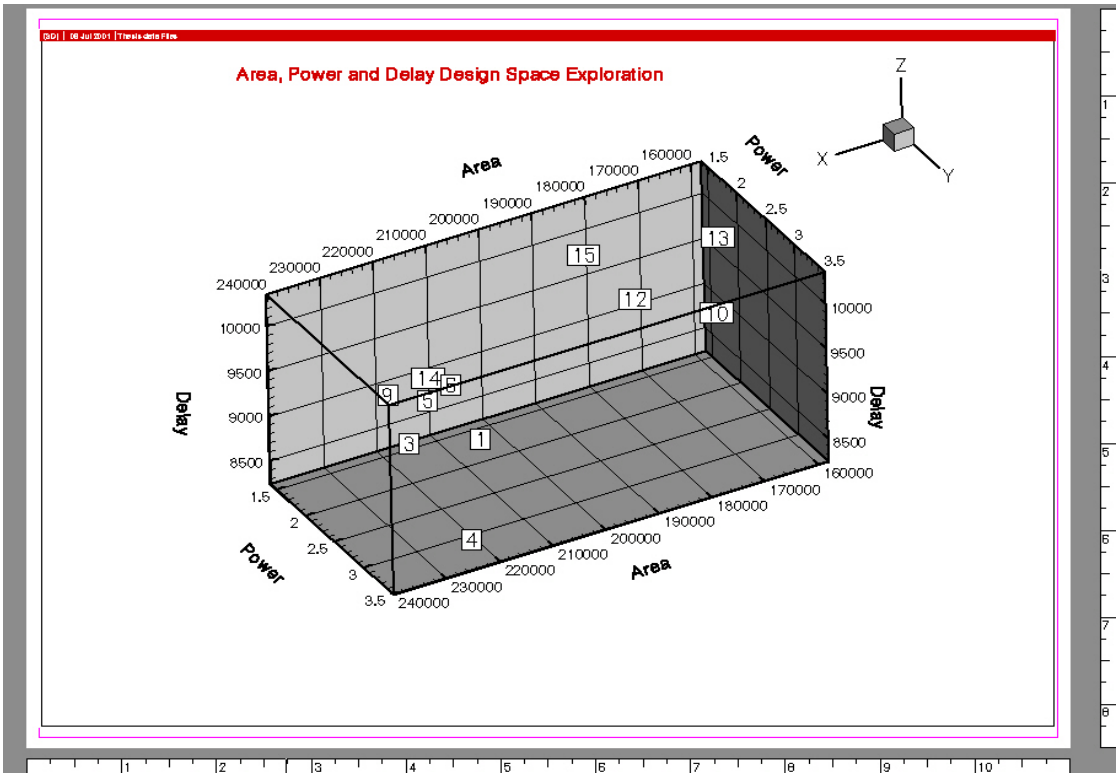


FIGURE 4.1 AREA, POWER AND DELAY PLOT

The numbers one to fifteen shown on the plot indicate the design specification used. The plot shown in Fig 4.2 illustrates the path of the changes in the objective functions. This plot is of interest in if a designer is interested in knowing the sequence of design constraints that needs to be set up to achieve a certain result. The Figures 4.3, 4.4 and 4.5 are specific for delay, power and area. If the designer is interested in one objective function and is willing to trade off the others for improving the function he is interested in, this plot would help the designer in specifying the constraints for altering the objective function. The Fig 4.6 gives the layout of the ASIC after place and route. Overall the results of the synthesis and optimization were close to what was expected, except in the case of flattening where the reason for the increase in area was not initially understood.

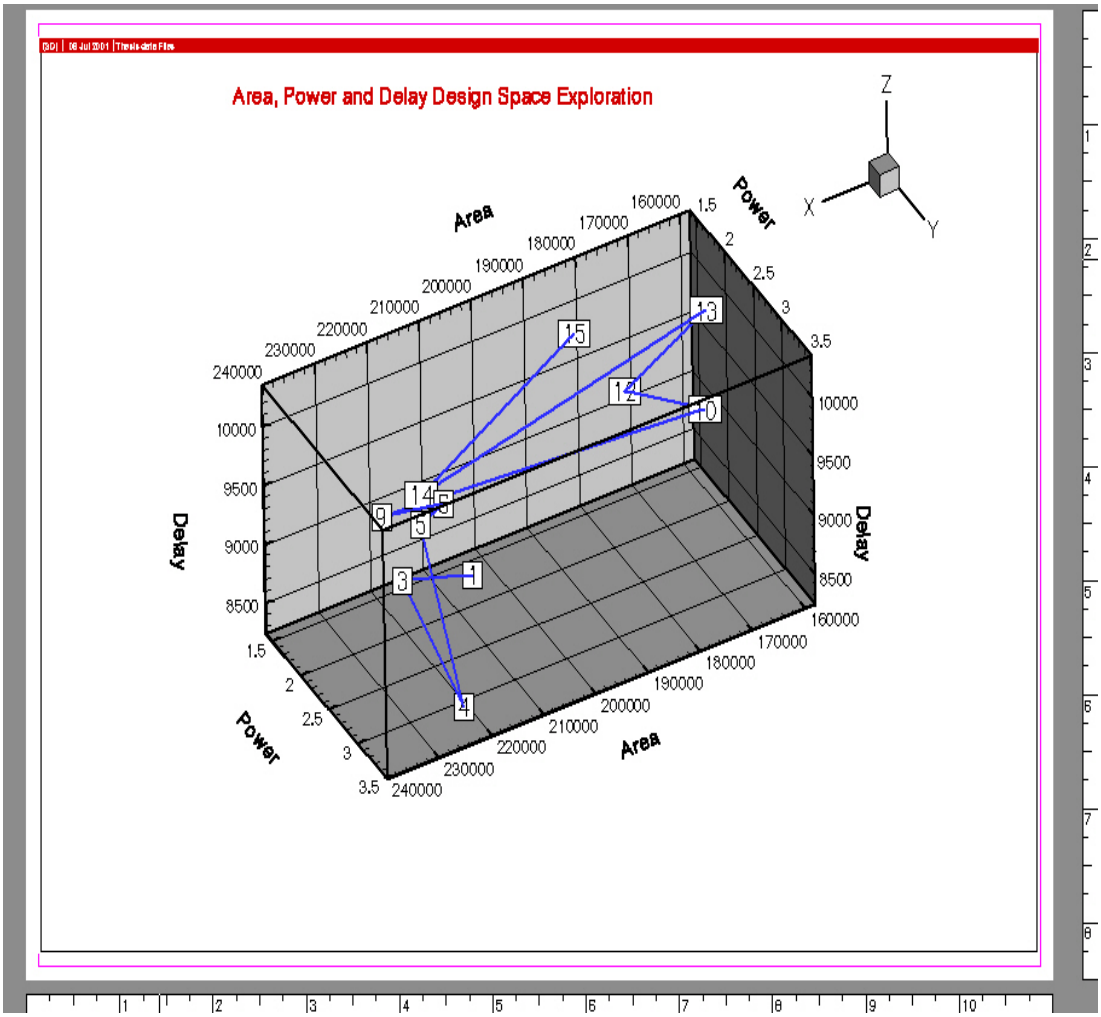


FIGURE 4.2 AREA, POWER AND DELAY PLOT, SHOWING THE PATH OF THE FLOW

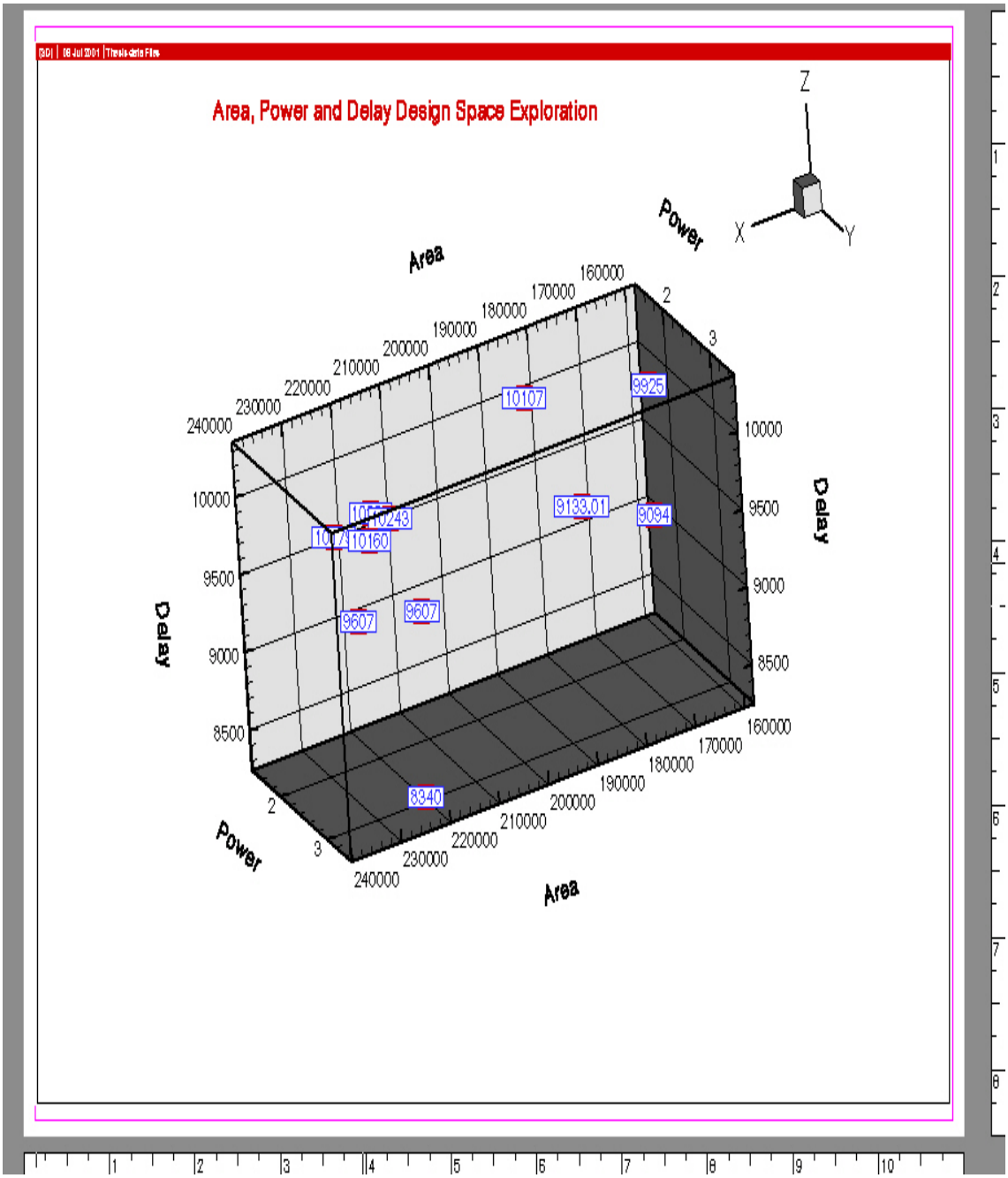


FIGURE 4.3 PLOT ILLUSTRATING THE CHANGE IN DELAY FOR VARIOUS DESIGN CONSTRAINTS

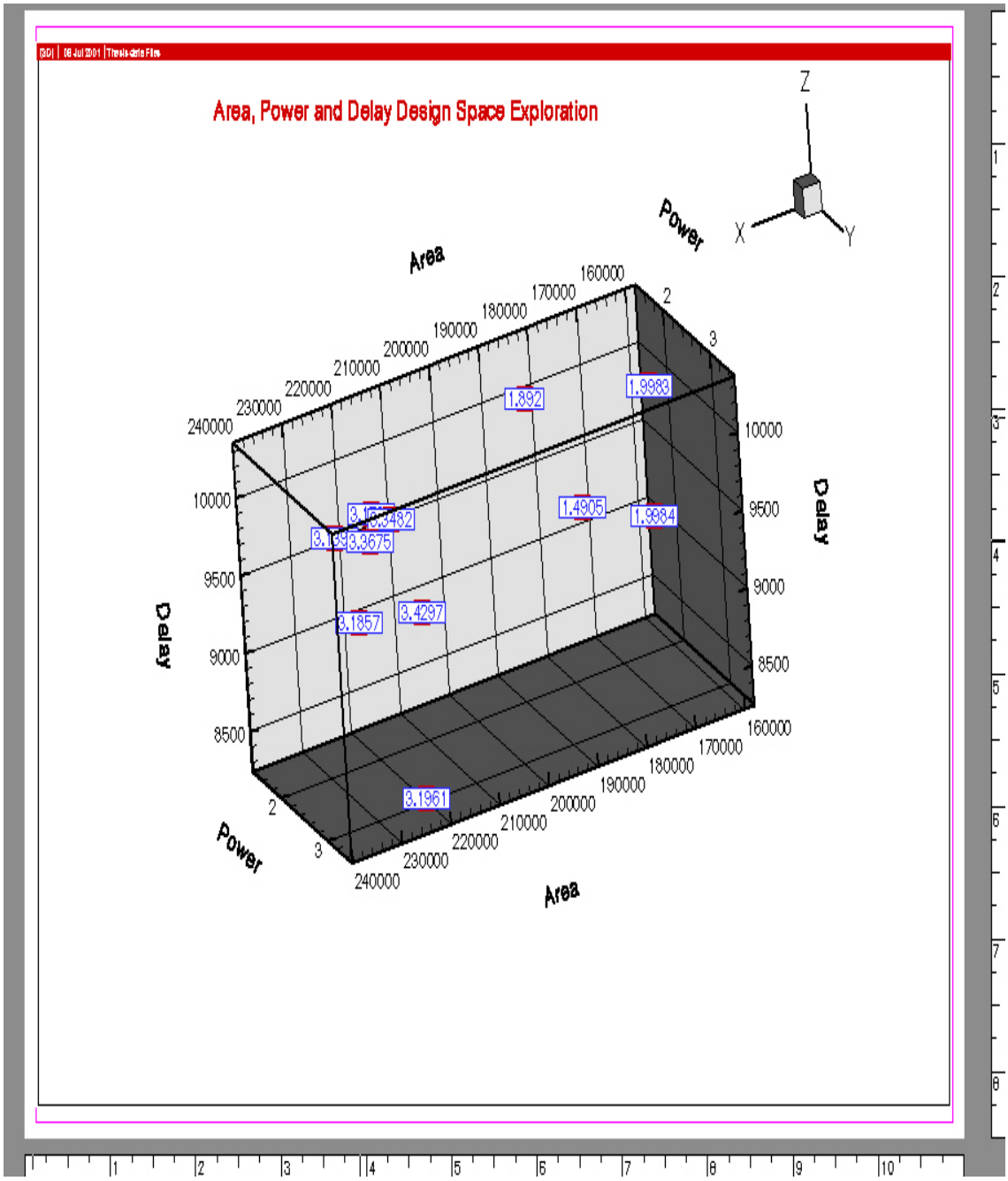


FIGURE 4.4 PLOT ILLUSTRATING THE CHANGE IN POWER FOR VARIOUS DESIGN CONSTRAINTS

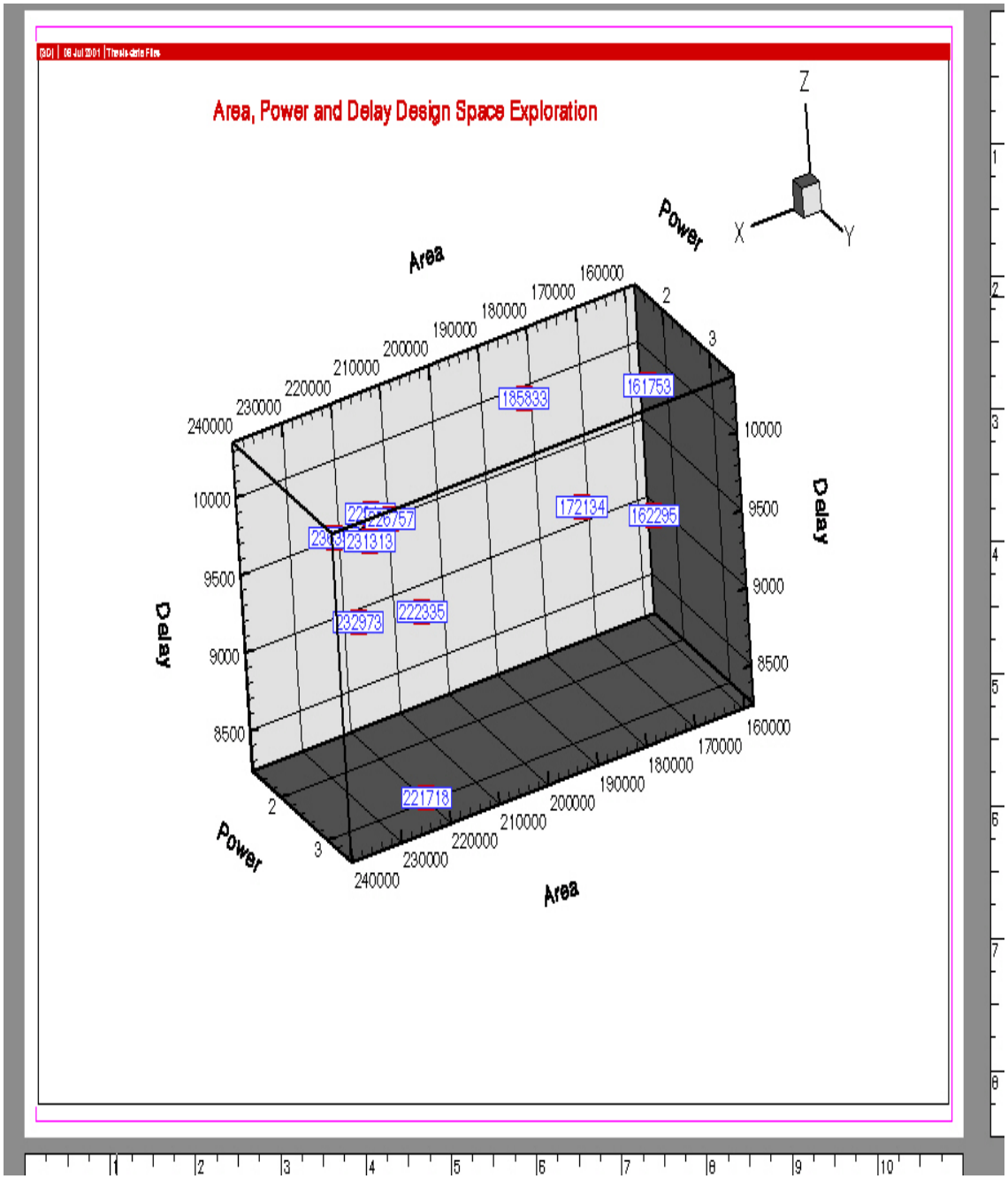


FIGURE 4.5 PLOT ILLUSTRATING THE CHANGE IN AREA FOR VARIOUS CONSTRAINTS

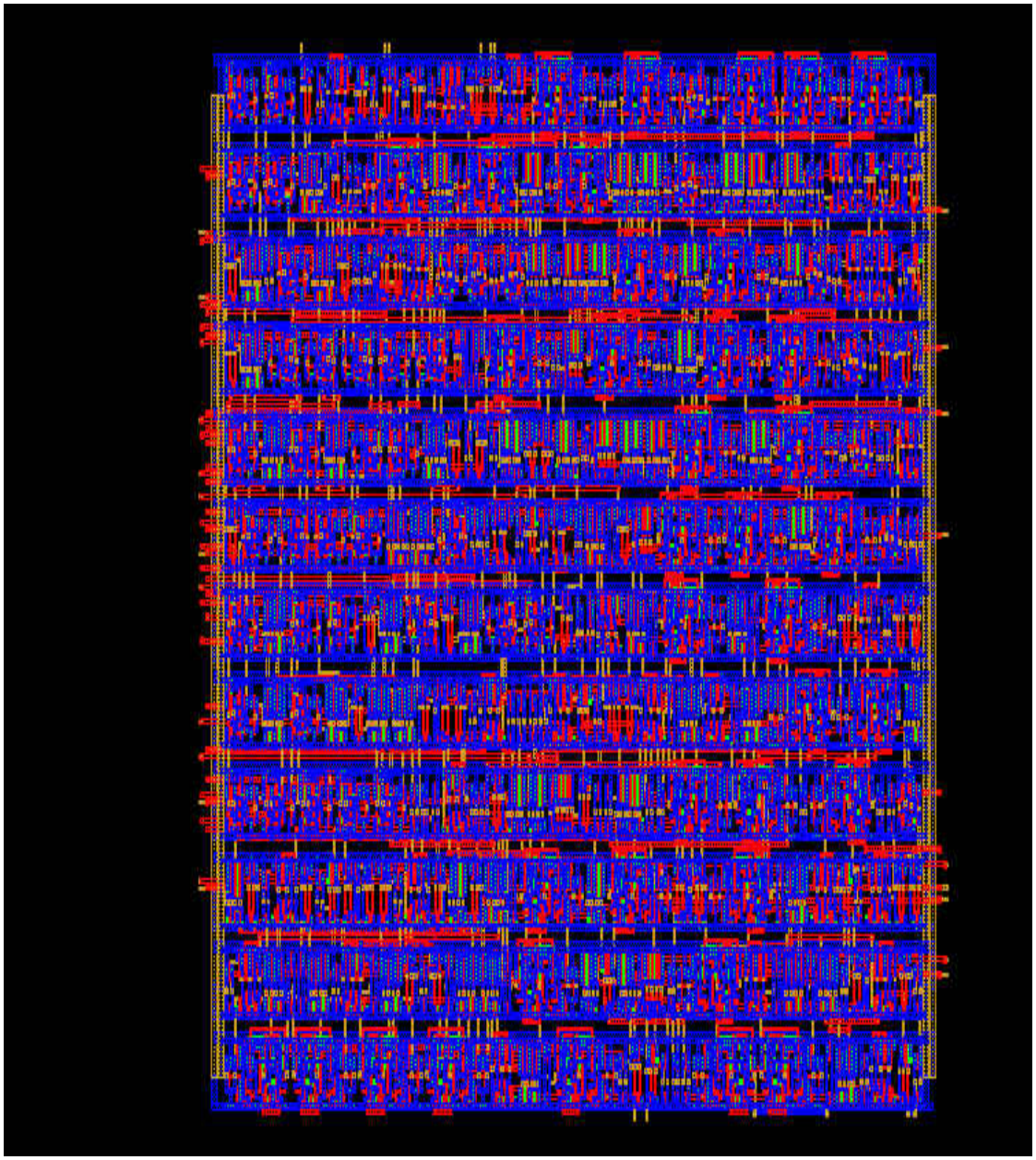


FIGURE 4.6 LAYOUT OF THE ASIC AFTER PLACE AND ROUTE

CHAPTER 5

Summary & Conclusion

This chapter gives a summary of the research followed by the conclusion on techniques for reducing area, power and delay. A brief discussion on future development in this field is also described.

5.1 Summary

In this research work, a 2-D Wavelet transform was implemented using VHDL. The ASICs were functionally tested, synthesized, optimized, placed and routed. The testing of the RTL was performed using Synopsys & Mentor graphics tool sets. The logic design level synthesis and optimization was performed using Synopsys Design analyzer 2000.11 and the physical level synthesis used EPOCH for place & route. The research was performed on a SUN Solaris 5.8 systems. The RTL for this research work was obtained from Honeywell Inc. Few modifications were done to the code such as adding in synopsys libraries, altering the modules etc., this was done to make the code compatible to the synopsys tool during synthesis & optimization.

The functional testing was done using two different methodologies. The first was about comparing the results of the WT on a set of image data implemented using VHDL and C. The root mean square error (RMSE), maximum pixel difference and signal to noise ratio were calculated. The results showed that the qualitative difference between the two types of implementation was small enough to prove the functionality of the RTL.

The second methodology use Mentor graphics tool, FPGA Advantage. The RTL was compiled and simulated for a set of values that were similar to the input of the first technique for functional testing. The values generated by simulation were then compared with the output generated by a commercially available software routine, which performed WT. The error and maximum pixel difference was calculated and functionality tested. The next step in the research was to perform Synthesis and optimization at logic level as well as physical design level. The constraints were understood and setup on the Synopsys tool. Gate level netlist of fourteen ASICs with maximum and minimum values of area, power and delay was the result of this synthesis. The gate level netlist were then placed and routed using a design tool EPOCH. Constraints were also setup at the physical design level. The time for optimization and synthesis varied between ten minutes to almost thirty minutes for the optimization and synthesis to be completed. The result of this research work was fourteen ASICs that were optimized for maximum and minimum values of area, power and delay.

5.2 Conclusions

Many market segments are driving the need for chips that are small, fast and those that consume lesser power. With the increasing use of chips for mobile applications the emphasis is now on power optimization. The optimization of the three axes has to be performed at all design phases for a better result. At the behavioral design level the designer has complete control over the way in which implementation of an ASIC specification is to be done and this includes the type of logic used, number of pins, redundancy of modules etc. When the number of pins in the layout was reduced, the synthesis resulted in a chip with smaller area and power consumption as see in table 1 of chapter 4.

Supply voltage to the circuit also affects the speed of the circuit. Since the propagation delay is dependent on the ratio of supply voltage and threshold voltage. So a low voltage/ low threshold technology would help in reducing the propagation delay.

Using pseudo-nMOS, domino logic, clocked CMOS logic or clocking strategies for some specific applications can help to reduce power dissipation of the circuit and increase the operating speed. Also reducing the device size used in design would result in both the reduction of area and power. The technology library used for the synthesis also plays a role in deciding the extent of optimization on a chip. Some of the libraries do not assist much in power optimization, so we may end up getting chips which loose their functionality during synthesis. So care must be taken on choosing a correct library for synthesis and optimization. The design tool used also influences the optimization. Using power, area and speed conscious tools and techniques at behavioral, logic design and at physical level would greatly improve the end result. Also if speed can be traded for power, then using techniques like partitioning the architecture would decrease the area and the power consumed by the chip. The power consumed by a chip is related to switching frequency of the network. This activity is a function of the input statistics, the network topology and the logic style. These factors also influence the speed and area of a chip since the area, power and speed of a chip is dependent on the number of devices a CMOS circuit is driving, the type of input etc.

If time to market is not a constraint and the size of the layout is very large, then full custom design would be a better option compared to automatic layout. Also reducing the diffusion gap and sidewall capacitance would make the device have minimum area. At the physical design level metal routing, supply lines for the transistors, number of metal layers used affect the size of the chip. New metal routing techniques like over-the cell routing, griddles routing etc. and multi metal layer technologies are used during the place and route for better result during optimization.

The routing area, for design rules with two metal layers, is about 50%. When we use three metal layers, and on using over-the-cell routing, this area could reduce to 25% of the chip area. The present-day technology uses eight metal layer layout design, and this provides a substantial reduction in routing area and hence the chip area.

At the manufacturing stage, use of silicides over the diffusion layer can alter the objective functions. Also by using technologies like advanced interconnect substrates such are multi-chip modules (MCM) and CMOS scaling to sub micron device sizes the chip and package capacitance can be reduced. This approach can be very effective but it is also very expensive and has its own pace of development.

In this research synthesis & optimization has been done predominantly at the behavioral and physical design level. Lately high-level optimization has gained importance with emphasis in logic level optimization because of the ease of optimization. The application for the research reported herein could be found in Design libraries. The user when designing a circuit could specify the range for area, power and delay of the devices that are to be referenced from the libraries. The devices in the library would be optimized for area, power and delay. Corresponding to the range of area, power and delay specified by the user, a device with those attributes could be used for the design.

With the tools becoming more and more efficient and new techniques developed, the scale of optimization in the coming years is going improve, resulting in smaller, faster and low power chips. Also the design flow would get more automated needing lesser assistance from a designer.

REFERENCES

[1] Amara Graps, “An Introduction to Wavelets”, IEEE Computational sciences and Engineering, summer 1995, vol2., No.2.

[2] C.Sidney Burrus, Ramesh A. Gopinath, Haitao Guo, “Introduction to Wavelets and Wavelet Transforms A primer”, Prentice Hall Inc, ISBN 0-13-489600-9

[3] Jan M. Rabaey, “Digital Integrated Design, A design perspective”, Prentice Hall Electronics and VLSI series, ISBN 0-13-1786092-1, 1996

[4] Srinivasa Vemuru, Norman Scheinberg and Edwyn Smith, ”Short-Circuit power dissipation formulae for CMOS gates”, IEEE transactions on circuits and systems, vol 41, no 11, Nov 1994

[5] Neil H.E. Weste, Kamran Eshraghian, “ Principles of CMOS VLSI Design, A systems perspective”, Second edition, Addison Wesley Longman, ISBN 981-235-880-3

[6] Robert Lawrence Lang, “ Parallel VLSI architecture for one, two and three dimensional discrete wavelet transform”, Department of Electrical and Computer Engineering, University of New castle, March 1996

[7] F. Moraes, L.Torres, M.Robert, D.Auvergne, “ Estimation of layout densities for CMOS digital circuits”

[8] J.Kim, S.M. Kang, “ A new triple-layer OTC channel router”, IEEE transactions on CAD, Vol.15, no.9, September 1996,pp 1059-1070

[9] K.Saito and E.Arai, “ Experimental analysis and new modeling of MOS LSI yield associated with the number of elements”, IEEE JSSC, vol. SC-17, no.1, Feb.1982.28-33

[10] R.B. Seeds, “ Yield and cost analysis of bipolar LSI, “ paper 1.1, Proc. IEEE International Electron Devices Meeting, Oct 1967

[11] B.T. Murphy, “ Cost-size optima of monolithic integrated circuits, “ Proc IEEE, vol. 52, Dec 1964, pp.1537-1545

[12] Charles Kooperberg, “ Circuit layout and Yield, “ IEEE JSSC, vol. 23, no.4, Aug.1988, pp.887-892

[13] Renu Mehra, Jan Rabaey, “ Behavioral level power estimation and exploration, “ Department of EECS, University of California at Berkeley

[14] M.Psilogeorgopoulos, M.Munteanu, T.S.Chuang, P.A.Ivey, L.Seed,”Contemporary techniques for lower power circuit design” PREST deliverable

[15] Synopsys user manuals, ver 1999.11

[16] Dr Bouldin’s class lectures.
“http://microsys6.engr.utk.edu/ece/bouldin_courses/551/overview.pdf”

APPENDIX

A1 VHDL Coding for Wavelet Transform

```
/*****************************************************************/
--
--   Copyright (November 1997) Honeywell Inc. Unpublished - All rights
--   reserved. This material may be reproduced by, or for, the U.S.
--   Government pursuant to the copyright license under the clause at
--   DFARS 252.227-7013 (Oct. 1988).
--
--
*****************************************************************/

*****************************************************************/
--   Version 1.1 - 3/21/1998  Original Version
--
*****************************************************************/

--
*****************************************************************/
--  Wavelet code (512x512)
--  Version 1.1
--
--  File         : wavelet512.vhd
--  Company      : Honeywell Technology Center, Minneapolis, MN 55412
--  Date         : March 21, 1998
--  Author       : Saed Wadi
--  Description  :This code reads an image (512x512) from the memory
and performs a wavelet on the image. The code performs a row
operation followed by a column operation for the image (512x512). Then
the image will be cut in half (256x256) and the same row and column
operation will be performed on that image. Then the image will be cut
again in half (128x128), a row AND column operation will be performed
on this image. By the end of the operation, an image of (128x128) will
be a compressed representation of the (512x512) image. This version has
been synthesized, placed, routed and tested and proved to be working.
*****************************************************************/
LIBRARY IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

 ENTITY  wavelet512 IS
     PORT ( PE_Pclk             : IN  std_logic;
          PE_Reset            : IN  std_logic;
            PE_MemBusGrant_n    : IN  std_logic;
            PE_MemData_InReg    : IN  std_logic_vector(15 DOWNTO 0);
            PE_MemData_OutReg   : OUT std_logic_vector(15 DOWNTO 0);
            DONE_wavelet        : OUT std_logic;
            PE_MemWriteSel_n    : OUT std_logic;
            PE_MemStrobe_n      : OUT std_logic;
            PE_MemAddr_OutReg   : OUT std_logic_vector(18 DOWNTO 0));
END wavelet512;
```

```
ARCHITECTURE rtl of wavelet512 IS

SIGNAL RdWrCnt              : Std_logic_Vector(2 DOWNTO 0);
SIGNAL MemStrobe_n          : std_logic;
SIGNAL MemWriteSel_n        : std_logic;
SIGNAL PE_MemStrobe_n1      : std_logic;
SIGNAL PE_MemWriteSel_n1    : std_logic;

SIGNAL rowR       : std_logic_vector(8 DOWNTO 0);
SIGNAL columR     : std_logic_vector(8 DOWNTO 0);
SIGNAL rowW       : std_logic_vector(8 DOWNTO 0);
SIGNAL columW     : std_logic_vector(8 DOWNTO 0);
SIGNAL done       : std_logic;

SIGNAL orit      : std_logic_vector(8 DOWNTO 0);
SIGNAL andit     : std_logic_vector(8 DOWNTO 0);
SIGNAL place     : INTEGER RANGE 0 TO 4;
SIGNAL address   : std_logic_vector(18 DOWNTO 0);

-----------2D stuff----------------------------
signal R0C1R     : std_logic;
signal R0C1R1    : std_logic;
signal R0C1R2    : std_logic;
signal R0C1R3    : std_logic;
signal R0C1R4    : std_logic;
signal R0C1R5    : std_logic;
signal CNTR      : std_logic_vector(8 DOWNTO 0);
signal CNTR1     : std_logic_vector(8 DOWNTO 0);
signal CNTR2     : std_logic_vector(8 DOWNTO 0);
signal CNTR3     : std_logic_vector(8 DOWNTO 0);
signal CNTR4     : std_logic_vector(8 DOWNTO 0);
signal CNTR5     : std_logic_vector(8 DOWNTO 0);
signal cutR      : std_logic;
SIGNAL doneR     : std_logic;


signal R0C1W     : std_logic;
signal R0C1W1    : std_logic;
signal CNTW      : std_logic_vector(8 DOWNTO 0);
signal CNTW1     : std_logic_vector(8 DOWNTO 0);
signal cutW      : std_logic;
SIGNAL doneW     : std_logic;
signal D1        : std_logic_vector(15 DOWNTO 0);
signal D2        : std_logic_vector(15 DOWNTO 0);
signal S1        : std_logic_vector(15 DOWNTO 0);
signal S2        : std_logic_vector(15 DOWNTO 0);
signal S3        : std_logic_vector(15 DOWNTO 0);
signal Snxt      : std_logic_vector(15 DOWNTO 0);
signal Dcrt      : std_logic_vector(15 DOWNTO 0);
signal Scrt      : std_logic_vector(15 DOWNTO 0);
signal Dpre      : std_logic_vector(15 DOWNTO 0);
signal Dpre1     : std_logic_vector(15 DOWNTO 0);
signal Dsnd1     : std_logic_vector(15 DOWNTO 0);
signal Ssnd1     : std_logic_vector(15 DOWNTO 0);
```

```
signal addrs0    : std_logic_vector(18 DOWNTO 0);
signal addrs1    : std_logic_vector(18 DOWNTO 0);
signal addrs2    : std_logic_vector(18 DOWNTO 0);
signal addrs3    : std_logic_vector(18 DOWNTO 0);
signal addrs4    : std_logic_vector(18 DOWNTO 0);

BEGIN
--------------------------------------------------------------------
--      000 ==> WRITE
--      001 ===========> Strobe = 1
--      010 ==> READ
--      011 ==> READ
--      100 ===========> Strobe = 1
--      101 ==> WRITE
--
--    Do 2 Reads and 2 Writes separated by a strobe = 1
--
--
--------------------------------------------------------------------
RdWrGen : PROCESS(PE_Pclk, PE_Reset)

  BEGIN
    IF (PE_Reset = '0') THEN
      RdWrCnt <= "001";
    ELSIF (PE_Pclk = '1' AND PE_Pclk'event) THEN
      IF ( PE_MemBusGrant_n = '0' ) THEN         -- Get the grant,
                                                 -- then start counting
        IF ( RdWrCnt = "101" ) THEN
          RdWrCnt <= "000" ;
        ELSE
          RdWrCnt <= RdWrCnt + '1';
        END IF;
      END IF;
     END IF;

END PROCESS;
--------------------------------------------------------------------
---
-- Generate strobe=1, when RdWrCnt= 1 0r 4, Or when done=1
-- When the strob is high, reading or writing to MEM is not allowed
--------------------------------------------------------------------
strobeGen : PROCESS(RdWrCnt, done)
  BEGIN
       IF ( done = '0' ) THEN
          IF ( RdWrCnt = "001" OR RdWrCnt = "100" ) THEN
            MemStrobe_n <= '1';
          ELSE
            MemStrobe_n <= '0';
          END IF;
       ELSE
          MemStrobe_n <= '1';
       END IF;

END PROCESS;
```

```
-------------------------------------------------------------------
-- Generate the Reading and Writing addresses
-------------------------------------------------------------------
COUNTER_RW  :PROCESS (PE_Pclk, PE_Reset)

  BEGIN
    IF (PE_Reset = '0') THEN             --  PE_Reset the READ/WRITE
COUNTER
      rowR    <= "000000000";
      columR  <= "000000000";
      R0C1R   <= '0';                    --  R0C1R<='0'  ROW READ
                                         --  R0C1R<='1'  COLUMN READ
      CNTR    <= "111111111";            --  Count the read to 512
      cutR    <= '0';                    --  Control halfing the
                                         --  image during reading
      doneR   <= '0';                    --  Stop reading from MEM
                                         --  when   it's '1'
      done    <= '0';                    --  Finish the wavelet
      rowW    <= "000000000";
      columW  <= "000000000";
      R0C1W   <= '0';                    --  R0C1W<='0'  ROW WRITE
                                         --  R0C1W<='1'  COLUMN WRITE
      CNTW    <= "111111111";            --  Count the write to 512
      cutW    <= '0';                    --  Control halfing the
                                         --  image during writing
      doneW   <= '1';                    --  Stop writing to  MEM
                                         --  when   it's '1'
      place   <=  0;
      orit    <= "100000000";
      andit   <= "011111111";

    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN
      IF ( MemStrobe_n = '0'  ) THEN
        IF ( done = '0' ) THEN
         IF ( RdWrCnt(1) = '1' ) THEN            --  READING COUNTER

                IF ( doneR = '0' ) THEN
                    IF ( R0C1R = '0' ) THEN      --  Row READ wavelet
                        columR <= columR  + '1';
                        IF ( columR = "000000011" ) THEN
                          doneW <= '0';          -- Start writing
                        ELSIF ( columR = CNTR ) THEN
                          rowR   <= rowR + '1';
                          columR <= "000000000";
                        END IF;
                    ELSE                         -- colum READ wavelet
                        rowR <= rowR + '1';
                        IF ( rowR = "000000011" ) THEN
                          doneW <= '0';          --  Start Writing
                        ELSIF ( rowR = CNTR ) THEN
                          columR <= columR + '1';
                          rowR   <= "000000000";
                        END IF;
                    END IF;
```

```
                                    --  Last pixel value
            IF ( rowR = CNTR  AND  columR = CNTR  ) THEN
                R0C1R  <= NOT R0C1R;
                cutR   <= NOT cutR;
                doneR  <= '1';                --  Stop Reading
                rowR   <= "000000000";   --  Reset the address
                columR <= "000000000";
                IF ( cutR = '1' ) THEN        --  Divide by 2
                  CNTR  <= '0' & CNTR(8 DOWNTO 1);

                  IF  ( CNTR(8 DOWNTO 7)= "00" ) THEN
                    CNTR   <= "111111111";
                  END IF;
                END IF;
            END IF;
          END IF;


ELSE                                        -- WRITING COUNTER
  IF ( doneW = '0' ) THEN
    IF ( R0C1W = '0' ) THEN             --  Row write wavelet
      IF ( columW = CNTW ) THEN
        rowW   <= rowW + '1';
        columW <= "000000000";
      ELSIF (columW(8- place) = '1') THEN
        columW <= (andit AND columW(8 DOWNTO 0)) + '1';
      ELSE
        columW <= (orit OR columW(8 DOWNTO 0));

      END IF;
    ELSE                              --  Colum write wavelet

      IF ( RowW = CNTW ) THEN
        columW <= columW + '1';
        rowW   <= "000000000";
      ELSIF ( rowW(8 - place) = '1' ) THEN
        rowW <= (andit AND rowW(8 DOWNTO 0))+ '1';
      ELSE
        rowW <= (orit OR rowW(8 DOWNTO 0));
      END IF;
    END IF;
                                       --  Last pixel value
    IF ( rowW = CNTW  AND  columW = CNTW  ) THEN
      R0C1W  <= NOT R0C1W;
      cutW   <= NOT cutW;
      doneR  <= '0';                     -- Start reading again
      doneW  <= '1';                           -- Stop writing
      rowW   <= "000000000";
      columW <= "000000000";
      IF ( cutW = '1' ) THEN
        CNTW  <= '0' & CNTW(8 DOWNTO 1);
        place <= place + 1;

        andit <= '0' & andit(8 DOWNTO 1);
```

```
            orit  <= '0' & orit(8 DOWNTO 1);
                  IF  ( CNTW(8 DOWNTO 7)= "00" ) THEN
                    done   <= '1';                  -- Finish the wavelet
                    CNTW   <= "111111111";
                    place  <=  0;
                    orit   <= "100000000";
                    andit  <= "011111111";
                    doneR  <= '1';
                  END IF;
                END IF;
              END IF;
         END IF;
        END IF;
      END IF;
     END IF;
   END IF;
END PROCESS;

-------------------------------------------------------------------------
---
-- Mux out the address. The select signal is the RdWrCnt(1), which
-- indicate reading or writing.
-------------------------------------------------------------------------
---
addrssmux :PROCESS ( R0C1W, R0C1R, rowW, rowR, columW, columR,
RdWrCnt(1)  )
  BEGIN
    IF ( RdWrCnt(1) = '1' ) THEN
      --  Row Read from MEM1, address MSB=0
      --  Column Read from MEM2, adress MSB=1
      address <=  R0C1R & rowR & columR ;
    ELSE
      --  Row Write to MEM2, address MSB =1
      --  Column Write to MEM1, address MSB=0
      address <= ( NOT R0C1W) & rowW & columW ;
    END IF;

END PROCESS;


-----------------------------------------------------------------------
-- Dmux the input data. Using the falling edge of the clk to latch the
-- data at the right time
-----------------------------------------------------------------------

DMUX16 :PROCESS (PE_Reset, PE_Pclk )
    BEGIN
    IF (PE_Reset = '0') THEN
       S1     <= "0000000000000000";
       Snxt   <= "0000000000000000";
       D1     <= "0000000000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '0') THEN
      IF ( done = '0' ) THEN
```

```
      IF ( RdWrCnt(2) = '1' ) THEN          -- Data ready after 2 clk
             IF ( RdWrCnt(0) = '0' ) THEN
               S1     <= PE_MemData_InReg(15 DOWNTO 0);
               Snxt   <= PE_MemData_InReg(15 DOWNTO 0);
             ELSE
               D1     <= PE_MemData_InReg(15 DOWNTO 0);
             END IF;
         END IF;
      END IF;
     END IF;
END PROCESS;


-------------------------------------------------------------------------
---
--  Delay the reading address associated with the wavelet part by a
clk.
-------------------------------------------------------------------------
---
Delayaddress  :PROCESS (PE_Pclk, PE_Reset)
  BEGIN
    IF (PE_Reset = '0') THEN
      addrs0 <= "0000000000000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN
      addrs0 <= R0C1R & rowR & columR;
    END IF;

END PROCESS;

-------------------------------------------------------------------------
-- Use 5 pipeline stages that controlled by a counter signal instead
-- of using 8 pipeline stages without control signal. This will
--- Pipeline
-- data, address and control signals until the next data is ready to
-- be read ( Snxt) before the wavelet part start doing the computation.
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-- Pipeline STAGE 1
-------------------------------------------------------------------------
 latchem :PROCESS (PE_Pclk, PE_Reset)
  BEGIN
    IF (PE_Reset = '0') THEN
      addrs1 <= "0000000000000000000";
      R0C1R1 <= '0';
      CNTR1  <= "000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN  -- Rising edge
      IF ( RdWrCnt(2) = '1') THEN            --  data in valide after 2
clk
        CNTR1  <= CNTR;
        R0C1R1 <= R0C1R;
        addrs1 <= addrs0;
      END IF;
    END IF;
END PROCESS;
```

```
-------------------------------------------------------------------
-- Pipeline STAGE 2
-------------------------------------------------------------------

Stage11 :PROCESS (PE_Pclk, PE_Reset)

  BEGIN
    IF (PE_Reset = '0') THEN
      S2     <= "0000000000000000";
        addrs2 <= "00000000000000000000";
        R0C1R2 <= '0';
        CNTR2  <= "000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN  -- Rising edge
      IF ( RdWrCnt(1 DOWNTO 0) = "00" ) THEN
        S2     <= S1;
        addrs2 <= addrs1;
        R0C1R2 <= R0C1R1;
        CNTR2  <= CNTR1;
      END IF;
    END IF;
  END PROCESS;

-------------------------------------------------------------------
-- Pipeline STAGE 3
-------------------------------------------------------------------

  Stage2 :PROCESS (PE_Pclk, PE_Reset)
  BEGIN
    IF (PE_Reset = '0') THEN
      D2   <= "0000000000000000";
      S3   <= "0000000000000000";
      addrs3 <= "00000000000000000000";
      R0C1R3 <= '0';
      CNTR3  <= "000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN -- Rising edge
     IF ( RdWrCnt(1 DOWNTO 0) = "00" ) THEN
      D2   <= D1;
      S3   <= S2;
      addrs3 <= addrs2;
      R0C1R3 <= R0C1R2;
      CNTR3  <= CNTR2;
     END IF;
    END IF;
  END PROCESS;

-------------------------------------------------------------------
-- Pipeline STAGE 4
-------------------------------------------------------------------

  Stage22 :PROCESS (PE_Pclk, PE_Reset)
  BEGIN
    IF (PE_Reset = '0') THEN
      Dcrt   <= "0000000000000000";
      Scrt   <= "0000000000000000";
```

```
      R0C1R4 <= '0';
      CNTR4  <= "000000000";
    ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN -- Rising edge
     IF ( RdWrCnt(1 DOWNTO 0) = "11") THEN
      Dcrt   <= D2;
      Scrt   <= S3;
      R0C1R4 <= R0C1R3;
      CNTR4  <= CNTR3;
     END IF;
    END IF;
  END PROCESS;


-----------------------------------------------------------------------
-- Pipeline STAGE 5
-----------------------------------------------------------------------

Stage5 :PROCESS (PE_Pclk, PE_Reset)

BEGIN
  IF (PE_Reset = '0') THEN
    Dpre   <= "0000000000000000";
    R0C1R5 <= '0';
    CNTR5  <= "000000000";
    addrs4 <= "0000000000000000000";
  ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN     -- Rising edge
    IF ( RdWrCnt(1) = '1' ) THEN
      Dpre   <= Dpre1;
      R0C1R5 <= R0C1R4;
      CNTR5  <= CNTR4;
      addrs4 <= addrs3;
    END IF;
  END IF;
END PROCESS;

-----------------------------------------------------------------------
-----------------------Wavelet code------------------------------------
-- It needs Scrt, Dcrt, Snxt, Dpre and addrs4 to perform the
-- computation.
-----------------------------------------------------------------------


Stage3   :PROCESS (PE_Pclk, PE_Reset)
 variable Dout      : std_logic_vector(15 DOWNTO 0);
 variable Dout1     : std_logic_vector(15 DOWNTO 0);

BEGIN

IF (PE_Reset = '0') THEN
    Dsnd1 <= "0000000000000000";
    Ssnd1 <= "0000000000000000";
    Dpre1 <= "0000000000000000";
ELSIF (PE_Pclk'event and PE_Pclk = '1') THEN  -- Rising edge
```

```
 --  First pixel evaluation

     IF (((addrs4(8 DOWNTO 0) = "000000000") AND R0C1R5= '0')  OR
         ((addrs4(17 DOWNTO 9) = "000000000") AND R0C1R5= '1')) THEN
       Dout := (Dcrt(14 DOWNTO 0) & '0') - (Scrt(15 DOWNTO 0) +
Snxt(15 DOWNTO 0));
          IF ( Dout(15) = '0') THEN
             Ssnd1 <= Scrt(15 DOWNTO 0) + ( "00" & Dout(15 DOWNTO 2));
            ELSE
               Ssnd1 <= Scrt(15 DOWNTO 0) + ( "11" & Dout(15 DOWNTO 2));
          END IF;
             Dsnd1 <= Dout;
               Dpre1 <= Dout;

 --  Last Pixel evaluation

     ELSIF (((addrs4(8 DOWNTO 0) = (CNTR5-1)) AND R0C1R5= '0') OR
            ((addrs4(17 DOWNTO 9) = (CNTR5-1)) AND R0C1R5= '1')) THEN
          Dout  := (Dcrt(14 DOWNTO 0) & '0') -(Scrt(14 DOWNTO 0) & '0');
            Dout1 := Dpre(15 DOWNTO 0) + Dout(15 DOWNTO 0);
          IF ( Dout1(15) = '0') THEN
             Ssnd1 <= Scrt(15 DOWNTO 0) + ( "000" & Dout1(15 DOWNTO 3));
          ELSE
             Ssnd1 <= Scrt(15 DOWNTO 0) + ( "111" & Dout1(15 DOWNTO 3));
          END IF;
             Dsnd1 <= Dout;

 --  All the pixels in between

     ELSE
        Dout  := (Dcrt(14 DOWNTO 0) & '0') - (Scrt(15 DOWNTO 0) +
Snxt(15 DOWNTO 0));
          Dout1 := Dpre(15 DOWNTO 0) + Dout(15 DOWNTO 0);
            IF ( Dout1(15) = '0') THEN
             Ssnd1 <= Scrt(15 DOWNTO 0) + ( "000" & Dout1(15 DOWNTO 3));
          ELSE
             Ssnd1 <= Scrt(15 DOWNTO 0) + ( "111" & Dout1(15 DOWNTO 3));
          END IF;
               Dsnd1 <= Dout;
               Dpre1 <= Dout;

     END IF;
END IF;
END PROCESS;
```

```
-------------------------------------------------------------------
---------------------- Writing to Memory--------------------------
-------------------------------------------------------------------

MUXOUT :PROCESS (doneW, MemStrobe_n, RdWrCnt, Ssnd1, Dsnd1)

  BEGIN

  IF ( doneW = '0' ) THEN
   IF ( MemStrobe_n = '0' ) THEN
    IF (RdWrCnt(1) = '0' ) THEN
      IF ( RdWrCnt(0) = '1') THEN
        PE_MemData_OutReg <= Ssnd1;
      ELSE
        PE_MemData_OutReg <= Dsnd1;
      END IF;
    END IF;
   END IF;
  END IF;

END PROCESS;

-------------------------------------------------------------------------
---
-- Assigning signals to the output signal names
-------------------------------------------------------------------------
---

outsig :PROCESS(RdWrCnt(1), MemStrobe_n , address, done)
  BEGIN

    PE_MemStrobe_n    <= MemStrobe_n;
    PE_MemAddr_OutReg <= address;
    PE_MemWriteSel_n  <= RdWrCnt(1);
    DONE_wavelet      <= done;       -- Finish 3-stages of wavelet

END PROCESS;

END rtl;

CONFIGURATION config_wavelet512 OF wavelet512 IS
   FOR rtl
   END FOR;
END config_wavelet512;
```

Vita

I was born in Madras, India and did most of my schooling in the same city. I completed my undergraduate studies in engineering, specializing in Electronics and Communication. In the fall of 99, I enrolled in the University of Tennessee to pursue my Master of Science degree in Electrical Engineering specializing in the design of Application-Specific Integrated Circuits.