MASTER'S THESIS

Matrix Inversion Using QR Decomposition by Parabolic Synthesis

By

Nafiz Ahmed Chisty

Supervisors:

Professor Peter Nilsson and MScEE Erik Hertz

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Abstract

Parabolic synthesis is one of the latest methodologies, proposed by Professor Peter Nilsson and Erik Hertz of EIT department at Lund University (LTH), for the implementation of unary functions in hardware. In the preceding research conducted at Lund University, it had been shown that parabolic synthesis is an effective solution, which is both fast and consumes less area compared to all the existing methods.

The goal of this Master's thesis is to develop hardware for the generation of three trigonometric functions (+sine, -sine and +cosine) using the novel approximation methodology, which is based on Parabolic synthesis, for the use in Givens rotations for implementing matrix inversion using QR decomposition. Two hardware designs has been developed, one for the parabolic synthesis and another for matrix inversion but due to time limitations the two hardware designs could not be integrated together. The paper mainly focuses on the implementation of the three trigonometric functions on both FPGA and ASIC and compares the result with metrics of speed and area and later a hardware solution for the overall system has been proposed.

Acknowledgements

The greatest gratitude after GOD goes to Professor Peter Nilsson, my Master's Thesis supervisor and Examiner, who not only helped me with kind insightful advice and encouragements but also made it possible for me to finish the thesis successfully by going beyond my expectations in assisting me with various academic and administrative issues.

I would also like to thank my other supervisor, MScEE Erik Hertz for his kind support and guidance.

Last but not least, I would like to thank my beloved family and all friends for their constant moral support and courage.

Nafiz Ahmed Chisty
Master's in System-on-Chip
Lund University (LTH)
Lund, Sweden
January 2012

Contents

CHAPTER 1

1 Introduction

The demand of fast and small hardware architectures are increasing day by day. Most hardware uses Unary functions like trigonometric functions, logarithms as well as square root and division functions. These functions are extensively used in applications like robotics, signal processing, communication systems, navigation, fluid physics, etc. The overall performance of the system is dependent on the methods of computing such functions. In many cases, software solutions are not sufficient and a hardware implementation is required [1].

For low precision computations, the simplest and faster method of implementations of such functions is by using Single look-up table. However, for High-precision computations this method gets inappropriate due to large table size and long execution time. Implementation using polynomial approximation also has large computational complexities and delays due to extensive use of multiplications and divisions [1].

The Coordinate Rotation Digital Computer (CORDIC) algorithm is a popular method for the fast computation of unary functions using only simple shift-add operation. Although this method is faster than a software solution but due to its iterative property it is slow and thus improper for high speed applications [1].

On the other hand Parabolic Synthesis, a methodology proposed by Professor Peter Nilsson and Erik hertz, is a method based on developing functions that performs approximation of original unary functions in hardware. This method uses parallelism to reduce execution time and employs low complexity operations thus making hardware implementation faster and simpler than all other existing methodologies [1].

This thesis mainly develops hardware for the generation of three trigonometric functions (+sine, -sine and +cosine) using parabolic synthesis. Later, hardware architecture is proposed for implementing matrix inversion using QR decomposition.

This paper consists of eight Chapters:

Chapter 1 deals with the motivation behind this thesis work.

Chapter 2 explains the basic of matrix inversion with details of QR decomposition using Given's rotations.

Chapter 3 introduces the novel Parabolic synthesis methodology.

Chapter 4 explains the development of architecture for the generation of trigonometric functions using parabolic synthesis. A hardware solution is also shown for implementing matrix inversion with QR decomposition using the implemented parabolic architecture.

Chapter 5 deals the Synthesis procedure and discusses the results obtained from FPGA and ASIC synthesis.

Chapter 6 discusses the result obtained from the thesis work.

Chapter 7 concludes the thesis work.

Chapter 8 suggests some future prospects of the implemented design with improvement.

CHAPTER 2

2 Matrix Inversion by QR decomposition using Givens rotation

2.1 Matrix inversion

In linear algebra, for an *n*-by-*n* square matrix A, matrix inversion is the process of finding the matrix B if (2.1) is satisfied

$$AB = BA = I_n \tag{2.1}$$

where I_n denotes the *n*-by-*n* identity matrix. The inverse of A is denoted by A^{-1}.

Non-square matrices (*m*-by-*n* matrices for which $m \neq n$) do not have an inverse but may have a left inverse or right inverse. A square matrix that is not invertible is called singular [3][4].

2.1.1 Properties of inverse matrix

Some of the most important properties for an invertible matrix A are:

$$(A^{-1})^{-1} = A \tag{2.2}$$

$$(kA)^{-1} = k^{-1} A^{-1} \quad \text{for nonzero scalar } k \tag{2.3}$$

$$(A^T)^{-1} = (A^{-1})^T \tag{2.4}$$

$$\det(A^{-1}) = \det(A)^{-1} \tag{2.5}$$

2.2 QR decomposition

QR decomposition is an efficient frequently used methodology when matrix inversion is needed. A typical application area is mobile communication systems using multiple antennas, i.e. Multi-Input Multi-Output (MIMO) systems. The QR decomposition factorizes a matrix into an orthogonal and an upper triangular matrix.

$$A = Q \times R \tag{2.6}$$

where *R* is an upper triangular matrix and *Q* is orthogonal, that is, the unity matrix is

$$I = Q \times Q^T \tag{2.7}$$

where Q^T is the transpose of *Q*, for real valued matrices.

MIMO systems often uses 4 transmit and 4 receive antennas. The inverse of a 4 by 4 matrix at the receiver side is therefore often practiced. We thus get a system like:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2.8}$$

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \tag{2.9}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix} \tag{2.10}$$

where R is an upper triangular matrix.

QR decomposition can be computed using several methods like the Gram-Schmidt process, Householder transformations, or Givens rotations. Each has a number of advantages and disadvantages. In this thesis, we will use Givens rotation method for computing QR decomposition since it can be parallelized and have a lower operation count [5].

2.3 Givens Rotations

Givens rotation is a rotation in the plane spanned by two coordinates axes, which is represented by a matrix of the form

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{2.11}$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. [6]

2.4 QRD Using Givens rotations

Givens rotations can be used to perform QR decomposition. The process utilizes a number of cycles of rotations whose function is to null an element in the sub-diagonal of the matrix, forming the *R* matrix as shown in (2.10). The orthogonal *Q* matrix, as shown in (2.9), can be obtained by the concatenation of all the Givens rotations [6].

2.4.1 Triangularization for QRD

A 3 by 3 input matrix, A_1, is given in (2.12)

$$A_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.12}$$

From A_1, we will determine the matrices A_2 and A_3, as well. In order to find a triangular matrix, *R*, three rotations are needed, where one element is set to zero after each rotation. It can for instance be done in the order (3, 1), (2, 1), and (3, 2). For that, three Givens rotation matrices are needed, G_1, G_2, and G_3, as defined below.

$$G_1 = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix} \tag{2.13}$$

$$G_2 = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$G_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix} \tag{2.15}$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. These values can be calculated as:

$$\begin{aligned} c_1 &= \frac{A_1(1,1)}{\sqrt{A_1(1,1)^2 + A_1(3,1)^2}} \\ s_1 &= \frac{A_1(3,1)}{\sqrt{A_1(1,1)^2 + A_1(3,1)^2}} \end{aligned} \tag{2.16}$$

$$c_2 = \frac{A_1(1,1)}{\sqrt{A_1(1,1)^2 + A_1(2,1)^2}}$$
$$s_2 = \frac{A_1(2,1)}{\sqrt{A_1(1,1)^2 + A_1(2,1)^2}} \tag{2.17}$$

$$c_3 = \frac{A_1(2,2)}{\sqrt{A_1(2,2)^2 + A_1(3,2)^2}}$$
$$s_3 = \frac{A_1(3,2)}{\sqrt{A_1(2,2)^2 + A_1(3,2)^2}} \tag{2.18}$$

However, these operations include square-root, square, and division, which is not feasible for hardware implementation.

In (2.19), the matrices A_2 and A_3 are determined.

$$A_2 = G_1 \times A_1$$
$$A_3 = G_2 \times A_2 \tag{2.19}$$

Finally, the Q and R matrices can be determined, as shown in (2.20).

$$Q = G_1^T \times G_2^T \times G_3^T$$
$$R = G_3 \times A_3 \tag{2.20}$$

2.4.2 The inverse matrix for QRD

In (2.21) the inverse of A is derived. For that, the inverse of R is needed, which is a straight forward operation since R is upper triangular. The transpose of Q is basically done with memory operations.

$$A = Q \times R$$
$$A^{-1} = (Q \times R)^{-1}$$
$$A^{-1} = Q^{-1} \times R^{-1}$$
$$A^{-1} = R^{-1} \times Q^T \tag{2.21}$$

2.5 Hardware for inverse matrix for QRD

Using formula (2.12) - (2.21), the basic hardware for obtaining the inverse matrix is demonstrated in Fig. 2.1.

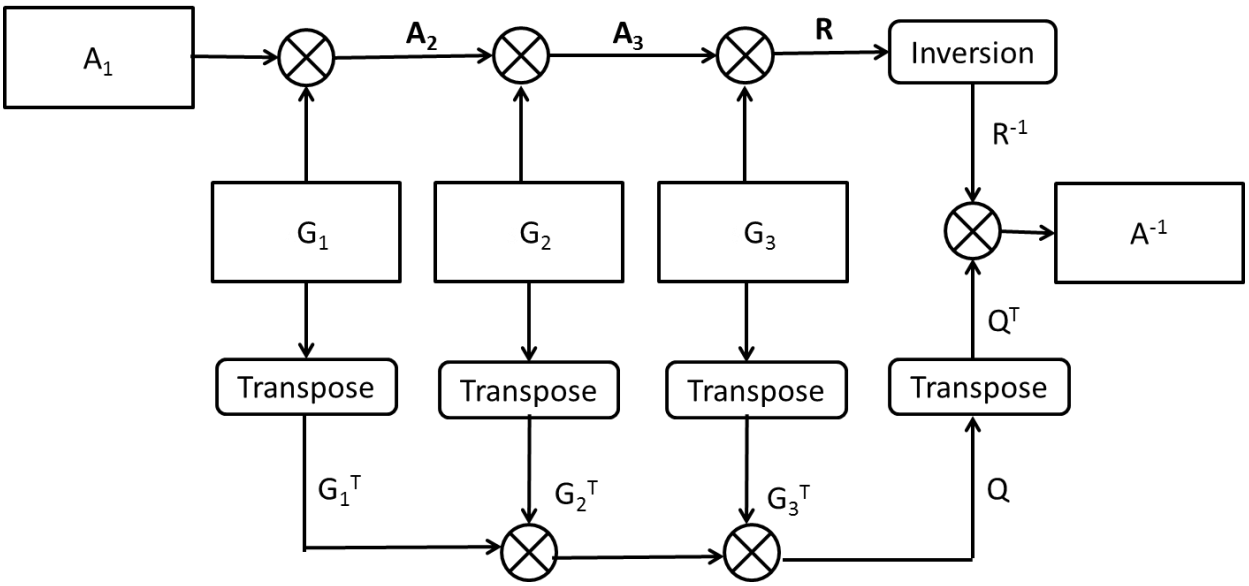

Figure 2.1. Basic hardware for obtaining matrix inversion using QRD.

CHAPTER 3

3 Parabolic Synthesis Methodology

3.1 Introduction

Parabolic Synthesis is a method based on developing functions that performs approximation of original unary functions in hardware. This method uses parallelism to reduce execution time and employs low complexity operations like shifts, additions, and multiplications that are simple to implement in hardware, thus making hardware implementation faster and simpler than all other existing methodologies [1].

3.2 Other Hardware Approximation Methods

The demand of hardware approximation for the implementation of elementary functions is increasing with the passage of time. The goal is to make the implemented hardware fast at the same time limiting the area consumption to a minimum level.

3.2.1 Advantage of Hardware over Software

Most hardware uses Unary functions like trigonometric functions, logarithms as well as square root and division functions. These functions are extensively used in applications like robotics, signal processing, communication systems, navigation, fluid physics, etc. The overall performance of the system is dependent on the methods of computing such functions. In many cases, software solutions are not sufficient and a hardware implementation is required [1].

Some popular hardware approximation methods include single lookup table, approximations using polynomials, CORDIC etc.

3.2.2 Disadvantage of Look up table

For low precision computations, the simplest and fastest method of implementations of such functions is by using Single look-up table. However, for High-precision computations this method gets inappropriate due to large table size and long execution time [1].

3.2.3 Disadvantage of using polynomials

This method is also known as ROM-less system. Implementation using polynomial approximation also has large computational complexities and delays due to extensive use of multiplications and divisions [1].

3.2.4 Disadvantage of the CORDIC

The Coordinate Rotation Digital Computer (CORDIC) algorithm is a popular method for the fast computation of unary functions using only simple shift-add operation. Although, this method is faster than a software solution but due to its iterative property it is slow thus improper for high speed applications [1].

3.3 Parabolic Synthesis methodology

This is a method for hardware implementation of approximations of unary functions using parallelism and low complexity operations. The method consists of three important steps: Normalization, Processing and Post Processing. Of these three steps, the processing step is the most important part but the other two steps are also necessary in some cases [1] [2].

3.3.1 Normalizing

This is the first step of the Parabolic synthesis methodology. The purpose of this step is to limit the numerical range in the interval $0 \leq x < 1$ on the x-axis and $0 \leq y < 1$ on the y-axis to facilitate the hardware implementation. The unary function is normalized to either a concave or convex function, known as the original function $f_{org}(x)$, with starting coordinate of (0,0) and ending coordinate smaller than (1,1) [1] [2].

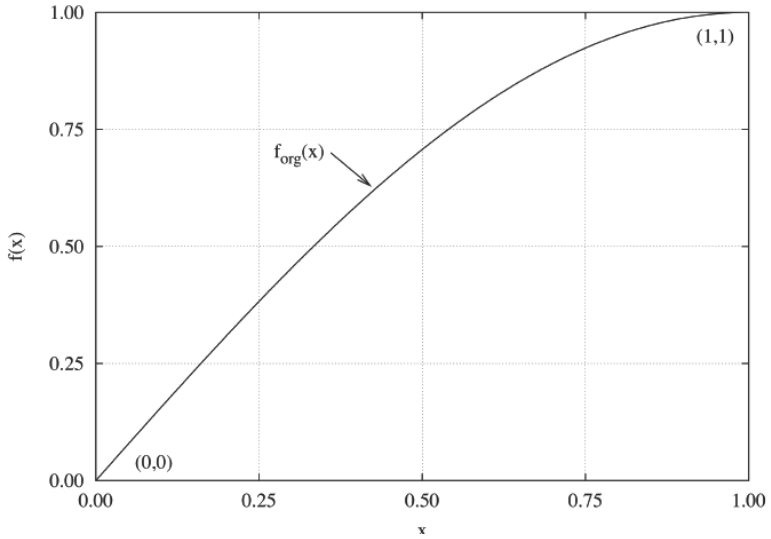

Figure 3.1. Example of normalized function [1].

3.3.2 Developing the Hardware Architecture

For efficient hardware architecture development, this methodology is founded on second order parabolic functions called sub-functions, $s_n(x)$, which uses low complexity operations like shifts, additions and multiplications. Multipliers are commonly used due to the ever going scaling down of the semiconductor technologies and fast development of efficient multiplier architecture which has led hardware implementation of multiplication operation efficient in both size and execution time. As shown in (3.1), the original function $f_{org}(x)$, can be obtained by multiplying the sub-functions and its accuracy depends on the number of sub-functions used [1] [2].

$$forg(x) = s_1(x) \times s_2(x) \times s_3(x) \times s_4(x) \tag{3.1}$$

A parabolic looking function called the first help-function, $f_1(x)$, is obtained by dividing the original function $f_{org}(x)$, with the first sub-function $s_1(x)$.

$$f_1(x) = \frac{forg(x)}{s_1(x)} \tag{3.2}$$

The rest of the functions is generated, as shown on (3.3).

$$f_{n+1}(x) = \frac{f_n(x)}{s_{n+1}(x)} \tag{3.3}$$

3.3.3 Methodology for developing sub-functions

Sub-functions are developed by the decomposition of the original function $f_{org}(x)$ by using second order parabolic functions within the interval $0 \leq x < 1.0$ and the sub intervals within the interval [1] [2].

3.3.3.1 The first sub-function

The first sub-function $s_1(x)$ can be obtained by dividing the original function $f_{org}(x)$ with x as a first order approximation. The division produces two possible results, one where $f(x)>1$ and one where $f(x)<1$ as shown on Fig. 3.2 [1] [2].

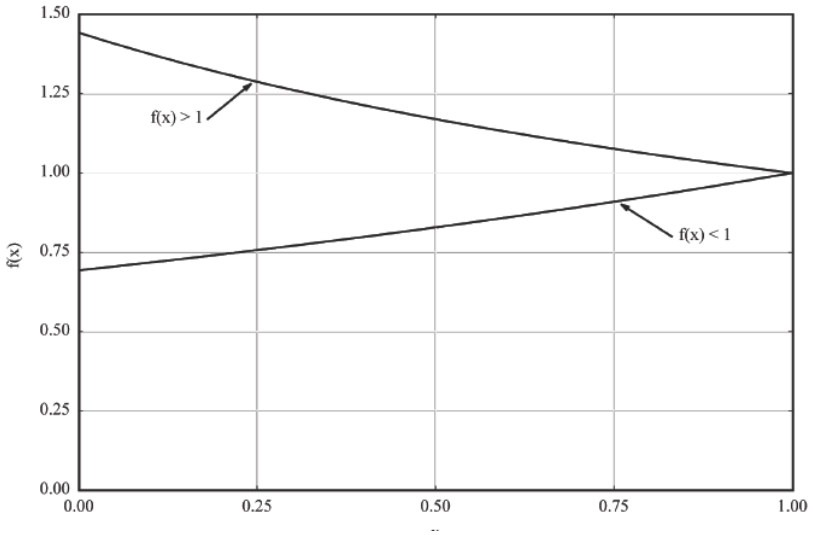

Figure 3.2. Two possible results after dividing an original function with x [1].

The first sub-function $s_1(x)$, as shown on equation (3.4), is achieved by multiplying x with the expression $1+(c_1 \cdot (1-x))$ where the coefficient c_1 is obtained from the limit from the division of the original function with x and subtracted with 1, according to (3.5) [1] [2].

$$s_1(x) = x \times (1 + [c_1 \times (1-x)]) = x + c_1 \times (x - x^2) \tag{3.4}$$

$$c_1 = \lim_{x \to 0} \frac{forg(x)}{x} - 1 \tag{3.5}$$

3.3.3.2 The second sub-function

The second sub-function $s_2(x)$, is chosen as a second order parabolic function as shown in (3.6) [1] [2].

$$s_2(x) = 1 + [c_2 \times (x - x^2)] \tag{3.6}$$

The coefficient c_2, is chosen in a such way that it satisfies with the quotient between the first function $f_1(x)$ and the second sub-function $s_2(x)$ is equal to 1 when x is set to 0.5, as shown below.

$$c_2 = 4 \times [f_1(0.5) - 1] \tag{3.7}$$

In this manner the second help-function $f_2(x)$, will get a shape of lying S shape as shown in figure (3.3). This help-function can be divided into a pair of parabolic looking shapes where the first interval are from $0 \leq x < 0.5$ and second interval from $0.5 \leq x < 1.0$ [2].

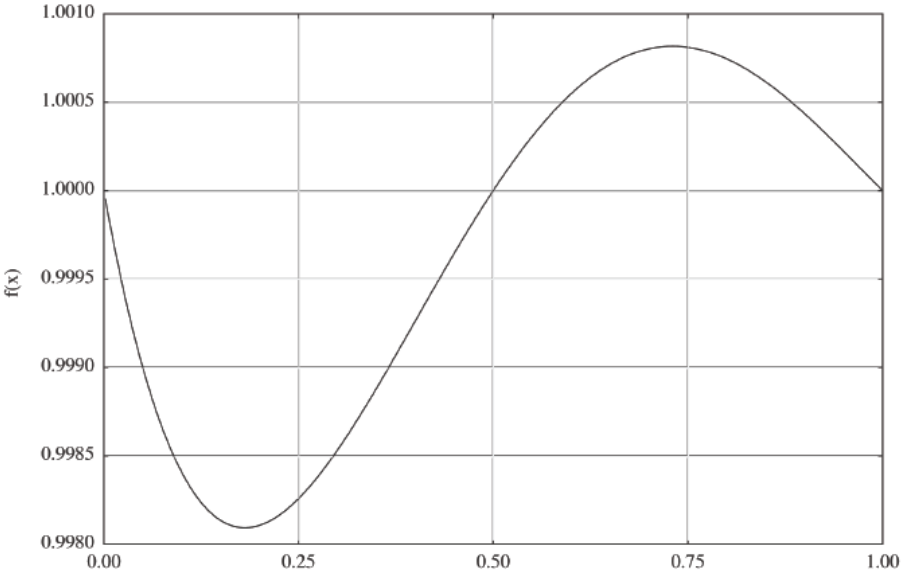

Figure 3.3 Example of the second help function [2].

For easy hardware implementation, the size of the sub-intervals are chosen as a power of 2 since the normalization of the interval can be performed as a left shift of x where the fractional part is the normalization of the two new intervals and the integer part is the addressing of the coefficients for the intervals [1] [2].

3.3.3.3 Sub-functions when n > 2

It is beyond the scope of this thesis to evaluate sub-functions for n>2.

3.3.4 Hardware Implementation

Two's complement representation is used for the hardware implementation. The implementation is divided into three hardware parts: preprocessing, processing, and postprocessing as shown in Figure 3.4.

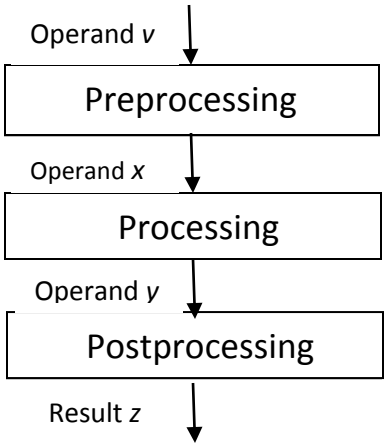

Figure 3.4 The hardware architecture of the methodology [1].

3.3.4.1 Preprocessing

In this part the input operand *v* is normalized for the processing part. For a large system the preprocessing part can be reduced or eliminated if the approximation is implemented together with other logic in the preceding block [1].

3.3.4.2 Processing

In this part, the approximation of the original function is implemented in either iterative or parallel hardware architecture. The iterative architecture as shown on figure (3.6) has the advantage of small chip area but at the expense of longer computation time [1].

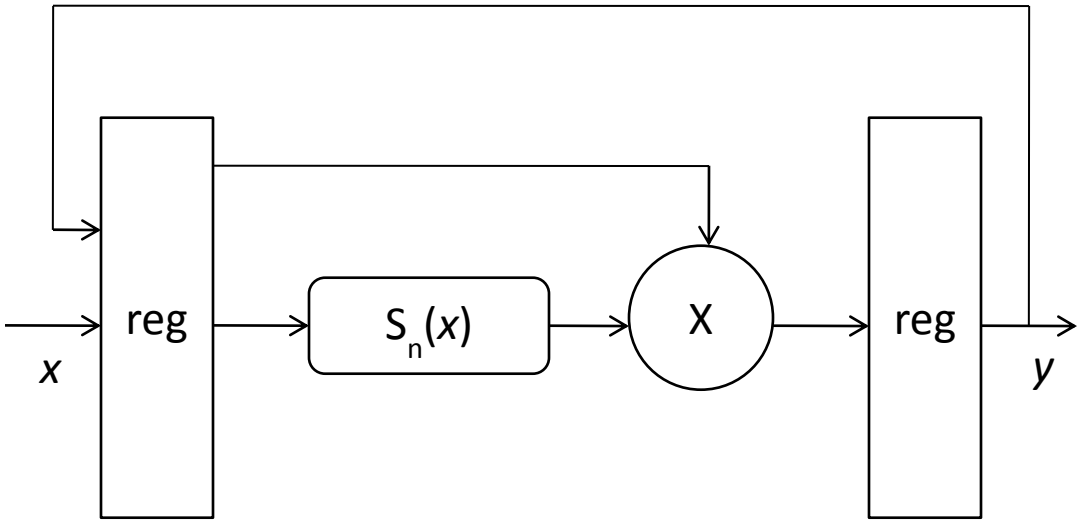

Figure 3.6 The principle of iterative hardware architecture [1].

On the other hand, the parallel hardware architectures as shown for four sub-functions on figure (3.7), give a short critical path and fast computation at the prize of a larger chip area. The throughput can be increased by pipelining.

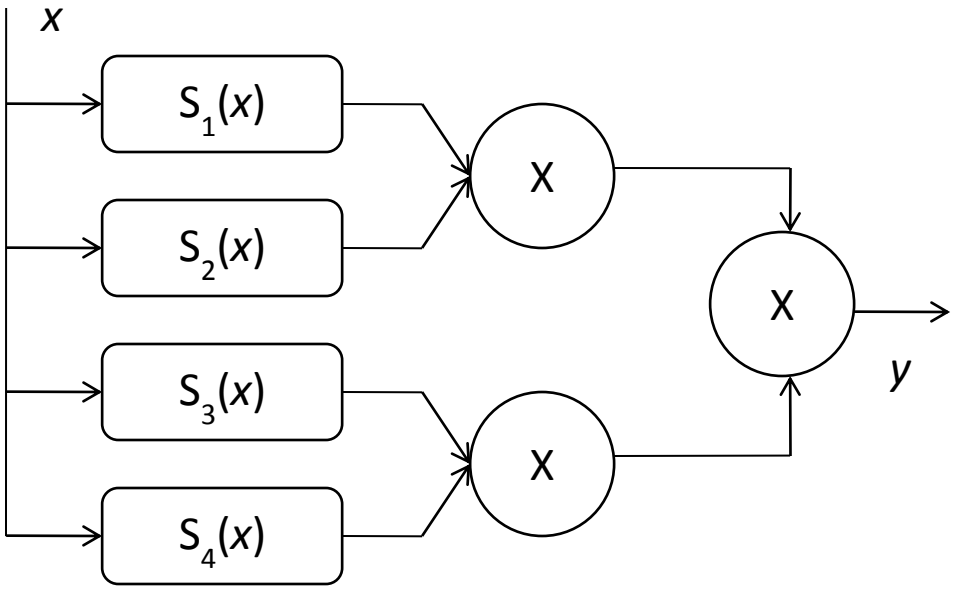

Figure 3.7 The architecture principle for four sub-functions [1].

3.3.4.3 The square Unit

Square components like x^2 and x_n^2 are reoccurring operations in the sub-functions. The square operation x_n^2 in the parallel hardware architecture is a partial result of x^2. That is why a unique squarer has been developed [1].

				x_3 x_3	x_2 x_2	x_1 x_1	x_0 x_0	
							x_0 x_0	
						p_1 x_1 x_0 x_0 x_1	p_0	p
				q_3 x_2 x_1 x_1 x_2	q_2 x_2 x_0 x_0 x_2	q_1	q_0	q
			r_5 x_3 x_2 x_2 x_3	r_4 x_3 x_1 x_1 x_3	r_3 x_3 x_0 x_0 x_3	r_2	r_1	r_0	r
s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0	s

Figure 3.8 Squaring algorithm for the partial product x_n^2 [1].

				x_3 x_3	x_2 x_2	x_1 x_1	x_0 x_0	
							x_0	
					x_1 x_1 x_0	p_1	p_0	p
			x_2 x_2 x_1	q_3 x_2 x_0	q_2	q_1	q_0	q
	x_3	r_5	r_4	r_3	r_2	r_1	r_0	r

		$x_3\ x_2$	$x_3\ x_1$	$x_3\ x_0$					
	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0	s

Figure 3.9 Simplified squaring algorithm for the partial product x_n^2 [1].

Vector *p*:

$$p_0=x_0x_0=x_0 \tag{3.8}$$

The result of component p_1 is equal to 0 as the result of p_0 does not contribute anything to p_1.

Vector *q*:

$$q_1=p_1.2^1+x_1x_0.2^1+x_0x_1.2^1=p_1.2^1+x_1x_0.2^2=p_1.2^1 \tag{3.9}$$

$$q_2=x_1x_0.2^2+x_1x_1.2^2=x_1.2^2+x_1x_0.2^2 \tag{3.10}$$

Vector *r*:

$$r_2=q_2.2^2+x_2x_0.2^2+x_0x_2.2^2=q_2.2^2+x_2x_0.2^3=q_2.2^2 \tag{3.11}$$

$$r_3=q_3.2^3+x_2x_1.2^3+x_1x_2.2^3+x_2x_0.2^3 = q_3.2^3+x_2x_1.2^4+x_2x_0.2^3=q_3.2^3+x_2x_0.2^3 \tag{3.12}$$

$$r_4=x_2x_1.2^4+x_2x_2.2^4=x_2.2^4+x_2x_1.2^4 \tag{3.13}$$

Vector *s*:

$$s_3=r_3.2^3+x_3x_0.2^3+x_0x_3.2^3=r_3.2^3+x_3x_0.2^4=r_3.2^3 \tag{3.14}$$

$$s_4=r_4.2^4+x_3x_0.2^4+x_3x_1.2^4+x_1x_3.2^4=r_4.2^4+x_3x_0.2^4+x_3x_1.2^5=r_3.2^3+x_3x_0.2^4 \tag{3.15}$$

$$s_5=r_5.2^5+x_3x_1.2^5+x_3x_2.2^5+x_2x_3.2^5=r_5.2^5+x_3x_1.2^5+x_3x_2.2^6=r_5.2^5+x_3x_1.2^5 \tag{3.16}$$

$$s_6=x_3x_2.2^6+x_3x_3.2^6=x_3.2^6+x_3x_2.2^6 \tag{3.17}$$

The value of component s_7 in the *s* vector is a possible carry from the s_6 component. The result of square *x*, x^2 is in the *s* vector and the partial products of the square are found for x_3^2 in *r* vector and in *q* for x_4^2 [1].

3.3.5 Postprocessing

The main motivation for the part is to transform the output to a feasible form for the proceeding parts in the system.

CHAPTER 4

4 Parabolic Architecture development for Trigonometric functions

4.1 Design Methodology

From chapter 3, we have seen that parabolic architecture uses parallelism to reduce the execution time and employs low complexity operations thus making hardware implementation faster and simpler than all other existing methodologies.

The first step of developing the architecture is to define the specifications and the requirements. The second step is to develop the behavioral descriptions using a hardware description language like VHDL, which will lead to the development of register transfer level (RTL) model. The RTL model is simulated using a testbench for verification of the defined logic and for finding possible errors. After successful simulation, the design is synthesized for either a FPGA or an ASIC. The synthesis converts the RTL model into a design implementation in terms of logic gates. These logic gates are further simulated for the actual implementation of the architecture in hardware and the process is known as post-synthesis simulation. On the success of the simulation, the layout is sent for fabrication. The top down design methodology is shown on figure 4.1.

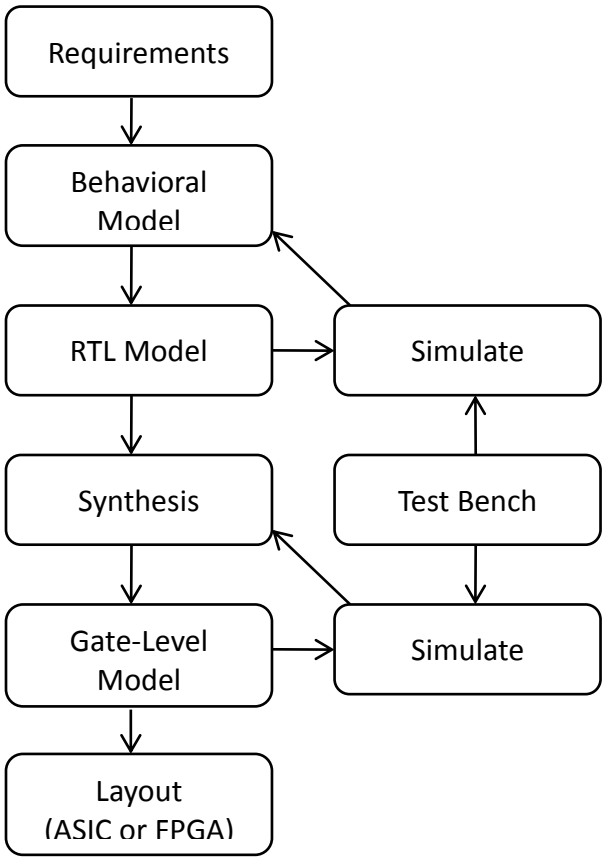

Figure 4.1 Top down design Methodology [7].

4.2 The sub-functions

Based on the concepts on chapter 3, the sub-functions, which lead to the approximated sine and cosine functions are shown in (4.1) and (4.2) respectively. The angle θ_f is the normalized fractional part of υ. It can be noted that it is only s_1 that differs for the sine and cosine functions.

$$\begin{aligned} s_1(\theta_f) &= \theta_f + c_1 \times (\theta_f - \theta_f^2) \\ s_2(\theta_f) &= 1 + c_2 \times (\theta_f - \theta_f^2) \\ \sin(\theta_f) &\approx s_1(\theta_f) \times s_2(\theta_f) \end{aligned} \tag{4.1}$$

$$\begin{aligned} s_1(\theta_f) &= 1 - \theta_f + c_1 \times (\theta_f - \theta_f^2) \\ s_2(\theta_f) &= 1 + c_2 \times (\theta_f - \theta_f^2) \\ \cos(\theta_f) &\approx s_1(\theta_f) \times s_2(\theta_f) \end{aligned} \tag{4.2}$$

Where c_1 and c_2 are the coefficients.

The optimal 7-bit coefficients are shown in (4.3). For obtaining parallelism in the architecture, c_1 and c_2 are multiplied at the same time. Figure 4.2 shows the basic Parabolic Synthesis architecture.

$$\begin{aligned} c_1 &= 0.5703125 = 0.1001001_2 \\ c_2 &= 0.4062500 = 0.0110100_2 \end{aligned} \tag{4.3}$$

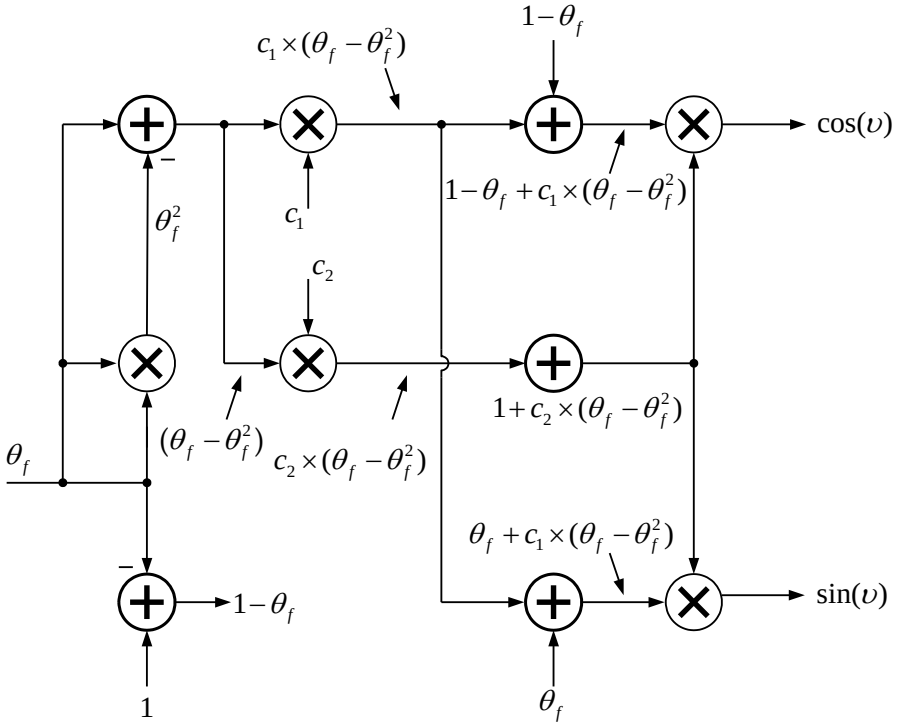

Figure 4.2 The first parabolic synthesis architecture

4.3 Angle transformation

The architecture shown in figure 4.2 is only valid for angles in the first quadrant. The other three quadrants must also be covered, which is done by transforming normalized angles larger than "1" to the first quadrant. Figure 4.3 illustrates the methodology. In figure 4.3a, the original angles are shown. These are normalized to Figure 4.3b, with a factor $2/\pi$, e.g. the angle $\upsilon = \pi/6$ is transformed to the normalized angle $\theta_f = 1/3$. The equations for the normalization are shown below.

$$\upsilon = \pi - \frac{\pi}{6} \tag{4.4}$$

$$norm = \frac{2}{\pi} \times \upsilon = \frac{2}{\pi} \times (\pi - \frac{\pi}{6}) = 01\frac{2}{3} \tag{4.5}$$

$$frac = \frac{2}{3} \tag{4.6}$$

$$\theta_f = 1 - frac = \frac{1}{3} \tag{4.7}$$

All the normalized angles in the first quadrant are less than "1". Figure 4.3b shows the integer part of the angle θ. The integer part, $\theta_1\theta_0$, will thus show which quadrant the angle is in, i.e. quadrant 1, 2, 3 or 4 and is represented by 0, 1, 2, 3 respectively in binary representation. That is useful in the selection of the angle when transforming to the first quadrant, as shown in figure 4.3c for the sine functions. This can be illustrated with the example in (4.4) – (4.7).

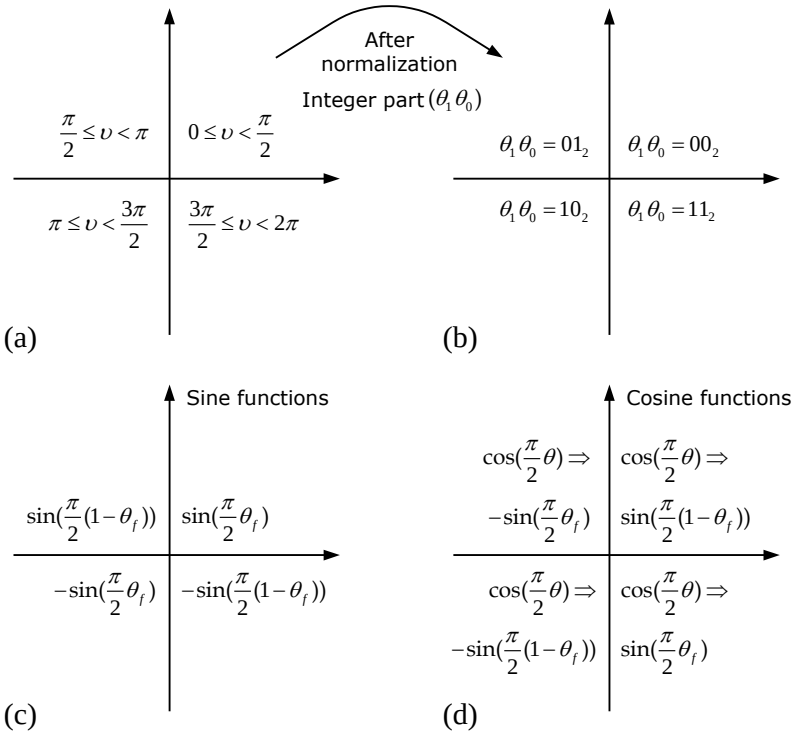

Figure 4.3 Transforming angles from quadrant 2-4 to quadrant 1.

An angle $\upsilon = \pi - \pi/6$ (4.4) in the second quadrant corresponds to the angle $\upsilon = \pi/6$ in the first quadrant. After normalization the angles will correspond to 5/3 and 1/3, where the first angel is larger than one (4.5). The idea is now to transform 5/3, in the second quadrant to $\theta_f = 1/3$ in the first quadrant. The integer part, $\theta_1\theta_0 = 01$, in (4.5) is taken away to get (4.6). Finally, the angle $\theta_f = 1/3$ is transformed to the first quadrant (4.6). However, the integer part is not thrown away. It will be used to select the quadrant for the initial angle and for two's complement conversion at the output. Figure 4.3d shows the transformation of the cosine functions in the four quadrants. All cosine functions are transformed to sine functions since the architecture is designed for sine functions.

Table I shows when the input transformations are needed. To select that, the integer value θ_0 is used. This is solved by using two MUXes.

Table I. Input transformations

	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
Sine	θ_f	$1-\theta_f$	θ_f	$1-\theta_f$
Cosine	$1-\theta_f$	θ_f	$1-\theta_f$	θ_f

Since all calculations are done in the first quadrant, the output has to be transformed back again. As an example, if we compute cos(υ) for the 2nd quadrant angle $\pi - \pi/3$, we get the value -0.5. However, the cosine value is determined in the 1st quadrant with the angle $\pi/3$, which gives the value 0.5. To correct that, a two's complement conversion is needed. Table II shows when the two's complement conversion is needed. To select that, the integer values θ_1 and θ_0 are used.

Table II. Output transformations

	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
Sine	+	+	-	-
Cosine	+	-	-	+

Figure 4.4 shows the updated architecture capable of transforming the angles.

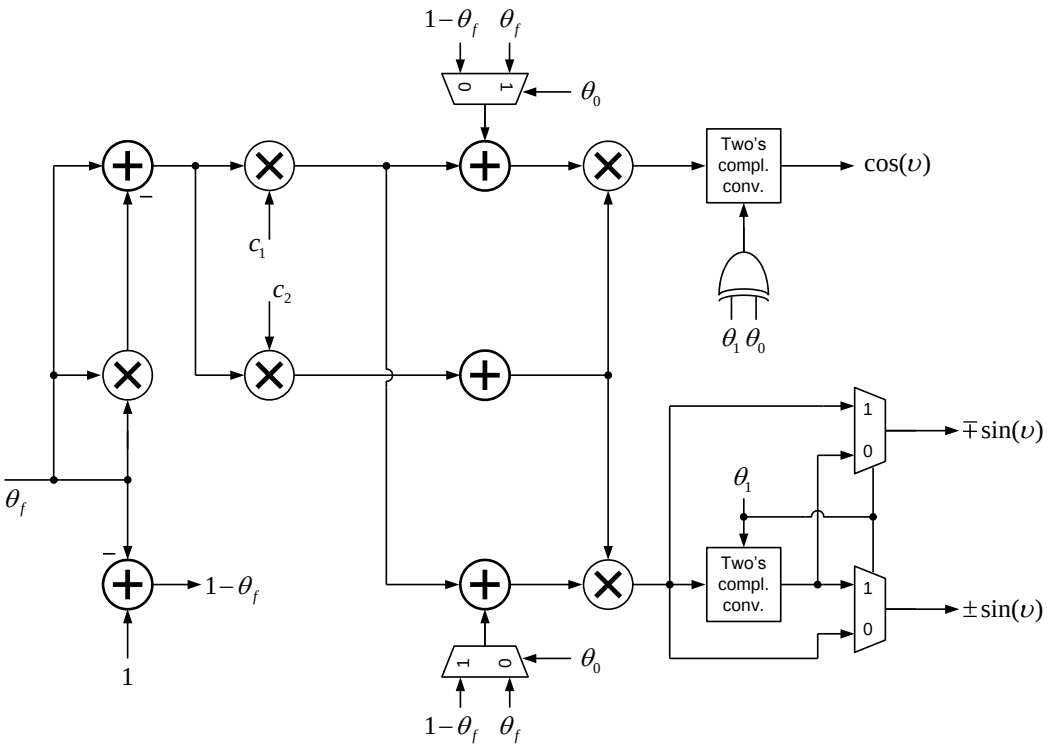

Figure 4.4 The second parabolic architecture

4.4 Simplifications

Figure 4.4 is further improved by optimizing some of the components.

4.4.1 The MCM unit

The design can be improved by optimizing two multipliers. The multipliers can be exchanged with three adders and some shifts as shown in Figure 4.5. The technique is called Multiple Constant Multiplication (MCM). There is one input, $\theta_f - \theta_f{}^2$, two outputs $c_1(\theta_f - \theta_f{}^2)$ and $c_2(\theta_f - \theta_f{}^2)$ in both figures. In the VHDL-code, we thus eliminated expressions like "*A*x*B"* for these two multipliers.

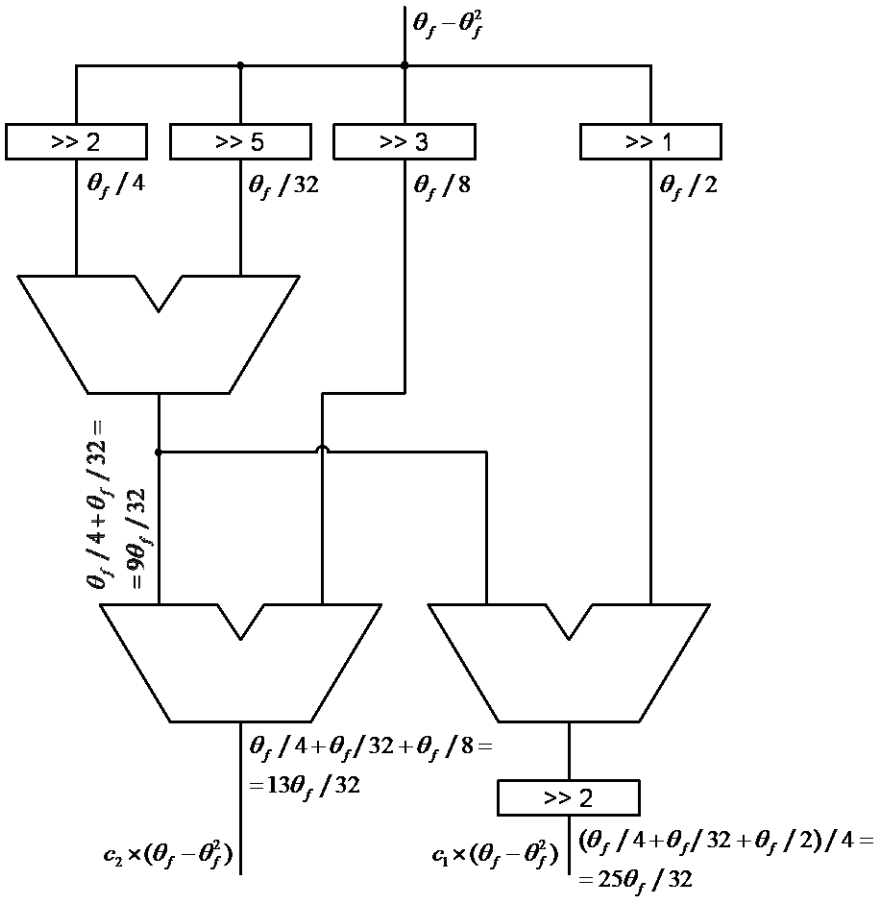

Figure 4.5 Architecture for Multiple Constant Multiplication (MCM)

4.4.1.1 Example

If we use a fractional positive (it cannot be negative) input value for the left part in figure 4.5 we get:

$\theta_f - \theta_f^2 = 1/7 \Rightarrow 001001001101$

000010010011|01000	Shifted by 2	$\theta_f/4$
000000010010|01101	Shifted by 5	$\theta_f/32$

000010100101|10101		
000001001001|10100	Shifted by 3	$\theta_f/8$

000011101111|01001		

In (4.8) we then get the result for the left part of the architecture in figure 4.5

$$c_2(\theta_f - \theta_f^2) = (\theta_f - \theta_f^2)/4 + (\theta_f - \theta_f^2)/32 + (\theta_f - \theta_f^2)/8 \quad (4.8)$$

4.4.2 Eliminating an adder

Adding a "1" to $c_2(\theta_f - \theta_f{}^2)$ can be simplified. Since c_2 is positive, $c_2(\theta_f - \theta_f{}^2)$ will never be larger than "1", i.e. $c_2(\theta_f - \theta_f{}^2) < 1$. The fractional part can thus be merged to the "1" directly with the wiring as shown in figure 4.6.

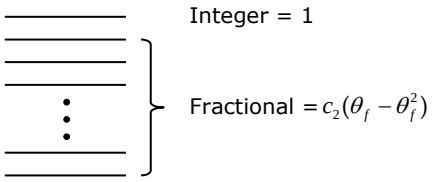

Figure 4.6 The fractional bus with an added integer "1"

4.4.3 Two's complement conversion

Figure 4.7 shows an architecture for two's complement conversion. The architecture uses half adders (HAs) and XOR gates. A control signal θ_1 or θ_1 XOR θ_0 is used to select when the conversion is to be done.

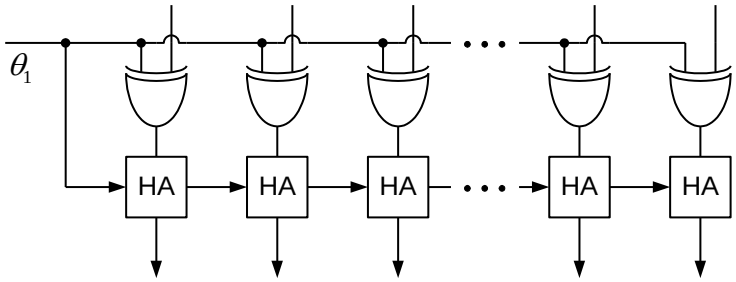

Figure 4.7 Two's complement conversion

4.4.4 Squarer

Instead of using a multiplier for the squaring, a simplified version can be used as shown in figure 4.8.

					*x*5	*x*4	*x*3	*x*2	*x*1	*x*0
					*x*5	*x*4	*x*3	*x*2	*x*1	*x*0
*x*5*x*4	*x*5*x*3	*x*5*x*2	*x*5*x*1	*x*5*x*0	*x*4*x*0	*x*3*x*0	*x*2*x*0	*x*1*x*0	0	*x*0
*x*5		*x*4*x*3	*x*4*x*2	*x*4*x*1	*x*3*x*1	*x*2*x*1		*x*1		
		*x*4	*x*3*x*2	*x*3*x*2´		*x*2				

Figure 4.8 A 6-bit squarer

4.5 Final architecture

The final architecture with all the simplifications is shown in figure 4.9. The architecture contains:

- Two multipliers,
- One squarer,
- Seven adders,
- Two two's conversion converter, and
- Four MUXes.

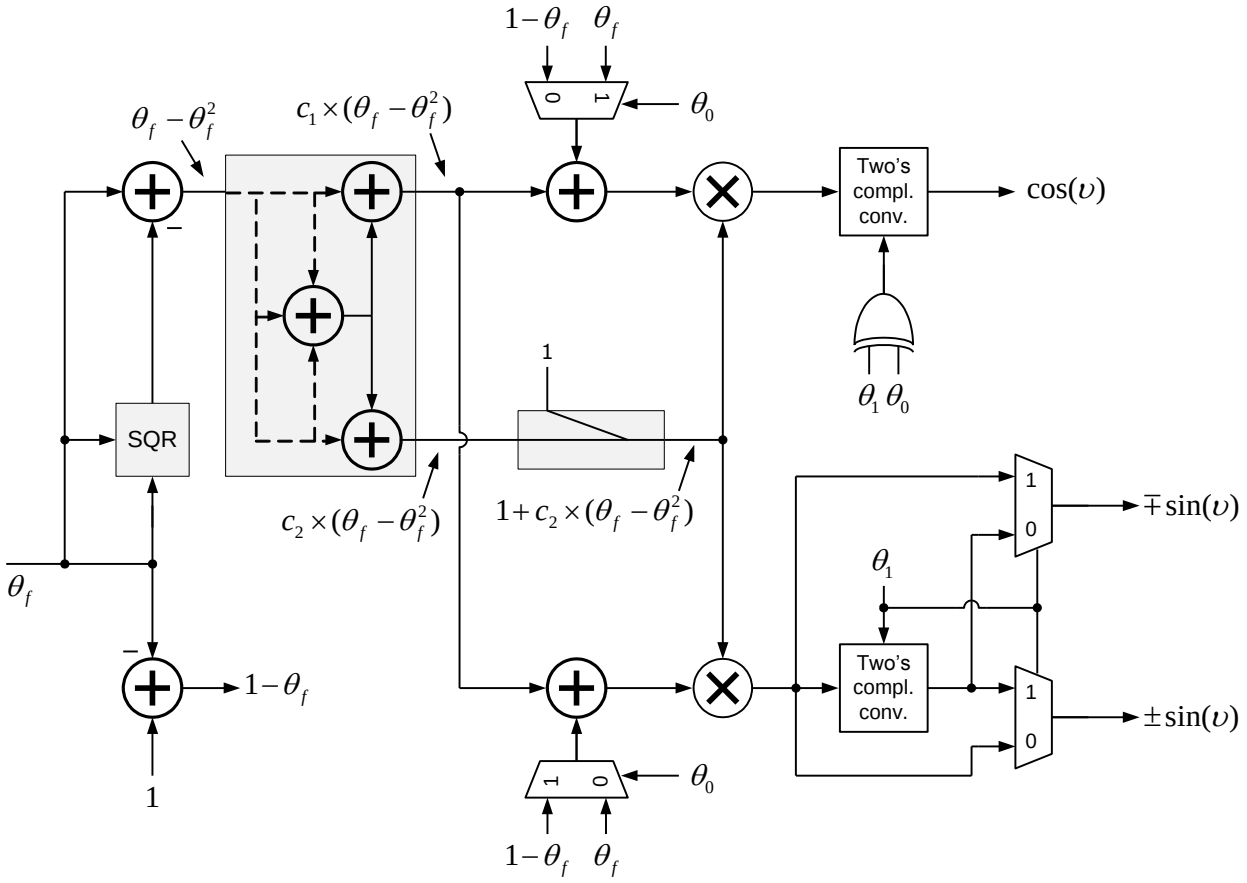

Figure 4.9 The final architecture.

The critical path goes through:

- One squarer
- Four adders
- One multiplier
- One two's conversion converter
- One MUX

4.6 Wordlengths

Figure 4.10 shows the needed internal bits to reach a 9-bit accuracy at the output.

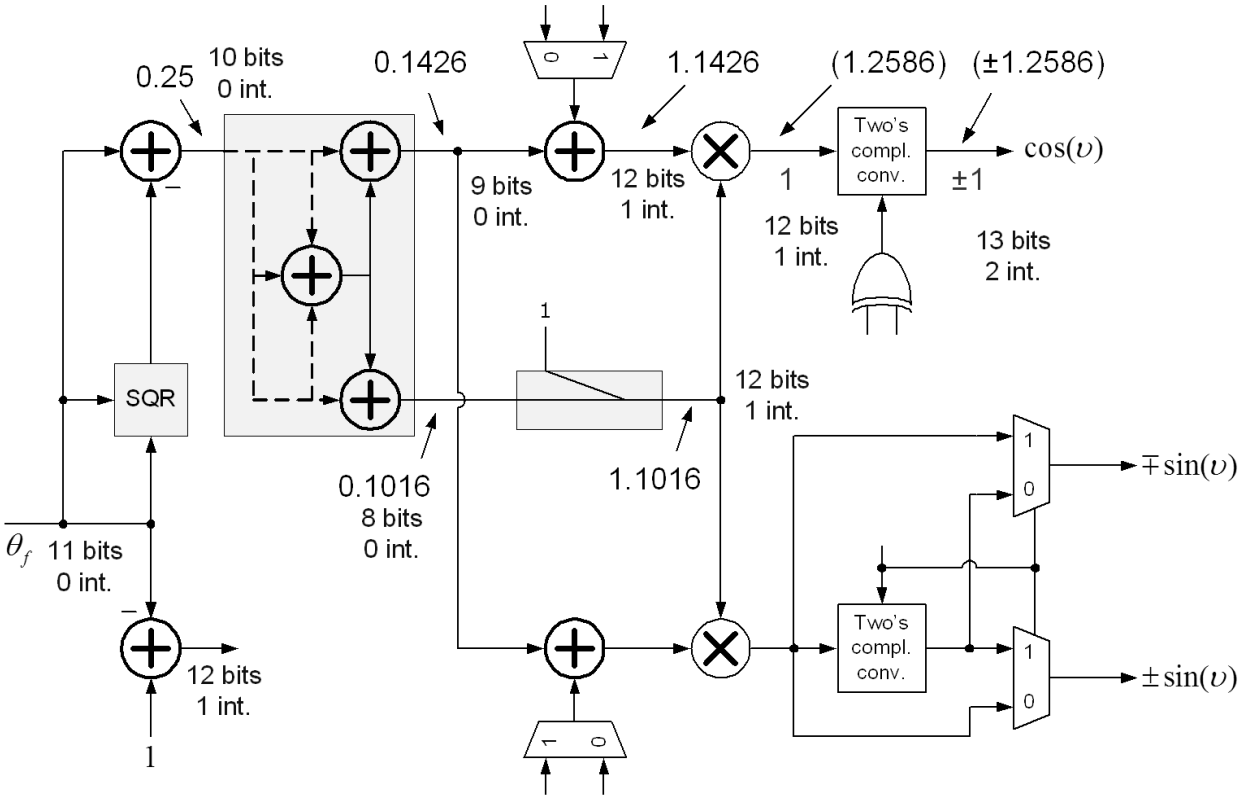

Figure 4.10 Internal wordlengths

4.7 Hardware for matrix inversion using QRD by Parabolic synthesis

Figure 4.11 shows the proposed hardware for doing matrix inversion using QRD. Due to time limitation the parabolic hardware could not be integrated with the QRD hardware.

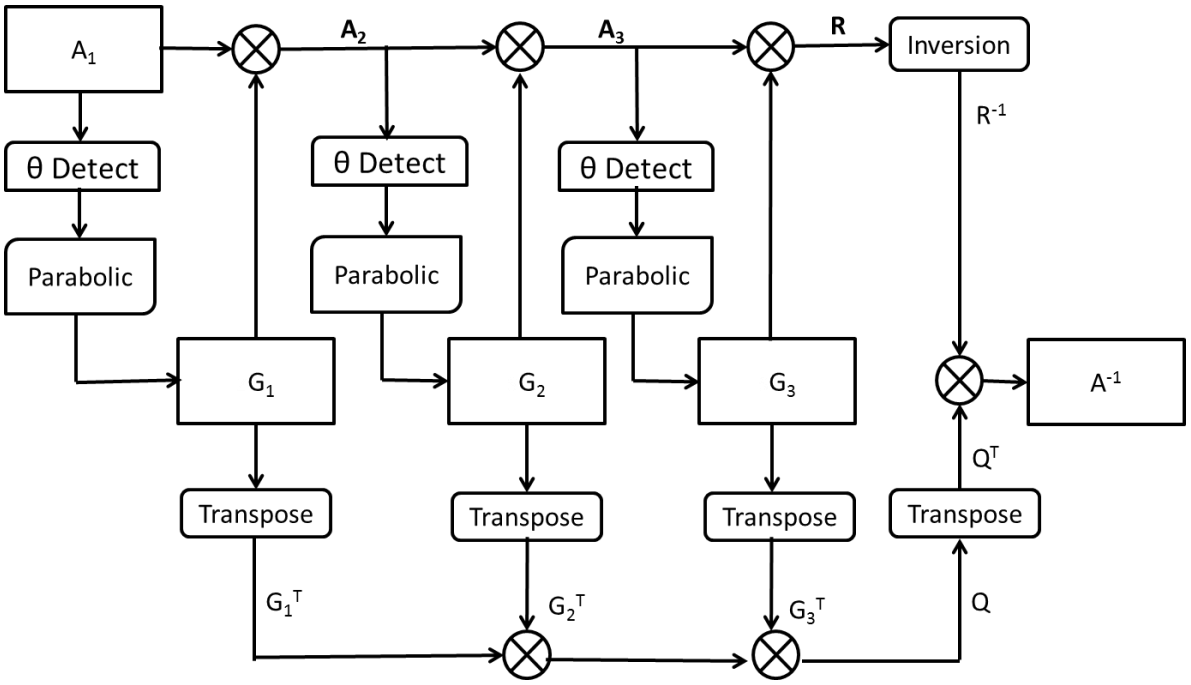

Figure 4.11. Proposed hardware for matrix inversion using QRD.

CHAPTER 5

5 Synthesis

5.1 Synthesis

Synthesis is a process by which conceptual description of the logic functions needed for the desired circuit behavior (typically register transfer level (RTL)) is turned into a design implementation in terms of logic gates [8].

The flow chart of the synthesis is:

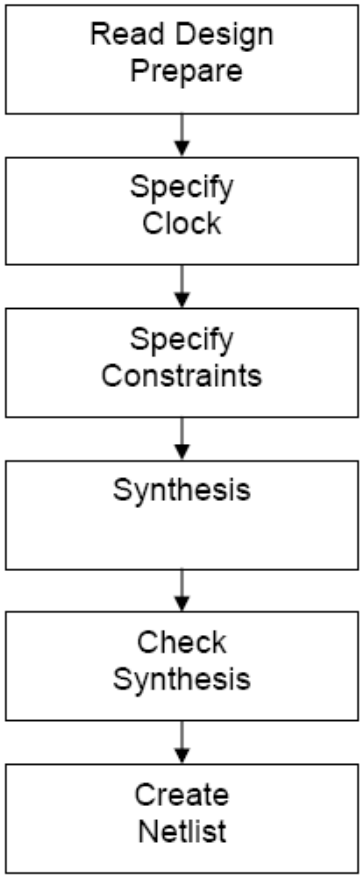

Figure 5.1 Flow chart of synthesis process [8].

5.2 Types of Synthesis

For the thesis work two types of synthesis have been performed. One targeted towards Virtex 2 pro FPGA using Xilinx ISE design suit and the other have been performed for an ASIC implementation using Synopsis Design Vision tool in STM 65nm technology. As mentioned in the previous chapter,

the Hardware Description Language (HDL) has been used for the design implementation. Some of the advantages of HDL for synthesis include:

1) Decrease in design time by permitting a high-level design specification,
2) Reduced errors for manual translation from HDL to schematic design,
3) Increased optimization and efficiency due to the utilization of the automation techniques used by the synthesis tool (such as, automatic I/O insertion and machine encoding styles) during the optimization to the original HDL code.

5.3 Synthesis for FPGA

The design has been synthesized on Xilinx ISE Design suit for Virtex 2 pro FPGA using VHDL for a speed grade of -7. The RTL schematic from the synthesis is shown in fig. 5.2.

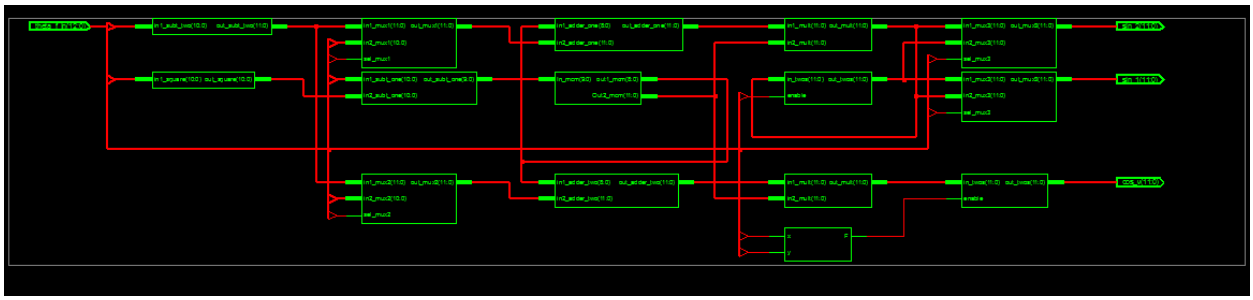

Figure 5.2 RTL Schematic obtained from synthesis on Virtex 2 pro FPGA.

The RTL schematic is shown in fig. 5.3.

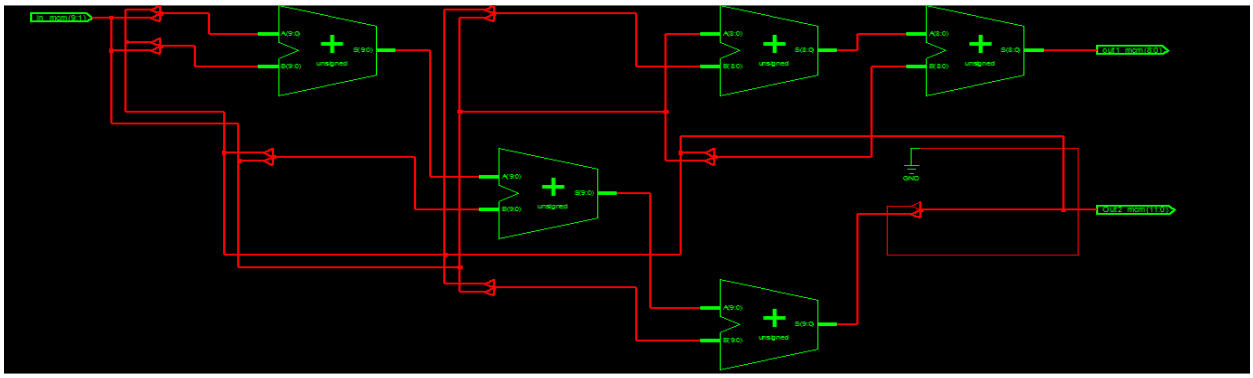

Figure 5.3 MCM block Schematic obtained from synthesis on Virtex 2 pro FPGA.

The technology schematic for the overall design is shown in fig. 5.4

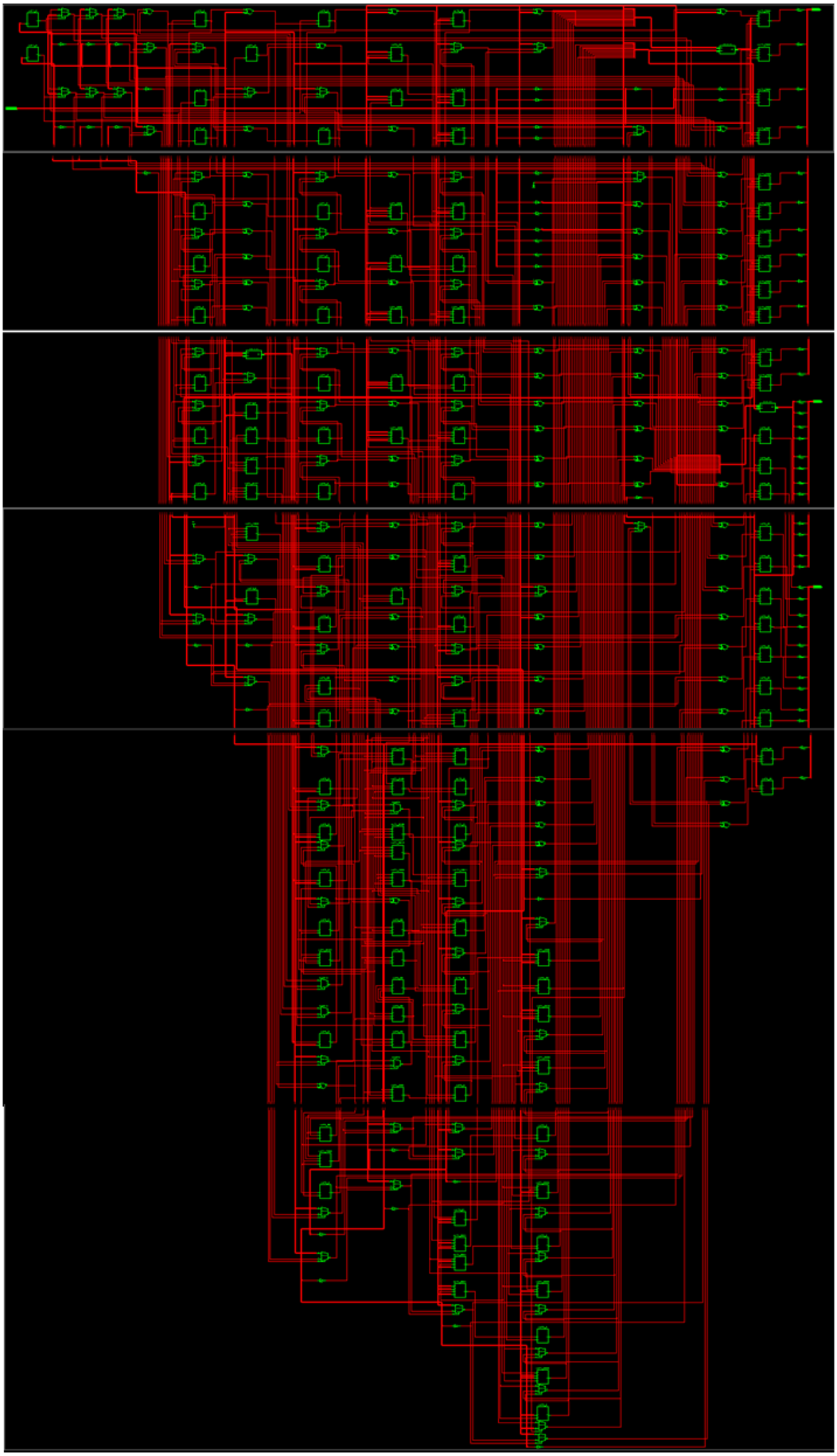

Figure 5.4 Technology Schematic obtained from synthesis on Virtex2pro FPGA.

5.3.1 Synthesis Report from FPGA

The critical path includes the squarer, one subtractor, one multiplier, one two's complement unit, one multiplexer and the MCM unit. The critical path time is 20.496 ns so the maximum clock frequency that is achievable is (1/20.496) = 48MHz. The individual delay of each component in the critical path is shown in Appendix 1, Table VIII.

Detailed synthesis reports containing the macro statistics, cell usage and device utilization are shown in Appendix 1, Table IX, X and XI respectively.

5.4 Synthesis for ASIC

The design has been synthesized towards an ASIC implementation on a 65nm technology using Synopsis Design Vision tool. Two constraints, area and speed, were emphasized on this synthesis. As a result, the design has been synthesized for both high speed and for minimum area. The scripts used for these two types of synthesis are shown on appendix 1, Table I.

5.4.1 Minimum Area Synthesis

While doing synthesis for minimum area, we have set the clock period to a very high value and set the maximum area to zero.

The constraints that were set for minimum area are:

i) Area, and
ii) Clock uncertainty time.

5.4.2 High Speed Synthesis

While synthesizing for high speed we have set the clock constraint to such a value so that we do not get any negative slack and no parameter for the area constraint.

The constraints that were set for High Speed are:

i) Clock speed, and
ii) Clock uncertainty time.

5.4.3 Results from ASIC Synthesis

The results obtained from the Synopsis Design Vision synthesis report are as follows:

5.4.3.1 Area

The total area of the design is 450893 of which 2893 is the combinational area and the remaining 448000 is the sequential area consisting of the I/O pads and the input and output registers used for determining the critical path. The individual component area is shown in Appendix 1, Table IV.

5.4.3.2 Implemented arithmetic blocks

The synthesis tool used the two libraries 'IO65LPHVT_SF_1V8_50A_7M4X0Y2Z' and 'CORE65LPHVT' for the implementation of the arithmetic blocks. The details of each implemented blocks are shown in Appendix 1, Table V.

5.4.3.3 Timing

The critical path includes one squarer, one subtractor, one adder, one multiplier, one two's complement unit, one multiplexer and the MCM unit. The critical path time is 11.33 ns so the maximum clock frequency that is achievable is (1/11.33) =88MHz. The individual delay of each component in the critical path is shown in Appendix 1, Table III.

5.4.3.4 Power

From synthesis we obtained the dynamic power to be 0.0794mW of which 54.71% is net switching power and 45.27% is cell internal power.

For obtaining a more accurate power report, the design has been simulated in Prime Time tool. The Script and the detailed report are shown in Appendix 1, Table VII.

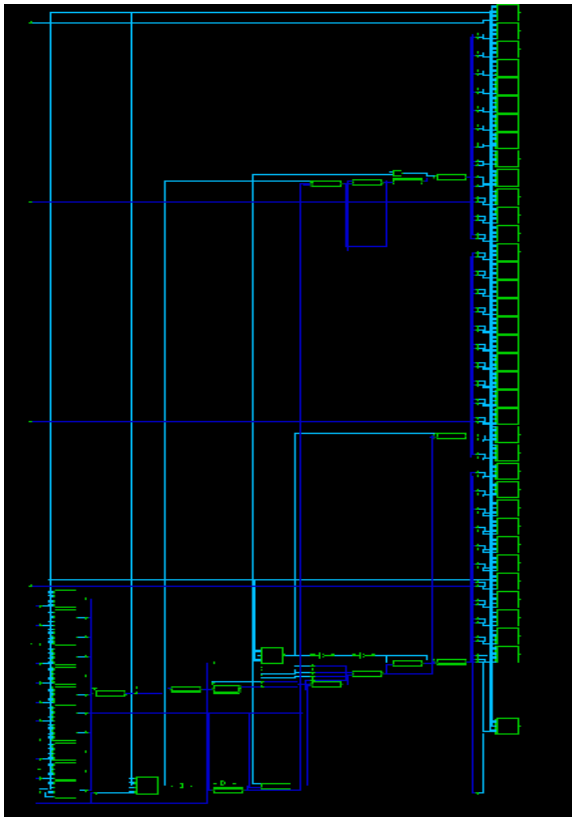

Figure 5.5 The parabolic architecture with I/O pads from Design Vision.

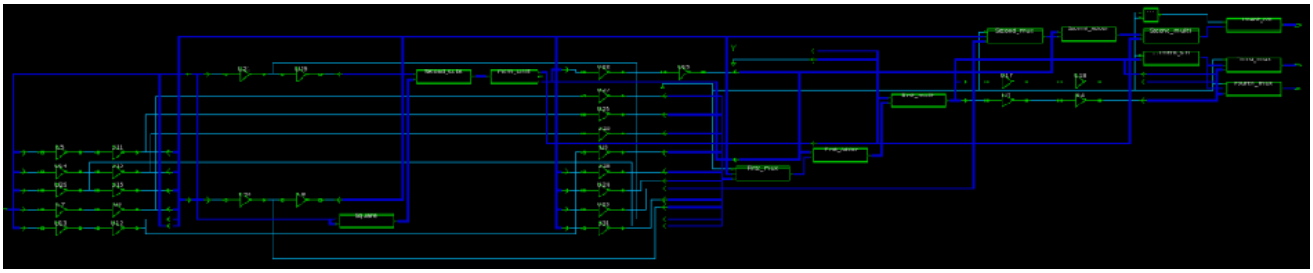

Figure 5.6 Parabolic architecture from Design vision.

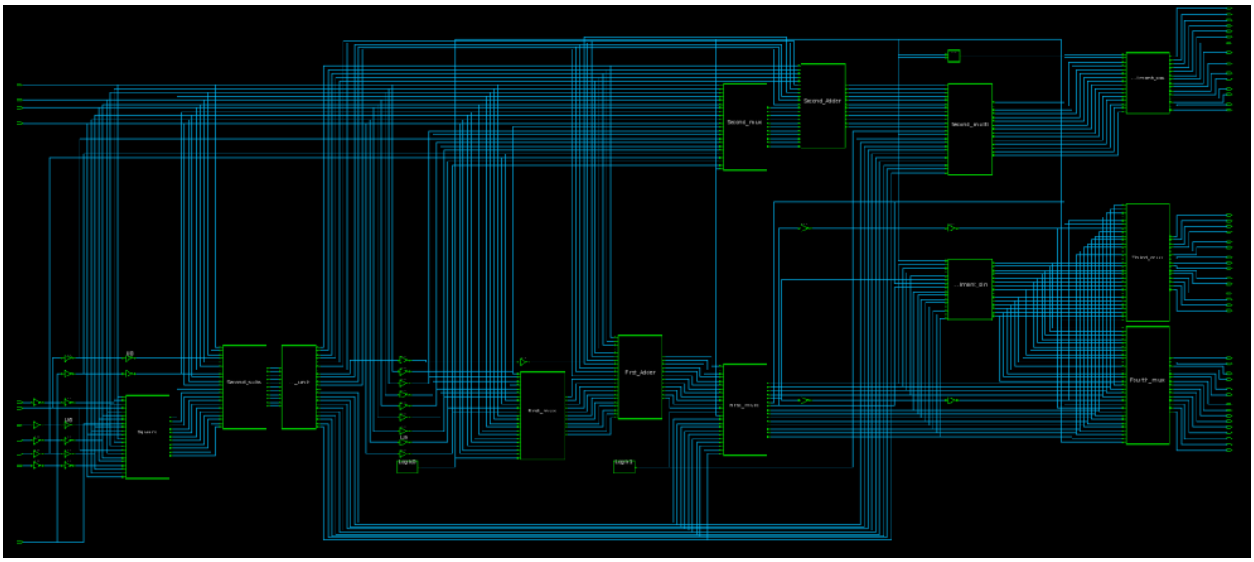

Figure 5.7 Parabolic architecture from Primetime

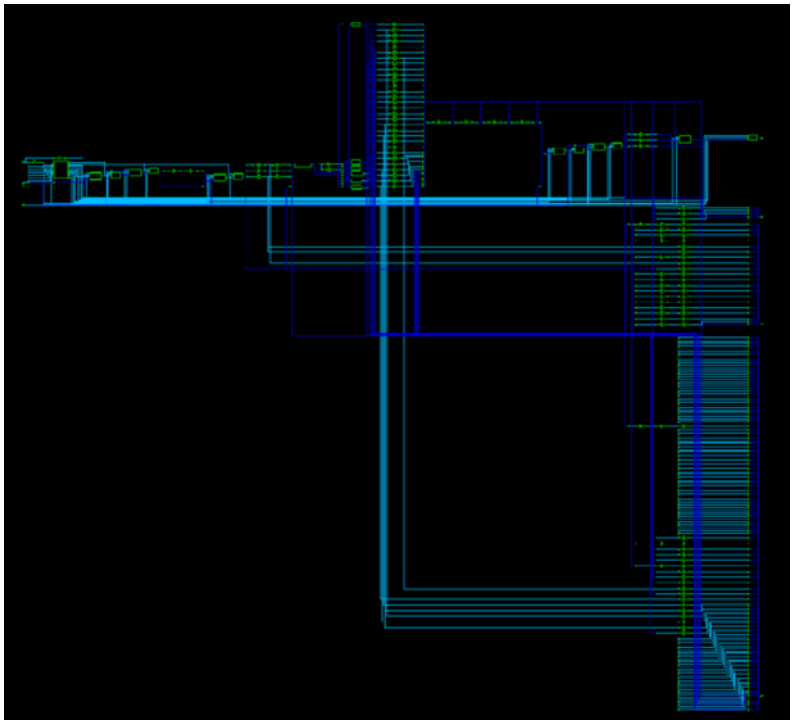

Figure 5.8 Schematic view of the Matrix Inversion from Design Vision

CHAPTER 6

6 Results and Conclusion

The aim of the thesis was to develop hardware for the generation of three trigonometric functions (+sine, -sine and +cosine) using the novel approximation methodology, which is based on Parabolic synthesis, for the use in Givens rotations for implementing matrix inversion using QR decomposition. Separate hardware for parabolic synthesis and matrix inversion using QRD is implemented. However, due to time limitation, the two modules could not be integrated. Although, a hardware solution had been presented at the end.

CHAPTER 7

7 Future Work

The matrix inversion unit only works for a fixed set of matrix. Furthermore, the division unit is limited to 4 bits.

In future, square root implementation could lead to matrix inversion for any set of values. The division logic can be scaled down at the beginning and again scaled up at the end to allow the usage of larger bits.

Reference

[1] http://www.eit.lth.se/fileadmin/eit/docs/Licentiate/lic_14.5_mac.pdf

[2] Erik Hertz and Peter Nilsson, "A Methodology for Parabolic Synthesis", a book chapter in VLSI, In-Tech, ISBN 978-3-902613-50-9, pp. 199-220, Vienna, Austria, September 2009.

[3] http://en.wikipedia.org/wiki/Invertible_matrix

[4] http://www.purplemath.com/modules/mtrxinvr.htm

[5] http://en.wikipedia.org/wiki/QR_decomposition

[6] http://en.wikipedia.org/wiki/Givens_rotation

[7] http://www.eit.lth.se/index.php?id=241&ciuid=475&coursepage=2602&L=1

[8] http://www.eit.lth.se/fileadmin/eit/courses/etin01/slides/Synthesis.pdf

[9] P.T.P. Tang (1991), "Table-lookup algorithms for elementary functions and their error analysis," Proc. of the 10th IEEE Symposium on Computer Arithmetic, pp. 232 - 236, ISBN: 0-8186-9151-4, Grenoble, France, June 1991

[10] J.-M. Muller (2006), Elementary Functions: Algorithm Implementation, second ed. Birkhauser, ISBN 0-8176-4372-9, Birkhauser Boston, c/o Springer Science+Business Media Inc., 233 Spring Street, New York, NY 10013, USA

[11] Erik Hertz and Peter Nilsson, “Parabolic Synthesis Methodology Implemented on the Sine Function”, in Proceedings of the 2009 International Symposium on Circuits and Systems (ISCAS’09), Taipei, Taiwan, May 24-27, 2009.

[12] Lei Wang, Hardware Implementation of Parabolic Synthesis Methodology for the Sine and Cosine Functions, Master’s thesis, Lund University 2009.

Appendix 1: For Synthesis

Table I: Synthesis Scripts for ASIC

For High Speed:
gui_start remove_design -all analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/para_generics_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/adder_block_pack1.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/adder_block_pack2.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/subtractor_block_pack1.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/subtractor_block_pack2.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mcm_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mult_block_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack1.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack2.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack3.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack4.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/squarrer_block_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/twos_comp_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/XOR_block_pack.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/parabolic_senthesis1.vhd} analyze -library WORK -format vhdl {/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/top_parabolic_senthesis1.vhd} elaborate top_parabolic_senthesis -lib WORK -arch structural compile -map_effort high report_constraint -all_violators change_names -rules verilog -hierarchy write -format verilog -hierarchy -output netlists/top_para.v write_sdf ./netlists/top_para.sdf write_sdc ./netlists/top_para.sdc
For Minimum Area:
gui_start remove_design -all analyze -library WORK -format vhdl

```
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/para_generics_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/adder_block_pack1.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/adder_block_pack2.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/subtractor_block_pack1.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/subtractor_block_pack2.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mcm_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mult_block_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack1.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack2.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack3.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/mux2by1_pack4.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/squarrer_block_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/twos_comp_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/XOR_block_pack.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/parabolic_senthesis.vhd}
analyze -library WORK -format vhdl
{/home/piraten/sx08nc4/Desktop/New_Parabolic/vhdl/top_parabolic_senthesis1.vhd}
elaborate top_parabolic_senthesis  -lib WORK -arch structural
set_max_area 0
compile -map_effort high
report_constraint -all_violators
change_names -rules verilog -hierarchy
write -format verilog -hierarchy -output netlists/par_min.v
write_sdf ./netlists/par_min.sdf
write_sdc ./netlists/par_min.sdc
```

Table II: Area Hierarchy of the design(ASIC)

Cell	Reference	Library	Area	Attributes
InPad_0	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_1	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_2	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_3	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_4	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_5	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_6	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	
InPad_7	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_8	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_9	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_10	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_11	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
InPad_12	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
pad_clock	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	d, n
pad_cos_v_0	BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z	4480.000000	

```
pad_cos_v_1             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_2             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_3             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_4             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_5             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_6             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_cos_v_7             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
 pad_cos_v_8             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_cos_v_9             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_cos_v_10            BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_cos_v_11            BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_sin_1_0             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_sin_1_1             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_sin_1_2             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
 pad_sin_1_3             BD2SCARUDQP_1V8_SF_LIN
                                 IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
```

```
pad_sin_1_4             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_5             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_6             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_7             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_8             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_9             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_10            BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                d, n
pad_sin_1_11            BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
pad_sin_2_0              BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                         4480.000000
                                                 d, n
pad_sin_2_1             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_2             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_3             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_4             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_5             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_6             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
                                                 d, n
pad_sin_2_7             BD2SCARUDQP_1V8_SF_LIN
                                IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                        4480.000000
```

```
pad_sin_2_8              BD2SCARUDQP_1V8_SF_LIN
                                        IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                                      4480.000000
                                                                d, n
pad_sin_2_9              BD2SCARUDQP_1V8_SF_LIN
                                        IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                                      4480.000000
                                                                d, n
pad_sin_2_10             BD2SCARUDQP_1V8_SF_LIN
                                        IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                                      4480.000000
                                                                d, n
pad_sin_2_11             BD2SCARUDQP_1V8_SF_LIN
                                        IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
                                                      4480.000000
                                                                d, n
parabolic_top            parabolic_senthesis          226893.799918
                                                                h, n
------------------------------------------------------------------------------
Total 51 cells                                        450893.799918
```

Table III: Components in the critical path with individual delays (ASIC)

```
Operating Conditions: nom_1.00V_1.80V_25C   Library: IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
Wire Load Model Mode: top

  Startpoint: theta_f_in_pad[0]
              (input port)
  Endpoint: sin_2_pad[11]
            (output port)
  Path Group: (none)
  Path Type: max

  Point                                                    Incr       Path
  ---------------------------------------------------------------------------
  input external delay                                     0.00       0.00 f
  theta_f_in_pad[0] (inout)                                0.00       0.00 f
  InPad_0/IO (BD2SCARUDQP_1V8_SF_LIN)                      0.00       0.00 f
  InPad_0/ZI (BD2SCARUDQP_1V8_SF_LIN)                      0.95       0.95 f
  parabolic_top/theta_f_in[0] (parabolic_senthesis)        0.00       0.95 f
  parabolic_top/InPad_0/IO (BD2SCARUDQP_1V8_SF_LIN)        0.00       0.95 f
  parabolic_top/InPad_0/ZI (BD2SCARUDQP_1V8_SF_LIN)        0.74       1.69 f
  parabolic_top/Square/in1_square[0] (squarrer_block_I_square11_O_square11)
                                                           0.00       1.69 f
  parabolic_top/Square/mult_54/a[0] (squarrer_block_I_square11_O_square11_DW_mult_tc_1_0)
                                                           0.00       1.69 f
  parabolic_top/Square/mult_54/U70/Z (HS65_LH_IVX9)        0.08       1.77 r
  parabolic_top/Square/mult_54/U51/Z (HS65_LH_NOR2X6)      0.05       1.82 f
  parabolic_top/Square/mult_54/U10/S0 (HS65_LH_HA1X4)      0.14       1.96 r
  parabolic_top/Square/mult_54/product[2] (squarrer_block_I_square11_O_square11_DW_mult_tc_1_0)
                                                           0.00       1.96 r
  parabolic_top/Square/out_square[2] (squarrer_block_I_square11_O_square11)
                                                           0.00       1.96 r
  parabolic_top/Second_subs/in2_subt_one[2] (subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10)
```

```
                                                  0.00       1.96 r
parabolic_top/Second_subs/sub_59/B[2] (subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10_DW01_sub_0)
                                                  0.00       1.96 r
parabolic_top/Second_subs/sub_59/U6/Z (HS65_LH_IVX9)
                                                  0.03       1.98 f
parabolic_top/Second_subs/sub_59/U2_2/CO (HS65_LH_FA1X4)
                                                  0.15       2.13 f
parabolic_top/Second_subs/sub_59/U2_3/S0 (HS65_LH_FA1X4)
                                                  0.22       2.35 f
parabolic_top/Second_subs/sub_59/DIFF[3] (subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10_DW01_sub_0)
                                                  0.00       2.35 f
parabolic_top/Second_subs/out_subt_one[3] (subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10)
                                                  0.00       2.35 f
parabolic_top/mcm_unit/In_mcm[3] (mcm_block)      0.00       2.35 f
parabolic_top/mcm_unit/U79/Z (HS65_LH_IVX2)       0.12       2.47 r
parabolic_top/mcm_unit/U78/Z (HS65_LH_NOR2AX3)    0.08       2.55 f
parabolic_top/mcm_unit/U77/Z (HS65_LH_PAOI2X1)    0.11       2.66 r
parabolic_top/mcm_unit/U75/Z (HS65_LH_IVX2)       0.12       2.77 f
parabolic_top/mcm_unit/U74/Z (HS65_LH_AOI12X2)    0.09       2.86 r
parabolic_top/mcm_unit/U73/Z (HS65_LH_AOI12X2)    0.07       2.93 f
parabolic_top/mcm_unit/U72/Z (HS65_LH_PAOI2X1)    0.12       3.05 r
parabolic_top/mcm_unit/U70/Z (HS65_LH_IVX2)       0.14       3.19 f
parabolic_top/mcm_unit/U69/Z (HS65_LH_AOI12X2)    0.11       3.30 r
parabolic_top/mcm_unit/U68/Z (HS65_LH_AOI12X2)    0.07       3.37 f
parabolic_top/mcm_unit/U67/Z (HS65_LH_PAOI2X1)    0.22       3.59 r
parabolic_top/mcm_unit/U64/Z (HS65_LH_AO12X4)     0.19       3.78 r
parabolic_top/mcm_unit/U63/Z (HS65_LH_OAI12X2)    0.10       3.88 f
parabolic_top/mcm_unit/U60/Z (HS65_LH_NOR2AX3)    0.15       4.03 f
parabolic_top/mcm_unit/U59/Z (HS65_LHS_XNOR2X3)   0.11       4.14 r
parabolic_top/mcm_unit/U58/Z (HS65_LH_IVX2)       0.09       4.23 f
parabolic_top/mcm_unit/U13/Z (HS65_LH_OAI12X2)    0.08       4.31 r
parabolic_top/mcm_unit/U12/Z (HS65_LH_OAI12X2)    0.13       4.45 f

 parabolic_top/mcm_unit/U9/Z (HS65_LH_OA12X4)       0.16       4.61 f
 parabolic_top/mcm_unit/U7/Z (HS65_LH_AOI12X2)      0.10       4.71 r
 parabolic_top/mcm_unit/U6/Z (HS65_LHS_XOR2X3)      0.18       4.89 f
 parabolic_top/mcm_unit/out1_mcm[6] (mcm_block)     0.00       4.89 f
 parabolic_top/First_Adder/in1_adder_one[6] (adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12)
                                                    0.00       4.89 f
 parabolic_top/First_Adder/add_57/A[6] (adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12_DW01_add_0)
                                                    0.00       4.89 f
 parabolic_top/First_Adder/add_57/U1_6/CO (HS65_LH_FA1X4)
                                                    0.20       5.09 f
 parabolic_top/First_Adder/add_57/U1_7/CO (HS65_LH_FA1X4)
                                                    0.15       5.24 f
 parabolic_top/First_Adder/add_57/U1_8/CO (HS65_LH_FA1X4)
                                                    0.15       5.39 f
 parabolic_top/First_Adder/add_57/U1_9/CO (HS65_LH_FA1X4)
                                                    0.15       5.54 f
 parabolic_top/First_Adder/add_57/U1_10/S0 (HS65_LH_FA1X4)
                                                    0.23       5.77 r
 parabolic_top/First_Adder/add_57/SUM[10] (adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12_DW01_add_0)
                                                    0.00       5.77 r
 parabolic_top/First_Adder/out_adder_one[10] (adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12)
                                                    0.00       5.77 r
 parabolic_top/First_multi/in1_mult[10] (mult_block_I1_mult12_I2_mult12_O_mult12_0)
                                                    0.00       5.77 r
 parabolic_top/First_multi/mult_64/a[10] (mult_block_I1_mult12_I2_mult12_O_mult12_0_DW_mult_tc_1_0)
                                                    0.00       5.77 r
 parabolic_top/First_multi/mult_64/U190/Z (HS65_LH_IVX9)
                                                    0.03       5.80 f
 parabolic_top/First_multi/mult_64/U70/Z (HS65_LH_NOR2X6)
                                                    0.04       5.84 r
```

```
  parabolic_top/First_multi/mult_64/U31/S0 (HS65_LH_HA1X4)
                                                  0.16       5.99 r
  parabolic_top/First_multi/mult_64/U27/S0 (HS65_LH_FA1X4)
                                                  0.24       6.24 r
  parabolic_top/First_multi/mult_64/U24/S0 (HS65_LH_FA1X4)
                                                  0.24       6.48 r
  parabolic_top/First_multi/mult_64/U23/S0 (HS65_LH_FA1X4)
                                                  0.23       6.71 r
  parabolic_top/First_multi/mult_64/U3/S0 (HS65_LH_FA1X4)
                                                  0.25       6.96 r
  parabolic_top/First_multi/mult_64/product[10] (mult_block_I1_mult12_I2_mult12_O_mult12_0_DW_mult_tc_1_0)
                                                  0.00       6.96 r
  parabolic_top/First_multi/out_mult[10] (mult_block_I1_mult12_I2_mult12_O_mult12_0)
                                                  0.00       6.96 r
  parabolic_top/twos_compliment_sin/in_twos[10] (twos_comp_I_twos12_1)
                                                  0.00       6.96 r
  parabolic_top/twos_compliment_sin/U23/Z (HS65_LH_OA12X4)
                                                  0.10       7.06 r
  parabolic_top/twos_compliment_sin/U22/Z (HS65_LHS_XOR2X3)
                                                  0.11       7.17 f
  parabolic_top/twos_compliment_sin/out_twos[11] (twos_comp_I_twos12_1)
                                                  0.00       7.17 f
  parabolic_top/Fourth_mux/in2_mux3[11] (mux2by1_3_I1_mux312_I2_mux312_O_mux312_1)
                                                  0.00       7.17 f
  parabolic_top/Fourth_mux/U10/Z (HS65_LH_MUX21X4)        0.19       7.36 f
  parabolic_top/Fourth_mux/out_mux3[11] (mux2by1_3_I1_mux312_I2_mux312_O_mux312_1)
                                                  0.00       7.36 f
  parabolic_top/pad_sin_2_11/IO (BD2SCARUDQP_1V8_SF_LIN)
                                                  1.88       9.24 f
  parabolic_top/sin_2[11] (parabolic_senthesis)           0.00       9.24 f
  pad_sin_2_11/IO (BD2SCARUDQP_1V8_SF_LIN)                2.08      11.33 f
  sin_2_pad[11] (inout)                                   0.00      11.33 f
  data arrival time                                                 11.33
```

Table IV: Area Report (ASIC)

```
Number of ports:                50
Number of nets:                102
Number of cells:                51
Number of references:            2

Combinational area:       2893.799918
Noncombinational area:    448000.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          450893.799918
```

Table V: Implemented Arithmetic blocks (ASIC)

top_parabolic_senthesis	
BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
parabolic_senthesis	
BD2SCARUDQP_1V8_SF_LIN	IO65LPHVT_SF_1V8_50A_7M4X0Y2Z
HS65_LH_IVX2	CORE65LPHVT
XOR_block	
HS65_LHS_XOR2X3	CORE65LPHVT
adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12	
adder_block_one_I1_adder_one9_I2_adder_one12_O_adder_one12_DW01_add_0	
HS65_LHS_XOR2X6	CORE65LPHVT
HS65_LHS_XOR3X2	CORE65LPHVT
HS65_LH_AND2X4	CORE65LPHVT
HS65_LH_FA1X4	CORE65LPHVT
adder_block_two_I1_adder_two9_I2_adder_two12_O_adder_two12	
adder_block_two_I1_adder_two9_I2_adder_two12_O_adder_two12_DW01_add_0	
HS65_LHS_XOR2X6	CORE65LPHVT
HS65_LHS_XOR3X2	CORE65LPHVT
HS65_LH_AND2X4	CORE65LPHVT
HS65_LH_FA1X4	CORE65LPHVT
mcm_block	
HS65_LHS_XNOR2X3	CORE65LPHVT
HS65_LHS_XOR2X3	CORE65LPHVT
HS65_LHS_XOR3X2	CORE65LPHVT
HS65_LH_AOI12X2	CORE65LPHVT
HS65_LH_AOI32X3	CORE65LPHVT
HS65_LH_IVX2	CORE65LPHVT
HS65_LH_NAND2X2	CORE65LPHVT
HS65_LH_NOR2AX3	CORE65LPHVT
HS65_LH_OA12X4	CORE65LPHVT
HS65_LH_OAI12X2	CORE65LPHVT
HS65_LH_OR2X4	CORE65LPHVT
HS65_LH_PAO2X4	CORE65LPHVT
HS65_LH_PAOI2X1	CORE65LPHVT
mcm_block_DW01_add_2	
HS65_LHS_XOR2X6	CORE65LPHVT
HS65_LH_AO12X9	CORE65LPHVT
HS65_LH_FA1X4	CORE65LPHVT
HS65_LH_OA112X9	CORE65LPHVT
HS65_LH_PAO2X9	CORE65LPHVT
mult_block_I1_mult12_I2_mult12_O_mult12_0	
mult_block_I1_mult12_I2_mult12_O_mult12_0_DW_mult_tc_1_0	r
HS65_LHS_XOR2X6	CORE65LPHVT
HS65_LHS_XOR3X2	CORE65LPHVT
HS65_LH_FA1X4	CORE65LPHVT
HS65_LH_HA1X4	CORE65LPHVT
HS65_LH_IVX9	CORE65LPHVT
HS65_LH_NAND2AX7	CORE65LPHVT
HS65_LH_NAND2X7	CORE65LPHVT
HS65_LH_NOR2X6	CORE65LPHVT
mult_block_I1_mult12_I2_mult12_O_mult12_1	
mult_block_I1_mult12_I2_mult12_O_mult12_1_DW_mult_tc_1_0	r
HS65_LHS_XOR2X6	CORE65LPHVT
HS65_LHS_XOR3X2	CORE65LPHVT
HS65_LH_FA1X4	CORE65LPHVT
HS65_LH_HA1X4	CORE65LPHVT
HS65_LH_IVX9	CORE65LPHVT
HS65_LH_NAND2X7	CORE65LPHVT
HS65_LH_NOR2X6	CORE65LPHVT
mux2by1_1_I1_mux112_I2_mux111_O_mux112	
HS65_LH_MUX21X4	CORE65LPHVT
mux2by1_2_I1_mux212_I2_mux211_O_mux212	
HS65_LH_MUX21X4	CORE65LPHVT

```
mux2by1_3_I1_mux312_I2_mux312_O_mux312_0
    HS65_LH_MUX21X4                                                    CORE65LPHVT
mux2by1_3_I1_mux312_I2_mux312_O_mux312_1
    HS65_LH_MUX21X4                                                    CORE65LPHVT
squarrer_block_I_square11_O_square11
    squarrer_block_I_square11_O_square11_DW_mult_tc_1_0                          r
        HS65_LHS_XOR2X6                                                CORE65LPHVT
        HS65_LHS_XOR3X2                                                CORE65LPHVT
        HS65_LH_AND2X4                                                 CORE65LPHVT
        HS65_LH_FA1X4                                                  CORE65LPHVT
        HS65_LH_HA1X4                                                  CORE65LPHVT
        HS65_LH_IVX9                                                   CORE65LPHVT
        HS65_LH_NOR2X6                                                 CORE65LPHVT
subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10
    subtractor_block_one_I1_subt_one11_I2_subt_one11_O_subt_one10_DW01_sub_0
        HS65_LHS_XNOR2X6                                               CORE65LPHVT
        HS65_LHS_XNOR3X2                                               CORE65LPHVT
        HS65_LH_FA1X4                                                  CORE65LPHVT
        HS65_LH_IVX9                                                   CORE65LPHVT
        HS65_LH_OR2X9                                                  CORE65LPHVT
subtractor_block_two_I1_subt_two11_O_subt_two12
    HS65_LHS_XNOR2X3                                                   CORE65LPHVT
    HS65_LHS_XOR2X3                                                    CORE65LPHVT
    HS65_LH_AO12X4                                                     CORE65LPHVT
    HS65_LH_IVX2                                                       CORE65LPHVT
    HS65_LH_NAND2X2                                                    CORE65LPHVT
    HS65_LH_NOR2X2                                                     CORE65LPHVT
    HS65_LH_OAI12X2                                                    CORE65LPHVT
twos_comp_I_twos12_0
    HS65_LHS_XNOR2X3                                                   CORE65LPHVT
    HS65_LHS_XOR2X3                                                    CORE65LPHVT
    HS65_LH_IVX2                                                       CORE65LPHVT
    HS65_LH_NAND2AX4                                                   CORE65LPHVT
    HS65_LH_NAND2X2                                                    CORE65LPHVT
    HS65_LH_NOR2AX3                                                    CORE65LPHVT
    HS65_LH_NOR2X2                                                     CORE65LPHVT
    HS65_LH_NOR3X1                                                     CORE65LPHVT
    HS65_LH_NOR4ABX2                                                   CORE65LPHVT
    HS65_LH_OA12X4                                                     CORE65LPHVT
    HS65_LH_OAI12X2                                                    CORE65LPHVT
twos_comp_I_twos12_1
    HS65_LHS_XNOR2X3                                                   CORE65LPHVT
    HS65_LHS_XOR2X3                                                    CORE65LPHVT
    HS65_LH_IVX2                                                       CORE65LPHVT
    HS65_LH_NAND2AX4                                                   CORE65LPHVT
    HS65_LH_NAND2X2                                                    CORE65LPHVT
    HS65_LH_NOR2AX3                                                    CORE65LPHVT
    HS65_LH_NOR2X2                                                     CORE65LPHVT
    HS65_LH_NOR3X1                                                     CORE65LPHVT
    HS65_LH_NOR4ABX2                                                   CORE65LPHVT
    HS65_LH_OA12X4                                                     CORE65LPHVT
    HS65_LH_OAI12X2                                                    CORE65LPHVT
```

Table VI: Prime Time Script

```
start_gui
remove_design -all
set power_enable_analysis  true
set search_path "$env(STM065_DIR)/IO65LPHVT_SF_1V8_50A_7M4X0Y2Z_7.0/libs \
          $env(STM065_DIR)/CORE65LPHVT_5.1/libs \
          $env(STM065_DIR)/CORE65LPSVT_5.1/libs \
          $search_path"

set link_library "* IO65LPHVT_SF_1V8_50A_7M4X0Y2Z_nom_1.00V_1.80V_25C.db \
            CORE65LPHVT_nom_1.20V_25C.db  CORE65LPSVT_nom_1.20V_25C.db"

set target_library "IO65LPHVT_SF_1V8_50A_7M4X0Y2Z_nom_1.00V_1.80V_25C.db \
              CORE65LPHVT_nom_1.20V_25C.db CORE65LPSVT_nom_1.20V_25C.db "
```

Table VII: Power analysis obtained from Prime Time

Synthesized Area	Synthesized Time 5ns			
	Net switching power	Cell Internal power	Cell Leakage power	Total Power
None (High Speed)	8.95e-5	1.0e-4	4.61e-8	1.9e-4
0 (Minimum Area)	9.42e-5	1.0e-4	4.22e-8	1.95e-4
8592	8.20e-5	9.4e-5	4.07e-8	1.72e-4

Table VIII: Timing report from FPGA

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 157937947 / 36
-------------------------------------------------------------------------
Delay:               20.496ns (Levels of Logic = 21)
  Source:            theta_f_in<1> (PAD)
  Destination:       sin_2<11> (PAD)

  Data Path: theta_f_in<1> to sin_2<11>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             10   0.878   0.511  theta_f_in_1_IBUF (First_subs/Madd_temp1_signal_sub2)
     MULT18X18:A1->P17      1   2.413   0.430  Square/Mmult_temp_squar_mult0000 (sq_s2_sig<6>)
     LUT2:I1->O             1   0.275   0.000  Second_subs/Msub_temp1_signal_sub1_Madd_lut<6> (Second_subs/Msub_temp1_signal_sub1_Madd_lut<6>)
     MUXCY:S->O             1   0.334   0.000  Second_subs/Msub_temp1_signal_sub1_Madd_cy<6> (Second_subs/Msub_temp1_signal_sub1_Madd_cy<6>)
     XORCY:CI->O           18   0.708   0.691  Second_subs/Msub_temp1_signal_sub1_Madd_xor<7> (s2_mcm_sig<7>)
     LUT2:I1->O             2   0.275   0.476  mcm_unit/Madd_out_c1_addsub0000_Madd_lut<3> (mcm_unit/Madd_out_c1_addsub0000_Madd_lut<3>)
     LUT3:I1->O             1   0.275   0.000  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<1>1_F (N12)
     MUXF5:I0->O            2   0.303   0.396  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<1>1 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<1>)
     LUT3:I2->O             3   0.275   0.415  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<2>11 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<2>)
     LUT3:I2->O             8   0.275   0.614  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<3>11 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_cy<3>)
     LUT4:I0->O             1   0.275   0.429  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>126 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>12
     LUT4:I1->O             1   0.275   0.429  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>149 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>14
     LUT2:I1->O             1   0.275   0.000  mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>158 (mcm_unit/Madd_Out2_mcm_addsub0001_Madd_xor<7>15
     MUXCY:S->O             1   0.334   0.000  mcm_unit/Madd_Out2_mcm_add0000_cy<7> (mcm_unit/Madd_Out2_mcm_add0000_cy<7>)
     XORCY:CI->O            2   0.708   0.378  mcm_unit/Madd_Out2_mcm_add0000_xor<8> (mcm_m12_sig<8>)
     MULT18X18:B8->P18      3   2.506   0.397  First_multi/Mmult_temp_signal_mult (m1_t1_sig<10>)
     INV:I->O               1   0.275   0.000  twos_compliment_sin/Madd_temp1_not0000<10>1_INV_0 (twos_compliment_sin/Madd_temp1_not0000<10>)
     MUXCY:S->O             0   0.334   0.000  twos_compliment_sin/Madd_temp1_cy<10> (twos_compliment_sin/Madd_temp1_cy<10>)
     XORCY:CI->O            1   0.708   0.430  twos_compliment_sin/Madd_temp1_xor<11> (twos_compliment_sin/temp1<11>)
     LUT3:I1->O             1   0.275   0.332  Fourth_mux/out_mux3<11>1 (sin_2_11_OBUF)
     OBUF:I->O                  2.592          sin_2_11_OBUF (sin_2<11>)
    ----------------------------------------
    Total                      20.496ns (14.568ns logic, 5.928ns route)
                                        (71.1% logic, 28.9% route)
```

Table IX: Macro Statistics report from FPGA

# Multipliers	: 3
11x11-bit multiplier	: 1
12x12-bit multiplier	: 2
# Adders/Subtractors	: 11
10-bit adder	: 3
10-bit subtractor	: 1
11-bit adder	: 1
12-bit adder	: 4
9-bit adder	: 2
# Xors	: 1
1-bit xor2	: 1

Table X: Cell Usage report from FPGA

# BELS	: 305
# GND	: 1
# INV	: 22
# LUT1	: 11
# LUT2	: 31
# LUT3	: 29
# LUT4	: 46
# MUXCY	: 78
# MUXF5	: 3
# MUXF6	: 1
# VCC	: 1
# XORCY	: 82
# IO Buffers	: 49
# IBUF	: 13
# OBUF	: 36
# MULTs	: 3
# MULT18X18	: 3

Table XI: Device utilization summary from FPGA

Selected Device :	2vp2fg256-7
Number of Slices:	76 out of 1408 5%
Number of 4 input LUTs:	139 out of 2816 4%
Number of IOs:	49
Number of bonded IOBs:	49 out of 140 35%
Number of MULT18X18s:	3 out of 12 25%