CHALMERS

Implementation and Optimization of High Speed Symbol Timing Recovery Algorithms

Master's Thesis in Integrated Electronic System Design

Tauseef Ahmad

Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, 2012

Implementation and Optimization of High Speed Symbol Timing Recovery Algorithms

Tauseef Ahmad

Examiner: Per Larsson-Edefors

Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, 2012
SE-412 96 Gothenburg Sweden
Telephone + 46 (0)31-772 1000

Abstract

Symbol synchronization has a cardinal role in high speed optical fiber communication systems. Accurate symbol synchronization is essential for reliable reception of data, whereas an erroneous synchronization mechanism can severely deteriorate the quality of the received signals and thus increase the bit error rate of the communication system.

In this thesis work, two feedforward (Maximum Likelihood, Oerder & Meyer) and one feedback (Gardner) symbol timing recovery algorithms are implemented on a 65-nm ASIC technology. These algorithms are specifically designed for optical fiber communication systems and modified to fulfill the throughput requirement of 112 Gbit/s. Area and power consumption are key parameters used as cost function. The analysis shows that the OM algorithm is power efficient, but occupies more area than Gardner's algorithm. Some future work optimizations have been suggested that will make the OM algorithm consume the least area and power. The ML algorithm is found to be the least effective option and recommended not to be used in power critical communication systems. Different orders of interpolation blocks are also implemented on hardware and found to be overshadowing the resources occupied by the symbol timing recovery algorithms. Comparisons are made for various clock frequencies; higher clock frequency designs are found to be more area and power efficient than lower clock frequency designs.

This master thesis report is a result of the project carried out along with two other Master thesis students, conducted in collaboration with the Computer Science and Engineering (CSE), Signals and Systems (S2) and Microtechnology and Nanoscience (MC2) departments at Chalmers.

Acknowledgments

I am sincerely thankful to Prof. Per Larsson-Edefors (CSE), Dr. Henk Wymeersch (SS2) and Dr. Pontus Johanisson (MC2) for their benevolent supervision, extensive guidance, kind support, wise suggestions, informative discussion and precious comments. I am also grateful to my family for their love, affection, patience and providing quality environment for studies.

Tauseef Ahmad

Göteborg, Sweden
June 2012

Contents

List of Figures

List of Tables

Nomenclature

ASICs	Application Specific Integrated Circuits
ASIP	Application-Specific Instruction-set Processor
BER	Bit Error Rate
CD	Chromatic Dispersion
CORDIC	COordinate Rotation DIgital Computer
FB	Feed Back
FF	Feed Forward
FPGA	Field Programmable Gate Array
GBaud	Giga Baud
IFT	Inverse Fourier Transform
IQ	In-phase Quadrature-phase
LUT	Look Up Table
MATLAB	MATrix LABoratory
ML	Maximum Likelihood
OM	Oerder & Meyr
QAM	Quadrature Amplitude Modulation
RTL	Register Transfer Level
RZ	Return to Zero
STR	Symbol Timing Recovery
VHDL	Very high speed integrated circuits Hardware Description Language

1

Introduction

Synchronization is an integral part of a communication system. Different kinds of synchronization, including carrier frequency synchronization, phase synchronization and symbol time synchronization, are done in a communication receiver for proper detection of a transmitted signal. The quality of the received signals is greatly affected by these synchronizers.

The role of the symbol synchronizer becomes even more critical in high speed fiber optical communication system. Extra signal processing is needed to cope with optical channel impairments like Polarization Mode Dispersion (PMD), Chromatic Dispersion (CD) and non-linearities. Efforts are being made to push the boundaries of digital signal processing further into the optical domain and replace the long and expensive optical filters with cheaper and power efficient digital filters [1], [2], [3]. Previously, a feasibility study was made on the implementation of an adaptive equalizer on an FPGA with a 16-QAM modulation scheme for optical communication systems [4].

A comparison of symbol timing recovery algorithms (OM and Gardner) was done for underwater acoustic data communication receivers [5]. The Gardner algorithm was also implemented and optimized for a DVB-S2 receiver on an Altera Stratix II EP2S180 FPGA [6]. As the previous projects do not address very high throughput demands, there is still a need to make similar evaluations for symbol timing recovery algorithms for optical fiber communication receivers.

1.1 Problem Description

In this thesis, focus is on symbol timing recovery (STR) for high speed fiber optical communication system with a 112 Gbit/sec data rate. A symbol timing recovery mechanism is essential for proper working of the communication system. Failure to achieve

synchronization can severely deteriorate the bit error rate (BER).

The goal of this thesis is to compare different symbol timing recovery algorithms on hardware in terms of power and area, and to give feedback to the algorithm developer [7] who can further modify the algorithm on the system level for improved hardware implementations. This thesis is intended to convey a feeling to a system developer how algorithms are converted into hardware and what are the trade offs involved during this process. This thesis will support the idea that direct conversions of algorithms are not always hardware efficient; we may need to do some modifications like converting divisions into multiplications, floating point calculations into fixed-point calculations, etc. where ever possible.

1.2 System Overview

A simulation environment is developed in MATLAB to evaluate the performance of different STR algorithms. This simulation setup also models the different channel impairments of optical fiber communication system. Two feedforward (OM and ML) and one feedback (Gardner) algorithms are developed and evaluated on this optical communication system simulator and then finally implemented on hardware.

The optical fiber simulation setup that was developed in MATLAB is outlined in Figure 1.1 [8]. This setup was established to understand the various trade offs in the system design. The yellow block in the simulation setup was replaced with different STR algorithms, and their performances were assessed in the presence of various channel impairments. These algorithms were also implemented in hardware for making a comparison of their area and power cost.

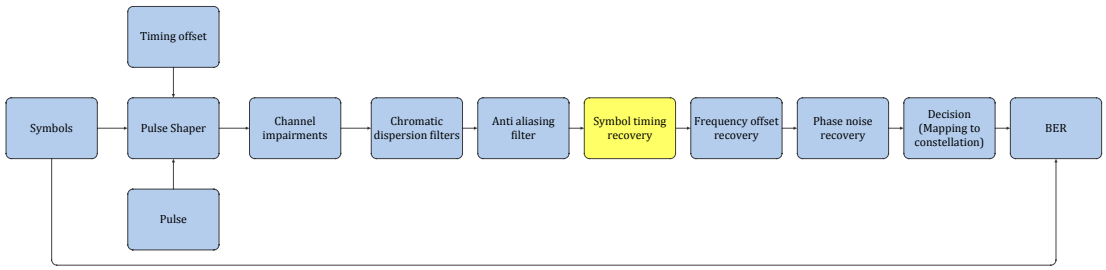

Figure 1.1: Evaluation Setup

1.2.1 System Specifications and Channel Impairments

The BER and the timing estimation error $e(n) = \tau - \hat{\tau}$ were the parameters used to compare the performance of different algorithms. The algorithms were examined in the

presence of channel disturbances like Additive White Gaussian Noise (AWGN), Chromatic Dispersion (CD), phase noise and frequency offset. The communication system under study has the following specifications [8].

- Baud rate: 28 Gbaud
- Modulation scheme: QAM
- No. of polarizations: 2
- No. of bits per symbol: 2

The bit rate for our communication system is calculated below

$$Bitrate = Baudrate \times No.of Polarization \times No.of bits per symbol. \tag{1.1a}$$
$$= 28 \times 10^9 \times 2 \times 2. \tag{1.1b}$$
$$= 112\,\text{Gbit/sec} \tag{1.1c}$$

1.3 Hardware Considerations

1.3.1 Choice of Hardware Platform

There are different kinds of platforms available for hardware implementations, including Application Specific Instruction-set Processors (ASIPs), Field Programmable Gate Arrays (FPGAs), and Application Specific Integrated Circuits (ASICs). These platforms have their own pros and cons, and the platform choice very much depends on the problem at hand.

FPGAs provide reconfigurability, high computational performance, limited development cost and short time to market, but consume more power and has less clock speed as compared to ASICs, which can deliver the highest performance and consume the least energy as compared to any other hardware platform. The downsides of ASICs are that they have high development costs and they require longest development time. Moreover, an ASIC lacks the ability of reconfiguration, so a lot of caution is needed during design, otherwise it will cost millions of dollars to fix bugs after an ASIC chip has been taped out.

Our initial plan was to implement all the three STR algorithms on an FPGA and select the best out of three to implement on ASIC. However, due to the limited number of I/O pins on the intended FPGA, the unavailability of latest models of Virtex 7 on the ISE software platform and the unavailability of the license of Virtex 7 to implement design, we chose a 65-nm ASIC platform for all our evaluations.

1.3.2 Pipelining and Parallelization

The concept of pipelining is very important for real time high speed hardware implementations. This concept enables designs to operate at a higher clock frequency, by splitting

a big logic block into two or more smaller logic blocks, which are separated by registers (flip-flops). It also means that the entire computational result is not available within one clock cycle but only a part of result is computed. The auxiliary result is stored in the flip-flops. With an increase in degree of pipelining, the latency of the design also increases.

Parallelization is another way to increase the throughput of the system. In this method, the received signals are demultiplexed and hardware is duplicated to process more data in parallel [2]. Area increases linearly with an increase in parallel streams/stages. Parallelization is an effective way to realize a high speed system on a lower clock frequency. The number of parallel streams is inversely proportional to clock the frequency.

As discussed in Section 1.2.1, the baud rate of the system under study is 28 Gbaud. To reach this high throughput, we have employed both the above mentioned methods

2

Method

Initially a prestudy was carried out, in which the Master thesis "Real-Time Signal Processing Implementation for 100 Gb/s Fiber Communication" was evaluated because of the similar nature of project [4]. The previous thesis involved the implementation of an equalizer on FPGA. After the pre-study phase, an optical communication system setup was developed in MATLAB. Different symbol timing recovery (STR) algorithms were integrated in this setup. These algorithms were step by step modified into a hardware amenable form, while keeping track of the resulting BER at the system's output. Simultaneously, block diagrams for each algorithm were developed. In the block diagram, each computation operation was mapped into a separate block for ease of testing. We started the implementation of these blocks in VHDL for hardware implementations with an approximative word length. The written VHDL codes were generic, so the fixed-point word lengths could be updated during any phase of the project. Logic simulations were made using Incisive Simulator [9] and the output of each block was verified by comparing its output with the golden reference from MATLAB. Once satisfied with the functionality, designs were synthesized in Cadence RTL compiler [10] with different timing constraints. Often VHDL codes were modified to fulfill the timing constraints set up during synthesis.

2.1 Work Flow

This project is carried out along with two other Master thesis students. The divided work flow is shown in figure 2.1.

- Ai Yun converted algorithm equations to executable MATLAB code [7].
- Pavithra Muralidharan developed a simulation environment for optical communication system in MATLAB which takes different sources of noise/impairments into consideration. The algorithms developed by Yun were inserted into this simulation

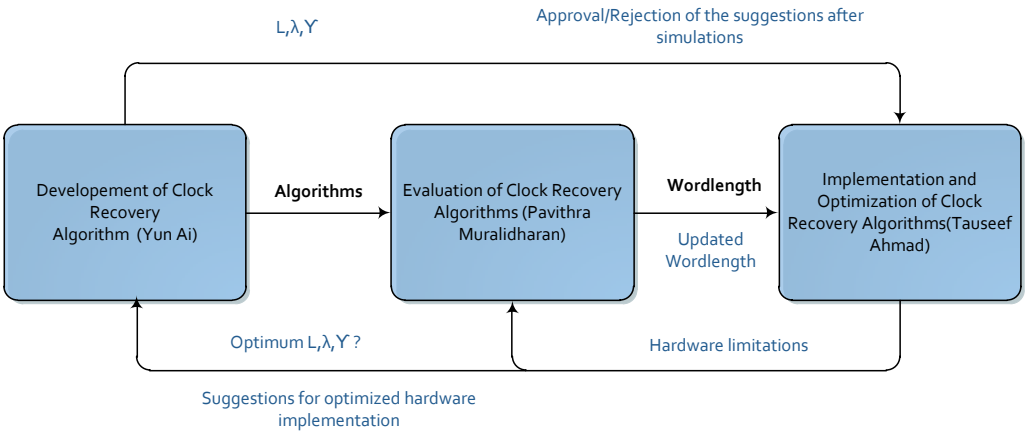

Figure 2.1: Work Flow

environment, and their performance were evaluated. Later, these algorithms were changed from floating point to fixed point for hardware implementation [8].

- The task of this thesis was to take algorithms from Ai Yun, modify the MATLAB code into more hardware friendly code, make block diagrams for hardware implementation, take fixed-point information from Pavithra Muralidharan, and implement the algorithm in VHDL. The next and final step was to analyze the area and power consumptions by these algorithms and propose the best algorithm to be used in the energy-efficient communication system.

3

Hardware Implementation

This chapter deals with implementation details of different symbol timing recovery (STR) algorithms.

3.1 Oerder & Meyr (OM) Algorithm

Oerder & Meyr (OM) is a feedforward algorithm based on heuristic reasoning [11]. This algorithm needs at least four samples per symbol for proper operation. The mathematical expression is given in equation (3.1) and block diagram is shown in figure 3.1

$$\hat{\tau} = -\frac{T}{2\pi} arg \left\{ \sum_{k=0}^{NL_0-1} |x(kT_s)|^2 \, e^{-2\pi ik/N} \right\} \tag{3.1}$$

where

- $\hat{\tau}$ is the estimated timing error
- T is the symbol time
- N is the oversampling rate, which equals 4 for the sake of implementation simplicity
- L_0 is the number of symbols used for estimation of $\hat{\tau}$
- T_s is the sampling time
- $x(k)$ is the input to the STR block

As discussed in section 1.2.1, the baud rate of the system per polarization is 28 GSymbol/s. As the OM algorithm requires four samples per symbol, the throughput

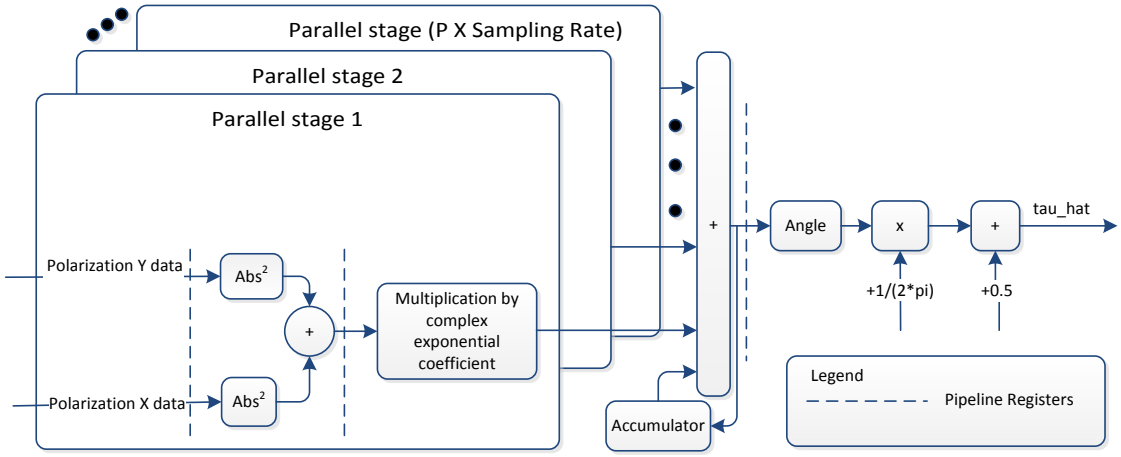

Figure 3.1: OM Block Diagram

requirement of the system becomes 28 GSymbol/s × 4 samples/symbol = 112 GSamples/s, so the algorithm must process data in parallel. These parallel streams are shown in the block diagram in figure 3.1 and expressed mathematically as

$$P \times SamplingRate = \frac{BaudRate \times SamplingRate}{Clock\ Frequency} = \frac{112\,GSample/s}{Clock\ Frequency} \tag{3.2}$$

The number of parallel streams is inversely proportional to clock frequency and decreases by increasing clock frequency. This algorithm was implemented for 2 GHz and 500 MHz clock frequencies, which requires 56 and 224 parallel streams respectively.

3.1.1 Determination of number of trial symbols L_0

The computation is based on a large number of symbols to increase the credibility of estimations and to suppress the effects of noise. Using more symbols increases the system complexity. This complexity translates into huge area and power consumption in hardware implementation. Thus, simulations were made to find the optimal number of trial symbols that could give acceptable accuracy with optimum hardware resource utilization.

Figure 3.2 shows the dependence of BER on the trial symbol count, L_0. It can be observed that L_0 has a negligible effect on BER. Although this graph is obtained for the ML algorithm, it is also valid for the OM algorithm. It can be concluded that 10 symbols are enough for determination of $\hat{\tau}$.

This result enables us to remove the dependence of parallel streams of STR algorithm on clock frequency. With an oversampling rate of four and an L_0 equal to 10, the OM

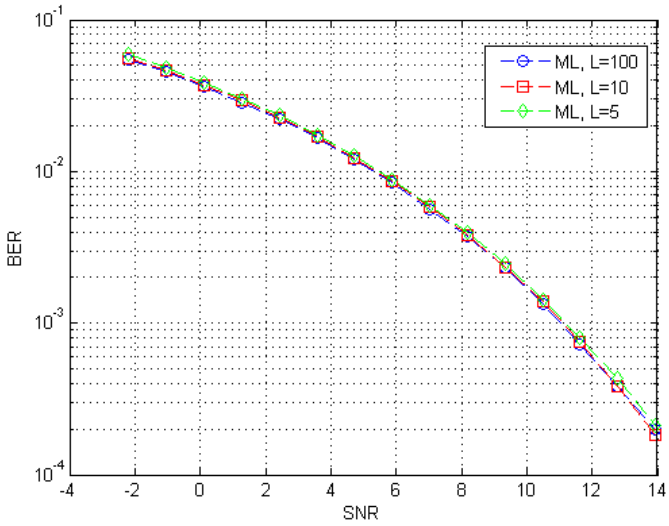

Figure 3.2: BER vs L_0

algorithm needs 40 (10×4) parallel streams for calculation of $\hat{\tau}$. Now, the OM algorithm implementation does not scale up with clock frequency and an accumulator for storing previous samples is no longer needed. A modified block diagram of the OM algorithm embracing the effects of the limited number of L_0 is shown in figure 3.3.

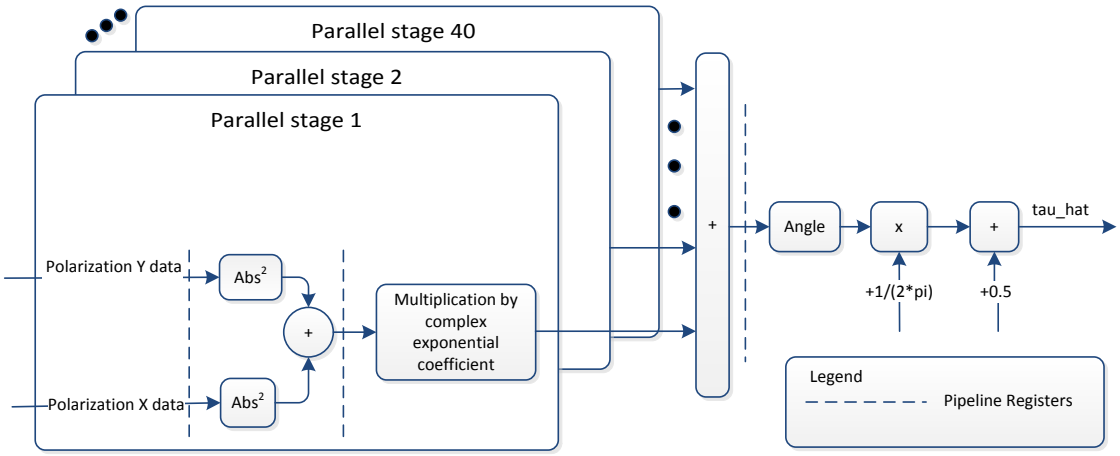

Figure 3.3: Optimized OM Block Diagram

3.1.2 Implementation details

A bottom-up approach is used for hardware implementation. The complete algorithm is divided into different modules for ease of implementation and functional verification. These modules are then written in VHDL, and their output is compared with output from MATLAB testbench. The division of the algorithm into modules is shown below.

- Absolute Square Block
- Multiplication by complex exponential coefficients $e^{-2\pi i \frac{k}{N}}$
- Cascaded Adder
- Angle Block

The absolute square block takes an I/Q input, which is complex in nature and computes a square of absolute value. The output of the 'Absolute Square Block' is fed into the 'Multiplication by $e^{-2\pi i \frac{k}{N}}$ Block' (figure 3.3). As $N = 4$ for the OM algorithm and k varies from 0 to 39, $e^{-2\pi i \frac{k}{N}}$ reduces to 1, -1, i and -i. This block does not have any multiplier. Later, outputs from all the parallel streams are added up, using a cascaded adder, and given to angle block. The output of the angle block is the estimation error $\hat{\tau}$, which is fed into interpolation blocks for symbol timing recovery of incoming samples.

Two different ways for implementation of angle block were considered.

- COordinate Rotation DIgital Computer (CORDIC) algorithm
- Look Up Table (LUT)

The CORDIC algorithm is iterative in nature, and mainly used for the calculation of trigonometric functions [12]. Its accuracy increases with each iteration. Choice of CORDIC algorithm for implementation of angle block would increase the latency (in cycles), area and power dissipation. Therefore, the LUT technique was used. LUTs are not only being used for determination of the angle but also for calculating $\frac{angle(input)+\pi}{2\pi}$. As the input to the angle block is complex data, the LUT is two dimensional. The LUT size, that is, $2^5 \times 2^5$ entries, was determined using MATLAB Fixed Point Tool Box.

3.2 Maximum Likelihood (ML) Algorithm

The Maximum Likelihood (ML) algorithm is a feedforward algorithm and needs at least two samples per symbol for proper functioning. The mathematical expression for the ML algorithm is written in equation (3.3) and the block diagram is shown in figure 3.4.

$$\hat{\tau} = -\frac{T}{2\pi} arg \left\{ \sum_{k=ND}^{N(L+D)-1} x[(k-ND)T_s]\, e^{-\frac{i\pi(k-ND)}{N}} z[(k-ND)T_s] \right\} \tag{3.3}$$

where

$$z(kT_s) = \left[x^*\left(k_2 T_s\right) e^{-\frac{\pi i k_2}{N}}\right] \otimes q(kT_s) \tag{3.4}$$

and $q(t)$ is the Inverse Fourier Transform (IFT) of

$$Q(f) = G\left(f - \frac{1}{2T}\right) G^*\left(f + \frac{1}{2T}\right) \tag{3.5}$$

and $G(f)$ is the Fourier transform of the used pulse $g(t)$. L is the length of trial signal (symbols), and N is the oversampling ratio ($\frac{T}{T_s}$). T and T_s are the symbol period and sampling period, respectively.

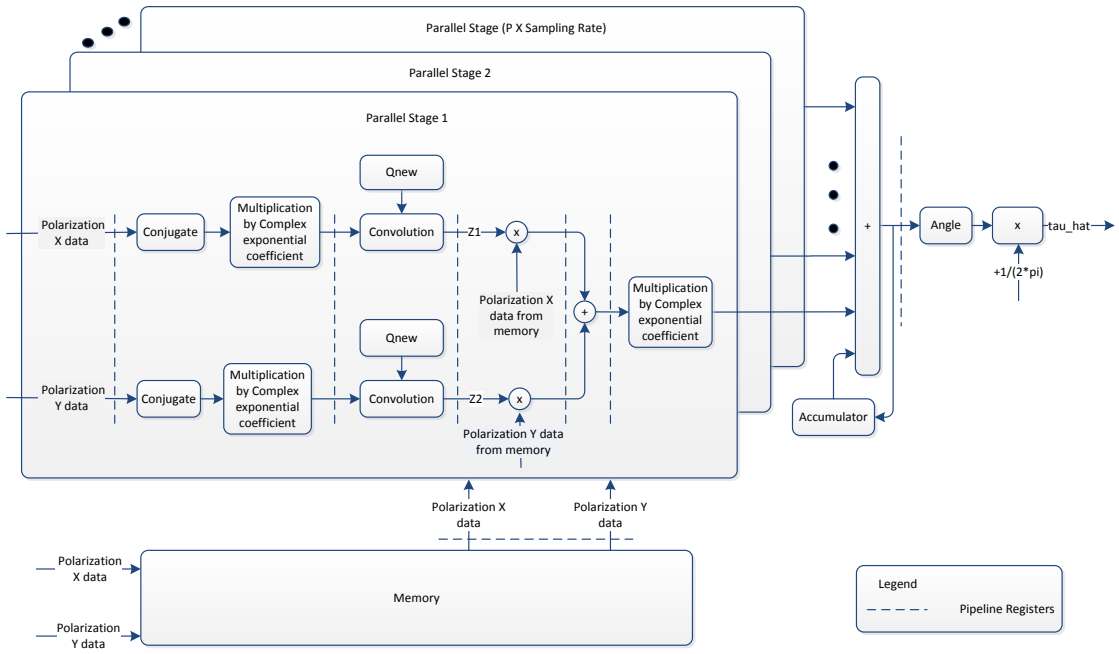

Figure 3.4: ML Block Diagram

We need 10 symbols for $\hat{\tau}$ recovery. We can make a similar optimization to ML algorithms that we made for the OM algorithm. An optimized block diagram for the ML algorithm is shown in figure 3.5.

3.2.1 Implementation of complex conjugate

The conjugate operation on a complex number $Z = a + ib$ is defined as

$$Conj(Z) = \bar{Z} = a - ib \tag{3.6}$$

In hardware, negative numbers are usually represented in two's complement form. In this format, any integer in range of -2^{n-1} to $2^{n-1} - 1$ can be represented with n bits.

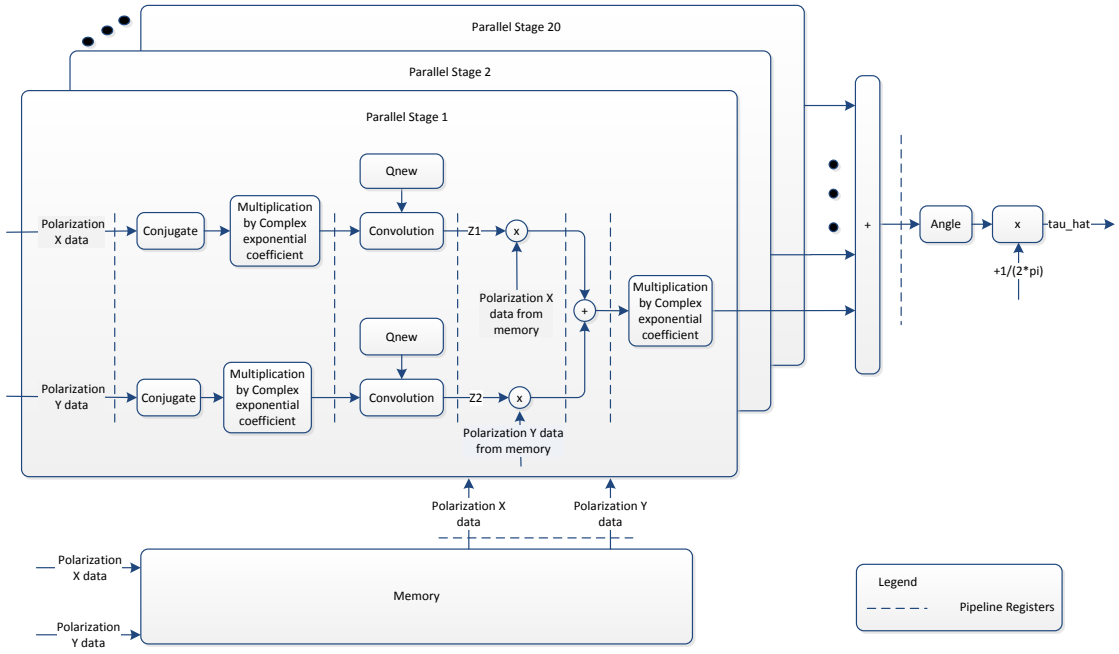

Figure 3.5: Optimized ML Block Diagram

Negation of -2^{n-1} either requires saturation of the result to $2^{n-1}-1$ or costs an extra bit. The saturation method is used in this block because, besides saving hardware, this method has a negligible effect on the final result.

3.2.2 Implementation of convolution operation

The convolution block is an implementation of equation (3.4), rewritten below for convenience.

$$z(kT_s) = \left[x^*\left(k_2T_s\right)e^{-\frac{\pi i k_2}{N}}\right] \otimes q(kT_s)$$

$\left[x^*\left(k_2T_s\right)e^{-\frac{\pi i k_2}{N}}\right]$ is output from 'Multiplication by complex exponential coefficient' block. $q(kT_s)$ is a function of pulse shaping filter used in a communication system. We chose Gaussian Return to Zero (RZ) pulse with 67% duty cycle. After selecting the pulse shaping filter, $q(t)$ reduces to a constant vector. We no longer need to calculate the IFT of $Q(f)$ defined in equation (3.5).

While converting MATLAB code to hardware implementation, the length of the $q(t)$ vector was reduced to 7 (from 15) and the word length used is 3 (determined using MATLAB Fixed Point Toolbox). As $q(t)$ is an even function, the number of multiplications were reduced from 7 to 4 owing to symmetric nature of $q(t)$. The implementation of the

convolution operation is done by writing equations in VHDL for each stage.

3.2.3 Implementation of complex multiplication

Complex multiplication is an area and power consuming operator in the ML algorithm's implementation. One stream of the ML algorithm needs two complex multiplications. The total number of parallel streams determines the total number of complex multiplications in the ML algorithm. As this algorithm needs 20 parallel streams, a total of 40 complex multipliers are required. The input to the clock recovery block is In-phase and Quadrature-phase (I/Q) data, represented by complex numbers. Two complex numbers $Z_1 = a + ib$ and $Z_2 = c + id$ when multiplied yield the result:

$$Z_1 \times Z_2 = (a + ib)(c + id) \tag{3.7a}$$
$$= (ac - bd) + i(bc + ad) \tag{3.7b}$$

A straightforward implementation of equation (3.7) in hardware requires four multipliers and two adders. As multipliers are area and power hungry, performing less multiplications in hardware implementation is desirable. Equation (3.7) can be changed to use three multipliers and five adders / subtractors.

$$\Re(Z_1 \times Z_2) = ac - bd \tag{3.8a}$$
$$\Im(Z_1 \times Z_2) = (a + b)(c + d) - ac - bd \tag{3.8b}$$

3.3 Implementation of Gardner Algorithm

Gardner algorithm is a feedback algorithm. The mathematical expression for the Gardner algorithm is shown in equation (3.9)[13] and the block diagram is presented in figure 3.6.

$$e(k) = Re\left\{r(kT - T + \hat{\tau}_{k-1}) - r(kT + \hat{\tau}_{k-1}) \times r^*(kT - \frac{T}{2} + \tau_{k-1})\right\} \tag{3.9}$$

where $e(k)$ is the output of the timing error detector. The spacing between $r(kT - T + \hat{\tau}_{k-1})$ and $r(kT + \hat{\tau}_k)$ is one symbol period and the spacing between $r(kT - T + \hat{\tau}_{k-1})$ and $r(kT - \frac{T}{2} + \hat{\tau}_{k-1})$ is $\frac{T}{2}$.

This timing error is passed through a loop filter which is a 2^{nd}-order low pass filter expressed mathematically as

$$\tau(n) = \lambda \times \tau(n-) + (1 - \lambda) \times e(n) \tag{3.10}$$

λ in equation (3.10) is known as the loop bandwidth. The choice of λ is very important because it determines the number of symbols required for convergence and error

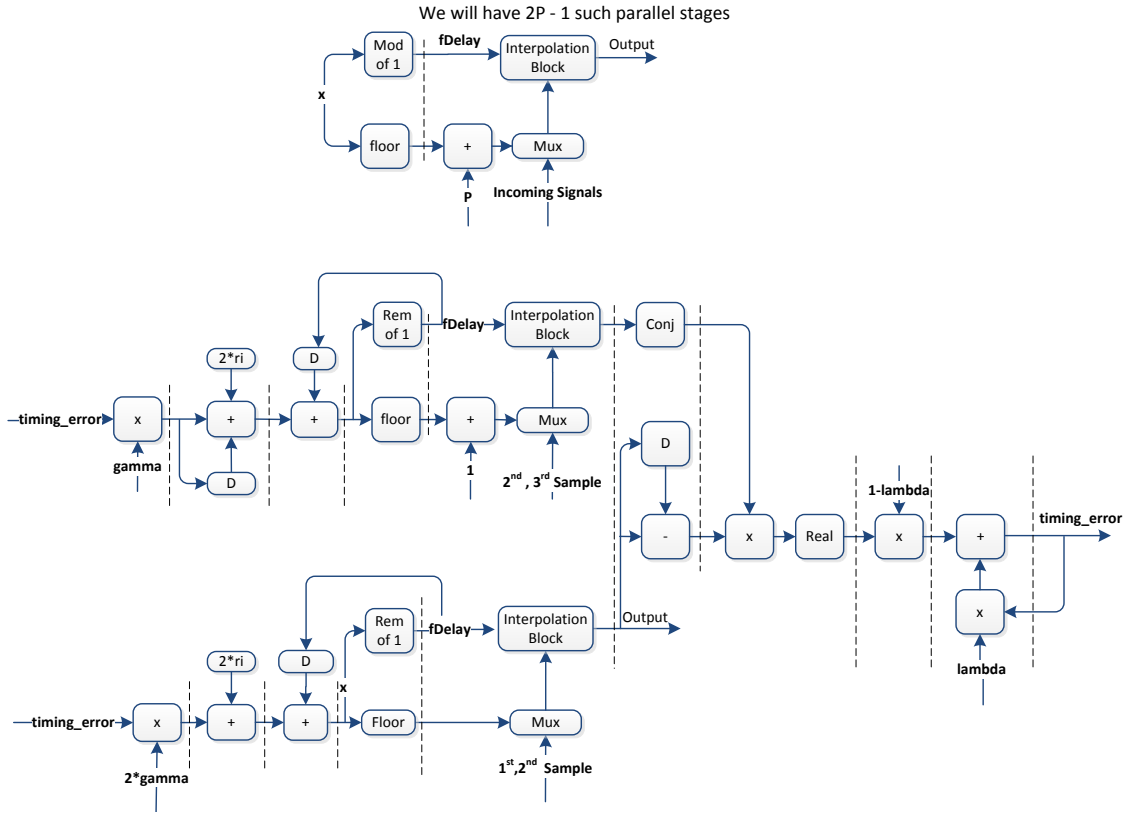

Figure 3.6: Gardner Block Diagram

variance. Figure 3.7 shows the trade off between convergence speed and stability of the timing estimation detector for different values of λ. A value of $\lambda = 0.92$ was found to be a good compromise between convergence speed and stability after convergence [7].

An optimized block diagram of the Gardner algorithm is presented in figure 3.8 utilizing the luxury of integer oversampling rate.

3.4 Implementation of Cubic Interpolator

Interpolation block takes the timing error information and the incoming samples, corrects the timing error and down samples the incoming signals into one symbol. The coefficients for cubic interpolation is given in equation (3.11) and the block diagram is shown in

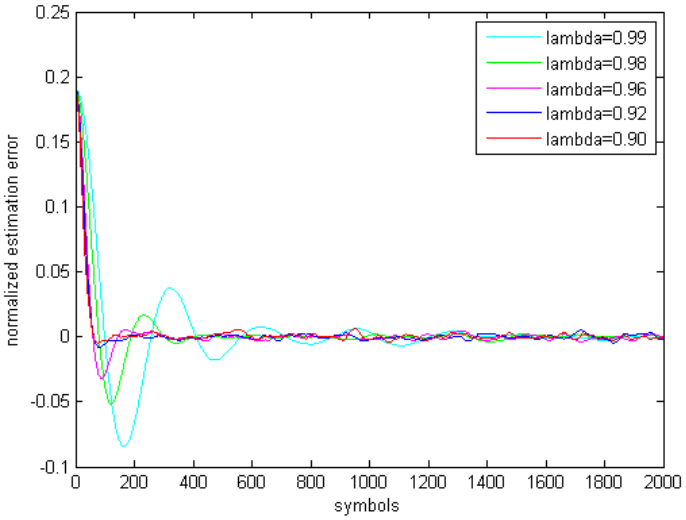

Figure 3.7: Convergence with Different Values of λ

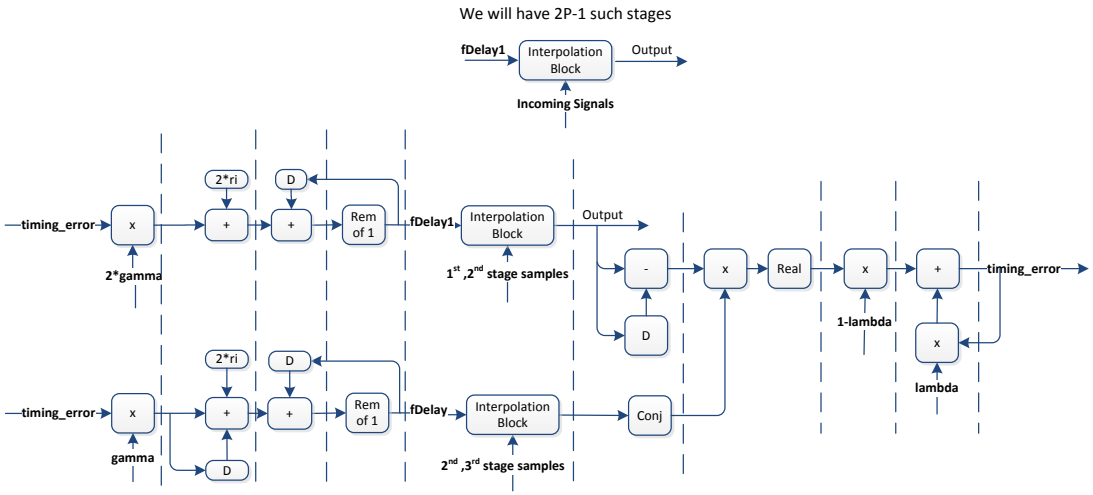

Figure 3.8: Optimized Gardner Block Diagram

figure 3.9.

$$C_{-2} = \frac{u^3 - u}{6} \tag{3.11a}$$

$$C_{-1} = \frac{-u^3 + u^2}{2} + u \tag{3.11b}$$

$$C_0 = \frac{u^3 - u}{2} - u^2 + 1 \tag{3.11c}$$

$$C_1 = \frac{-u^3}{6} + \frac{u^2}{2} - \frac{u}{3} \tag{3.11d}$$

where u is $rem(\hat{\tau},1)$ from the STR algorithms. P in block diagram is defined as

$$P = \frac{BaudRate}{ClockFrequency} \tag{3.12}$$

For hardware implementation, we have divided the computation of the coefficient's calculation into two blocks. One block calculates the square and cube of u. The second block named 'Interpolation coefficient calculator' block computes equations (3.11). Once the coefficients are calculated, they are multiplied by the arriving samples. As coefficients are real and incoming signals are complex, the cubic interpolation requires eight multipliers per stage. As our design can operate at a top frequency of 2 GHz, the interpolation block requires at least 14 parallel streams per polarization of interpolation blocks. Thus, at a 2-GHz clock frequency, in total 112 multipliers are required for each polarization. The number of parallel stages, thereby the number of multiplications, increases with a decrease in clock frequency. The results of multiplications of each stage are added and sent as a recovered symbol to the output.

3.5 Implementation of Linear Interpolator

The coefficients for the linear interpolator are given in equation (3.10) and the block diagram is shown in figure 3.10.

$$C_{-1} = u \tag{3.13a}$$

$$C_0 = 1 - u \tag{3.13b}$$

Linear interpolation is much simpler than cubic interpolation and requires four multiplications per stage. Following the same reasoning as in the cubic interpolation block, we require 56 multiplications for each polarization at an operating clock frequency of 2 GHz. The choice of either cubic or linear interpolators is a neat examples of a trade off between performance and computational overhead.

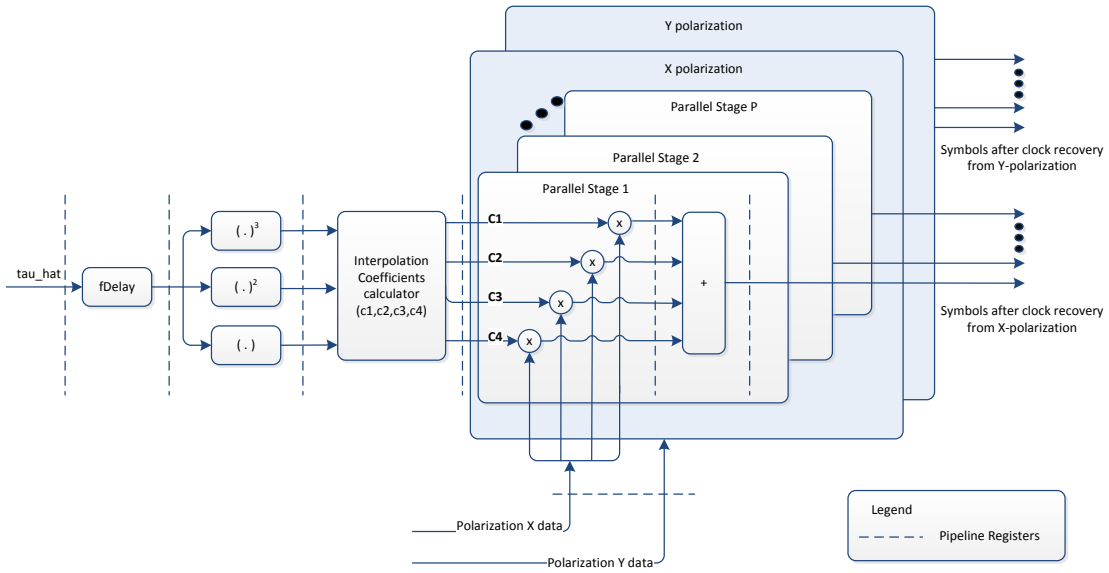

Figure 3.9: Cubic Interpolation Block

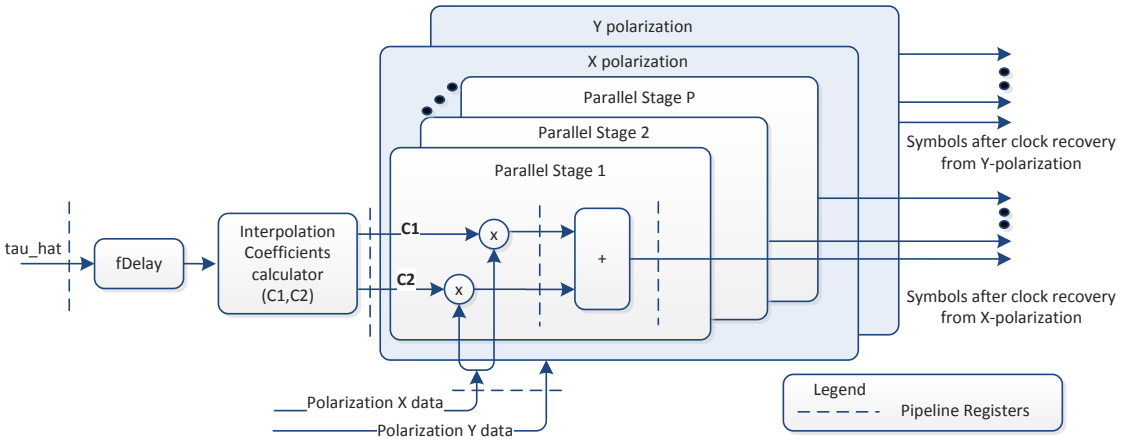

Figure 3.10: Linear Interpolation Block

4

Results

These results are obtained after synthesizing algorithms that are described in VHDL on a 65-nm CMOS technology (called the CORE65GPSVT cell library) with 0.9 volt power supply. Encounter RTL Compiler v09.10-p104_1 was used for synthesizing the design with nominal operating conditions.

4.1 Fixed Point Representation

This section deals with conversion of algorithms from floating point representation into fixed point representation. Table 4.1 and 4.2 shows the fixed point word length used for STR algorithms and interpolation blocks, respectively.

The method used for determination of fixed point word length is described below:

- An evaluation setup using a model of the system was established in MATLAB that could determine the bit error rate (BER) at the output of the communication system, for different channel impairments.
- Different STR algorithms were inserted in this evaluation setup and logically divided into different modules.
- MATLAB Fixed Point Toolbox was used for conversion of floating point format in a fixed point format. For each logical module, the word length was reduced from 64 bits to fewer bits until the BER value goes above the threshold of an acceptable BER (which is a function of the error correction that is done downstream).

Table 4.1: Fixed Point Word Lengths for OM, ML and Gardner Algorithms

Algorithm	Module	Input	Output	Rounding Mode
OM	Abs_square	3	7	FULL PRECISION
	Complex coeff	7	7	FULL PRECISION
	Cascaded adder	7	5	MSB
	Angle block	5	5	FULL PRECISION
ML	Convolution block	3	3	MSB
	Complex mul	3	7	FULL PRECISION
	Cascaded adder	7	5	MSB
	Angle block	5	5	FULL PRECISION
Gardner	Feedback loop	6	6	MSB

Table 4.2: Fixed Point Word Length for Cubic/Linear Interpolator

Module	Input	Output	Rounding Mode
Coefficient calculator	5	6	MSB
Complex multiplication and addition	6	7	MSB

4.2 Area and Power Consumption for STR Algorithms

We used a bottom-up approach for hardware design and implementation. Two different timing constraints, corresponding to 2 GHz and 500 MHz, were set to see the effect of higher and lower system clock frequencies on area and power consumption. Input vectors used for the obtaining the power values were generated using the communication system setup. As all the implemented STR algorithms include the interpolation block, a separate section explains the area and power consumption for the interpolation blocks.

4.2.1 ML Algorithm's Area and Power

Table 4.3 shows the area and power consumption for the ML algorithm. The design synthesized for the 2-GHz clock frequency requires more pipeline stages and has more latency cycles as compared to a design synthesized for 500 MHz to satisfy timing constraints. As the numbers of pipeline stages are fixed for the timing error estimation block and do not scale with clock frequency, the area and power statistics improve for 500 MHz. The angle block is implemented using LUTs. Owing to the static nature of LUTs, stricter timing constraints do not have much effect on power and area for angle block. The complex multiplication block seems to be the most area and power hungry module in the ML algorithm implementation.

Table 4.3: Area, Power and Latency (Cycles) for the ML Algorithm

ML Algorithm	Clock rate: 500 MHz			Clock rate: 2 GHz		
	Area (μm^2)	Latency (cycles)	Power (mW)	Area (μm^2)	Latency (cycles)	Power (mW)
Convolution	26850	1	4.55	39600	3	11.0
Complex multiplications	30600	1	10.7	56400	4	24.0
Cascaded adders	5519	1	0.75	7531	2	1.36
Angle	1364	1	0.18	1409	1	0.20
Memory	4186	1	0.7	13080	1	2.04
Total Area / Power	68519	-	16.88	118020	-	38.6

4.2.2 OM Algorithm's Area and Power

Table 4.4 shows the area and power consumption for the OM algorithm. As the OM algorithm is a simplification of the ML algorithm, it consumes 4.26 times less area at 2 GHz and 3.58 times less area at 500 MHz. We observe that power does not scale in the same way as area for these two algorithms, because the ML algorithms involve computationally-expensive operations (dynamic power) and need registers and standard logic cells of high driving strength (static power) to fulfil timing constraints.

Table 4.4: Area, Power and Latency (Cycles) for the OM Algorithm

OM Algorithm	Clock rate: 500 MHz			Clock rate: 2 GHz		
	Area (μm^2)	Latency (cycles)	Power (mW)	Area (μm^2)	Latency (cycles)	Power (mW)
Absolute square	12160	1	1.93	18720	1	3.31
Cascaded adders	5519	1	0.75	7531	4	1.36
Angle	1421	1	0.18	1454	1	0.21
Total Area / Power	19100	-	2.86	27705	-	4.88

4.2.3 Area and Power of the Gardner Feedback Loop

Table 4.5 shows the area and power consumption for the Gardner algorithms's feedback loop. This feedback loop will take its input from two interpolators and give its output to multiple interpolators depending on the operating clock frequency (see figure 3.6). The area decreases 1.52 times and power decreases 1.85 times when clock frequency is reduced from 2 GHz to 500 MHz.

Table 4.5: Area and Power for Gardner Feedback Loop

	Clock rate: 500 MHz		Clock rate: 2 GHz	
	Area(μm^2)	Power(mW)	Area(μm^2)	Power(mW)
Feedback loop	4692	0.74	7143	1.37

4.3 Area and Power Consumption for Interpolation blocks

Table 4.6 shows the area and power values of the cubic interpolator. A comparison between the interpolator block and the timing error estimation blocks shows that the interpolator has a dominant role in STR algorithms. It can be observed from the table that multiplications consume the majority of the area and power. The number of multiplications is directly related to the number of coefficients. In order to balance distribution of resources between the interpolator and the timing error estimator, we went for a less complex interpolator.

Table 4.7 shows the area and power values of the linear interpolator. Comparing the cubic with the linear interpolator shows that the linear interpolator consumes half of the area and power, because of the reduction of multiplications.

Table 4.6: Area and Power for Cubic Interpolator

Cubic Interpolator	Clock rate: 500 MHz		Clock rate: 2 GHz	
	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)
Complex multiplication and addition	583968	93.1448	194880	49.981736
Power block and coeff calculator	1898	0.263757	1690	0.321225
Complete block	585866	93.40	196570	50.3

Table 4.7: Area and Power for Linear Interpolator

	Clock rate: 500 MHz		Clock rate: 2 GHz	
	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)
Linear Interpolator	269360	45	88760	24

4.4 Area and Power Consumption for STR Algorithms

Table 4.8 shows the area and power consumptions for STR algorithms with different cubic and linear interpolation block. An interpolation block with constant word length was used while making these comparisons. It can be seen from the table that the least power is being consumed by the OM algorithm with the linear interpolator at a 2-GHz clock frequency. The ML algorithm is totally defeated by the other two algorithms and is not suggested to be used in area and power critical systems.

The OM algorithm is superior to Gardner's algorithm as it is power efficient and its output stabilizes quickly. Gardner's algorithm requires hundreds of cycles to lock the timing estimation error, while OM algorithm can rapidly calculate its timing estimate and provide a valid result.

Table 4.8: Area and Power for STR Algorithms with Different Interpolators

Algorithm	Interpolator	Clock rate: 500 MHz		Clock rate: 2 GHz	
		Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)
OM	Cubic	604966	96.27	224275	55.18
ML	Cubic	654385	111.71	327670	88.97
Gardner	Cubic	604478	102.75	210673	60.45
OM	Linear	288460	47.90	116465	28.90
ML	Linear	337879	61.98	206780	62.69
Gardner	Linear	276502	50.38	99073	30.24

5

Future Work

There exist significant opportunities in bringing improvements to the current design. Three suggestions for future work are presented in this chapter. The first two suggestions were not carried out because the size of symbol timing recovery blocks were negligible in comparison to the interpolation blocks. The third suggestion was not enforced to keep things simple and realizable within time.

5.1 Serializing the OM Algorithm

As we know from previous chapters, the OM algorithm needs a limited numbers of samples for calculating $\hat{\tau}$. So, instead of using 40 parallel stages, we can use a single stage, store its result in an accumulator and give its output to the angle block for calculation of $\hat{\tau}$. A block diagram to serialize the OM algorithm is shown in figure 5.1.

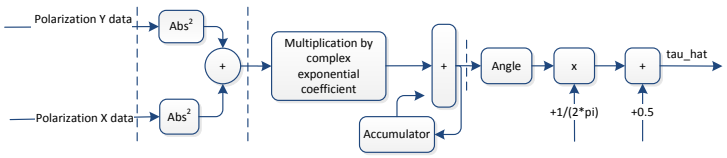

Figure 5.1: Serialized OM Block

5.2 Serializing the ML Algorithm

Similar to the OM algorithm, the ML algorithm can also be serialized and reduce area by 20 times. A block diagram to serialize ML algorithm is shown in figure 5.2.

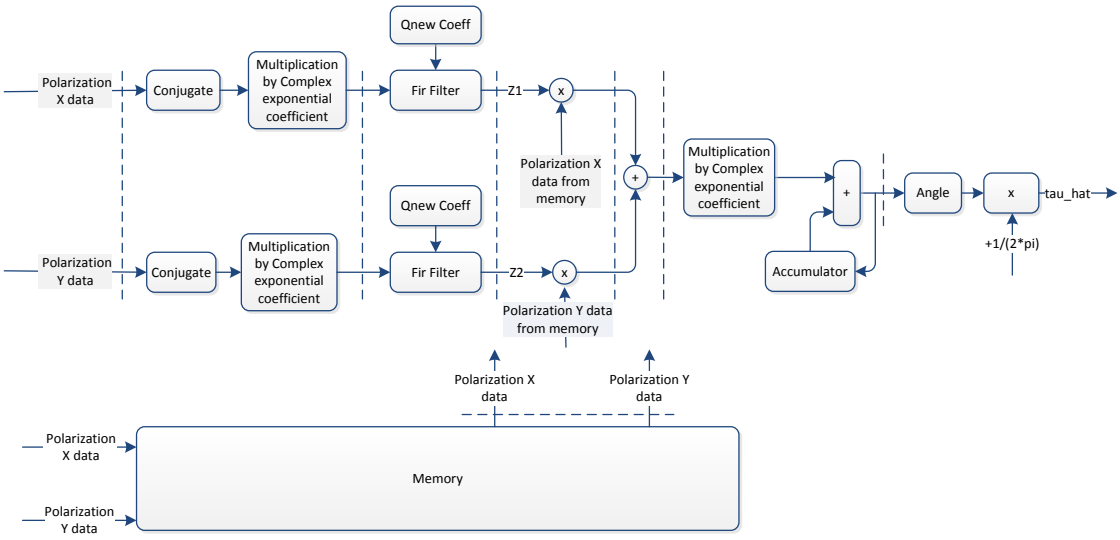

Figure 5.2: Serialized ML Block

5.3 Non-Integer (Fractional) Sampling Rate

Our present hardware designs are for fixed integer sampling rate on which we have made several simplification, for example, complex coefficient multiplications have been reduced to $1, -1, i$ and $-i$ and many simplifications have been made in the interpolation blocks. Real world communication systems are not that simple. They may take any non-integer sampling rate. So future work may include modifying the system for a generic sampling rate.

6

Conclusion

Three algorithms were successfully converted from equations to MATLAB code and then finally implemented in hardware. The BER and the timing estimation error were key parameters used for determining their performance at system level. After hardware implementation, area and power values were used as cost functions.

The Gardner algorithm occupies the least area, but the OM algorithm dissipates the least power. The OM algorithm can easily overcome this shortcoming by an optimization suggested in section 5.1. As the ML algorithm is proved to be needing the largest area and consuming the most power of the STR algorithms, we suggest not to use it in any power and area critical application. The OM algorithm is proved to be superior to the Gardner's algorithm in our evaluation setup as it is power efficient and its output stabilizes quickly. The Gardner's algorithm requires hundreds of clock cycles to lock the timing estimation error, while the OM algorithm can rapidly calculate a timing estimate and provide a valid result.

The interpolation blocks are found to dominate the hardware resources. Compared to a cubic interpolator, a linear interpolator can provide reasonable performance with approximately half the area and power consumption. It was also noted that interpolation blocks operating at higher clock frequency require less number of parallel streams which consequently results in less overall area and power consumption.

Bibliography

[1] S. J. Savory, G. Gavioli, R. I. Killey, P. Bayvel, Electronic compensation of chromatic dispersion using a digital coherent receiver, Opt. Express 15 (5) (2007) 2120–2126.

[2] T. Pfau, S. Hoffmann, R. Noe, Hardware-efficient coherent digital receiver concept with feedforward carrier recovery for M-QAM constellations, Journal of Lightwave Technology 27 (8) (2009) 989–999.

[3] S. J. Savory, Digital filters for coherent optical receivers, Opt. Express 16 (2) (2008) 804–817.

[4] F. Toft, N. Rousk, Real-Time Signal Processing Implementation for 100 Gb/s Fiber Communication, Master's thesis (Jul. 2011).

[5] G. Eynard, C. Laot, Non data aided timing recovery algorithm for digital underwater communications, in: IEEE/OES OCEANS Conference, 2007.

[6] W. Xin, Z. Ni, Optimization of FPGA design and implementation of timing recovery in DVB-S2, in: International Conference on Communications, Circuits and Systems, 2008, pp. 1265–1269.

[7] Y. Ai, Clock Recovery for High-Speed Fiber-Optic Communication Systems, Master's thesis (Jun. 2012).

[8] P. Muralidharan, Evaluation of Symbol Timing Recovery Algorithms in a High-Speed Fiber-Optic Communication System, Master's thesis (Jun. 2012).

[9] Cadence Design Systems, Inc. (Cadence), Incisive Enterprise Simulator Overview (Jun. 2009).

[10] Cadence Design Systems, Inc. (Cadence), Quick Reference for Encounter RTL Compiler (Dec. 2010).

[11] M. Oerder, H. Meyr, Digital filter and square timing recovery, IEEE Transactions on Communications 36 (5) (1988) 605–612.

[12] R. Andraka, A survey of CORDIC algorithms for FPGA based computers, in: ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays, FPGA '98, 1998, pp. 191–200.

[13] F. M. Gardner, Phaselock techniques, 3rd Edition, John Wiley, Hoboken, NJ, 2005.