University of Alberta

AREA AND POWER EFFICIENT ECHO CANCELLATION FOR HIGH-SPEED WIRE-LINE SYSTEMS

by

Saraswathi Sachidananda 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2004

Your file Votre référence
ISBN: 0-612-95844-2
Our file Notre référence
ISBN: 0-612-95844-2

Dedicated to,
My family

Acknowledgements

I would like to sincerely thank all the people who made this thesis possible.

My utmost regards to my Supervisor, Dr. Stephen Bates for his valuable guidance, patient assistance and constant encouragement. I would like to thank my colleagues Pranavi Anand and Ram Swamy for their useful feedback and suggestions. I also express my gratitude to Amir Alimohammad and Kamlesh Raiter who helped me get familiar with some of the CAD tools. I am indebted to the Department of Electrical and Computer Engineering, University of Alberta for having provided me with Research and Teaching assistantships.

Lastly, I would like to extend my gratefulness to my family and friends for all their support and encouragement.

Table of Contents

List of Figures

List of Tables

Acronyms

ADC	Analog-to-Digital Converter
ASIC	Application-Specific Integrated Circuit
CAD	Computer-Aided Design
CLB	Configurable Logic Block
DA	Design Analyzer
DAC	Digital-to-Analog Converter
DC	Design Complier
DLL	Delay-Locked Loop
DSP	Digital Signal Processing
DUT	Design Under Test
FEXT	Far-End Crosstalk
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
Gbps	Giga bits per second
GHz	Gigahertz
HDL	Hardware Description Language
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronic Engineers
IOB	Input-Output Block
ISI	Inter-Symbol Interference
LAN	Local Area Network
LMS	Least Mean Square
LSB	Least Significant Bit
LSW	Least Significant Word
LUT	Look Up Table
MBA	Modified Booth Algorithm
MHz	Megahertz
Mbps	Mega bits per second
Msps	Mega symbols per second
MSW	Most Significant Word
NEXT	Near-End Crosstalk
PAM	Pulse Amplitude Modulation
PSC	Parallel-to-Serial Converter
RAM	Random-Access Memory

RF	Radio Frequency
RTL	Register Transfer Level
UTP	Unshielded Twisted Pair
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale Integration
db	Decibel
ms	Millisecond
mw	Milliwatt
ns	Nanosecond
um^2	Micrometersquare
us	Microsecond
uw	Microwatt

Chapter 1

Introduction

1.1 Gigabit Ethernet systems

Ethernet is the dominant protocol for local area data communications. Recent growth in network traffic and the demand for greater speeds, have paved the way for faster Ethernet technologies. This chapter begins with an introduction to Ethernet systems and goes on to describe the migration from Ethernet to Gigabit Ethernet. We introduce the 1000BASE-T Gigabit Ethernet system in terms of its advantages, channel model and throughput. We then discuss the noise sources in full-duplex high-speed wire-line systems and finally define the channel impairment that we are trying to minimize. This chapter thus presents a brief introduction to the thesis.

1.1.1 Ethernet to Gigabit Ethernet

Ethernet protocols are based on the specifications in the IEEE 802.3 standard [3]. Since the 1970s, *Ethernet* has been the most commonly used data exchange method in Local-Area Networks (LANs). It supports data rates of up to 1000 Mbps and is also called as the XBASE-T system.

The use of improved signal processing and VLSI techniques has resulted in steady increases in speed. The original XBASE-T system was 10BASE-T and is capable of operating at speeds of 10 Mbps. A newer version is called 100BASE-T or *Fast Ethernet*, and supports data transfer rates of 100 Mbps. The most recent version is *Gigabit Ethernet* [1] which has a speed of 1 Gigabit per second (Gbps).

Gigabit Ethernet is also termed 1000BASE-T.

The 1000BASE-T Gigabit Ethernet protocol [1] is fully compatible with the existing Ethernet protocols and was standardized in June 1998. It has a tenfold increase in speed over Fast Ethernet and is destined to play a major role in high-speed LAN backbones and server connectivity. The following section discusses the advantages of upgrading to 1000BASE-T systems.

1.1.2 Benefits of Gigabit Ethernet

The most important advantage of Gigabit Ethernet is its complete compatibility with the existing base of Ethernet and Fast Ethernet products. 10BASE-T and 100BASE-T have already proved profitable in terms of performance, reliability and network installations. Since Gigabit Ethernet standards are based on the same standards, its use becomes not only easy and convenient, but also reliable. Some of the benefits of the 1000BASE-T system are listed below:

- Category-5 (CAT-5) Unshielded Twisted Pair (UTP) copper wires, which have been used in the cabling of the existing 10BASE-T and 100BASE-T systems, are also used in the 1000BASE-T Gigabit Ethernet system. This avoids the need for re-wiring.
- 1000BASE-T is ten times faster than Fast Ethernet and one hundred times faster than regular Ethernet.
- Full duplex capacity allows data to be transmitted and received at the same time. This way, the effective bandwidth is virtually doubled. Increased bandwidth results in higher performance.

Having seen the advantages of 1000BASE-T Gigabit Ethernet, let us see how this 1000 Mbps is attained.

1.1.3 Achieving 1000 Mbps

A 1000BASE-T system [6] consists of four transceiver sections, i.e., four transmitters and four receivers; each transceiver transports data over a pair of UTP copper

cables. The symbol rate of every transceiver section is 125 Mega-symbols per second (Msps), with two bits in every symbol. Hence the data transfer rate in every cable is 250 Mbps. In the 1000BASE-T system, there are four transceiver sections and the total throughput is therefore 4×250 Mbps = 1000 Mbps.

1.1.4 Modulation scheme of 1000BASE-T

The baud rate in the 100BASE-T full duplex baseband system is 125 Mega-baud. To retain the same baud rate in 1000BASE-T systems, the number of bits encoded onto each symbol was increased. The modulation scheme used in 1000BASE-T is Pulse Amplitude Modulation-5 (PAM-5), which uses five signalling levels. Using a regular 4-dimensional modulation scheme, all possible 8-bit data words can be coded into $4^4 = 256$ unique symbols. The Gigabit Ethernet system adopts the 5-level PAM modulation scheme, and hence the number of possible symbols increases to $5^4 = 625$. This provides additional symbols for coding error control.

1.2 Channel impairments in the Gigabit Ethernet system

The main channel impairments [3] in Gigabit Ethernet transmission systems include insertion loss, interference and noise from other sources. These noise sources are shown in Figure 1.1 and are briefly discussed below:

- Insertion loss: Insertion loss occurs due to the insertion of a device within a transmission line. In the 1000BASE-T system, insertion loss occurs due to the placement of the hybrid device as shown in Figure 1.1. It is usually expressed in decibels (dB).

- Interference: Interference mainly occurs in the form of echo and unwanted signal coupling, i.e., crosstalk. These forms are described briefly.

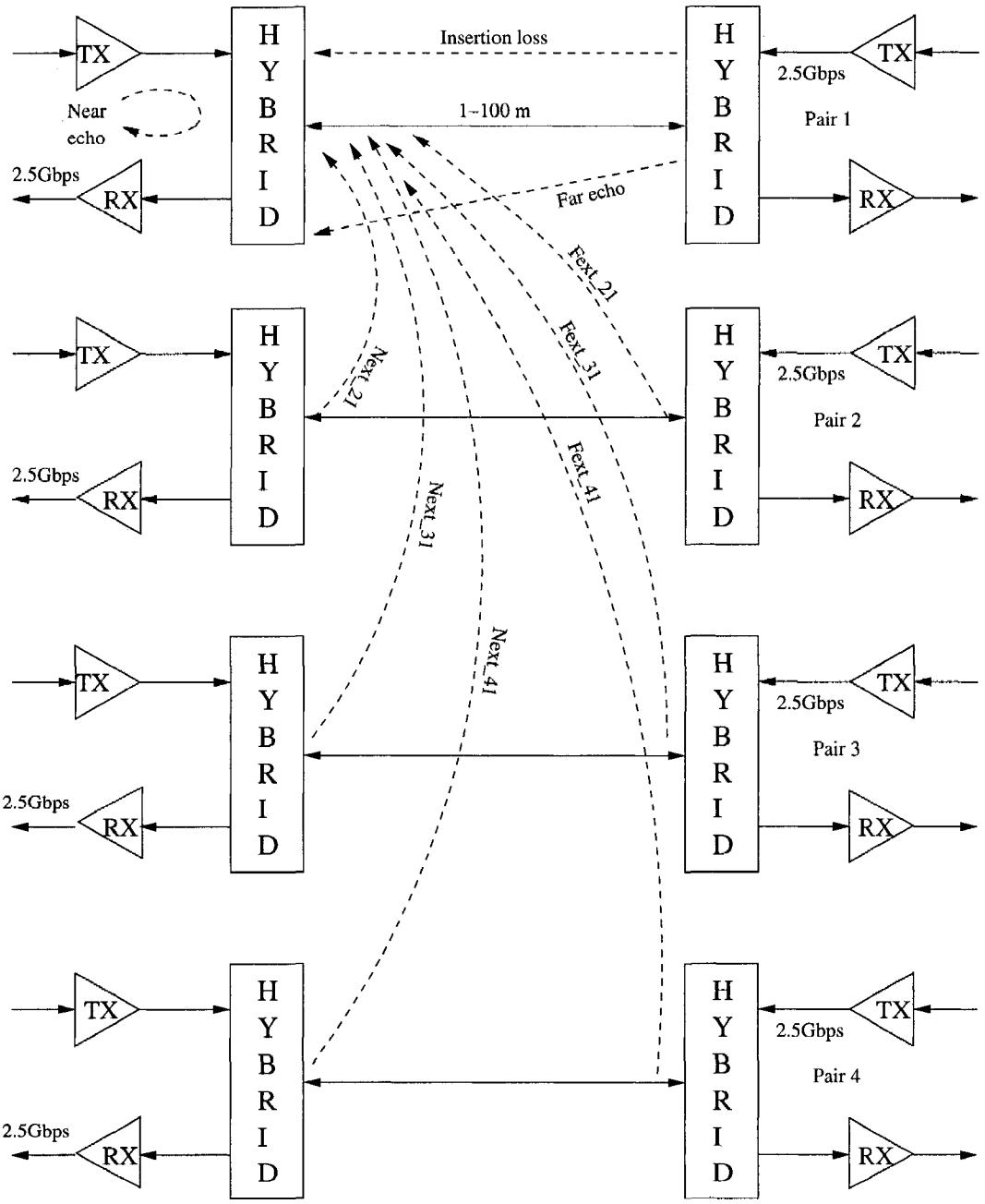

Figure 1.1: Channel model in the 1000BASE-T system

- Near-End Crosstalk (NEXT): NEXT is defined as the crosstalk that appears at the output of a wire pair, with respect to a transmitter which is at the near end of that same wire pair.
- Far-End Crosstalk (FEXT): FEXT is a measure of the crosstalk from a transmitter at the far-end, with respect to a neighboring cable measured at the near-end.
- Echo: Echo occurs due to impedance mismatches at the hybrid device. Echoes can occur at both the local hybrid and also the remote hybrid, and accordingly result in near and far echoes.

- Noise: Disturbances such as thermal noise, noise from analog components, Radio Frequency (RF) pick up, etc also contribute to channel impairments.

The channel model diagram for the full duplex Gigabit Ethernet transmission system [3], indicating all the channel impairments, is shown in Figure 1.1. We see that channel degradation takes place due to insertion loss, near and far echoes, NEXT and FEXT. We see how near echoes originate at the local hybrid and far echoes at the remote hybrid. We also see how interference occurs between the first cable and the other three cables. In the case of NEXT, the received data at the first local receiver interferes with the transmitted data of the other three local transmitters. FEXT occurs due to the interference of the received signal at the first local receiver with the remotely transmitted signals from the other three remote transmitters.

To obtain accurate data transmission, it is crucial that the channel be free from all of the above noise sources. In this thesis, we make an attempt to deal with echoes.

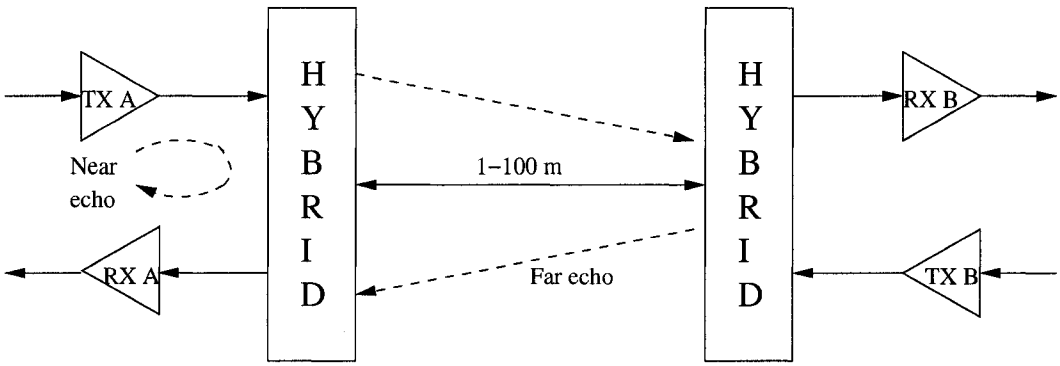

Figure 1.2: The channel model for echo in high-speed wire-line systems

1.3 The echo phenomenon

1.3.1 Echo in the channel

The channel model for high-speed wire-line systems is illustrated in Figure 1.2. It contains two transmitters and two receivers separated by a channel; in this case a cable of length 1-100 m. The cable length of 1-100 m, is as defined by the IEEE standard. Like in any full-duplex communication system, a signal enters a Gigabit Ethernet transmission system through a hybrid device. A hybrid device is responsible for the two-to-four-wire conversion. Due to impedance mismatches at this conversion junction, echo signals originate at the local transceiver side of the communication system.

A part of the signal transmitted by the local transmitter-*TX A* gets reflected into the receiver-*A* when it encounters the local hybrid. This reflected signal is termed as *near* echo. Near echo is therefore defined as the echo that is generated due to impedance mismatches at a local hybrid. As the transmitted signal travels along its data path towards receiver-*RX-B*, it encounters the remote hybrid. This results in some more reflection of the transmitted signal into the local receiver-*RX-A*, thus generating the *far* echo component. Far echo is defined as the echo that arises due to mismatches at a remote hybrid. Near and far echoes have been depicted in the channel model of Figure 1.2.

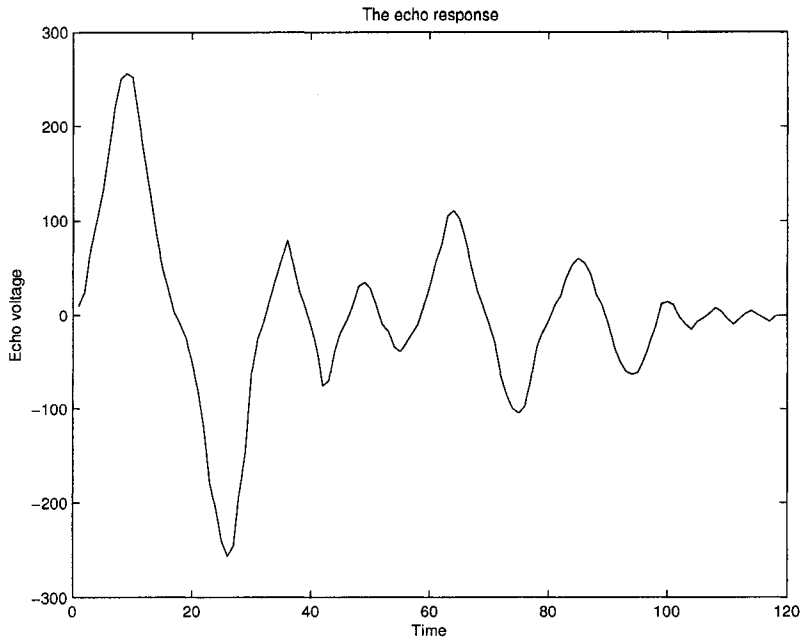

Figure 1.3: Echo response

1.3.2 The echo response

The echo response represents a variation of the echo voltage with time. This is shown in Figure 1.3. The variation is linear and constant over time, but varies in size.

Echo sizes basically translate to echo voltages. Near echo occurs due to the local hybrid which is situated just a few millimeters away from the local transceiver section. Near echo definitely occurs no matter what the cable length is. This kind of echo occurs when the transmitted signal is in its early stages of transmission and is therefore large in size. Near echoes are typified by the large initial transients in the echo response of Figure 1.3. As opposed to near echoes, far echoes occur when the locally transmitted signal meets the remote hybrid. Far echoes occur when the transmitted signal has crossed the channel. By the time the signal reaches the remote hybrid it has already been slightly attenuated. Hence, the energy of far echoes is significantly lower as compared to near echoes. Far echoes are represented in Figure 1.3, by the smaller transients which occur relatively away from the origin.

Since the classification of echoes as near and far depends on the amount of

distance travelled by the locally transmitted signal, we can say that echo size is a function of cable length. The random separation in time between echoes of varying sizes is also a function of the cable length. As defined in the IEEE standards [3], the cable length in 1000BASE-T systems is between 1 and 100 m.

Having studied echo as a channel impairment, let us now begin to look at its cancellation.

1.4 Adaptive echo cancellation

At the local receiver we are trying to receive the data sent by the remote transmitter. However, this data is mixed with the near and far echo components generated due to the encounter of the locally transmitted signal with the two hybrids. One of the important tasks in signal reconstruction is to filter out the echo components.

It is therefore crucial to incorporate an echo cancellation unit in the receiver section of a transceiver. The role of the echo canceller is to filter out the echo component and generate an adjusted output signal. The important aspect of our design is its *adaptive* nature, meaning that the echo canceller is able to filter out echoes of different sizes in a different manner. We will see in the following sections what this idea means and how our design methodology actually minimizes area and power estimates.

The block diagram of an echo canceller unit in the Gigabit Ethernet communication system is represented in Figure 1.4. Its main components are the PAM generator, adaptive filter, hybrid device and data converters.

A PAM generator is used as a signal generator to generate symbols, while the Digital-to-Analog Converter (DAC) is used to convert the symbols to analog form for transmission over the channel. The PAM generator and DAC constitute the transmitter division in the local transceiver side, while the *Incoming signal* represents the information from the remote transmitter. The local receiver obtains the remotely transmitted signal and tries to reconstruct it by cancelling the echo elements. Before entering the transceiver section, the received signal is converted to digital form using an Analog-to-Digital Converter (ADC).

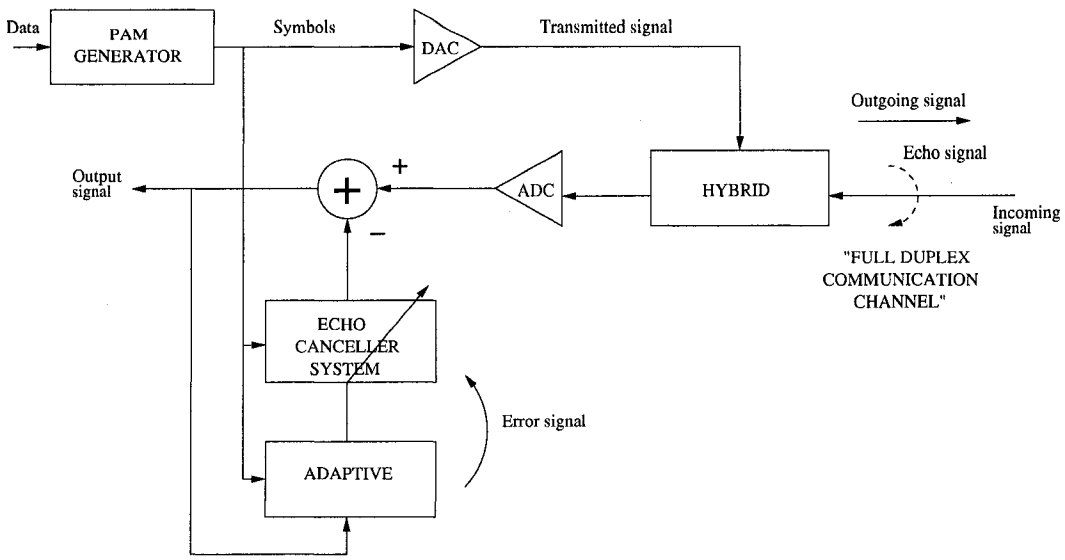

Figure 1.4: Block diagram of the echo canceller system

Since we want to remove the echo generated by the locally transmitted signal, we replicate the transmission path filter in the local receiver. This implies that the echo canceller filter in the receiver section must have the same filter response as the transmit filter. The degree of similarity between the two responses determines the quality of the echo canceller. If the responses of the two filters are identical or, in other words, if the difference between them is zero, then the echo cancellation can be described as effective.

The filter in the transmit path determines its response by using the locally transmitted signal as data input and converted values of echo voltages as tap coefficients. The proposed adaptive filter, which should have the same response as the transmit filter consists of a canceller filter with an adaptive mechanism. It is important to note in Figure 1.4, that the locally transmitted data is known to the adaptive filter. The filter derives its tap coefficients from the transmit filter using the Least Mean Square Steepest descent algorithm [7]. An error signal, which is the difference between the two reponses, is calculated. The LMS algorithm recursively minimizes this error by updating the coefficients. In an ideal situation, the error is zero and the echo cancellation would be complete.

1.5 Goals of this thesis

The aim of this thesis is to design an echo cancellation unit for high-speed wire-line communication systems. The two main design issues are area and power. It is essential that the echo cancellation unit does not dominate the receiver section of a Gigabit Ethernet transmission system. It is also important that the echo canceller unit be low-power device.

To meet the application requirements, first we need to be able to clock our design at extremely high frequencies. We also need to meet design considerations in order to have minimum area and power consumptions. This thesis therefore presents the design of an echo cancellation unit for high-speed wire-line systems, withminimization of area and power as the optimization goals.

In the design of our cancellation unit, we need to select an appropriate architectural pattern to be able to meet the design requirements. In Chapter 2, we do an architectural comparison between bit-parallel and bit-serial implementations. Based on the findings, we select an architecture mode for our filter. The design, implementation and comparison results of the serial and parallel architectures are discussed in Chapter 2.

Chapter 2

Architectural issues

Before selecting an architecture for the design of the echo canceller system, a comparison between the serial and parallel modes was made. For this, we implemented two versions of a mutiplier unit: A *bit-serial* version and a *bit-parallel* version.

Hardware realization can be based on two methods of data processing, namely, the serial and parallel approaches. In the former approach, one bit of the input sample is processed in one clock cycle while in the latter approach all the bits contained in the data word are processed in a single clock cycle. A comparison between the two schemes has been made by applying the bit-serial and bit-parallel formats in an arithmetic unit, namely, a multiplier block. The comparison between the two multipliers with respect to Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) implementations is presented in this chapter.

Architectural descriptions of serial and parallel multiplier units are presented and the two designs are compared in terms of their FPGA and ASIC characteristics. The logic designs have been translated to hardware using the Very High-Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL). The designs have been implemented using the *Xilinx-ISE-6.2.02i* software version, for the *Spartan* FPGA device family *Spartan*. The *ModelSim* simulator tool [4] which is compatible with the *Xilinx-ISE-6.2.02i* software, was used for simulation and verification of the design.

The bit-serial and bit-parallel multipliers transformed to ASIC designs by syn-

thesizing the VHDL codes using the *Synopsys Design Compiler*, version 2003.06. The area, power and timing reports for the designs were analyzed and compared.

Architectural descriptions, techniques employed, circuit operation and comparison results from FPGA and ASIC simulations for the bit-serial and bit-parallel multipliers are presented in this chapter. A compromise between these two approaches is the digit-serial approach [12], [11]. This has been described in the Appendix section.

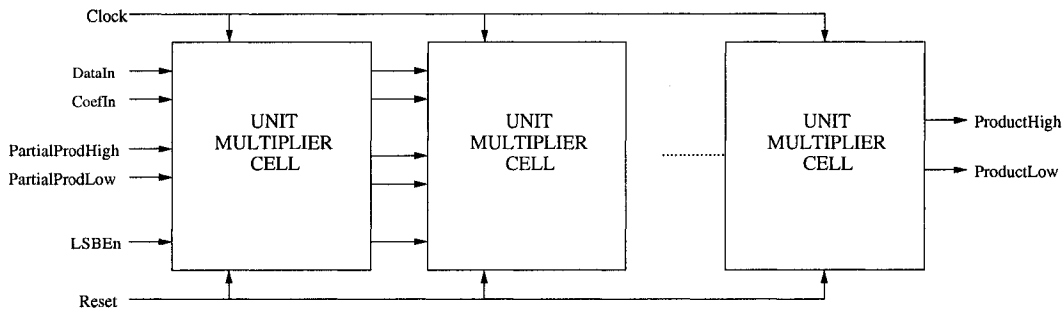

Figure 2.1: Block diagram of the bit-serial multiplier

2.1 Bit-serial multiplier

2.1.1 Introduction

The two main inputs to the bit-serial multiplier are the operand terms i.e. *data* and *coefficient*. The data and coefficient term bits are both applied to the multiplier block in a serial manner.

2.1.2 Bit-serial multiplier

A bit-serial multiplier is designed as a set of cells connected in tandem as shown in Figure 2.1. The inputs to the first cell are the following:

- *DataIn*: Data term
- *CoefIn*: Coefficient term
- *PartialProdHigh* and *PartialProdLow*: Incoming partial products *PartialProdHigh* and *PartialProdLow* represent respectively, the most significant and least significant sections of the incoming partial product
- *LSBEn*: A control signal which indicates the arrival of the Least Significant Bits (LSBs) of *DataIn* and *CoefIn*
- *Clock*: Clock signal
- *Reset*: Reset signal.

The product of multiplication is obtained from the output of the last cell in the serial multiplier. The result is spread out on two lines; namely *ProductHigh* and *ProductLow*. These signals contain the high and low sections of the product word. In an n-bit multiplier, the first least significant n bits of the result appear on the *ProductLow* line while the subsequent bits of the result appear on the *ProductHigh* line. Each cell interprets one or more bits of the multiplier coefficient and forms the partial product. The partial product is then added to the input signal entering the multiplier cell on the *PartialProdHigh* line. In each individual cell, one or more of the least significant partial product sum bits get appended to the Least Significant Word (LSW) of the product arriving on the *PartialProdLow* line, and the result leaves as the *ProductLow* signal. The remaining section of the partial product leaves as the *ProductHigh* signal.

In direct bit-serial multiplication between an n-bit multiplicand X and an m-bit multiplier Y, the product P can be computed according to Equation 2.1,

$$P = Y \times X = -Z_{m-1} + \sum_{i=0}^{m-2} Z_i \quad \text{where} \quad \mathrm{Z_i = y_i 2^i X} \tag{2.1}$$

The serial two's complement multiplication defined in Equation 2.1, is illustrated in Equation 2.2 as an example.

Equation 2.2 demonstrates the multiplication between a 4-bit multiplicand term $X = X_3X_2X_1X_0$ and a 6-bit multiplier term $Y = Y_5Y_4Y_3Y_2Y_1Y_0$. The partial products formed at the i^{th} stage is represented as Z_i. The partial products are added up at each level, to form the resultant partial products, represented by S_i. The final product P is obtained from the result of the last computation stage. A single bit P_i gets appended to the LSW of the product at each step and the remaining bits of the final partial product get appended as the MSW section of the product P.

We see that, in total, six steps are involved in the computation of the final product, i.e., S_5. Hence, six unit-multiplier cells need to be connected in series and, in every cell a single bit is appended to the LSW of the product. An m-bit multiplier therefore requires an implementation involving m cells, where each cell interprets a single coefficient bit and computes a partial product component Z_i.

$$
\begin{array}{cccccccccc}
 & & & & & & & X_3 X_2 X_1 X_0 & & X \\
 & & & & & & & Y_5 Y_4 Y_3 Y_2 Y_1 Y_0 & & Y \\
\hline
 & & & & & & 0 & 0 \quad 0 \quad 0 & & S_{-1} \\
 & & & & & & X_3Y_0 & X_2Y_0 \quad X_1Y_0 \quad X_0Y_0 & & Z_0 \\
\hline
 & & & & & S_{4,0} & S_{3,0} & S_{2,0} \quad S_{1,0} \quad S_{0,0} & & S_0 \\
 & & & & & X_3Y_1 & X_2Y_1 & X_1Y_1 \quad X_0Y_1 \quad 00 & & Z_1 \\
\hline
 & & & & S_{4,1} & S_{3,1} & S_{2,1} & S_{1,1} \quad S_{0,1} \quad 00 & & S_1 \\
 & & & & X_3Y_2 & X_2Y_2 & X_1Y_2 & X_0Y_2 \quad 00 \quad 00 & & Z_2 \\
\hline
 & & & S_{4,2} & S_{3,2} & S_{2,2} & S_{1,2} & S_{0,2} \quad 00 \quad 00 & & S_2 \\
 & & & X_3Y_3 & X_2Y_3 & X_1Y_3 & X_0Y_3 & 00 \quad 00 \quad 00 & & Z_3 \\
\hline
 & & S_{4,3} & S_{3,3} & S_{2,3} & S_{1,3} & S_{0,3} & 00 \quad 00 \quad 00 & & S_3 \\
 & & X_3Y_4 & X_2Y_4 & X_1Y_4 & X_0Y_4 & 00 & 00 \quad 00 \quad 00 & & Z_4 \\
\hline
 & S_{4,4} & S_{3,4} & S_{2,4} & S_{1,4} & S_{0,4} & 00 & 00 \quad 00 \quad 00 & & S_4 \\
 & -X_3Y_5 & -X_2Y_5 & -X_1Y_5 & -X_0Y_5 & 00 & 00 & 00 \quad 00 \quad 00 & & Z_5 \\
\hline
P_9 & P_8 & P_7 & P_6 & P_5 & P_4 & P_3 & P_2 \quad P_1 \quad P_0 & & S_5 \\
\hline
\end{array}
$$

(2.2)

Table 2.1: The Booth encoding table

Coefficient bits	Operation of each unit-cell	Booth encoded digit
Y_i,Y_{i+1}	Function	Z_i
0,0	add	0
0,1	add	1
1,0	add	$\bar{1}$
1,1	add	0

2.1.3 Booth algorithm

A.D Booth [8], proposed a method of reorganizing the partial products in Equation 2.1, so that all of them could be computed in an identical fashion. Consider the multiplication between two numbers X and Y where the multiplicand X is n bits wide and the multiplier term Y is m bits wide. The equation for multiplication employing the Booth algorithm is indicated in Equation 2.3.

$$P = Y \times X = \sum_{i=0}^{m-1} Z_i \quad \text{where} \quad Z_i = \left(-y_i + y_{i-1}\right)2^i X \tag{2.3}$$

Equation 2.3 leads to a multiplier implementation containing m identical cells, where the i^{th} cell interprets two successive multiplier coefficient bits Y_i and Y_{i+1}. The Booth algorithm [8] is shown in Table 2.1.

The advantage of incorporating this algorithm in ordinary bit-serial multiplication is that it leads to a regular multiplier structure with all the hardware cells being identical to each other. However, it does not decrease the number of generated partial products and hence does not decrease the computation time. The Booth algorithm is therefore not advantageous in high-speed applications. Since the number of computation stages remain the same, the number of cells in the series connection is not reduced. Hence the Booth algorithm does not comply with low area needs.

2.1.4 Modified Booth algorithm

The Modified Booth Algorithm (MBA) [8] was proposed by MacSorley as an improvement to the Booth algorithm.

Table 2.2: The Modified Booth encoding table

Coefficient bits	Encoded digit
Y_{2i+1},Y_{2i},Y_{2i-1}	Z_i
0,0,0	0
0,0,1	1
0,1,0	1
0,1,1	2
1,0,0	$\bar{2}$
1,0,1	$\bar{1}$
1,1,0	$\bar{1}$
1,1,1	0

MacSorley modified the Booth algorithm in such a way that the number of partial products in the computation condensed to half of that in regular Booth multiplication. The advantages of applying the MBA in the multiplier design are summarized as follows:

- The computation speed increases twofold.
- The architectural complexity and amount of hardware resources get reduced as fewer blocks are required to implement the reduced computation steps.

With this technique, every three consecutive bits of the multiplier coefficient, namely, Y_{2i+1}, Y_{2i}, Y_{2i-1} are encoded into a single bit Z_i. The encoded bit is formed as per the encoding scheme shown in Table 2.2.

The data term is multiplied with the encoded bit Z_i, where $Z_i \in \{\bar{2}, \bar{1}, 0, 1, 2\}$ and the generated partial products are added together. Multiplication by two is just a simple shift-left operation. For the multiplication between an n-bit multiplicand X and an m-bit multiplier Y, the product P can be expressed as shown in Equation 2.4.

$$P = Y \times X = \sum_{i=0}^{(m-2)/2} K_i \quad \text{where} \quad \mathrm{K_i} = \mathrm{Z_i} 4^{\mathrm{i}} \mathrm{X} \tag{2.4}$$

K_i represents the i^{th} partial product, where $Z_i \in \{\bar{2}, \bar{1}, 0, 1, 2\}$. The multiplication process incorporating the MBA is illustrated with an example involving a 4-bit multiplicand X and a 6-bit multiplier term Y. The 6-bit coefficient term is encoded with just three bits Z_2, Z_1 and Z_0. The multiplicand term is therefore multiplied with the encoded bits of the multiplier term. This is shown step-wise in Equation 2.5.

$$
\begin{array}{cccccccccc}
 & & & & & & X_3 & X_2 & X_1 & X_0 \quad X \\
 & & & & & & Y_5 & Y_4 & Y_3 & Y_2 \; Y_1 \; Y_0 \quad Y \\
\hline
 & & & & & 0 & 0 & 0 & 0 & S_{-1} \\
X_3Z_0 & X_3Z_0 & X_3Z_0 & X_3Z_0 & X_3Z_0 & X_2Z_0 & X_1Z_0 & X_0Z_0 & & K_0 \\
\hline
S_{7,0} & S_{6,0} & S_{5,0} & S_{4,0} & S_{3,0} & S_{2,0} & S_{1,0} & S_{0,0} & & S_0 \\
X_3Z_1 & X_3Z_1 & X_3Z_1 & X_2Z_1 & X_1Z_1 & X_0Z_1 & 00 & 00 & & K_1 \\
\hline
S_{5,1} & S_{4,1} & S_{3,1} & S_{2,1} & S_{1,1} & S_{0,1} & 00 & 00 & & S_1 \\
X_3Z_2 & X_2Z_2 & X_1Z_2 & X_0Z_2 & 00 & 00 & 00 & 00 & & K_2 \\
\hline
P_9 & P_8 & P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 \; P_0 & S_2 \\
\hline
\end{array}
\tag{2.5}
$$

K_i represents the partial product computed at each stage and S_i represents the resultant partial product component. At every computation level, two bits are appended to the LSW section of the result. Hence the LSW of the result, the *ProductLow* line, contains six LSB bits and the MSW of the result, *ProductHigh*, contains the remaining four bits.

Table 2.3: Implementation of the MBA in the bit-serial multiplier

Coefficient bits	Operation of each unit-cell	Modified Booth encoded digit
Y_{2i+1},Y_{2i},Y_{2i-1}	Operation	Y_s,Y_b,Y_a
0,0,0	add 0	0,0,0
0,0,1	add 1 $\times 4^i$	0,0,1
0,1,0	add 1 $\times 4^i$	0,0,1
0,1,1	add 2 $\times 4^i$	0,1,0
1,0,0	subtract 2 $\times 4^i$	1,1,0
1,0,1	subtract 1 $\times 4^i$	1,0,1
1,1,0	subtract 1 $\times 4^i$	1,0,1
1,1,1	subtract 0	1,0,0

Table 2.4: Implementation of the MBA in the first cell

Coefficient bits	Operation of each unit-cell	Modified Booth encoded digit
Y_{2i+1},Y_{2i}	Operation	Y_s,Y_b,Y_a
0,0	add 0	0,0,0
0,1	add 1 $\times 4^i$	0,0,1
1,0	subtract 2 $\times 4^i$	1,1,0
1,1	subtract 1 $\times 4^i$	1,0,1

By using the MBA, the multiplication of data with a 6-bit coefficient term is complete with three computation levels. Hence, only three unit-multiplier cells are required for the implementation as compared to six cells needed for regular bit-serial multiplication. The encoding of the coefficient term is responsible for this reduction. The hardware implementation of the MBA therefore, involves a set of $m/2$ identical cells whereas ordinary Booth multiplication requires m cells. The operation of the unit-cell in the Modified Booth bit-serial multiplier is shown in Table 2.3. The requirement that $Y_{-1} = 0$ is satisfied by modifying Table 2.3 for the first cell. The resulting reduced table is shown in Table 2.4.

Now, two shifted versions of the multiplicand term are created and these shifted versions are multiplied with the encoded bits of the coefficient term. The multiplication of the data term X with the encoded digit Z_i is performed in each unit-cell, based on the signals Y_s,Y_b and Y_a. All the cells in the tandem connection are identical and the top-level symbol of the unit-cell is shown in Figure 2.2.

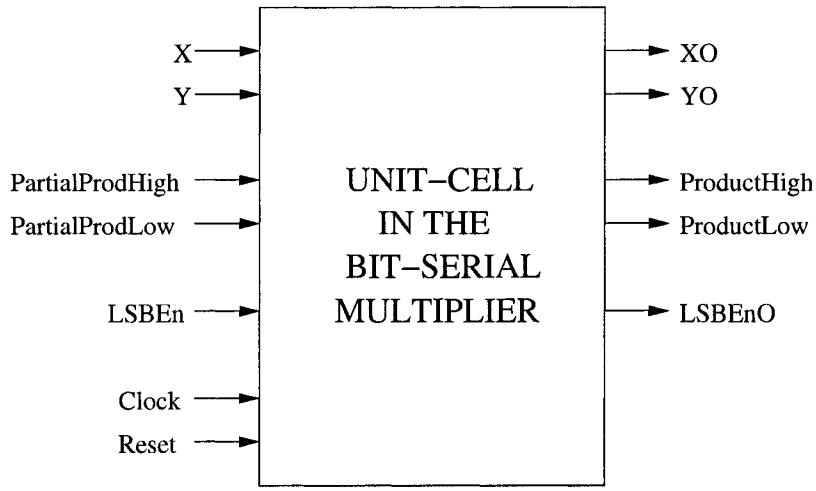

Figure 2.2: Unit-cell of the bit-serial multiplier

The inputs to the unit-cell of the bit-serial multiplier are the following:

- X: Data term
- Y: Coefficient term
- *PartialProdHigh* and *PartialProdLow*: Incoming partial product words
- *LSBEn*: Signal which indicates the arrival of the LSB bits of the operands
- *Clock*: Clock input
- *Reset*: Reset input.

The schematic of the unit-cell is shown in Figure 2.3. It is seen that the unit-cell is made of logic gates, D flip-flops and 2:1 multiplexers. The logic diagram of the bit-serial adder that is used to add the partial products is shown in Figure 2.4.

2.1.5 Operation of the unit-cell in the Modified Booth bit-serial multiplier

The inputs to the multiplier cell are the data signal X and the coefficient signal Y. The signal X is shifted to form the $x4^i$ and $2x4^i$ components. The output X_o represents the individual bits of X shifted by $2i$ bits. Three signals Y_a, Y_b and Y_s are

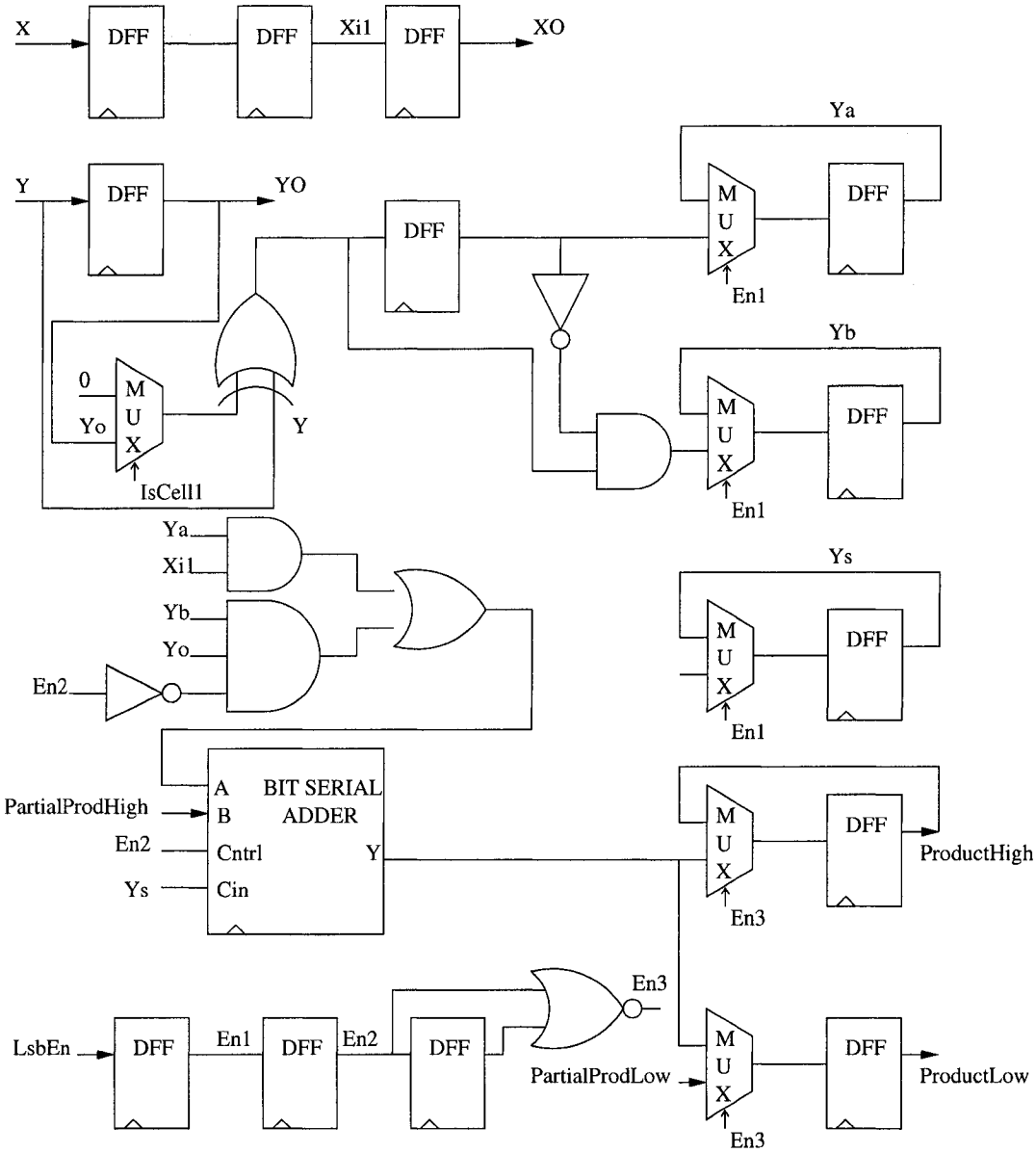

Figure 2.3: Schematic of the unit-cell

created, based on which the unit-cell of the serial multiplier operates. Every three consecutive coefficient bits Y_{2i+1}, Y_{2i} and Y_{2i-1} are mapped onto the latched signals Y_s, Y_b, and Y_a. The signal Y_s is used to indicate the sign of the encoded digit while Y_a and Y_b indicate the magnitude.

According to the MBA, it is required that the bit period prior to the LSB of the coefficient term bits arriving at the first cell = logic 0. This condition is satisfied by implementing Table 2.4. Both Table 2.3 and Table 2.4 are incorporated in the unit-

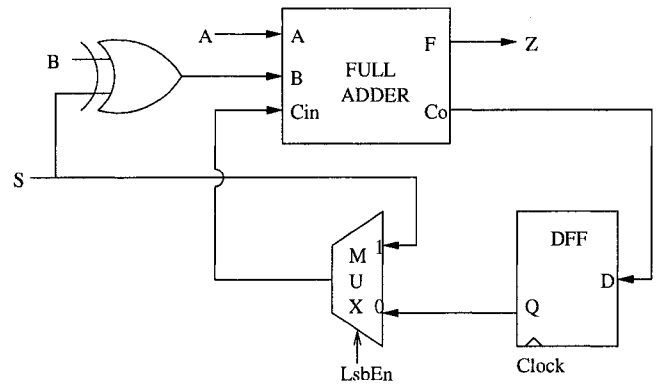

Figure 2.4: Schematic of the bit-serial adder

cell of the bit-serial multiplier by means of a 2:1 multiplexer. The select signal for the 2:1 multiplexer is a signal *IsCell1* which indicates if the unit-cell is the first one in the series connection or not. If it is the first cell, then the multiplier coefficient term bit $Y_{-1} = 0$.

The partial product leaves the unit-cell on the *ProductLow* and *ProductHigh* lines and enters the next cell on the *PartialProdLow* and *PartialProdHigh* respectively. The *ProductLow* and *ProductHigh* lines of the last block contain the product of multiplication.

2.1.6 Bit-serial adder

The bit-serial adder represented in Figure 2.4 is used to add the partial products generated as a result of multiplication. The inputs to the bit-serial adder are the following:

- *A*: The partial product term generated in the i^{th} cell by the multiplication between the data signal X and the encoded digit of the multiplier term Y
- *B*: The incoming partial product from the $(i-1)^{th}$ cell
- *S*: The sign bit
- *LsbEn*: The signal that indicates the arrival of the LSB of the inputs
- *Clock* and *Reset* signals.

Table 2.5: Components in the bit-serial multiplier

Component name	Equation
D Flip-flop	$7M$†
2:1 Mux	$3M+M/2$†
2-input Xor	$2M$†

†M: Size of the coefficient

2.1.7 Number of components

The number of components in each unit-cell of the bit-serial multiplier has been generalized into equations. The components in a unit-cell include D flip-flops, 2:1 multiplexers, 2-input *Xor* gates and basic gates. The equations for these components are listed in Table 2.5.

2.2 Bit-parallel multiplier

In bit-parallel multiplication [8], all the bits contained in the input sample are processed in one clock cycle. This section describes a multiplier unit based on such a parallel architecture. In the bit-parallel multiplier, both the data and coefficient term bits enter the multiplier unit on separate lines. The subsequent sections explain its operation and design.

2.2.1 Array multiplication

Consider two unsigned binary integers $A = a_{m-1}...a_1a_0$ and $B = b_{n-1}...b_1b_0$ with values A_v and B_v respectively. A is m bits wide and B is n bits wide. Equations 2.6 and 2.7 represent the A and B terms respectively.

$$A = \sum_{i=0}^{m-1} a_i 2^i \tag{2.6}$$

$$B = \sum_{j=0}^{n-1} b_j 2^j \tag{2.7}$$

In binary multiplication, the $(m+n)$-bit product $P = P_{m+n+1}...P_1P_0$ is formed

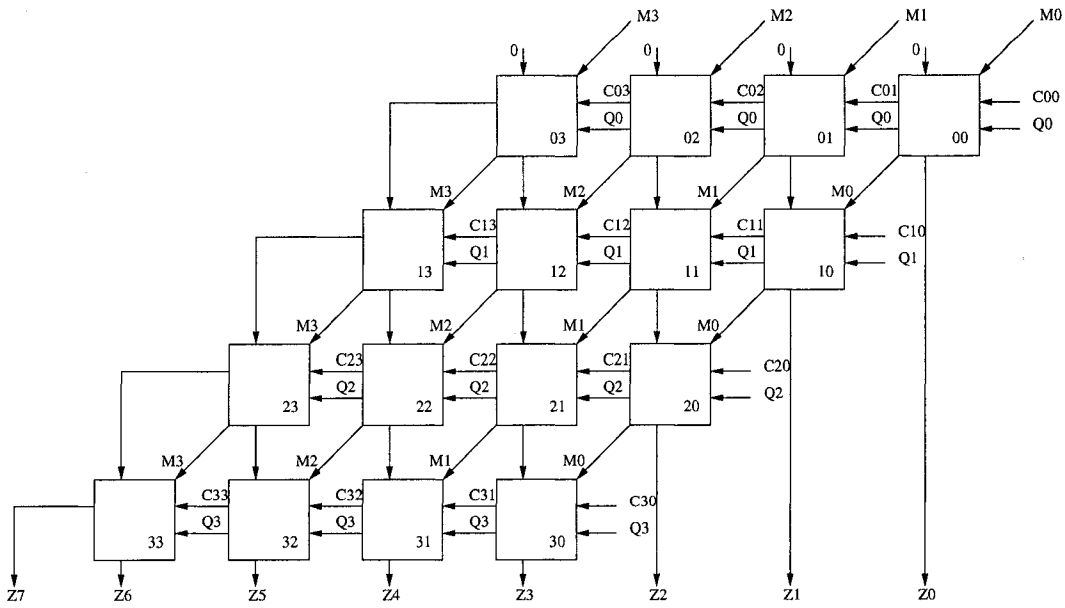

Figure 2.5: Schematic of the 4-bit-parallel multiplier

by multiplying the multiplicand A by the multiplier B. The value of product P is determined according to Equation 2.8.

$$P_v = A_v B_v = \left(\sum_{i=0}^{m-1} a_i 2^i\right)\left(\sum_{j=0}^{m-1} b_j 2^j\right) \qquad (2.8)$$

$$= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(a_i b_j 2^{i+j}\right) = \sum_{k=0}^{m+n+1} P_k 2^k \quad \text{where} \quad \mathrm{P_k = a_i \times b_j; k = i + j}$$

The operations involved in array multiplication are basically add-shift operations. Each of the partial product terms is called a summand. The $m \times n$ summands are generated in a parallel manner by $m \times n$ *And* gates. The data and coefficient term bits propagate through the array vertically and horizontally, respectively. The carry signal also propagates at every stage horizontally and the partial products travel from one level to the next. The partial product bits from the cells of the last stage represent the product of multiplication. The circuit diagram of a 4×4 array multiplier is shown in Figure 2.5.

The 4-bit data term M feeds into the array multiplier as individual bits, M_{3-0}. The coefficient term bits also enter the multiplier unit in parallel form i.e. Q_{3-0}.

These bits are propagated through the array as shown in Figure 2.5. The carry

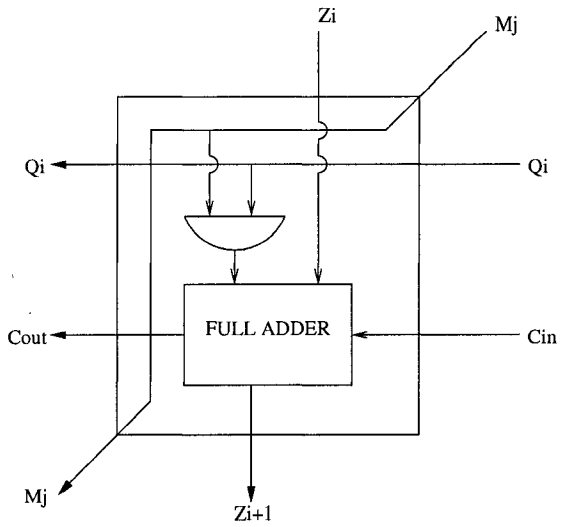

Figure 2.6: Unit-cell of a bit-parallel multiplier

inputs to the cells in the first column are initialized to zero and, in the subsequent stages, the carry out from the previous cell forms the carry input to the present cell. The final product appears on the Z_{7-0} lines. Each cell in the bit-parallel multiplier is identical and is represented in Figure 2.6.

The main component of the unit-cell is the single-bit full-adder. Based on the value of the multiplier coefficient bit Q_i, the *And* gate determines whether a multiplicand bit M_j is to be added to the incoming partial product bit or not. If $Q_i = 1$, the i^{th} row adds the shifted version of the multiplicand term to the incoming partial product Z_i, and generates the outgoing partial product Z_{i+1}. If $Q_i = 0$, then Z_i is passed vertically downward unchanged. At first the partial products are initialized to zero. The multiplicand is shifted left one position per row by the diagonal signal path. The partial products from the last row represent the result of the multiplication.

An $n \times n$ array multiplier consists of an $n \times n$ matrix of unit-multiplier cells. Hence, a 4×4 bit multiplier contains 16 cells; an 8×8 bit multiplier is made up of 64 cells and so on. The partial products from the first row are inputs to the cells present in the second row and so on. The outputs of the cells in the last row generate the result of multiplication.

Table 2.6: Components in the bit-parallel multiplier

Component name	Equation
And gate	n^2 †
Full-adder	n^2 †

†M: Size of the coefficient

2.2.2 Number of components

The number of components in each unit-cell can be generalized into equations. As seen in Figure 2.6, each unit-cell of an array multiplier is made up of an *And* gate and a full-adder circuit, which in turn is made of *And*, *Or* and *Xor* gates. The number of full-adders and *And* gates can be formulated as shown in Table 2.6.

The next section describes the FPGA implementation of the bit-serial and bit-parallel multipliers and presents a performance comparison.

2.3 FPGA implementation: Bit-serial and bit-parallel multipliers

2.3.1 Introduction to FPGA implementation

The *Xilinx-ISE* tools allow a design to be entered as a VHDL entity [4]. The serial and parallel multipliers were implemented using the *Xilinx-ISE-6.2.02i* software tool and the simulator tool *ModelSim-XE-II-5.7c* [4] was used to verify the correctness of the designs. The VHDL codes were synthesized for FPGA devices and the serial and parallel designs were compared with regard to their FPGA characteristics. An FPGA device is made up of an array of Configurable Logic Blocks (CLBs), surrounded by a perimeter of Input-Output Blocks (IOBs). The actual function of the design is implemented through Look Up Tables (LUTs). The device utilization of an FPGA implementation is analyzed with respect to the percentage usage of CLBs, LUTs and IOBs. For our implementation, the *Spartan* FPGA device family has been selected. Details of the *Spartan* FPGA device have been omitted in this section since it has been extensively covered in Chapter 4.

Table 2.7: Device utilization summary for the 4-bit multipliers

Multiplier type	4-bit-serial	4-bit-parallel
Number of slices	9%	8%
Number of LUTs	4%	7%
Number of IOBs	7%	17%
Gate count	358	168

Table 2.8: Device utilization summary for the 8-bit multipliers

Multiplier type	8-bit-serial	8-bit-parallel
Number of slices	16%	35%
Number of LUTs	8%	31%
Number of IOBs	11%	35%
Gate count	660	720

2.3.2 Performance comparison

Serial and parallel multipliers for sizes of 4, 8 and 16 bits have been implemented for the *Xilinx* XC2S15 *Spartan-II* FPGA device [5]. Table 2.7 compares the device utilization of a 4-bit-serial and 4×4 array multiplier in terms of the number of CLB slices, LUTs, IOBs and logic gates used. Tables 2.8 and 2.9 compare the device usage for the 8-bit and 16-bit multipliers respectively.

From Tables 2.7, 2.8 and 2.9, we observe that the bit-serial multiplier has lower device utilization than the bit-parallel multiplier. The difference in percentage becomes apparent as we go from the 4-bit multipliers to the 16-bit multipliers.

The difference in the device utilization characteristics of the two designs is significant enough to say that the serial design is more area efficient than the corresponding parallel one. It is important to note that the area efficiency of a serial architecture over its corresponding parallel architecture becomes more significant with the increase in design size.

A graph of the gate count comparison is plotted in Figure 2.7. The gate count of a design is a direct indication of the amount of hardware resources required in the actual implementation. The bit-serial designs use less hardware resources than their parallel counterparts and the difference becomes greater as we go from the 4-bit to

Table 2.9: Device utilization summary for the 16-bit multipliers

Multiplier type	16-bit-serial	16-bit-parallel
Number of slices	33%	148% †
Number of LUTs	17%	129% †
Number of IOBs	12%	71%
Gate count	1310	2976

†More than 100 % of device resources are used

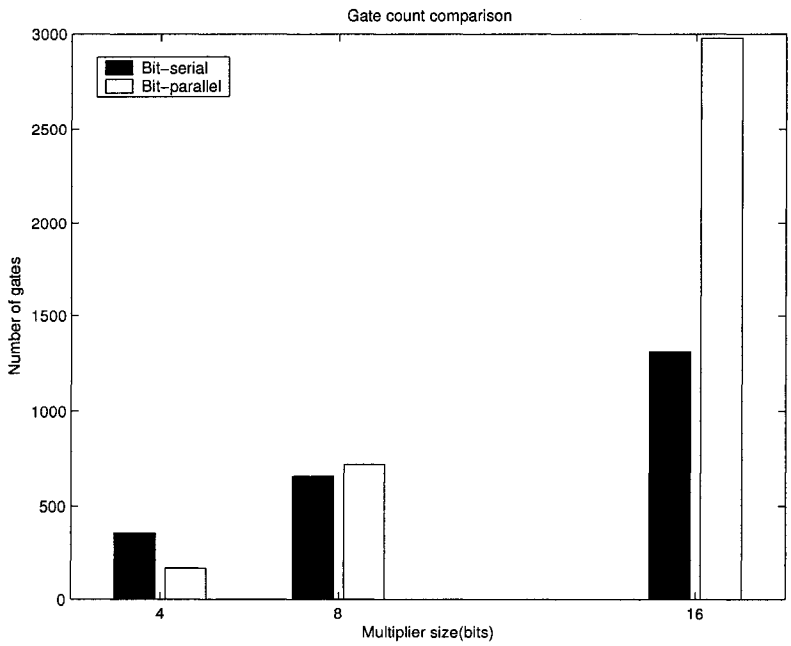

Figure 2.7: Gate count comparison of the serial and parallel multipliers

the 16-bit designs.

Having compared the serial and parallel multipliers with regard to area usage, let us study the frequency and power estimates when the designs are simulated as ASICs.

2.4 ASIC implementations

2.4.1 Introduction to *Synopsys* Design Compiler

Synopsys Design Compiler(DC) [2] is a Computer-Aided Design (CAD) tool that is used to synthesize a design from Register Transfer Level (RTL) to gate level.

Table 2.10: DC report of the 4-bit-parallel multiplier

Design: 4-Bit-parallel multiplier	Report
Frequency of operation	218.34 MHz
Global operating voltage	1.6 V
Total dynamic power	1.461 mW

Table 2.11: DC report of the 4-bit-serial multiplier

Design: 4-Bit-serial multiplier	Report
Frequency of operation	800.0 MHz
Global operating voltage	1.6 V
Total dynamic power	2.6538 mW

VHDL codes for the multiplier units were synthesized using the DC, version 2003 and the generated synthesis reports were analyzed and compared.

2.4.2 Comparison issues

The bit-serial multiplier is a synchronous design and area and power readings can be recorded over a range of clock frequencies. The maximum clock frequency of the bit-serial multiplier was observed by decreasing the clock period until the timings specifications were violated. The bit-parallel multiplier is an asynchronous design and the minimum frequency of operation was calculated as the reciprocal of the critical path delay. At this frequency, the area and power reports were obtained.

In the bit-serial scheme, one bit of the data word is processed every clock cycle, while in the parallel scheme the entire set of data bits are processed in the same clock cycle. Hence, a like to like comparison should be made when the frequency of operation of the serial design is k times higher than that of its parallel counterpart, where k is the word length. For instance, the characteristics of the 4-bit-serial and the 4×4 array multiplier have been compared when the clock frequency of the serial multiplier is four times faster than the operating frequency of the parallel multiplier. Similarly, the 8-bit parallel and serial designs have been compared at frequencies of k and $8 \times k$ respectively.

Table 2.12: DC report of the 8-bit-parallel multiplier

Design: 8-Bit-parallel multiplier	Report
Frequency of operation	94.607 MHz
Global operating voltage	1.6 V
Total dynamic power	9.1334 mW

Table 2.13: DC report of the 8-bit-serial multiplier

Design: 8-Bit-serial multiplier	Report
Frequency of operation	756.80 MHz
Global operating voltage	1.6 V
Total dynamic power	4.3418 mW

2.4.3 Performance comparion

2.4.3.1 4-Bit multipliers

The 4-bit-parallel multiplier of Figure 2.5 reported a critical path delay of 4.58 ns. Hence, the lowest operating frequency = 1/4.58 ns = 218.34 MHz. At this frequency, the power consumption was noted and is listed in Table 2.10.

The 4-bit-serial multiplier operated at a maximum clock frequency of 800 MHz. The power measurement made at the 800 MHz operating frequency is listed in Table 2.11.

2.4.3.2 8-Bit multipliers

The critical path delay of the 8-bit array multiplier was found to be 10.57 ns. Hence, the critical frequency of operation = 1/10.57 ns = 94.607 MHz. The power usage at 94.607 MHz is listed in Table 2.12.

The maximum clock frequency of the 8-bit-serial multiplier was found to be 800 MHz. In an 8-bit-serial multiplier, each multiplication takes place once in every eight clock cycles while in the bit-parallel multiplier, each multiplication takes place every clock cycle. Hence, in order to compare the efficiencies of the two multipliers, measurements are made when the frequency of the 8-bit-serial multiplier is eight times that of the 8-bit-parallel multiplier.

Table 2.14: DC report of the 16-bit-parallel multiplier

Design: 16-Bit-parallel multiplier	Report
Frequency of operation	41.42 MHz
Global operating voltage	1.6 V
Total dynamic power	43.9895 mW

Table 2.15: DC report of the 16-bit-serial multiplier

Design: 16-Bit-serial multiplier	Report
Frequency of operation	662.72 MHz
Global operating voltage	1.6 V
Total dynamic power	13.1983 mW

The 8-bit-serial multiplier is thus analyzed at a frequency of $94.607 \times 8 = 756.8$ MHz. Table 2.13 contains the power estimates at an operating frequency of 756.8 MHz.

2.4.3.3 16-Bit multipliers

The lowest operating frequency of the 16-bit parallel multiplier was calculated to be 41.42 MHz. At this frequency, the power consumption was recorded and is listed in Table 2.14.

The 16-bit-serial multiplier sustained a maximum clock frequency of 800 MHz. In a 16-bit-serial multiplier, each multiplication takes place once in every sixteen clock cycles while in the bit-parallel multiplier, each multiplication takes place every clock cycle. Hence, the 16-bit-serial multiplier is analyzed at a frequency of $41.42 \times 16 = 662.72$ MHz. The characteristics at the 662.72 MHz operating frequency are listed in Table 2.15.

The comparison in power consumption of the serial and parallel multipliers of 4, 8 and 16 bit size can be observed in the graph of Figure 2.8.

From the FPGA and ASIC simulations, we learn that the serial multiplier has better characteristics with regard to area, power and timing. We also infer that as the design size increases, the area and power characteristics of a serial design considerably improve with respect to a corresponding parallel design.

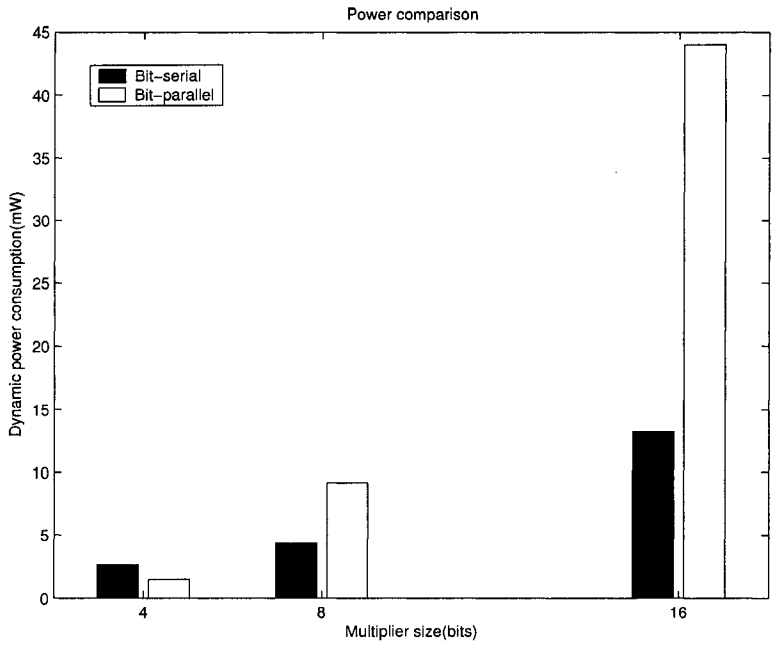

Figure 2.8: Power comparison of the serial and parallel multipliers

2.5 Conclusion

The purpose of comparing the serial and parallel multipliers was to choose an architecture for the design of the echo cancellation unit. The objective is to make the echo canceller an area and power-efficient device. The application of our design lies in high-speed wire-line communication systems and hence it is imperative to drive the design at high clock speeds.

The comparison results obtained from the FPGA and ASIC simulations demonstrated the following:

- The bit-serial multiplier required fewer gates compared to the corresponding bit-parallel multiplier. The advantage enjoyed by the serial design was more apparent with increases in design size.

- Synthesis of the serial and parallel multipliers for FPGA devices indicated that the bit-serial multiplier has a lower device utilization percentage. This means that employing the serial approach would decongest the receiver section in a digital communication system.

- Power measurements also considerably favoured the bit-serial multiplier.
- Most importantly, the clock frequencies of the bit-serial multiplier were found to be significantly high.

These results helped us in selecting an architectural pattern for the design of the echo cancellation unit. The bit-serial architecture has thus been chosen as the design methodology.

Chapter 3

Design of the echo cancellation unit

This thesis presents the design of an echo cancellation unit for high-speed wire-line communication systems. The central component of an echo canceller is the adaptive filter. In this chapter, we put forth an adaptive filter that is designed to cope with echoes of varying sizes, at the same time ensuring an efficient usage of hardware resources. The design goals are twofold. The first aim is to make sure that the echo canceller does not spatially dominate the receiver section in a digital communication system. The second goal is to maintain it as a low-power device. We are therefore trying to develop an echo cancellation unit for high-speed wire-line systems, keeping area and power as the optimization goals. To meet the speed requirement of 1000BASE-T Gigabit Ethernet, it is essential to be able to clock the echo cancellation unit at extremely high rates.

Based on the results in Chapter 2, bit-serial arithmetic has been chosen as the architecture mode of the design. This thesis is an attempt to make architectural and algorithmic considerations while designing an adaptive echo cancellation unit for high-speed communication systems, with area and power minimizations.

3.1 Echo canceller system

The block diagram of the echo cancellation unit for 1000BASE-T Gigabit Ethernet systems was presented in Figure 1.4 of Chapter 1. The chief component of the echo canceller unit is the adaptive filter [7]. This chapter presents the design of the bit-serial adaptive filter as a hierarchical model. The following subsections go on to describe the role of the *adaptive* nature of the filter, followed by optimization techniques and architecture descriptions.

3.1.1 The adaptive filter

The echo response is a random curve with echoes varying in size. It is advantageous to have an echo canceller that is able to adapt to the variations in echo size and which treats large and small echoes differently. By being able to filter out large and small echoes in a different manner, the behavior of the echo canceller can be described as *adaptive*.

3.1.2 Architecture mode of the filter

The major concerns in the architecture selection of our filter design are area, power and clock speeds. For application of the echo cancellation unit in Gigabit Ethernet systems, it is essential to be able to clock the design at frequencies close to the gigahertz range. The clock frequencies reached by the serial multiplier were high. It is also critical that the layout of the echo canceller does not dominate the receiver design and consume heavy power. The comparison of serial and parallel multipliers in Chapter 2 suggested that bit-serial designs are more area and power efficient. The following section explains how we addressed the area, power and speed issues.

3.1.3 Design goals

The designs goals with respect to area, power and speed were dealt with at the architectural and algorithmic levels.

3.1.3.1 Improving speed of operation

The operating speed of the design has been tackled at the algorithmic stage by incorporating the Modified Booth Algorithm (MBA) in the multiplier block of the filter unit. The MBA is basically a speed-up technique; encoding the coefficient term bits reduces the number of multiplication steps. The overall computation speed of the multiplication process therefore increases.

3.1.3.2 Minimizing area utilization

Results from the synthesis of the bit-serial and bit-parallel multipliers for FPGA devices showed that the serial multiplier required a much lower gate count. This is indicative of fewer hardware resources in the actual implementation. The FPGA device utilization of the two multipliers demonstrated that the serial design makes better usage of the FPGA device resources. Thus, adopting the serial architecture has naturally reduced the area utilization.

Also, by using the MBA in the multiplier design, the number of blocks required to implement the reduced computation steps gets halved.

3.1.3.3 Minimizing power consumption

Power consumption depends on the amount of switching activity of the signals on the chip. A section of the block is kept active only when the filter operates in 8-bit mode, i.e., only when dealing with large echoes. That part of the hardware unit required for 8-bit mode operation only is kept inactive when dealing with small echoes. This way, the power expenditure gets reduced.

Also, preserving the signal word length as short as possible has eliminated use of unnecessary arithmetic blocks and memory.

3.2 Design of the bit-serial adaptive filter

The design of the adaptive filter based on bit-serial arithmetic and the interaction of the filter with echoes is presented in this section.

The basic principle in the operation of our echo canceller is that values of echo voltage are interpreted as filter tap coefficients. The echo response requires filter taps that can be small or large. Large values of echo voltage translate to filter tap coefficients of 8-bit size which implies that the filter is working in 8-bit mode. Similarly, small echoes imply that the filter tap coefficients are four bits wide and that the filter is working in 4-bit mode. The filter in our design is capable of functioning in 4-bit mode and/or in 8-bit mode simultaneously, and continuously processes frames every eight clock cycles. In the case of small echoes, the filter operates in 4-bit mode while it adapts to working in 8-bit mode when the echoes are large.

The continuous processing of a series of frames that are 4-bits and/or 8-bits in precision, translates to the cancelling of a continuous chain of small and large echoes in a digital communication system.

The typical size of an echo canceller filter for the 1000BASE-T receiver [6] is 160 taps. This number is derived as follows:

The round-trip distance of the locally transmitted signal, i.e., the maximum distance covered = 200 m.

Velocity of electrons in the copper cable $\approx 0.7 \times 3 \times 10^8$ m/s.

The maximum time period for a far echo to occur is calculated to be 953.2 ns.

In the 1000BASE-T system, the signal changes its value every 8 ns.

The estimate of the number of echoes = $953.2/8 \approx 120$.

Hence, we approximate the maximum number of filter taps required in the 1000BASE-T echo cancelation filter to not exceed 160.

In this section, the bit-serial adaptive filter is considered as a two-tap filter entity. The top-level symbol view of the two-tap filter unit is shown in Figure 3.1.

As seen in Figure 3.1, the inputs to the two-tap filter are the following:

- $DataA_{3\_0}$: 4-bit data term A
- $DataB_{3\_0}$: 4-bit data term B
- $CoefficientA_{7\_0}$: 8-bit coefficient term A
- $CoefficientB_{7\_0}$: 8-bit coefficient term B

The A and B, data and coefficient inputs are fed into the 4-bit multipliers A and B, respectively. These inputs enter the circuit on parallel lines. The filter outputs that are the results of 4-bit and 8-bit mode operations are listed below:

- *ResultHighA*: MSW of the product of 4-bit multiplier A
- *ResultLowA*: LSW of the product of 4-bit multiplier A
- *ResultHighB*: MSW of the product of 4-bit multiplier B
- *ResultLowB*: LSW of the product of 4-bit multiplier B
- *ResultHighC*: MSW of the product of 8-bit multiplier C
- *ResultLowC*: LSW of the product of 8-bit multiplier C
- *Error signal*: The error signal

The widths of the filter outputs are specified in the symbol view of Figure 3.1. In the two-tap filter, there are three 4-bit multipliers A, B and C. Multipliers A and B are always active and continuously and simultaneously deal with small echoes. The 4-bit tap coefficients of the multipliers A and B correspond to small echo voltages. When a large echo arrives, a signal is used to indicate which one of the multipliers A or B would connect to the 4-bit multiplier C and work in 8-bit mode. In this case, the coefficient tap size would be eight bits wide. When both the multipliers A and B try to join C, an error occurs, for which purpose we have the *Error signal* output.

The top-level filter circuit instantiates the bit-serial adaptive filter and a few other blocks for data conversion and signal generation functions. The internal structure of the filter in Figure 3.1 is shown in Figure 3.2.

The components contained in the top-level circuit are the following:

- Data Parallel-to-Serial Converters (PSC) for each set of data terms
- Coefficient PSCs for each set of coefficient terms
- Signal generator component

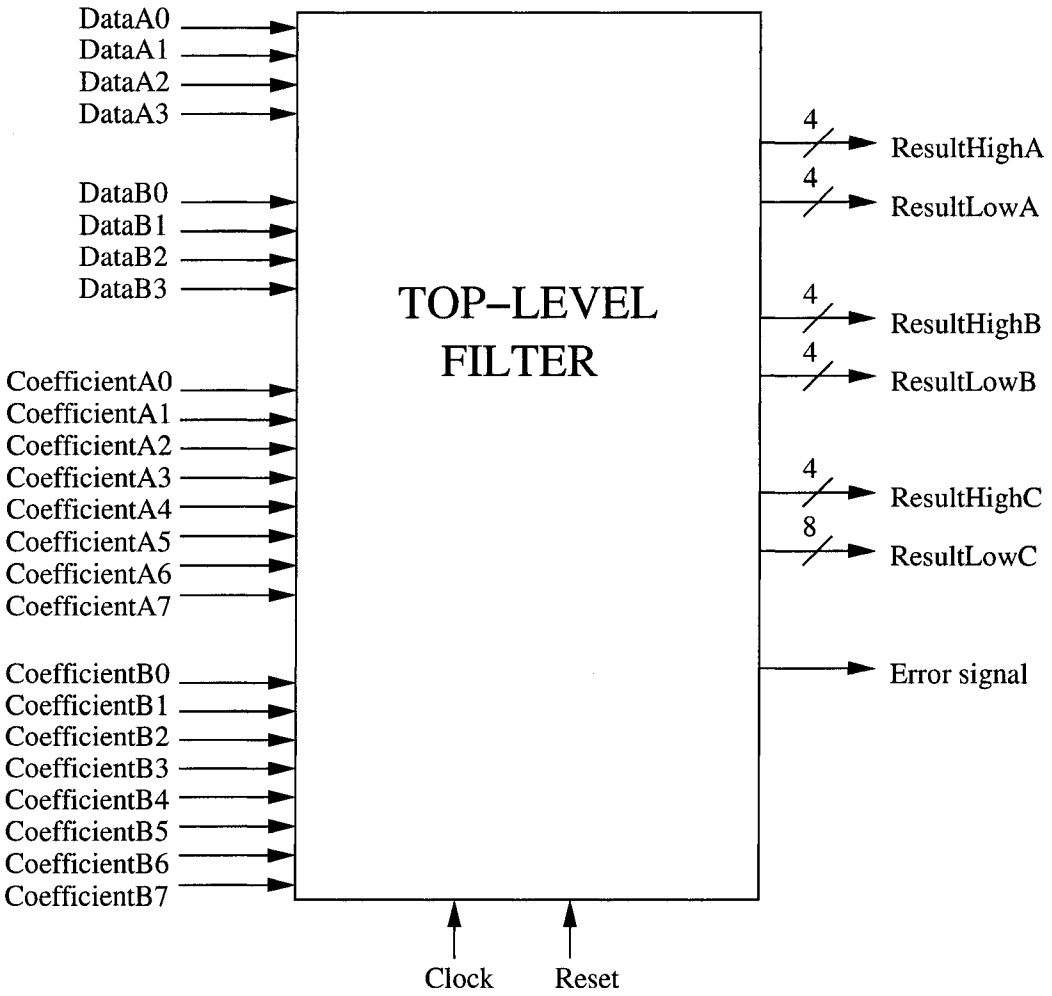

Figure 3.1: Block diagram of the bit-serial adaptive filter

- Clock generator
- Bit-serial adaptive filter

The bit-serial adaptive filter is referred to as the Design Under Test (DUT). For processing of the operands by the DUT, the parallel inputs have to be converted to a serial form. The incoming data and coefficients term bits are fed to PSCs and these outputs are fed to the actual filter component. The filter component requires certain signals for its functioning. These signals are all generated internally in the top-level schematic. It is the signal generator component that computes all the inputs required by the filter and the PSCs.

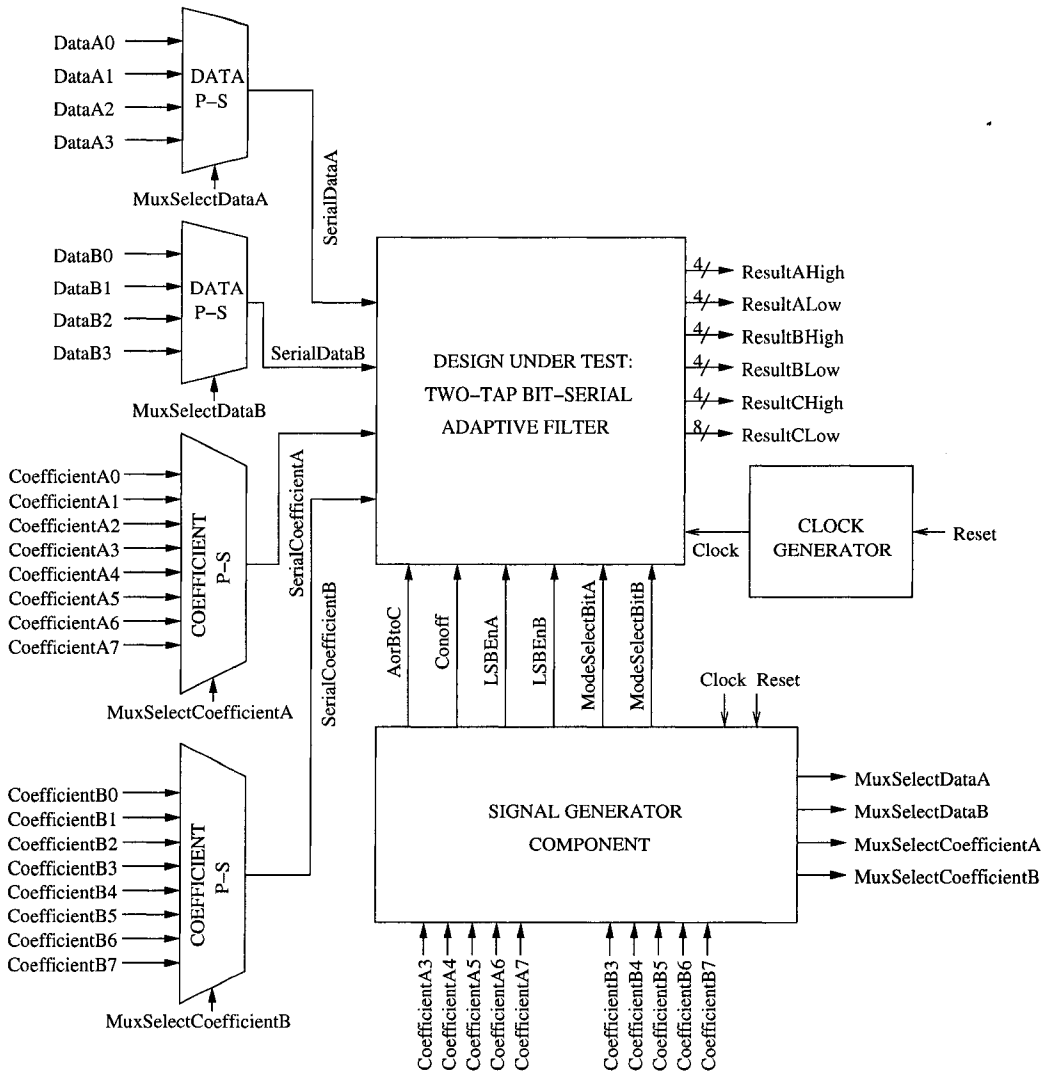

Figure 3.2: Schematic diagram of the bit-serial adaptive filter

3.2.1 Parallel-to-serial converter for the data term

A PSC has been designed to convert the 4-bit parallel inputs of the data term to a serial form such that the format of the data term is suited to the bit-serial nature of the filter. The block diagram of the data PSC is represented in Figure 3.3 and its schematic diagram is shown in Figure 3.4.

The PSC which is used for the transformation of the data term is basically a 4:1 multiplexer. The four single bit inputs, $Data_{3-0}$, are first stored in D flip-flops. The select inputs to the 4:1 mux, namely $MuxSelectData_{2-0}$, are provided by the signal

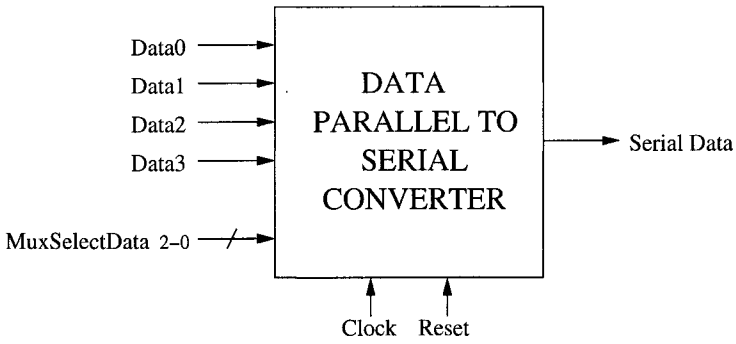

Figure 3.3: Block diagram of the data PSC

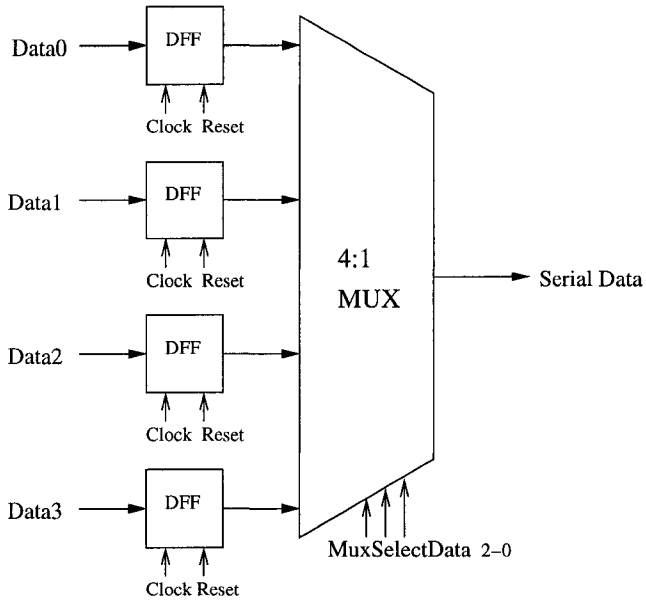

Figure 3.4: Schematic diagram of the data PSC

generator component. The output of the data PSC is the serial representation of the data term, i.e., the *SerialData* signal.

3.2.2 Parallel-to-serial converter for the coefficient term

The 8-bit coefficient term also enters the top-level circuit as a parallel input and is required to be converted to serial form for compatibility with the bit-serial filter. The block diagram of the PSC for the coefficient term is shown in Figure 3.5 and its schematic is represented in Figure 3.6.

Similar to the data converter, the coefficient PSC is also constructed from D

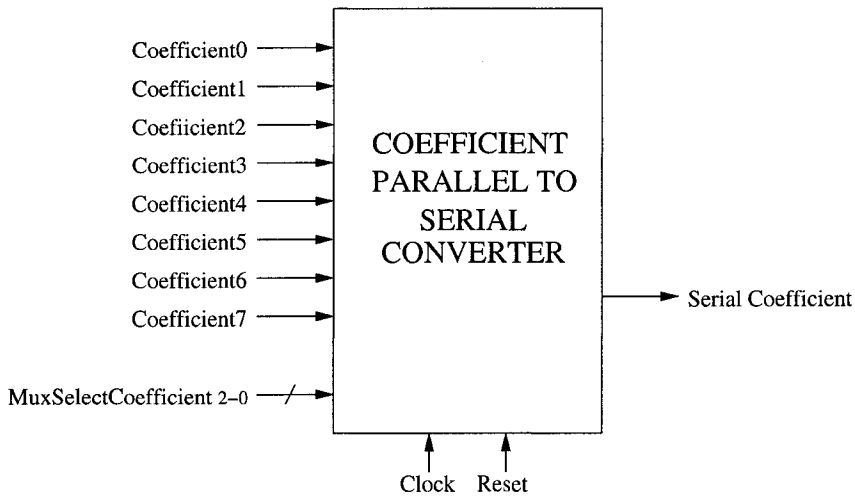

Figure 3.5: Block diagram of the coefficient PSC

flip-flops and a multiplexer. The select inputs to the 8:1 mux are provided by the signal generator. The output of the coefficient PSC, namely *SerialCoefficient*, is the serial form of the coefficient term. In an N-tap filter there would be N data PSCs and N coefficient PSCs. In the two-tap bit-serial adaptive filter, we have two of each.

Having described the generation of the inputs to the filter, the next section introduces the central component of the design, i.e., the bit-serial adaptive filter. The circuit illustration begins with an N-tap filter, followed by a detailed description of a two-tap version.

3.2.3 Filter component

An N-tap bit-serial adaptive filter component is presented in Figure 3.7.

The filter in our design is capable of functioning in 4-bit mode and/or in 8-bit mode simultaneously, and continuously process one frame every eight clock cycles. In the case of small echoes, the filter operates in 4-bit mode while when the echoes are large, it adapts to working in 8-bit mode. The section of the block that is required for 8-bit mode operation only, remains inactive when dealing with small echoes. This contributes to the area and power efficiency of the circuit. The power utilization issue has thus been addressed at the architectural level by making

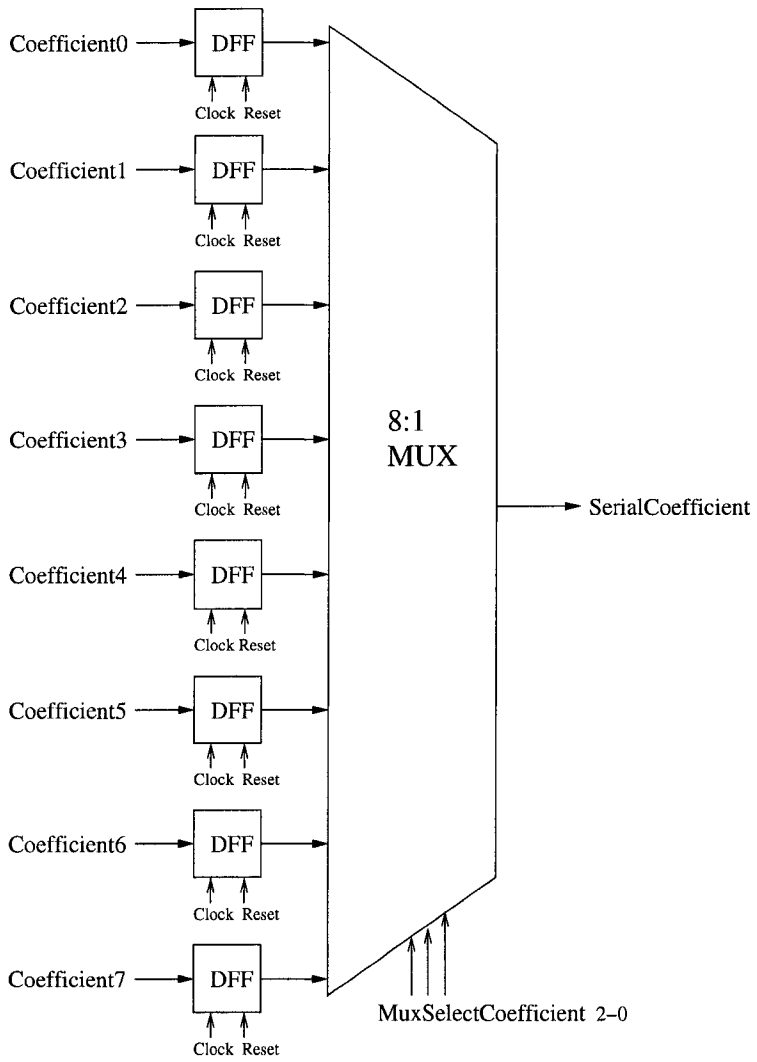

Figure 3.6: Schematic diagram of the coefficient PSC

a section of the block active only when the filter operates in 8-bit mode, i.e., only when dealing with large echoes.

Recalling from Chapter 1, large echoes arise due to impedance mismatches at the local hybrid. At this stage, the locally transmitted signal is early in its transmission path and hence the reflection of signal energy from the local hybrid are classified as large echoes. The residual echoes are those which occur due to mismatches at the remote hybrid. The source of these echoes is the locally transmitted signal which is in its latter stages of transmission. Hence, the far echoes are relatively smaller in size than the large echoes. Near or large echoes occur due to reflections

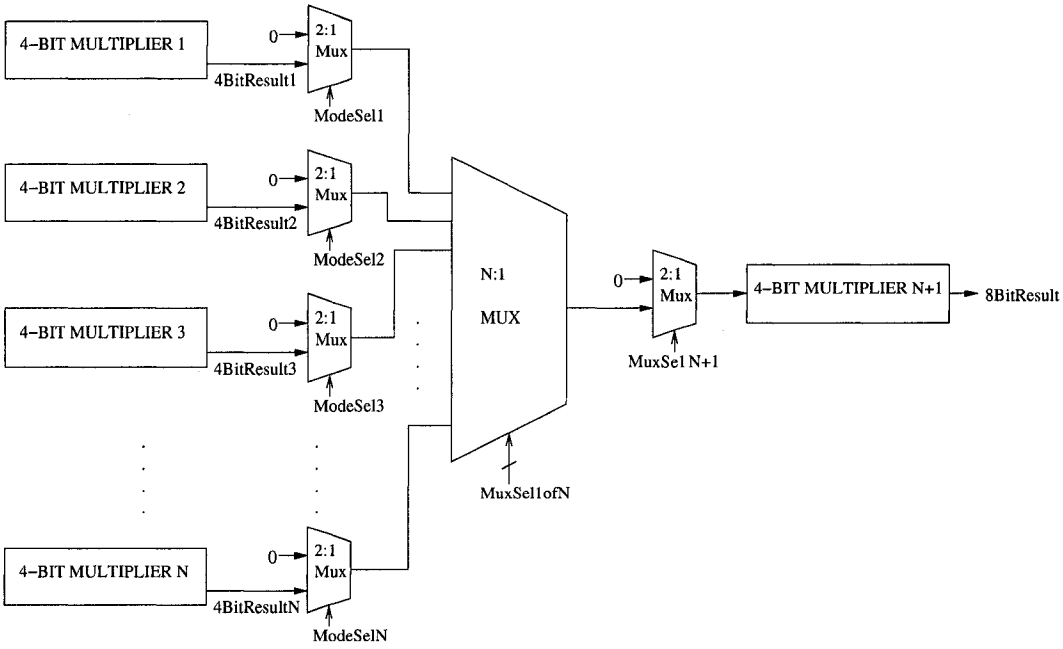

Figure 3.7: Bit-serial N-tap filter

at the first local hybrid and all the residual echoes occur due to far or small echoes caused by the mismatches at the remote hybrid, i.e., after the signal travels over the channel. Hence we infer that in the Gigabit Ethernet transmission system, the probability of encountering small echoes is greater than that of encountering echoes of larger sizes.

Accordingly in our filter unit we have a set of 4-bit multipliers, 1 to N that work independently in 4-bit mode. These are equipped to handle small echoes. N instances of $2:1$ multiplexers are used to indicate whether or not any of the 4-bit multipliers are used for concatenation with the 4-bit multiplier $N+1$. The select signals $ModeSel_{N-0}$ help make this selection. When a large echo signal is encountered, only one of the 4-bit multipliers is chosen to work in tandem with the 4-bit multiplier $N+1$. An $N:1$ multiplexer with select signal $MuxSel_{1ofN}$ is used to select which of the N multipliers joins the 4-bit multiplier $N+1$. If two of N taps try to simultaneously function in 8-bit mode, then there will be an error. However, when no large echo is present, the multiplier $N+1$ remains dormant. To choose between the active and inactive states of multiplier $N+1$, we have a $2:1$

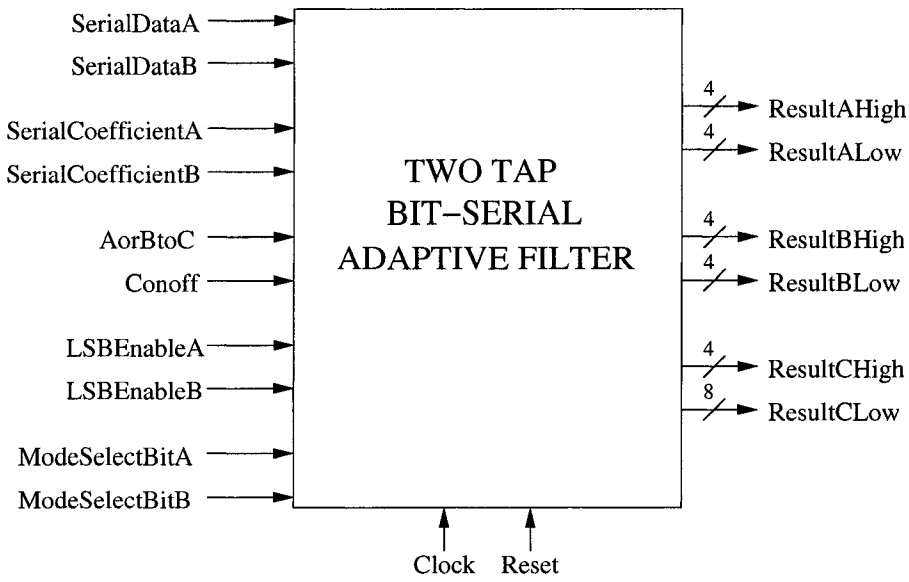

Figure 3.8: Block diagram of the bit-serial two-tap filter

multiplexer with select input $MuxSel_{N+1}$.

The multiplier component of the filter block is the same bit-serial multiplier design that was described in Chapter 2. The multiplier block incorporates the MBA, thereby increasing the efficiency of the design in terms of area utilization and speed of operation. By encoding the coefficient term bits using the MBA, the number of multiplication steps between the data and coefficient terms reduces. The reduction in the number of computation steps naturally, increases the frequency of operation. Fewer computation steps also implies fewer blocks in the series connection of individual multiplier cells, which in turn implies a reduction in hardware investment.

The top-level filter in Figure 3.1 instantiates a two-tap filter unit. The block diagram of the two-tap bit-serial filter is shown in Figure 3.8 and its internal diagram is represented in Figure 3.9.

The two-tap unit, which is a combination of three 4-bit multipliers A, B and C, incorporates the following functionality:

- A and B operate as 4-bit multipliers and C is off; or
- A operates as a 4-bit multiplier while B and C operate together as one 8-bit multiplier; or

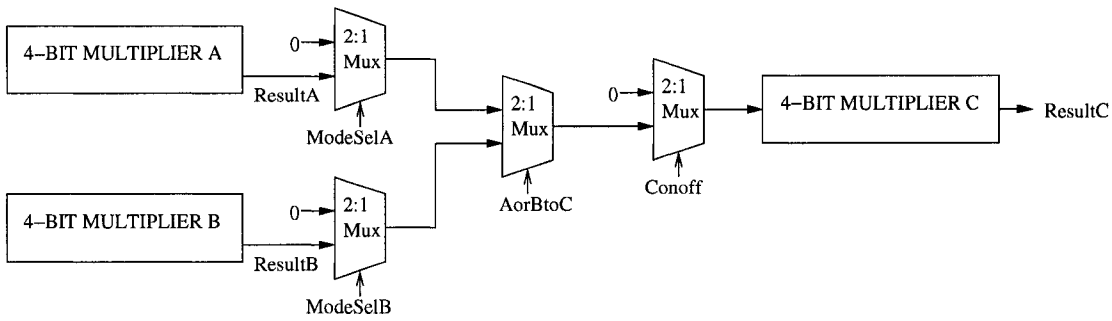

Figure 3.9: Schematic of bit-serial two-tap filter

- *B* operates as a 4-bit multiplier while *A* and *C* operate together as one 8-bit multiplier.

The above functions are carried out by means of 2:1 multiplexers.

The inputs to the filter entity are listed as follows:

- *Clock* signal
- *Reset* signal
- *SerialDataA*: The serial form of the dataA input which is generated by the PSC
- *SerialDataB*: The serial form of the dataB input
- *SerialCoefficientA*: The serial form of the coefficientA input which is generated by the PSC
- *SerialCoefficientB*: The serial form of the coefficientB input
- *LSBEnableA*: LSB enable for operands A
- *LSBEnableB*: LSB enable for operands B
- *ModeSelectBitA*: Mode select bit for multiplier *A*
- *ModeSelectBitB*: Mode select bit for multiplier *B*
- *Conoff*: A select signal which decides if multiplier *C* is operating or not

- *AorBtoC*: A select signal which selects which of the multipliers *A* or *B* works in series with multiplier *C*, for 8-bit mode.

A delay of four clock cycles was produced in the circuit. The outputs of the DUT, which are the most and least significant sections of the partial product terms, are listed as follows:

- *ResultAHigh*, *ResultALow*: High and low sections of the product of 4-bit multiplier A
- *ResultBHigh*, *ResultBLow*: High and low sections of the product of 4-bit multiplier B
- *ResultCHigh*, *ResultCLow*: High and low sections of the product of 8-bit multiplier C.

At the top-level entity, however, the only inputs are the individual bits of the data and coefficient terms. The filter unit described in this section, constitutes the actual filter component contained in the top-level filter circuit.

Multipliers *A* and *B* are 4-bit multipliers and are trained to handle small echoes. The outputs of multipliers *A* and *B* could either be used as results of 4-bit mode operation or they could be used as inputs to the 4-bit multiplier *C*. Connection to multiplier *C* is made only when a large echo is being handled. The mode select bits *ModeSelectBitA* and *ModeSelectBitB* aid the 2 : 1 multiplexers in deciding the operation mode of the two 4-bit multipliers *A* and *B*. The signal *AorBtoC* is the select input to the 2 : 1 multiplexer that selects one of the multiplier outputs *A* or *B* for concatenation with multiplier *C*. Multiplier *C* can either draw the outputs from one of the 4-bit multipliers or just remain inactive. Remaining inactive during small echoes is a power saving technique. The 2 : 1 multiplexer having a select input, *Conoff* determines the on/off state of multiplier *C*.

The filter inputs *ModeSelectBitA*, *ModeSelectBitB*, *AorBtoC* and *Conoff* are provided by the signal generator component. The LSB enable inputs namely, *LSBEnableA* and *LSBEnableB* are also computed in the signal generator box.

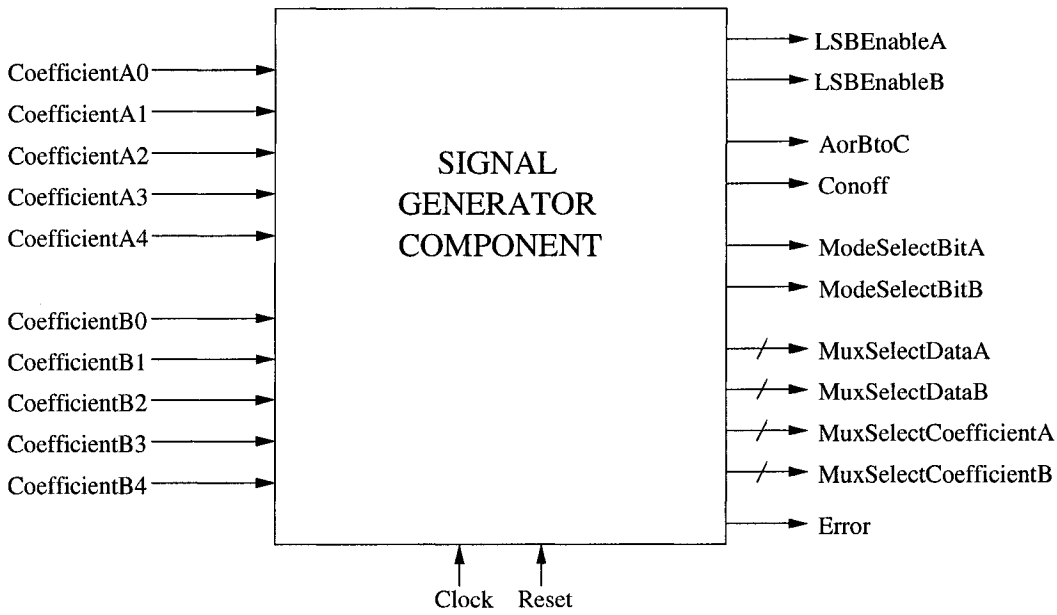

Figure 3.10: Block diagram of the signal generator

3.2.4 Signal generator component

The signal generator component has been designed to generate all the signals required by the filter unit and the PSCs. The block diagram of the signal generator component is shown in Figure 3.10.

The inputs to this entity are *Clock*, *Reset* and the first five MSB bits of the coefficient terms. From these inputs, the following signals are derived:

- *LSBEnableA*: LSB enable signal for the data and coefficient terms *A*
- *LSBEnableB*: LSB enable signal for the data and coefficient terms *B*
- *ModeSelectBitA*: Mode select bit for multiplier *A*
- *ModeSelectBitB*: Mode select bit for multiplier *B*
- *Conoff*: A select signal that decides if multiplier *C* is operating or not
- *AorBtoC*: A select signal that selects which of the multipliers *A* or *B* works in series with multiplier *C*, for 8-bit mode

Table 3.1: Working of the *ModeSelectBit*

ModeSelectBit	Operation of the multiplier
0	4-bit mode
1	8-bit mode

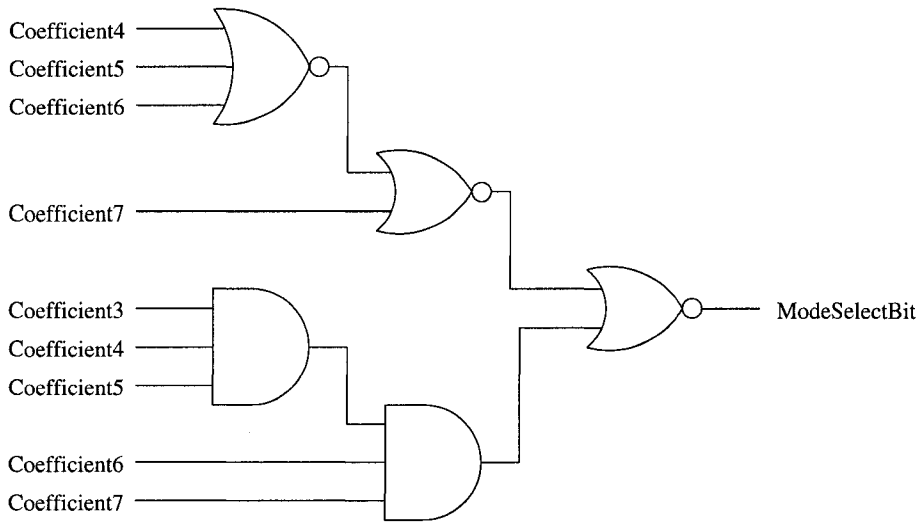

Figure 3.11: Generation of the *ModeSelectBit* signal

- *Error*: An error signal that goes high when both multipliers *A* and *B* want to operate in 8-bit mode
- *MuxSelectDataA*: The select signals for the dataA PSC
- *MuxSelectDataB*: The select signals for the dataB PSC
- *MuxSelectCoefficientA*: The select signals for the coefficientA PSC
- *MuxSelectCoefficientB*: The select signals for the coefficientB PSC.

The generation of each signal is explained in the following section.

3.2.4.1 *ModeSelectBit* signal

The *ModeSelectBit* is used to indicate if the 4-bit muliplier is working individually in 4-bit mode or in conjunction with another 4-bit multiplier for 8-bit mode operation. Its function is as listed in Table 3.1.

Table 3.2: Working of the *Error signal*

ModeSelectBitA	*ModeSelectBitB*	*Error signal*
0	0	0
0	1	0
1	0	0
1	1	1

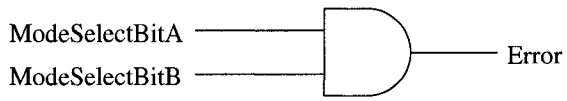

Figure 3.12: Generation of the *Error* signal

The *ModeSelectBit* signal is generated by comparing the first five MSB bits of the coefficient term. Generation of the *ModeSelectBit* is shown in Figure 3.11. In the two-tap filter, there are two signals *ModeSelectBitA* and *ModeSelectBitB*, which determine the mode of operation of the 4-bit multipliers *A* and *B* respectively. That 4-bit multiplier that is operating in 8-bit mode joins 4-bit multiplier *C*.

3.2.4.2 *Error* signal

When any two 4-bit multipliers in an N-tap filter try to work in 8-bit mode, the *Error* signal indicates an error. In the case of a two-tap filter, when both the 4-bit multipliers *A* and *B* try to join multiplier *C* for 8-bit mode operation, the *Error* signal goes high. The working of this signal is indicated in Table 3.2. The error signal is generated by means of a simple 2-input *And* gate. The logic operation is seen in Figure 3.12.

3.2.4.3 *Conoff* signal

In an N-tap filter, the $(N+1)^{th}$ 4-bit multiplier is used only in the case of a large echo. Likewise in a two-tap filter, the multiplier *C* is used only in the case of a requirement for 8-bit mode. For this purpose, we have a signal *Conoff*. The *Conoff* signal takes a value 1 when either one of the multipliers *A* or *B* wants to work in 8-bit mode. The mode select signals *ModeSelectBitA* and *ModeSelectBitB* are used to compute the signal *Conoff*. This select bit is generated as shown in Table 3.3.

Table 3.3: Working of the *Conoff* signal

ModeSelectBitA	*ModeSelectBitB*	*Conoff*
0	0	0
0	1	1
1	0	1
1	1	0

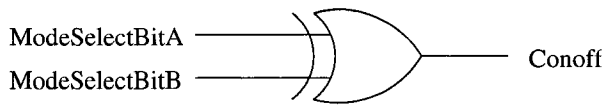

Figure 3.13: Generation of the *Conoff* signal

The logic is implemented in the form of a simple *Xor* gate, as shown in Figure 3.13.

3.2.4.4 *AorBtoC* signal

We have a signal to indicate which of the two 4-bit multipliers *A* or *B* would join multiplier *C* and execute in 8-bit mode. The truth table is shown in Table 3.4.

Table 3.4: Working of the *AorBtoC* signal

ModeSelectBitA	*ModeSelectBitB*	*AorBtoC*
0	0	X
0	1	1
1	0	0
1	1	1

3.2.4.5 Generic-counter

To provide the input select signals to the multiplexers present in the PSCs, we have designed a special counter and called it the *generic-counter*. Based on the mode select bit i.e. the *ModeSelectBit* input signal, the generic-counter also generates the LSB enable signals to the multiplier units in the filter. The generic counter itself instantiates two counters: the *data-counter* and the *coefficient-counter*. The block diagram of the generic-counter is shown in Figure 3.14.

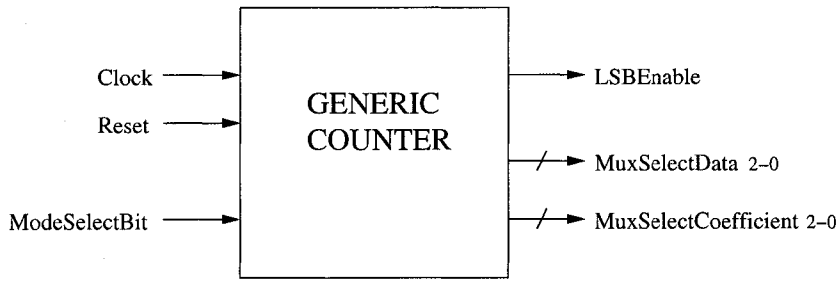

Figure 3.14: Block diagram of the generic-counter

Table 3.5: Counting action of the generic-counter

mode of operation	Data-counter	Coefficient-counter
4-bit mode	01230123	01230123
8-bit mode	01233333	01234567

The inputs to this block are the *Clock*, *Reset* and *ModeSelectBit* signals. The following signals are calculated:

- *LSBEnableA* and *LSBEnableB*: The LSB enable signals
- *MuxSelectDataA* and *MuxSelectDataB*: The select signals for the dataA PSC and dataB PSC converters
- *MuxSelectCoefficientA* and *MuxSelectCoefficientB*: The select signals for the coefficientA and coefficientB converters.

The generic-counter has four counter outputs. The data-counter provides the inputs to the data PSC, and the coefficient-counter provides the inputs to the coefficient converter. The counting results of the generic-counter for every eight clock cycles are listed in Table 3.5.

The signal generator component contains N generic-counters in the case of an N-tap filter and two for our two-tap implementation.

3.2.4.6 Data-counter

The block diagram of the data-counter is shown in Figure 3.15. Table 3.6 ilustrates the counting pattern of the data-counter in the two modes. The data-counter is a

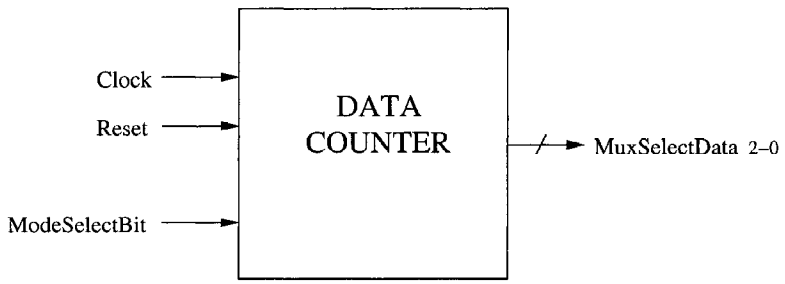

Figure 3.15: Block diagram of the data-counter

Table 3.6: Truth table of the data-counter

4-bit mode	8-bit mode
000	000
001	001
010	010
011	011
000	011
001	011
010	011
011	011

modified 3-bit up-counter. It provides the select signals to the multiplexer present in the PSC for the 4-bit data term. The counter counts differently when the multiplier coefficient term is four or eight bits wide. The inputs to the data-counter are *Clock*, *Reset* and the mode select bit, i.e., the *ModeSelectBit* signal. When $ModeSelectBit = 0$ the counter operates in 4-bit mode, and when $ModeSelectBit = 1$, the counter functions in 8-bit mode.

3.2.4.7 Coefficient-counter

The block diagram of the coefficient-counter is shown in Figure 3.16. The truth table in Table 3.7 describes the operation of the same. The coefficient-counter is also a modified 3-bit up-counter. It generates the select signals to the 8:1 multiplexer contained in the PSC for the 8-bit coefficient term. The coefficient-counter also generates the *LSBEnable* signals. The *LSBEnable* signal is a filter input which indicates to the filter, the arrival of the LSB bits of the data and coefficient term.

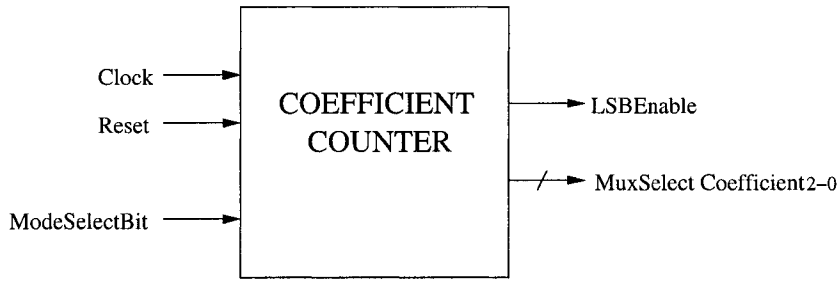

Figure 3.16: Block diagram of the coefficient-counter

Table 3.7: Truth table of the coefficient-counter

4-bit mode	*LSBEnable(4-bit mode)*	8-bit mode	*LSBEnable(8-bit mode)*
000	1	000	1
001	0	001	0
010	0	010	0
011	0	011	0
000	1	100	0
001	0	101	0
010	0	110	0
011	0	111	0

The coefficient-counter behaves differently during 4-bit and 8-bit modes. From the *Clock*, *Reset* and *ModeSelectBit* inputs, the coefficient-counter calculates the select inputs to the 8:1 multiplexer and the *LSBEnable* input to the filter component.

3.3 Testing the design

Testing of the DUT involved instantiating the DUT in a special code called as the *testbench*. Testing modules were instantiated in the testbench in order to set-up testing conditions and environment. This section goes on to describe the testbench entity in terms of the modules created for validating purposes. The following components were contained in the testbench of the top-level filter:

- Clock generator
- Databank
- Stimulator

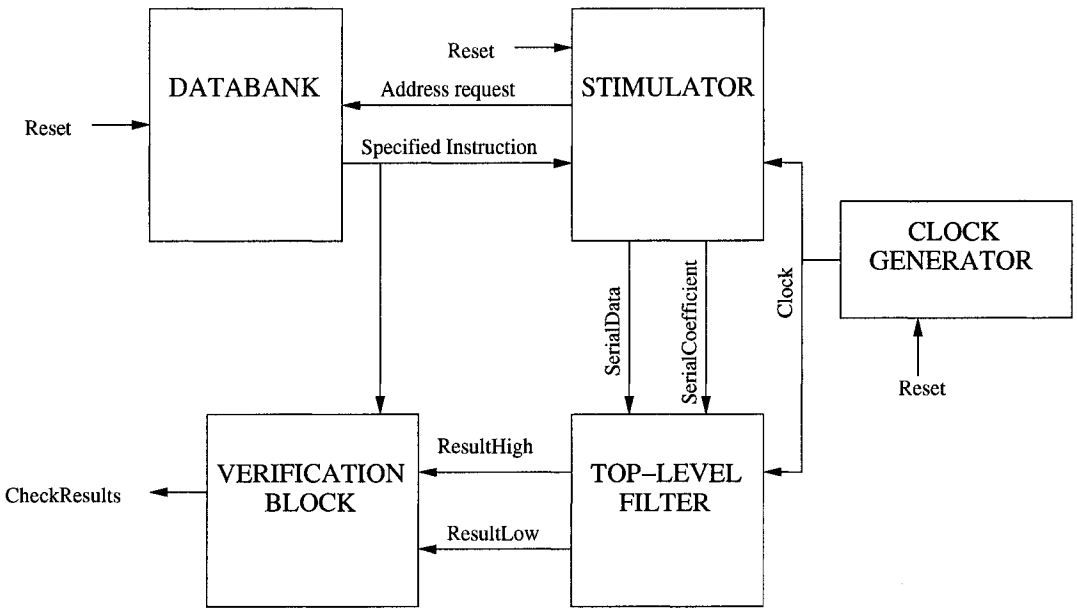

Figure 3.17: Block diagram of the testbench

- Verification block

The block diagram of the testbench with all the above components, is shown in Figure 3.17.

A *Matlab* code was used to generate a set of data and coefficient terms. The collection of operands were contained together in an array referred to as the *instruction array* and placed in a package code. The operation of the testbench is as follows: The stimulator block issues an address request to the databank. Based on the address request, the databank chooses a particular *instruction* from the *instruction array* and returns it to the stimulator block. The stimulator block processes the *instruction* and provides the top-level filter component with the necessary inputs. The filter outputs are then sent to the verification block. The verification block performs a mathematical calculation of the operands based on the *instruction* input and displays a signal *CheckResults*. This signal compares the actual results (from the DUT) and the expected results (calculated version) and helps in verifying the correctness of the DUT.

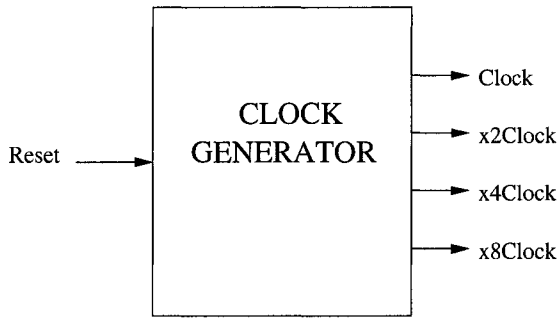

Figure 3.18: Block diagram of the clock generator

3.3.1 Clock generator

Stimulated by the reset signal, the clock generator was made to generate a base clock. By manipulating the base clock, this block was also capable of generating clocks of multiple sampling rates, i.e., x2, x4, x8, etc. The clock generator is shown in Figure 3.18.

3.3.2 Databank

The databank defines an array which is an integrated collection of the data term, coefficient term, mode select bit and the array index term, which represents the position of an *instruction* in the databank array. This array is referred to as the *instruction array*. Our databank contained about 1000 terms.

The databank refers to the bit-serial package where the *instruction array* has been defined. The databank receives an address request from the stimulator block to fetch a particular instruction from the *instruction array* and inturn provides the specified instruction to the stimulator block. The block diagram of the databank block is represented in Figure 3.19.

3.3.3 Stimulator

The stimulator block executes the instruction that it receives from the databank. The stimulator block processes the instruction and submits the data and coefficient terms to the top-level filter unit. The stimulator works as a synchronous Finite State Machine (FSM), whose states are defined below:

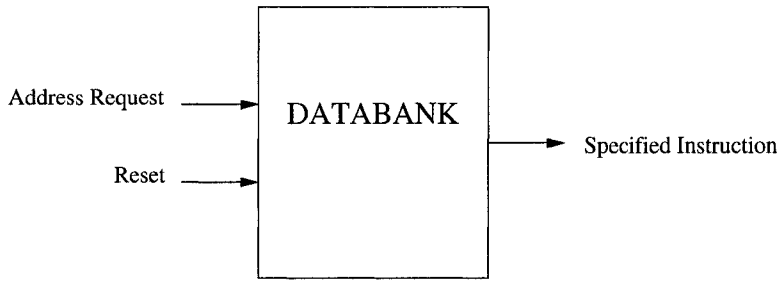

Figure 3.19: Block diagram of the databank

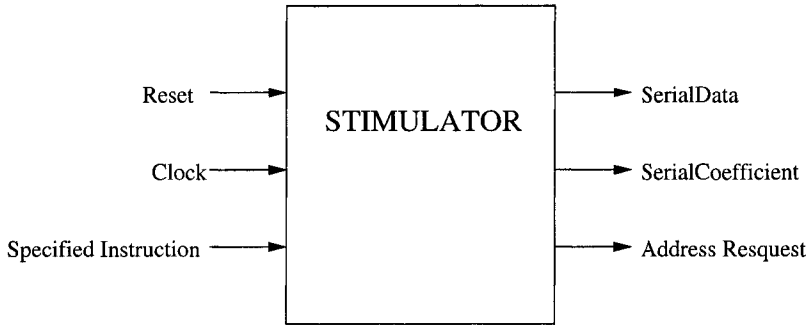

Figure 3.20: Block diagram of the stimulator

- Get the first instruction
- Process the first instruction
- Fetch the next instruction.

The FSM is a synchronous design and begins on *Reset*. Internal registers are used to store the intermediate records. The block diagram of the stimulator is represented in Figure 3.20.

3.3.4 Verification block

The verification block is a post-processing entity and is used to verify the correctness of the DUT. It obtains its inputs from the databank, DUT, and stimulator and generates an output log file that lists the outputs of the multiplication process in integer notation. It calculates the expected results and places it in the log file as well. The output log file displays the expected versus actual results, thereby validating the design. The block diagram of the verification unit is represented in Figure 3.21.

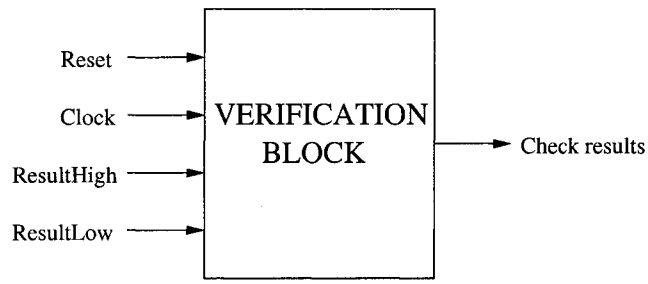

Figure 3.21: Block diagram of the verification block

3.3.5 Package

The package code basically defines a two-dimensional *instruction array*, the two parameters being the array dimensions: length and height. The length of the *instruction array* is the length of each instruction while the height of the array refers to the total number of instructions. The *instruction array* thus contains the inputs to the DUT, i.e., the data and coefficient term bits. Each *instruction* consists of the data term, coefficient term, mode select bit and the array index. The databank addresses the *instruction array* by referring to the package code.

3.3.6 Use of *Matlab*

The *instruction array* is generated by a *Matlab* code. It is generated as a set of random numbers, having two user inputs. One is for the number of instructions and the other is for the biggest data size. The *instruction array* thus obtained is fed to the VHDL codes by means of a package code. The DUT refers to the package code and executes based on the inputs provided in the *instruction array*. The results of the DUT are processed by the verification block, which compares the actual and expected results. Another *Matlab* code is used to plot the errors in the DUT computation. This *Matlab* file accepts the log file generated from VHDL, plots the expected versus actual results, finds the errors, calculates the size of the errors and plots the absolute value of the errors. Also, the code finds those results which do not contain errors, calculates this size and plots the absolute value of the same.

3.4 Conclusion

Having designed and tested the bit-serial adaptive filter, let us now see its characteristics when implemented on FPGAs. To check if the bit-serial architecture has really made a considerable advantage in terms of area utilization, we have compared our serial design with a corresponding parallel filter. FPGA implementation of the bit-serial adaptive filter and a comparison of the serial and parallel filters with respect to FPGA characteristics, are presented in the following chapter.

Chapter 4

FPGA implementation of the bit-serial adaptive filter

This chapter describes the FPGA hardware implementation of the bit-serial adaptive filter that was presented in the previous chapter. The top-level circuit of the filter component has been considered as a hierarchical model and described using an HDL. For the design to be read by physical design tools, the design entry can be made using HDL or schematics. The HDL codes or schematics are then synthesized to obtain an optimized net-list.

The serial adaptive filter was implemented for up to seven tap coefficients using the *Xilinx-ISE-6.2.02i* tool, and synthesized for the *Spartan* FPGA device. The design characteristics have been studied and the actual advantages of our filter have been determined by comparing them with a corresponding filter based on parallel architecture.

In this chapter, an introduction to logic design, translation of our design to hardware using VHDL, synthesis of our design for the *Spartan* FPGA device and the performance comparison of our serial implementation with an equivalent parallel realization are presented.

4.1 Introduction to logic design

In this section, we briefly review some of the concepts in logic design [10].

4.1.1 Top-down design process

In the top-down design methodology, we start with the topmost block and connect below each block, those blocks from which it is made. The highest level circuit is repeatedly divided into blocks and sub-blocks in order to complete the design. This makes the designing process organized and modular.

4.1.2 The hierarchical approach

The primary advantage in adopting the hierarchical approach is the reduced complexity required to represent the schematic diagram of a circuit. Another important benefit is the property of reusability. A block is reusable in the sense that it can be used in more than one location in the circuit and perhaps, in other circuits as well.

4.1.3 Computer aided design

Computer-Aided Design (CAD) tools are used for the design of simple or complex digital circuits and ICs. The *Xilinx-ISE-6.2.02i* software was utilized for the synthesis of a circuit from its high-level textual descriptions. Verification of the behavior of the hierarchical blocks was done by making use of the *ModelSim* logic simulator tool.

4.2 Hardware description languages

Hardware Description Languages (HDL) are similar to programming languages but are specifically tailored to describe hardware structures and their behaviors. The most important feature of HDLs is that they exhibit extensive parallel operations while most programming languages work sequentially. Hardware languages are used to give structural description by describing the interconnection of components in a circuit. These structural descriptions are then used as input to logic simulations.

Using the top-down approach, an HDL can be be used to make a complete high-level structural description of an entire system. Some of the reasons responsible for the popularity of HDLs are summarized below:

- Use of the top-down design methodology makes HDLs modular and convenient.
- Compatibility with logic synthesis tools permits the conversion of an HDL description into an interconnection of primitive components that implement the circuit.
- Portability across various CAD tools.

There are two standard HDLs, namely, VHDL [9] and Verilog. Throughout this thesis VHDL has been used in the translation of our design to hardware. VHDL was developed under contract for the U.S Department of Defense as a part of their Very-High-Speed Integrated Circuits (VHSIC) program and subsequently became a standard IEEE language.

There is a typical set of procedures [4] when using the HDL as a simulation input. These processes include analysis, elaboration, initialization and finally simulation. The *analysis* step checks the HDL for errors in syntax and semantic rules and provides an intermediate representation of the design. In the *elaboration* procedure, the design hierarchy is flattened to an interconnection of modules and a simulation model is generated and passed to the simulator. The simulator executes the simulation model with inputs specified by the user. In order to test the operation of the design, a testbench is used to specify the inputs and testing conditions.

After the schematic or HDL code has been entered, the design is read by the *Xilinx-ISE* [4] software. The *Xilinx-ISE* synthesis tool automatically partitions the design into logic blocks, finds an optimal placement for each block, and selects the interconnect routing. The user can enforce specific area, power and timing constraints and also, selectively edit critical path portions of the design. Once the design is compiled, a detailed timing report is generated. A serial bitstream is produced which can be downloaded into the FPGA device.

4.3 Introduction to FPGA devices

FPGAs are user-programmable devices that come in multiple families, different sizes and demonstrate different characteristics with respect to speed, temperature ranges, costs etc. In an FPGA implementation, a user selects an appropriate device, enters the logic design into the *Xilinx-ISE* development system software, and then loads the resulting configuration file into the FPGA device.

An FPGA device is made up of a regular array of Configurable Logic Blocks (CLBs), surrounded by a set of programmable Input-Output Blocks (IOBs). The design function is implemented by means of Look Up Tables (LUTs). The device utilization of an FPGA implementation is analyzed with respect to the percentage usage of CLBs, LUTs and IOBs. For our implementation, the *Spartan* FPGA device family which is fully supported by *Xilinx-ISE* development system- Foundation ISE series, has been selected.

A brief introduction to *Spartan* FPGA devices is presented in the subsequent section.

4.3.1 Introduction to the *Spartan* FPGA device family

The $Spartan^{TM} - II$ FPGA is a six member family, from which the *Spartan* XC2S15 device has been selected for our implementations. The *Xilinx* XC2S15 device consists of an 8×12 array of CLBs, each containing 4-input LUTs, registers, special purpose features such as RAM cells and fast carry logic, and also routing paths between the CLBs and IOBs. Table 4.1 contains the features of this device, as quoted in the $Spartan^{TM} - II$ *2.5V FPGA Family: Complete Data Sheet* [5].

The basic $Spartan - II$ Family FPGA block diagram, as illustrated in the $Spartan^{TM} - II$ *FPGA data sheet* [5], is represented in Figure 4.1.

Table 4.1: *Spartan – II* FPGA device: XC2S15

Feature	Details
Logic Cells	432
System Gates(RAM and logic)	15,000
CLB Array	8 x 12
Total CLBs	96
Max available User I/O	86
Total distributed RAM bits	6144
Total block RAM bits	16K

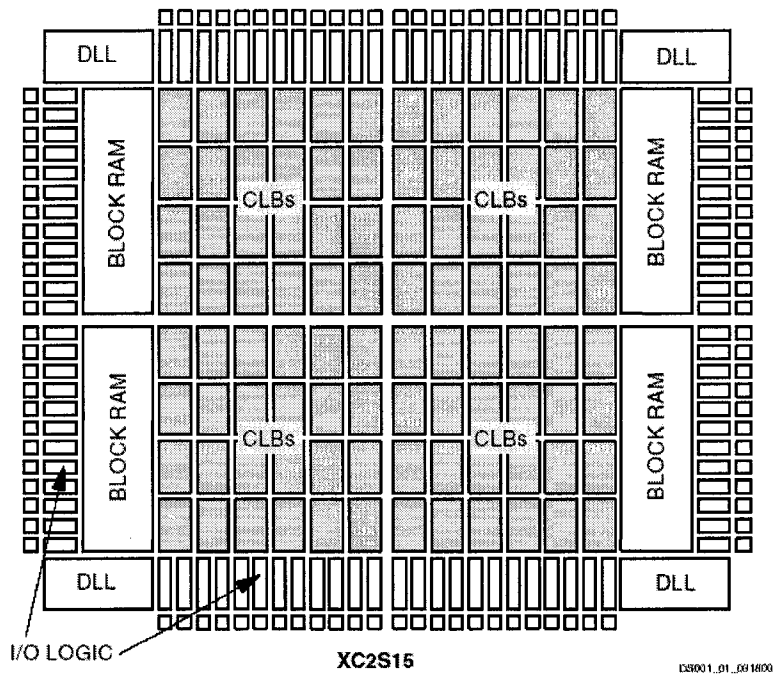

Figure 4.1: Block diagram of the *Spartan*^{TM} *– II FPGA* device; Courtesy: *Spartan*^{TM} *– II FPGA data sheet*

4.4 FPGA implementation of the bit-serial filter

The VHDL hardware description language was used to synthesize the bit-serial adaptive filter using the *Xilinx-ISE-6.2.02i* software for the *Spartan* FPGA device. The serial adaptive filter unit was tested using the *ModelSim* software for a continuous stream of 1000 frames. A 4-bit mode operation takes place once in every four clock cycles, while an 8-bit mode operation takes place once in every eight clock cycles. The latency in the circuit was found to be four clock cycles.

Recalling the filter operation, small echoes imply that the filter is operating in 4-bit mode and large echoes indicate 8-bit mode operation. The processing of consecutive frames translates to the cancellation of a continuous stream of large and small echoes.

4.5 Performance comparison of the serial and parallel filters

The bit-serial adaptive filter was studied in terms of FPGA device utilization. In order to realize the advantages of our bit-serial filter, a standard bit-parallel filter was used and the two filters were compared for gate count and usage of device resources. The two filters have been compared likewise, for structures of two to seven tap coefficients. Table 4.2 presents a gate count for the two filters. The comparison is made graphically in Figure 4.2.

Tables 4.3- 4.4 and Figures 4.3- 4.5 estimate device utilization in terms of the number of CLB flip-flops, LUTs, IOBs and actual gate count. Table 4.3 represents the device utilization of the bit-serial filter, while Table 4.4 represents the device utilization of the corresponding bit-parallel filter.

Table 4.2: Gate count comparison

No. of filter taps	No. of gates in the serial filter	No. of gates in the parallel filter
2	1084	1440
3	1559	2160
4	1953	2880
5	2402	3600
6	2790	4320
7	3184	5040

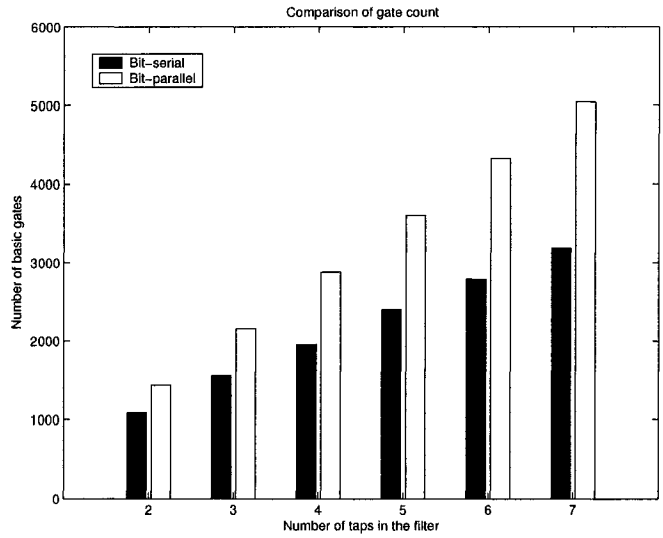

Figure 4.2: Gate count comparison

Clearly, our serial adaptive filter has a much lower gate count. This is indicative of use of fewer hardware resources compared to the equivalent parallel design.

Device utilization in terms of the number of CLB slices refers to the number of columns used in the CLB array. The area efficiency is mainly measured in terms of the CLB usage. *Xilinx* FPGAs implement the combinational logic in small LUTs (16 x 1 ROMs). Each such table either feeds the D-input of a flip-flop or drives other logic or I/O, thus responsible for the actual implementation of the function. Tables 4.3- 4.4 contain the exact utilization statistics of the two filters.

Table 4.3: Device utilization of the bit-serial filter

No. of filter taps	No. of slices(%)	No. of LUTs(%)	No. of IOBs(%)
2	27	17	25
3	39	26	33
4	49	33	42
5	60	40	49
6	70	47	52
7	79	54	70

Table 4.4: Device utilization of the bit-parallel filter

No. of filter taps	No. of slices(%)	No. of LUTs(%)	No. of IOBs(%)
2	71	62	71
3	107 †	93	106 †
4	145 †	125 †	142 †
5	179 †	156 †	177 †
6	215 †	187 †	213 †
7	251 †	218 †	218 †

†: More than 100 % of device resources are used

Tables 4.3 and 4.4 show that the serial adaptive filter makes better usage of the device resources, in terms of the number of CLBs, LUTs and IOBs. The percentage of CLB slices used in the parallel approach indicates more than 100% usage of a single FPGA device. This suggests that a function which requires multiple FPGAs for its parallel implementation can be conveniently fitted on a single FPGA device using a corresponding serial architecture.

The comparison in CLB usage between the serial and parallel filters is shown in Figure 4.3, while the comparison of LUT usage is seen in Figure 4.4. The two filters have also been compared in terms of their IOBs. This is shown in Figure 4.5. The advantage enjoyed by the bit-serial design is that all of the bits pass through the same logic, resulting in a huge reduction in the required hardware.

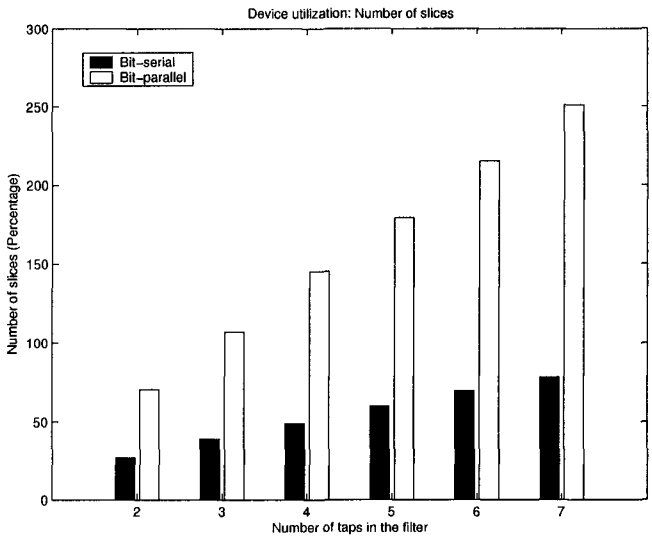

Figure 4.3: Device utilization: CLB usage

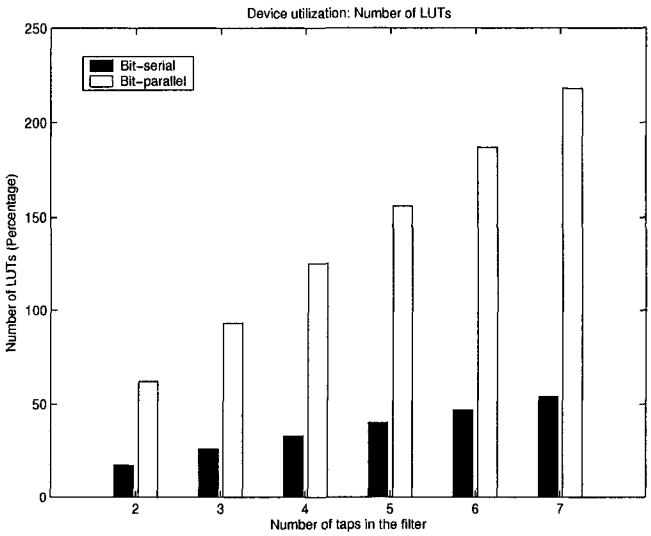

Figure 4.4: Device utilization: LUT usage

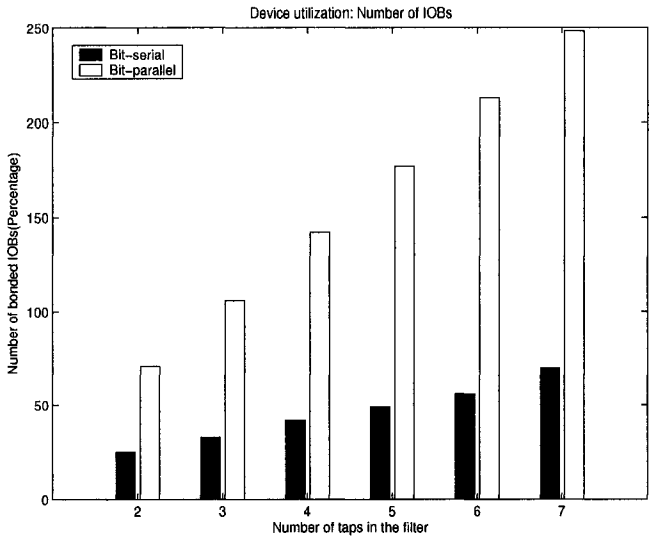

Figure 4.5: Device utilization: IOB usage

4.6 Conclusion

A parameterized unit has been realized on the *Xilinx-ISE* platform for the XC2S15 *Spartan* FPGA device. Earlier, it was seen that the bit-serial multiplier demonstrated superior characteristics as compared to the bit-parallel multiplier with respect to area, power and timing. The comparison between the serial and parallel filters presented in this chapter has shown that the serial filters are also better suited to FPGA implementations. It has been shown that a complex function can be fitted onto a single FPGA using the bit-serial approach, while the bit-parallel approach may require a set of FPGAs.

Bit-parallel designs simultaneously process all the bits contained in an input sample at the cost of significant hardware resources. A serial realization processes the input one bit at a time, using the results of the operations on the first bits to influence the processing of the subsequent bits. This results in a huge reduction in the use of hardware resources. It is important that complex architectures of an echo cancellation unit do not dominate the receiver sector of a digital communication system. The serial approach definitely caters to this need.

Chapter 5

Mapping the filter to an ASIC

The performance characteristics obtained from the FPGA implementations, suggest that the proposed echo cancelling filter design is a promising candidate for effecient ASIC implementation.

The bit-serial adaptive filter has been synthesized using the *Synopsys* Design Compiler (DC) tool, version 2003.06. DC [2] synthesizes the HDL description into a technology-dependent, gate-level design. Design optimization can be done with respect to speed, area, power and routability.

The area, timing and power characteristics of our design was studied by synthesizing the design using the DC tool. The area and timing constraints were varied to obtain an optimized gate-level net-list. The critical path in each module was traced and reduced by means of pipelining. An analysis of the hierarchical model of the bit-serial adaptive filter is presented in this chapter.

5.1 Motivation

The target of this ASIC implementation is to establish the echo cancellation unit as area and power efficient and, simultaneously have it running at frequencies suitable for Gigabit Ethernet. Area, speed and power constraints, therefore, direct the optimization steps of the synthesis process.

Area minimization is performed since it is important that the architecture of the echo cancellation unit does not crowd the receiver section of a digital communication system. In order to make it a low power device, efforts have been made to reduce the power consumption. Upper limits to constrain the area and power consumption can be defined, or a general directive to minimize the area or power usage can be specified. Because our application interests lie in high-speed wire-line transmission systems, achieving high clock rates is a must.

5.2 The synthesis process

The general steps followed by the synthesis process, as listed in the *Synopsys^{TM}Manual*, [2] are summarized below.

- Read in the design and its sub-designs.
- Set design attributes for the top-level design.
- Define realistic optimization goals for timing, area or power and set the design constraints.
- Run *check-design* to identify and correct errors.
- Perform optimizations.
- After compilation, obtain the constraint reports to determine whether design goals are met or not.

The synthesis process is also represented by means of a flowchart in Figure 5.1.

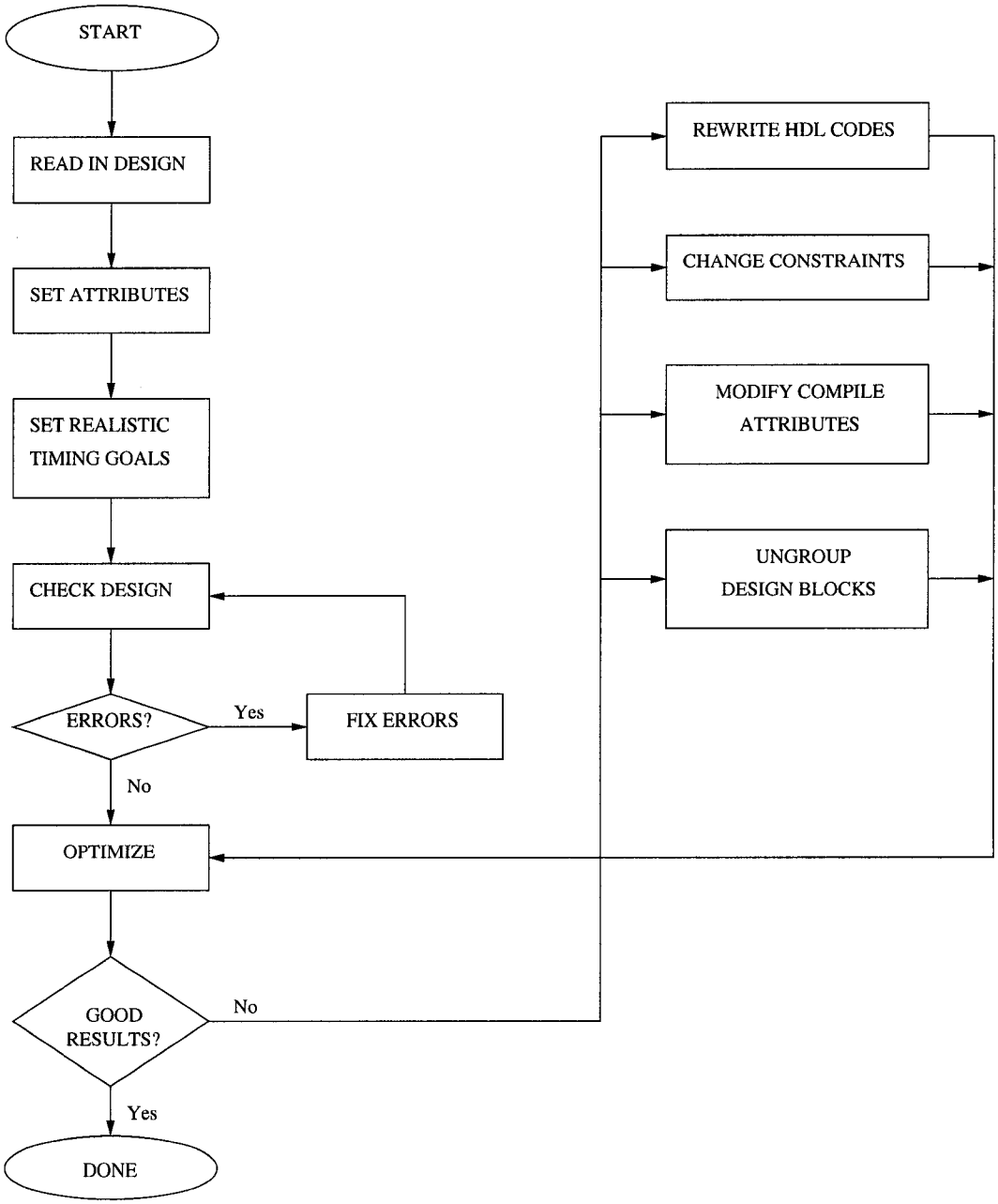

Figure 5.1: The synthesis process- Courtesy: *Synopsys^{TM}Manual*

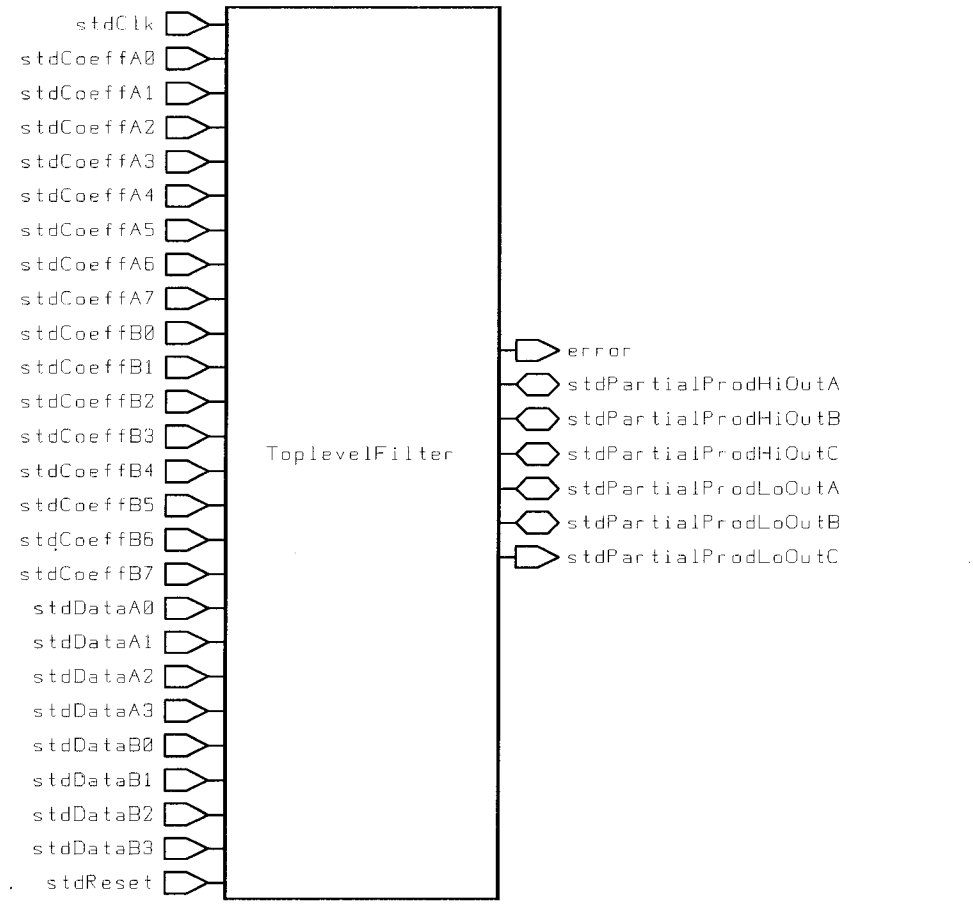

Figure 5.2: Symbol view of the top-level filter

This chapter presents simulation models of the top-level filter in the form of a hierarchical design. It also presents an analysis of the area, speed and power characteristics of all the modules.

5.3 Top-level filter

The top-level filter assumes the highest level in the hierarachy. The bit-serial adaptive filter, the signal generator component and the two parallel-to-serial converters are contained in it. VHDL codes for the hierarchical model of the top-level filter were loaded into the DC and the synthesis steps, i.e., *analyze, elaborate and compile* were carried out. The end result is a simulation model of the original HDL description. Figure 5.2 represents the symbol view of the top-level filter unit.

The inputs to the top-level filter block are the data and coefficient terms that

enter the circuit in the form of parallel lines. The top-level filter in Figure 5.2 instantiates a two-tap filter and hence there are two sets of inputs, i.e, A and B for the data and coefficient terms. The design is synchronous and also has a Reset input. The outputs of the top-level entity are the results of 4-bit and/or 8 bit operations. The error signal indicates if any two of the N multipliers are simultaneously trying to function in 8-bit mode.

The schematic diagram of the top-level filter for a two-tap filter unit, obtained by synthesizing the design using the *Synopsys* DC tool, is shown in Figure 5.3. The schematic diagram shows the interconnection of all the sub-blocks in the top-level filter. The schematic also illustrates the critical path in the circuit.

5.3.1 Critical path

Critical path is defined as the longest path in a circuit. To drive a design at higher clock frequencies, it is essential to trace and reduce the critical path delay. One of the widely used methods to reduce the criical path delay is *pipelining* [8].

Pipelining uses registers to improve the throughput of logic circuits. Throughput is defined as the rate at which each new output appears. In the pipelining process, the computation is divided into stages that take approximately the same time. These stages are separated by pipeline latches in order to isolate them. The critical path in the top-level circuit has been highlighted in the schematic of Figure 5.3. This is the final critical path in the circuit that gets retained after pipelining. It appears in the generation of the most significant section of the partial product. This path is inevitable and cannot be reduced further.

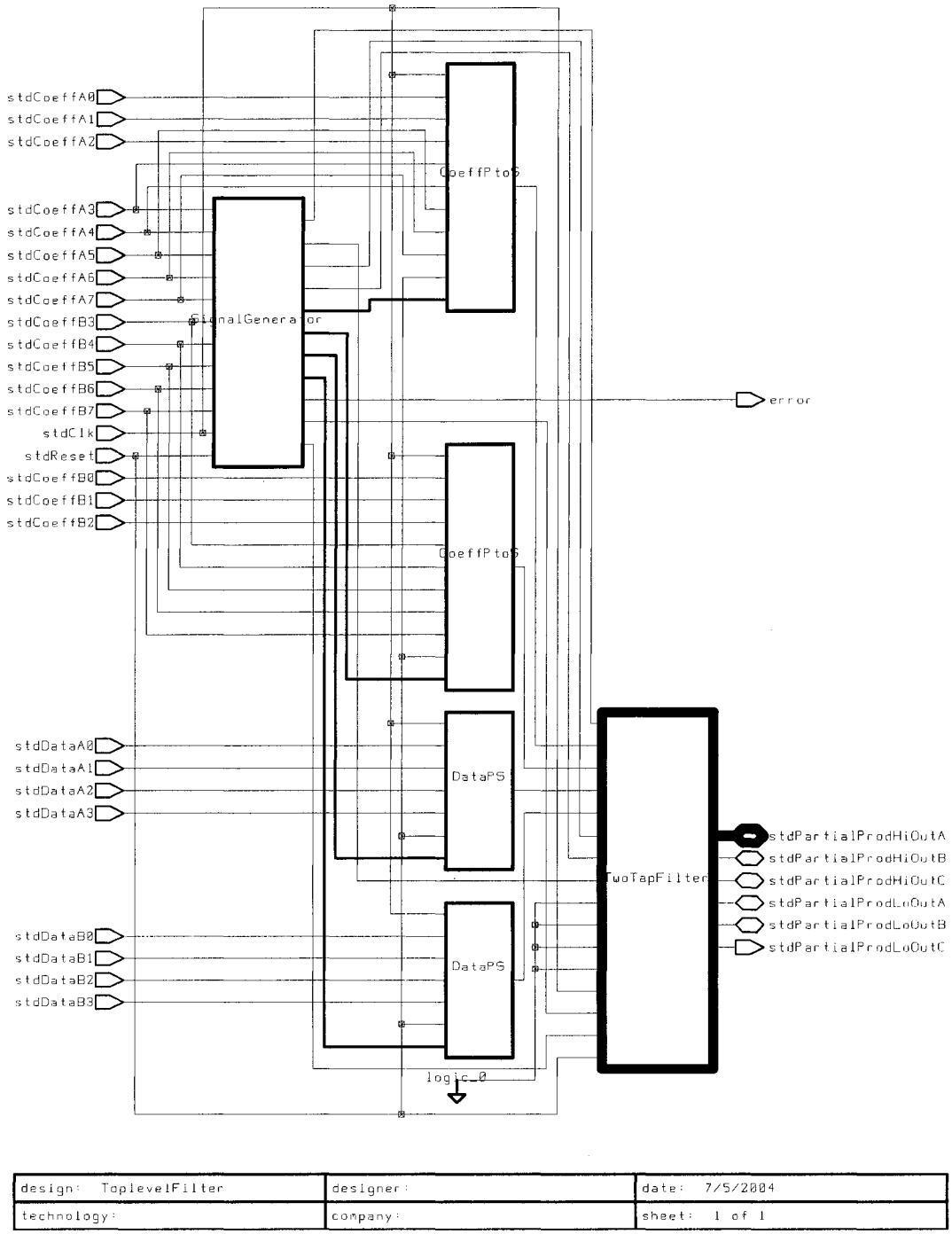

Figure 5.3: Schematic view of the top-level filter

Table 5.1: DC report of the top-level filter

Design:Top-level filter	Report
Total cell area	19124.6 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	19.6443 mW
Maximum clock frequency	0.7936 GHz

5.3.2 Frequency of operation

To obtain the maximum clock frequency of the circuit, the input clock period was decreased until the timing parameters were violated. The top-level filter was able to operate at clock frequencies close to 800 MHz. This high clock frequency of the top-level filter circuit supports the intended application of our design in high-speed communication systems. The cell area and power consumption at the highest achieved clock frequency are summarized in Table 5.1. The detailed area, power and timing reports obtained from the *Synopsys* DC have been attached in the Appendix section.

The variation of area and power with frequency has been studied, for the hierarchical model of the bit-serial adaptive filter. The values of cell area and dynamic power consumption were noted at decreasing clock periods and plotted as a function of frequency.

5.3.3 Area variation with frequency

Starting at the maximum frequency of operation, the cell area of the top-level filter was measured for a frequency range of 793.65 MHz down to about 330 MHz. These values are shown in Table 5.2. The same is graphically represented in Figure 5.4.

From Table 5.2 and Figure 5.4, it is seen that as the clock period increases or when the clock frequency decreases, the cell area also decreases. After a certain clock frequency, the cell area stabilizes and remains constant at all lower frequencies. The cell area at the maximum frequency of 793.65 MHz is 19124.58 um^2. It gradually decreases to attain a stable value of 16799.95 um^2 at 400 MHz.

Table 5.2: Area variation with frequency

Clock period (ns)	Clock frequency (MHz)	Cell area (um^2)
1.26	793.65	19124.58
1.28	781.25	18795.27
1.29	775.19	18750.55
1.30	769.23	18616.38
1.31	763.36	18453.26
1.40	714.28	18047.20
1.50	666.67	17880.51
1.60	625.00	17791.07
1.70	588.24	17303.19
2.00	500.00	17152.56
2.50	400.00	16799.95
2.80	357.14	16799.95
3.00	333.33	16799.95

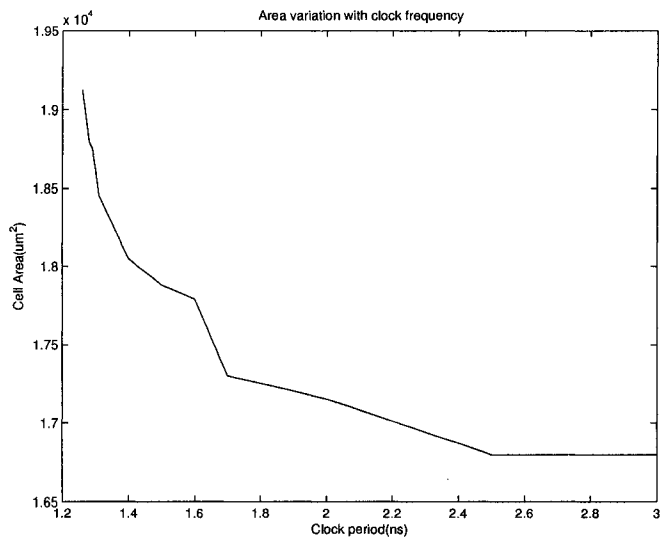

Figure 5.4: Variation of area with frequency

Having specified the design as *area critical*, the DC makes an effort to make the design as compact as possible. It does so, at the cost of speed. Hence, a trade-off is observed between frequency and area.

Table 5.3: Power variation with frequency

Clock period (ns)	Clock frequency (MHz)	Total dynamic power (mW)
1.26	793.65	19.6443
1.30	769.23	19.0331
1.40	714.28	17.4706
1.50	666.67	16.3125
1.60	625.00	15.2392
1.70	588.24	14.2400
2.00	500.00	12.0170
2.50	400.00	9.5710
3.00	333.33	7.9669
5.00	200.00	4.7802
10.0	100.00	2.3901
20.0	50.000	1.1950
50.0	20.000	0.4780
100	10.000	0.2390
200	5.0000	0.1195
500	2.0000	0.0478
1000	1.0000	0.0239

5.3.4 Power variation with frequency

The variation of dynamic power consumption was observed between the highest operating frequency of 793.65 MHz and the considerably lower value of 1 MHz. The variation of power consumption with clock frequency shown in Table 5.3. The power values have been plotted against a clock period range of 1.26 ns to 200 ns and the power variation with frequency is seen in Figure 5.5.

We see that the power usage decreases with decreases in clock speed. At the maximum clock frequency of 793.65 MHz, the power consumption was 19.6443 mW. The power value falls to a relatively stable value at a frequency of 100 MHz. The difference in power utilization at the maximum frequency of 793.65 MHz and that at 100 MHz is a significant value of about 18 mW. We notice that power consumption is at its peak when the circuit operates at its maximum frequency and takes on decreasing values with the decline in frequency. Hence, we can say that there exists a trade-off between power consumption and frequency of operation.

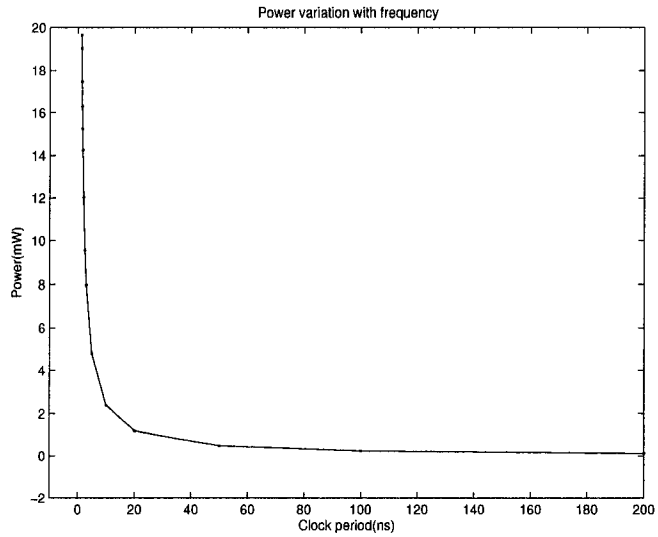

Figure 5.5: Variation of power with frequency

In order to conform to a standard, we normalize the values of power with respect to the maximum clock frequency. The normalized power values are therefore used to make a like-to-like comparison. At each frequency step, the raw power values are expressed as a percentage increase or decrease with respect to the power at the maximum operating frequency. Normalized values for the power readings in Table 5.3 have been calculated. Figure 5.6 is a graph wherein the normalized values of power are plotted for increasing clock periods.

We observe that as the operating frequency decreases, the power consumption also decreases. After a frequency of about 200 MHz, the power consumption normalizes to a stable value. If the goal is to minimize power then we need to run the circuit as slowly as possible while if the goal is to run the circuit at the maximum possible frequency, then a compromise on power consumption is made. Depending on the design application, the optimization constraints can be set in order to achieve optimum values for clock frequency and power consumption.

From the experiments of this section, we realize that at the highest clock speed of the design, area utilization and power consumption assume highest values. According to the design application, optimum values for area, power and speed can be achieved.

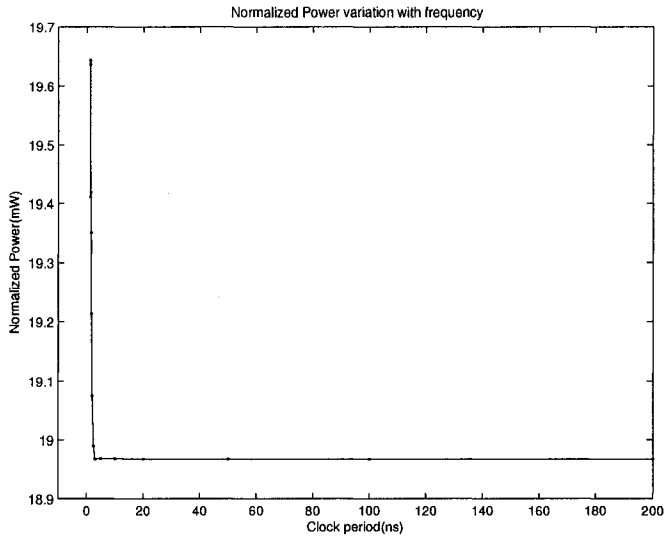

Figure 5.6: Variation of normalized power with frequency

Table 5.4: Maximum clock frequency of components in the top-level

Component	Maximum frequency (GHz)
Parallel-to-serial converter for data	2.703
Parallel-to-serial converter for coefficient	2.703
Signal generator box	0.862
Two-tap bit-serial adaptive filter	0.833

5.4 Components of the top-level filter

The components instantiated at the top-level have been analyzed individually in order to obtain the maximum clock frequency and the area, power and timing characteristics. Detailed synthesis reports obtained from the DC for each of the components are attached in the Appendix secion. The components present in the top-level entity are the two Parallel-to-Serial Converters (PSC), the signal generator component and the filter unit. Table 5.4 indicates the maximum frequencies of these components.

All the components in the top-level filter operate at frequencies that are in the gigahertz (GHz) range. As observed in the frequency listings of Table 5.4, the filter component operates at a maximum frequency of 833 MHz. The latency in our circuit is four clock cycles, and hence the delay can be calculated to be $4 \times$

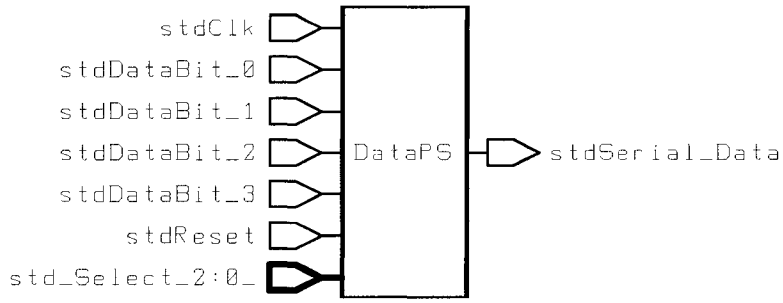

Figure 5.7: Symbol view of the data PSC

$(1/833 \times 10^6) = 4.8$ ns. The effective frequency is therefore 833/4 = 208.25 MHz. The baud rate of 1000BASE-T systems is 125 Mbaud. That means, the signal changes its value every 8 ns at a frequency of 125 MHz. The frequency of our circuit is greater than 125 MBaud. So, our design is suitable for applications in 1000BASE-T Gigabit Ethernet. This is one of the important results of our work since it supports the usage of our echo cancellation unit in high-speed wire-line systems, i.e., Gigabit Ethernet.

The following sub-sections introduce the components contained in the top-level with respect to their schematics, area and power characteristics.

5.4.1 Parallel-to-serial converter for the data term

The symbol view generated by the DC for the PSC used for the conversion of the 4-bit data term is shown in Figure 5.7. This block is used to convert the 4-bit data term that enters the top-level filter on separate parallel lines to a 4-bit-serial output. The serial output of this block is the data input to the bit-serial filter. The schematic diagram of the data PSC is shown in Figure 5.8. It is made up of four D flip-flops and one 4:1 multiplexer.

The highest clock frequency supported by the data converter was 2.703 *GHz*. The cell area and power consumption at this frequency are shown in Table 5.5.

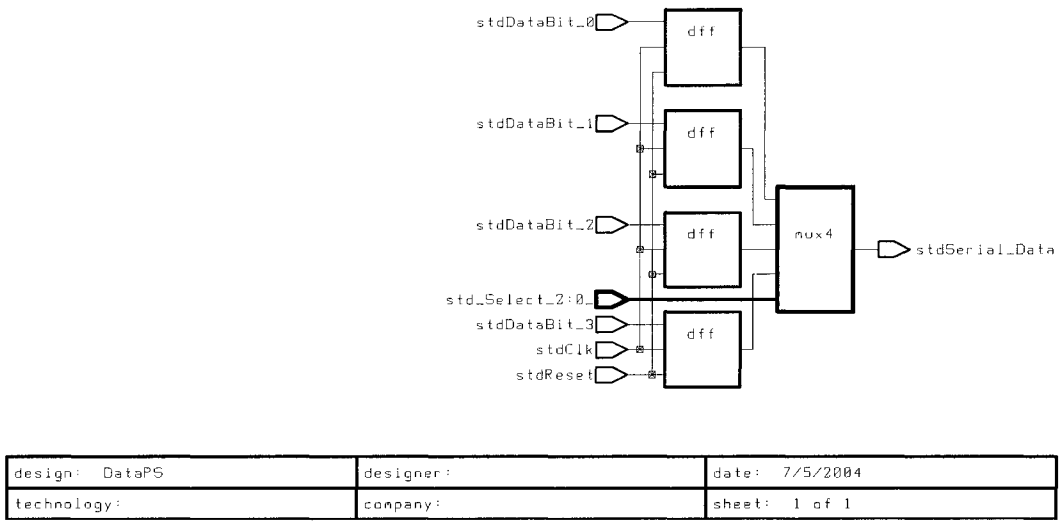

Figure 5.8: Schematic view of the data PSC

Table 5.5: DC report of the data PSC

Design:Data PSC	Report
Total cell area	504.133 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	2.7619 mW
Maximum clock frequency	2.7027 GHz

5.4.2 Parallel-to-serial converter for the coefficient term

The symbol view of the PSC used for the conversion of the 8-bit coefficient term is shown in Figure 5.9 and its schematic is depicted in Figure 5.10. In the schematic diagram, we see that the coefficient PSC is made up of eight D flip-flops and one 8:1 multiplexer.

Just like the PSC of the 4-bit data term, the coefficient PSC could also run at a maximum clock frequency of 2.703 GHz. The cell area and power usage at 2.703 GHz is listed in Table 5.6.

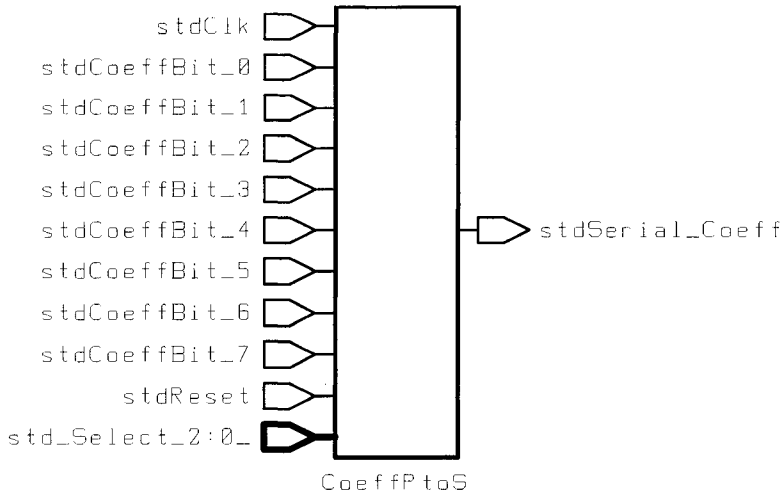

Figure 5.9: Symbol view of the coefficient PSC

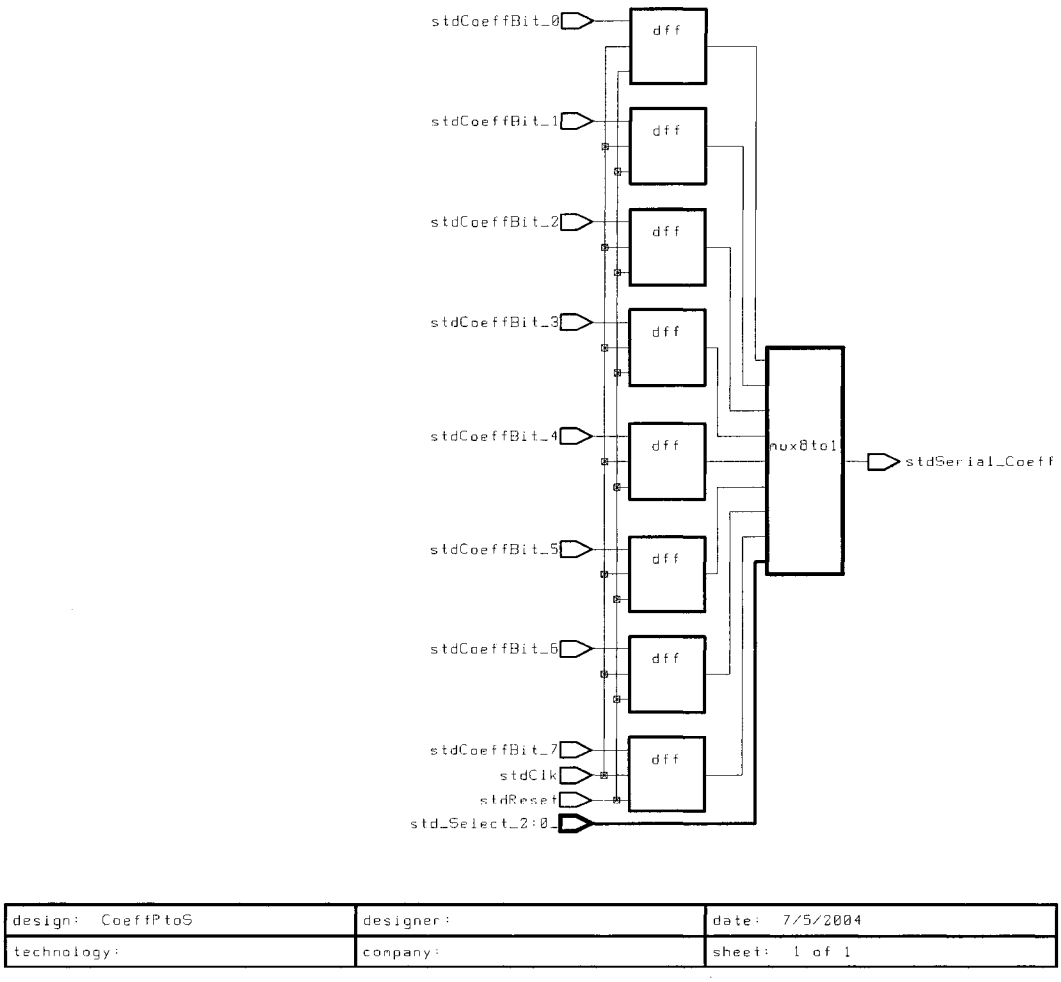

Figure 5.10: Schematic view of the coefficient PSC

Table 5.6: DC report of the coefficient PSC

Design:Coefficient PSC	Report
Total cell area	829.38 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	4.9531 mW
Maximum clock frequency	2.7027 GHz

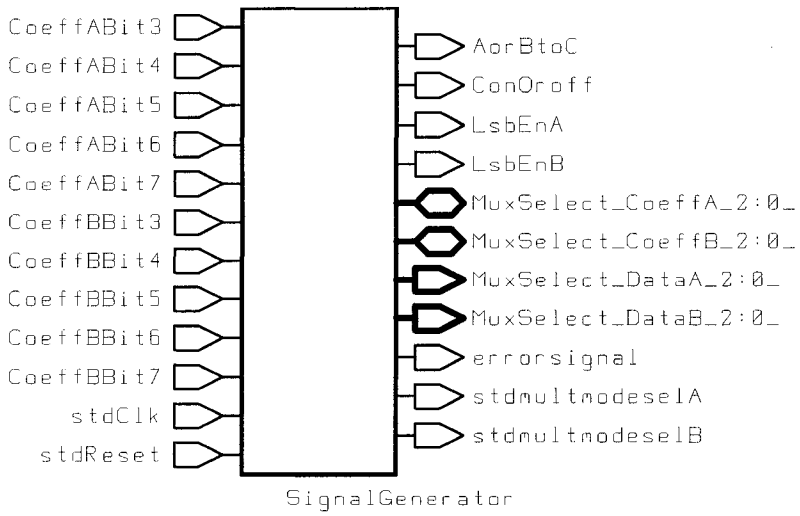

Figure 5.11: Symbol view of the signal generator component

5.4.3 Signal generator component

The signal generator component is used to generate all the input signals that are required by the serial adaptive filter and the two PSCs. The inputs to the signal generator box are the first five MSB bits of the coefficient terms. The signal generator is responsible for the computation of the input select signals to the parallel-to-serial converters and all the input signals to the filter unit. The symbol view of the signal generator component presented in Figure 5.11 is specifically designed for a two-tap filter unit. A two-tap filter unit has two coefficient terms and hence a corresponding signal generator would have its inputs as the five MSB bits of the two coefficient terms, i.e. A and B.

The schematic diagram of the signal generator component is shown in Figure 5.12.

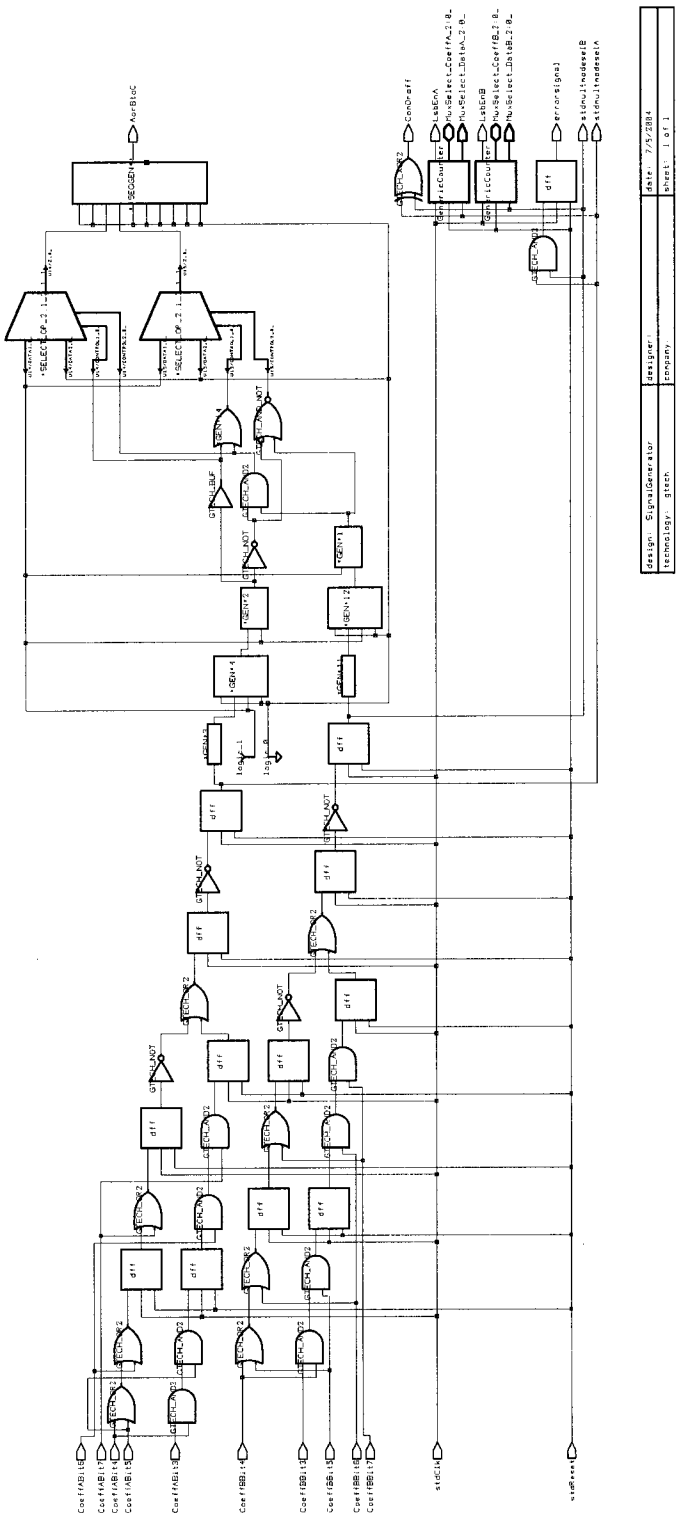

Figure 5.12: Schematic view of the signal generator component

Table 5.7: DC report of the signal generator component

Design:Signal generator	Report
Total cell area	5195.82 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	0.5352 mW
Maximum clock frequency	0.862 GHz

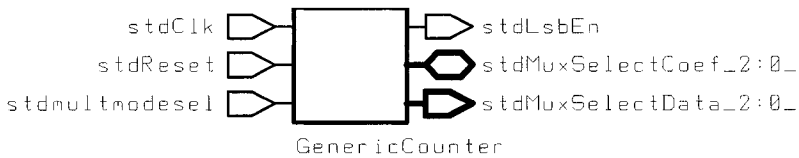

Figure 5.13: Symbol view of the generic-counter

The critical path in the top-level circuit previously originated in the signal generator component. This was traced and reduced by means of pipelining. The highest clock speed of the signal generator was found to be 862.07 MHz. The area and power characteristics at this operating frequency are listed in Table 5.7.

The signal generator arithmetically computes the input signals to the filter unit and these functions have been translated to gate-level implementations. In order to generate the input select signals to the multiplexers in the converters, a component that has been designed and named *generic-counter* is instantiated in the signal generator block.

5.4.3.1 Generic-counter

The role of the generic-counter is to generate the select signal inputs for the multiplexers in the two PSCs. The symbol view of the generic-counter is shown in Figure 5.13.

The generic-counter itself instantiates two counters. One of the counters is the data-counter which generates the select signals for the 4:1 multiplexer present in the data PSC. The other counter is the coefficient-counter which generates the select signals for the 8:1 multiplexer present in the coefficient PSC. The schematic view of the generic-counter in Figure 5.14 shows the connection of its two components.

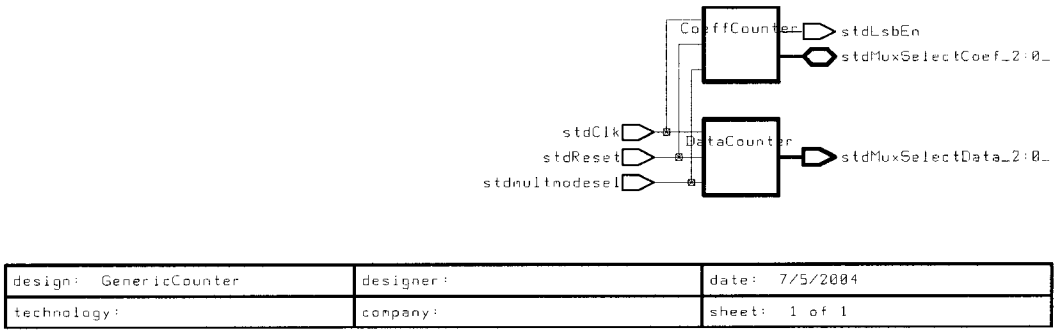

Figure 5.14: Schematic view of the generic-counter

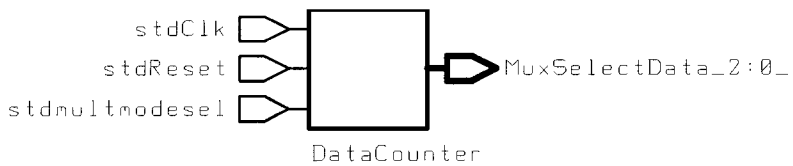

Figure 5.15: Symbol view of the data-counter

5.4.3.2 Data-counter

The data-counter is used to generate the 3-bit select input for the 4:1 multiplexer present in the data PSC. Its symbol view is shown in Figure 5.15.

5.4.3.3 Coefficient-counter

Figure 5.16 illustrates the symbol view of the coefficient-counter. The coefficient-counter is responsible for the generation of the 3-bit select signals for the 8:1 multiplexer which is present in the coefficient PSC. The coefficient-counter also computes the *LsbEn* signal which indicates the arrival of the LSB bits of the data and coefficient terms.

5.5 Bit-serial adaptive filter

The bit-serial adaptive filter is the main component in the top-level entity. This section presents the analysis of the two-tap filter unit which has been instantiated in the top-level module. In order to obtain the maximum operating frequency of the filter unit, the clock period was gradually reduced until the timing specifications were violated. The minimum allowable clock period was 1.20 *ns*. Hence the maximum

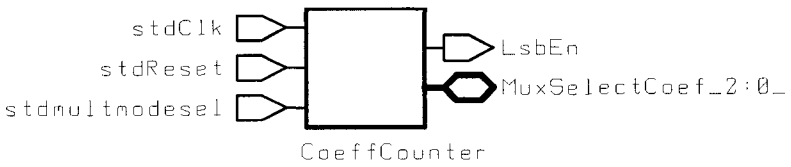

Figure 5.16: Symbol view of the coefficient-counter

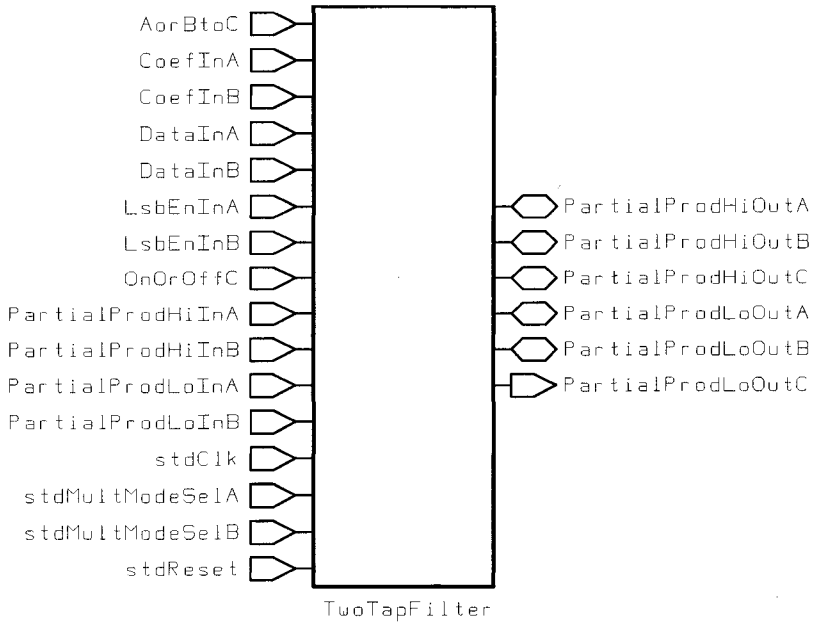

Figure 5.17: Symbol view of the two-tap bit-serial filter

frequency of operation was calculated to be 1/1.20 ns = 833.33 MHz.

The symbol view of the two-tap serial adaptive filter is shown in Figure 5.17 and its schematic diagram is represented in Figure 5.18.

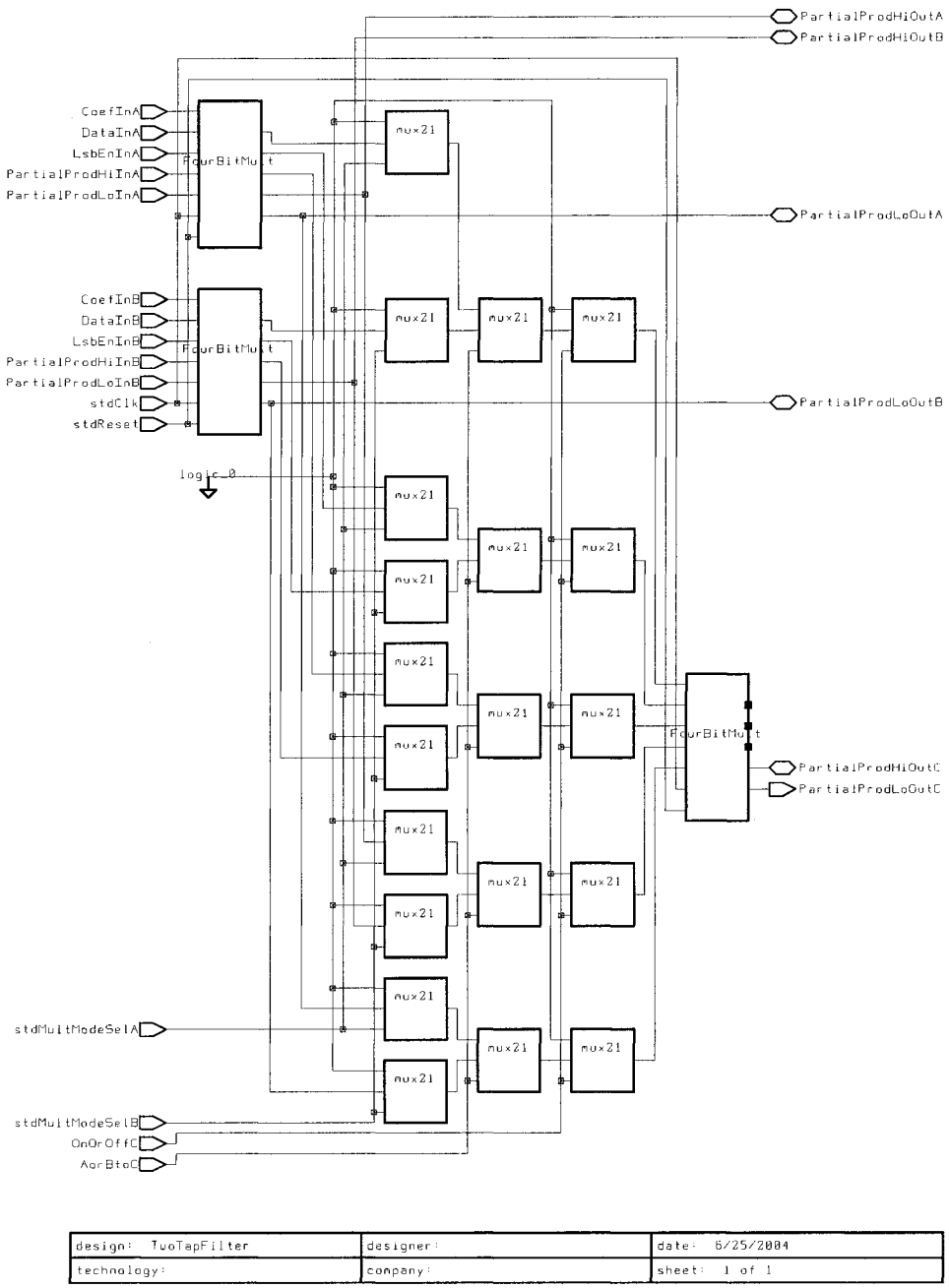

Figure 5.18: Schematic view of the two-tap bit-serial filter

Table 5.8: DC report of the bit-serial filter

Design:Bit-serial adaptive filter	Report
Total cell area	10184.35 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	7.3169 mW
Maximum clock frequency	0.833 GHz

The inputs to the two-tap filter are the data and coefficient terms in serial mode and all the other signals that are required by the filter for its operation. At the top-level, the signal generator is responsible for the generation of all the inputs of the filter unit, such that only the data and coefficient terms are fed into the top-level entity.

The main component of the filter unit is the 4-bit-serial multiplier. Because the above schematic represents a two-tap filter, it shows how two multiplier units at the first level are connected to another 4-bit multiplier at the next level using a set of 2:1 multiplexers.

The area and power measurements at the highest operating frequency of 833.33 MHz are listed in Table 5.8. A detailed area, power and timing report obtained from the DC has been attached in the Appendix section.

5.5.1 Area variation with frequency

The clock frequency was varied from 833 MHz down to 500 MHz and the variation of area with clock rate was observed. Table 5.9 shows the variation of cell area with respect to the clock period. The same has been plotted in Figure 5.19. We notice that as the frequency of operation decreases, the cell area also decreases. The total cell area at the maximum frequency of 833 MHz was found to be 10184.35 um^2. The cell area decreases and maintains a constant value of 9245.18 um^2 at all frequencies below 645 MHz.

Table 5.9: Variation of area with frequency

clock period (ns)	Clock frequency (MHz)	Cell area (um^2)
1.20	833.33	10184.35
1.23	813.01	10086.77
1.25	800.00	9944.478
1.26	793.65	9920.084
1.28	781.25	9802.181
1.30	769.23	9761.525
1.32	757.57	9570.442
1.35	740.74	9533.852
1.36	735.29	9432.210
1.38	724.64	9387.486
1.40	714.29	9375.287
1.43	699.30	9359.028
1.46	684.93	9253.320
1.50	666.67	9245.190
1.55	645.16	9245.188
2.00	500.00	9245.188

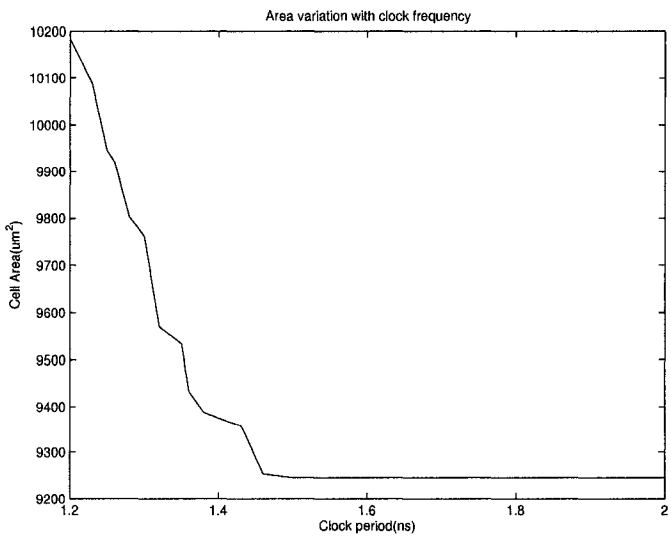

Figure 5.19: Variation of area with frequency

Table 5.10: Variation of power with frequency

Clock period (ns)	Clock frequency (MHz)	Total dynamic power (mW)
1.20	833.33	7.3169
1.23	813.01	6.0831
1.25	800.00	5.9479
1.30	769.23	5.6503
1.35	740.74	5.3631
1.40	714.29	5.1415
2.00	500.00	3.5943
5.00	200.00	1.4377
10.0	100.00	0.7188
30.0	33.333	0.2396
60.0	16.667	0.1198
100.	10.000	0.0718
300.	3.3333	0.0239
600.	1.6666	0.0119
1000	1.0000	0.0071

5.5.2 Power variation with frequency

The clock frequency was varied from 833.33 MHz down to 1 MHz and the variation of power with clock frequency was studied. The results are shown in Table 5.10.

The power consumption has been plotted against increasing clock periods in Figure 5.20. As the clock period increases, the power usage in the circuit decreases. Normalized values for dynamic power consumption have been calculated and plotted as a function of frequency. In Figure 5.21, we see that the maximum power consumption takes place at the highest frequency of operation. With a decrease in frequency, we observe a decrease in the power usage. After a certain frequency level, the power normalizes to a stable value.

Just like in the characteristics of the top-level filter, there is a trade-off between speed and power. Depending on the application, the design constraints can be varied to settle for optimum power and speed.

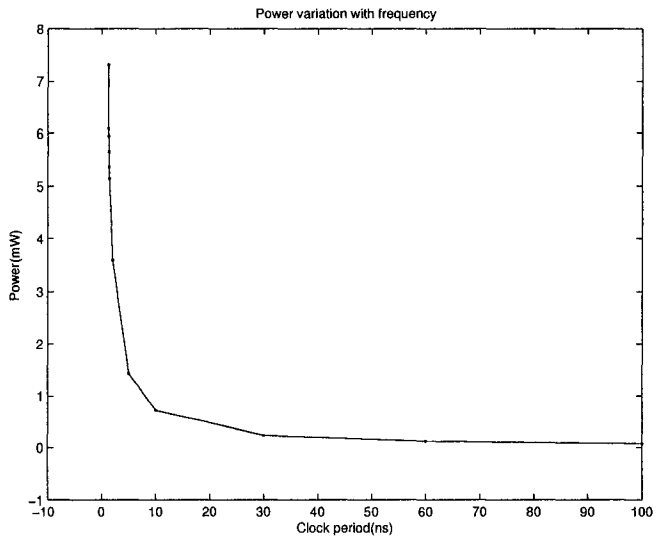

Figure 5.20: Variation of power with frequency

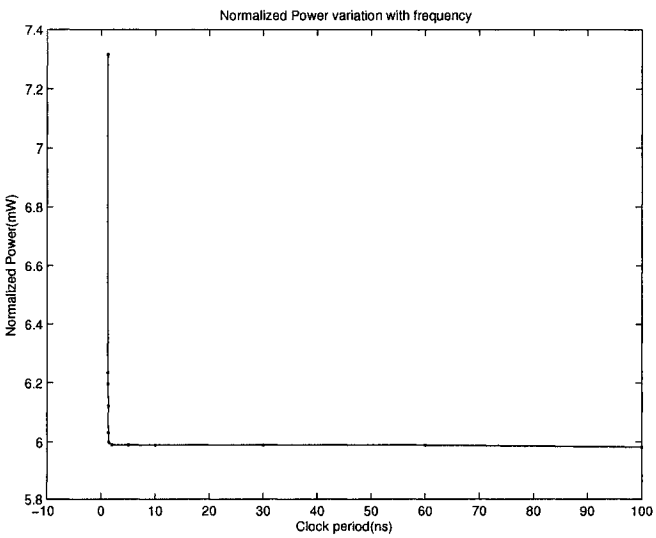

Figure 5.21: Variation of the normalized power with frequency

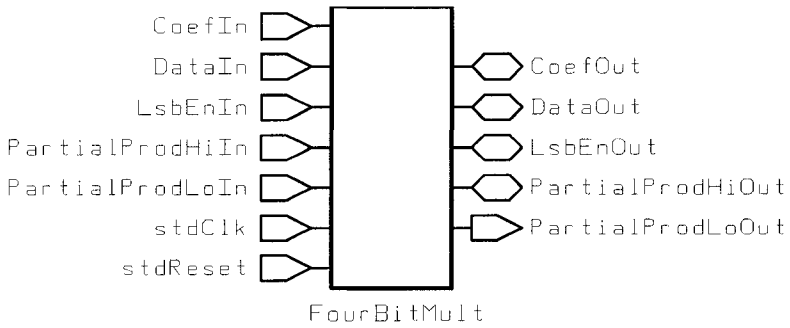

Figure 5.22: Symbol view of the bit-serial multiplier

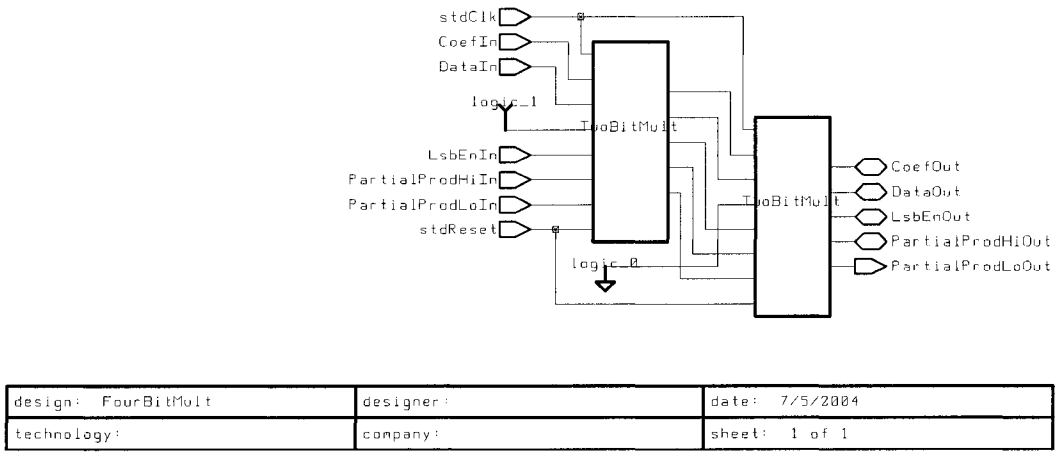

Figure 5.23: Schematic view of the bit-serial multiplier

5.6 Bit-serial multiplier

The main component of the bit-serial adaptive filter is the 4-bit-serial multiplier module. The symbol view of the 4-bit-serial multiplier generated by DC is shown in Figure 5.22 and its corresponding schematic is illustrated in Figure 5.23. A bit-serial multiplier is made up of a set of cells connected in tandem, with the outputs of the present stage generating the inputs to the next stage. As seen in the schematic diagram of Figure 5.23, the 4-bit-serial multiplier is made up of two 2-bit multiplier cells.

In order to determine the maximum allowable clock frequency, the clock period was gradually decreased until the timing constraints were violated.

Table 5.11: DC report of the bit-serial multiplier

Design:Bit-serial multiplier	Report
Total cell area	3260.61 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	2.588 mW
Maximum clock frequency	0.833 GHz

Table 5.12: Variation of area with frequency

Clock period (ns)	Clock frequency (MHz)	Cell area (um^2)
1.20	833.33	3260.61
1.22	819.67	3228.09
1.24	806.45	3207.76
1.40	714.28	3195.56
1.50	666.67	3195.56
2.00	500.00	3195.56

The minimum possible clock period of the 4-bit-serial multiplier was found to be 1.20 ns and hence the maximum clock frequency was calculated to be 833.33 MHz. The total cell area and dynamic power consumption at this frequency are shown in Table 5.11.

5.6.1 Area variation with frequency

The values of cell area for decreasing clock periods are listed out in Table 5.12.

Starting at the maximum frequency of 833 MHz, the values of cell area were measured at falling clock frequencies. The cell area maintains a constant value at all frequencies below 714 MHz. The area variation with clock speed is presented in Table 5.12. A graphical illustration of the same is presented in Figure 5.24.

5.6.2 Power variation with frequency

The dynamic power consumption of the bit-serial multiplier has been noted at declining clock periods and is listed in Table 5.13.

The power variation with clock frequency is presented in Figures 5.25- 5.26.

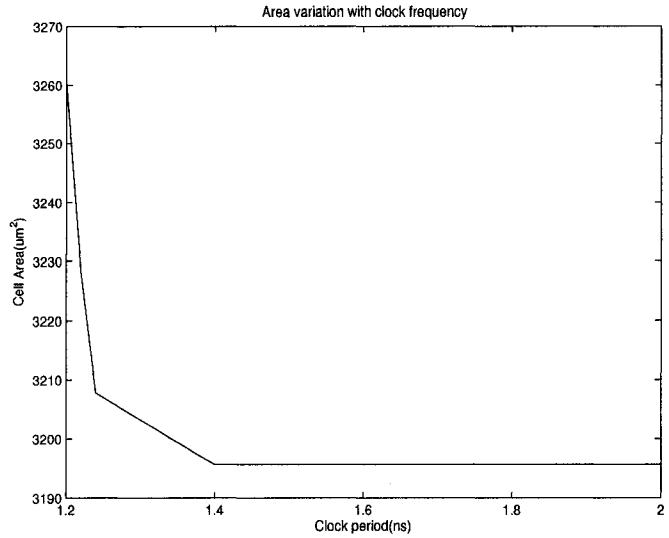

Figure 5.24: Variation of area with frequency

Table 5.13: Variation of power with frequency

Clock period (ns)	Clock frequency (MHz)	Total dynamic power (mW)
1.20	833.33	2.5889
1.22	819.67	2.5122
1.50	666.67	1.6817
2.00	500.00	1.2613
5.00	200.00	0.5045
10.0	100.00	0.2522
30.0	33.333	0.0841
60.0	16.667	0.0420
100.	10.000	0.0252
300.	3.3333	0.0084
600.	1.6667	0.0042
1000	1.0000	0.0025

The graphs in Figures 5.25 and 5.26 utilize the values of power and normalized power respectively. The decrease in power consumption with the decrease in clock frequency is once again, verified.

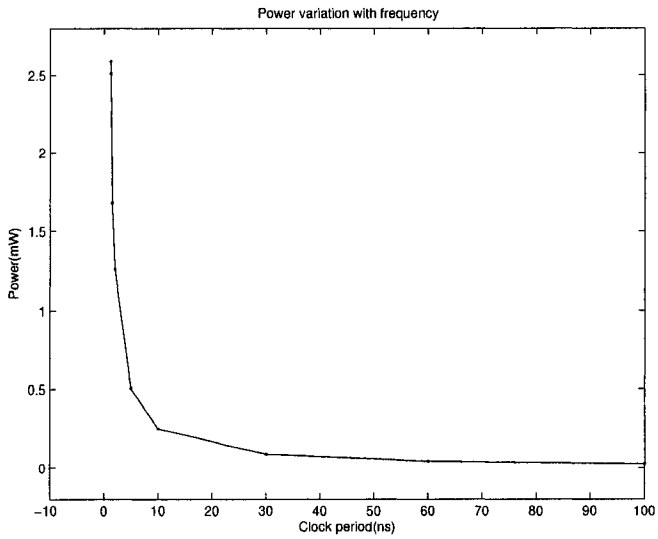

Figure 5.25: Variation of power with frequency

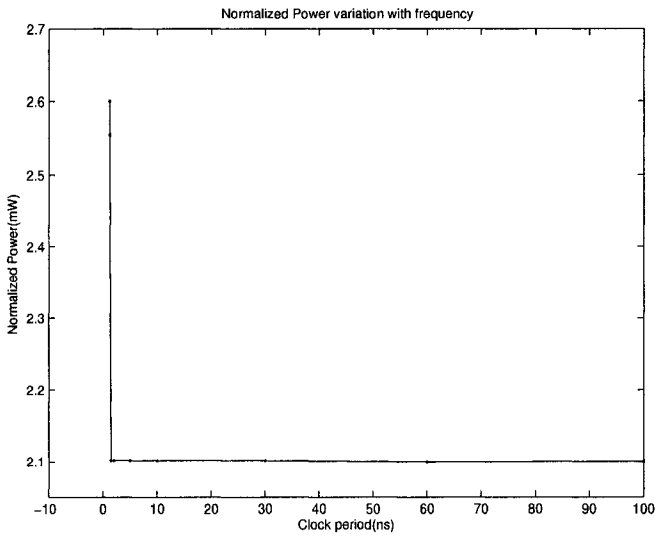

Figure 5.26: Variation of normalized power with frequency

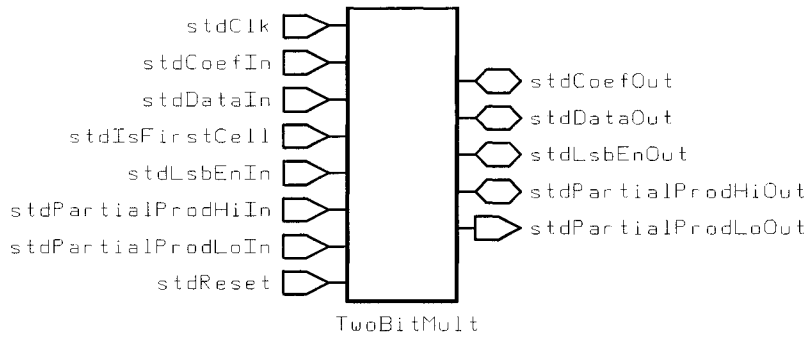

Figure 5.27: Symbol view of the unit-cell

Table 5.14: DC report of the unit-cell in the bit-serial multiplier

Design: Unit-cell	Report
Total cell area	1699.42 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	1.5508 mW
Maximum clock frequency	0.833 GHz

5.7 Unit-cell in the bit-serial multiplier

The unit-cell in the bit-serial multiplier is the 2-bit multiplier module. The symbol view of the 2-bit-serial multiplier is shown in Figure 5.27.

This module could run at a maximum frequency of 833.3 MHz. The area and power characteristics at this maximum frequency are shown in Table 5.14. The area, timing and power reports have been attached in the Appendix section.

The unit-cell in the bit-serial multiplier has been designed using basic components such as the D flip-flop, 2:1 multiplexer and a full-adder. A bit-serial adder has been instantiated for the addition of the partial products. The schematic diagram for the unit-cell in the bit-serial multiplier has been depicted in Figure 5.28.

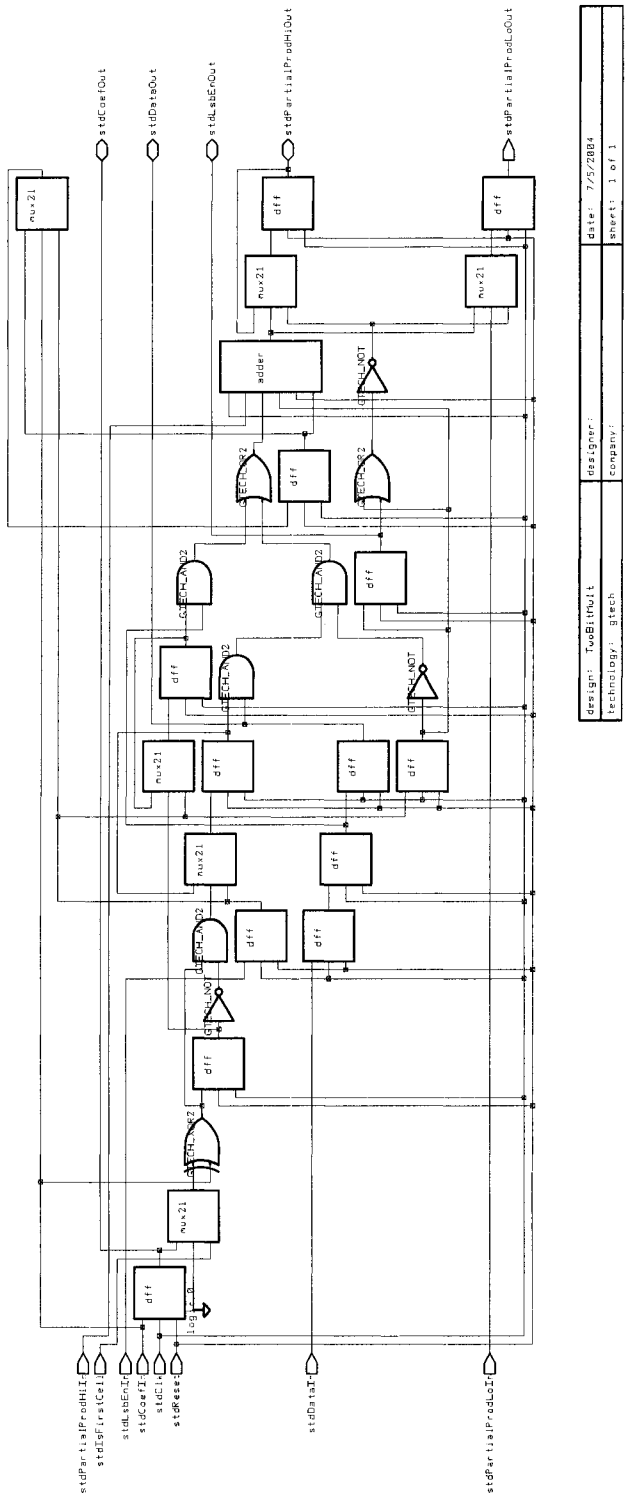

Figure 5.28: Schematic view of the unit-cell

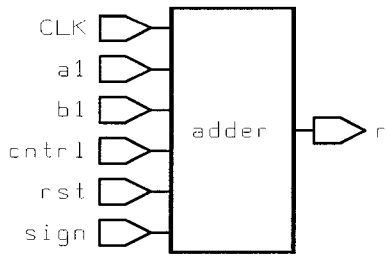

Figure 5.29: Symbol view of the bit-serial adder

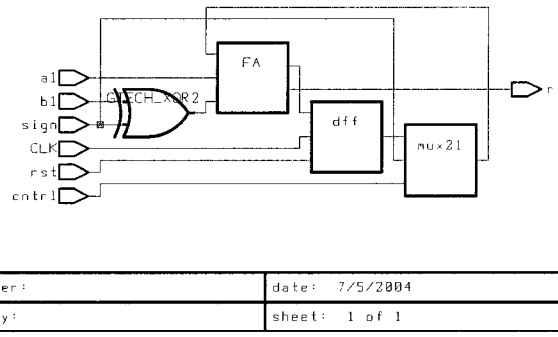

Figure 5.30: Schematic view of the bit-serial adder

5.8 Bit-serial adder

The bit-serial adder is used to add up the resulting partial products generated by the multiplication process. It is present in the unit-cell of the bit-serial multiplier. The symbol view of the bit-serial adder is shown in Figure 5.29.

In the schematic diagram of Figure 5.30, we see that the bit-serial adder is made up of fundamental units such as the D flip-flop, single-bit full-adder and 2:1 multiplexer.

The bit-serial adder reported a maximum frequency of operation of 1 GHz. The area and power characteristics at the maximum operating frequency are listed in Table 5.15.

Table 5.15: DC report of the bit-serial adder

Design:Bit-serial adder	Report
Total cell area	528.52 um^2
Global Operating Voltage	1.6 v
Total Dynamic Power	0.84 mW
Maximum clock frequency	1.00 GHz

5.9 Basic components

The basic and library components that have been instantiated several times in the design include the following:

- D flip-flop
- Single-bit full-adder
- 2:1 multiplexer
- 4:1 Multiplexer
- 8:1 Multiplexer

The area, timing and power reports of each of these blocks have been attached in the Appendix section.

5.10 Conclusion

One of the most important goals of our ASIC implementation was to clock the design at very high speeds. All the components in the top-level filter were able to operate in the Giga Hertz range. This justifies the application of our echo canceller design in high-speed communication systems.

We also studied the area and power variation with frequency. Based on consistent results, it was observed that the maximum area and power consumption takes place when the circuit operates at its highest speed. As the frequency of operation

decreases, the area and power consumption also decrease. After a certain frequency, the cell area stabilizes and maintains a constant value at all lower frequencies. The dynamic power consumption also normalizes to a stable value after a certain stage. These results highlight the trade-off between area, power and speed.

The most important issues in the design of our echo canceller were to meet the clocking requirements and cater to low area and power usage. The results obtained from the ASIC implementations satisfy these goals.

Chapter 6

Conclusion and Future work

A bit-serial adaptive filter has been designed to cancel a continuous echo signal. The novel aspect of our design is its ability to adapt to echoes of different sizes and to treat them appropriately. The primary intended application of our echo cancellation unit lies in Gigabit Ethernet communication systems.

A parameterized unit has been synthesized for *Spartan* FPGA devices. When compared with a corresponding parallel filter, the bit-serial adaptive filter has proved to be more efficient in terms of the amount of hardware resources required. The performance characteristics obtained from the FPGA realizations motivated an ASIC implementation.

The design was simulated as an ASIC and area, power and timing reports were analyzed. The most important application requirement was to be able to clock the filter at high speeds. The maximum operating clock frequency of the filter component was found to be 833 MHz. Taking into account the over-clocking requirement, the effective frequency of operation was calculated to be 208 MHz. Comparing this with the symbol rate of 125 Megabaud, it is clear that our design is suitable to application in 1000BASE-T Gigabit Ethernet systems.

The other performance criteria pertained to area and power characteristics. The congestion of the receiver section in high-speed communication systems is usually a major concern. The statistics of total cell area utilization obtained from the ASIC simulations led us to believe that our echo canceller unit would not dominate the receiver architecture. The other goal was to make the device suitable for low-power

application areas. The derived power consumption characteristics, suggest that our design would comply with low-power standards. The area and power measurements made at varying clock periods indicate the trade-offs between area versus frequency and power versus frequency.

Extension of the serial adaptive filter to about 160 taps would be useful in dealing with transmission systems that involve long cable lengths. The Gigabit Ethernet [6] system would require a filter tap size of 160, which can be implemented as an 8×20 tap-filter. This design stands as a potential candidate for successful ASIC implementation.

On future technology platforms, say 0.06 micron semiconductor technology, it will be possible to clock the cancellation filter at much higher clock speeds. With this, it may be possible to apply the proposed bit-serial adaptive filter in 10GBASE-T systems [1]. With some design effort it should be possible to operate the cancellation unit at faster clock speeds and the design could then be taken to chip-level.

Bibliography

[1] *http://www.ieee802.org/.*

[2] *Design Compiler Tutorial, Version 2000.05.* Synopsys, 2000.

[3] *IEEE Std 802.3-2002.* The Institute of Electrical and Electronics Engineers, Inc., 2002.

[4] *ISE 5 In-Depth Tutorial.* Xilinx, 2002.

[5] *Spartan-II 2.5V FPGA Family: Complete Data Sheet.* Xilinx, 2003.

[6] S. Bates and B. Murray. Issues for the design and implementation of a 1000BASE-T Gigabit Ethernet Transceiver. In *Proceedings of the International Conference on Signal Processing applications*, 2000.

[7] Simon Haykin. *Adaptive Filter Theory.* Prentice Hall, 2000.

[8] Kai Hwang. *Computer Arithmetic: Principles, Architecture and Design.* Wiley.

[9] Peter J.Ashenden. *The Designer's Guide to VHDL, 2nd edition.* Morgan Kaufmann, 2001.

[10] Charles R. Kime M. Morris Mano. *Logic And Computer Design Fundamentals, 2nd edition.* Prentice Hall, 2000.

[11] J.H. Sathyanarayana N. Chang and K. Parhi. Design and implementation of digit-serial multiplier. In *Proc. of the IEEE International Conference on Computer Design (ICCD)*, pages 186–195, Oct 1997.

[12] K.K. Parhi. A systematic approach for design of digit-serial signal processing architectures. *IEEE Trans. on Circuits and systems*, 38(4):358–375, April 1991.

Appendix A

Case study: Digit-serial multiplier

A.1 Introduction to digit-serial arithmetic

In the digit-serial approach [12], successive input words or successive groups of bits are processed in a serial manner. The number of bits processed in every clock cycle is referred to as the digit size D. When $D = 1$, digit-serial is equivalent to bit-serial and when $D = W$, where W is the word length, digit-serial is the same as bit-parallel.

Parhi K.K explained that an arbitrary two's complement number can be decomposed into D radix 2^D components where D is the digit size. In his paper [12], it has been stated that,
A J bit 2s complement number $W = -w_{J-1}2^{J-1} + \sum_{j=0}^{J-2} w_j 2^j$ *can be decomposed into D radix* 2^D *components Wi in accordance with,* $W = \sum_{i \in I} W_i 2^i$.

The decomposition of the complete data word enables the parallel processing of the constituent components, thereby permitting a fast processing speed in a corresponding digit-serial implementation. A generic digit-serial cell is shown below in Figure A.1.

A.2 Digit-serial multiplier

Parhi K.K and Sathyanarayana J.H have extensively researched the areas of digit-serial arithmetic and have proposed the Modified Booth digit-serial multiplier [11]. The same is presented in this section, as a case study.

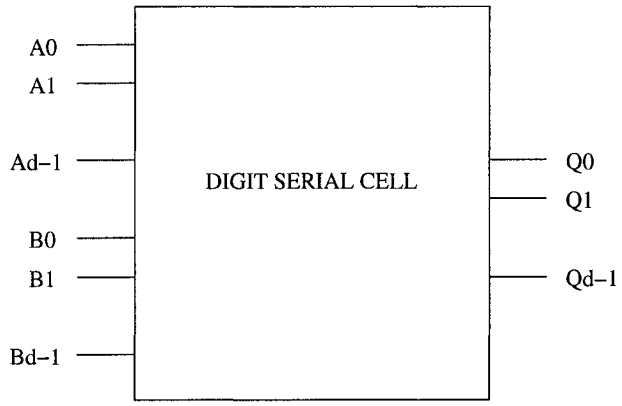

Figure A.1: Generic digit-serial cell

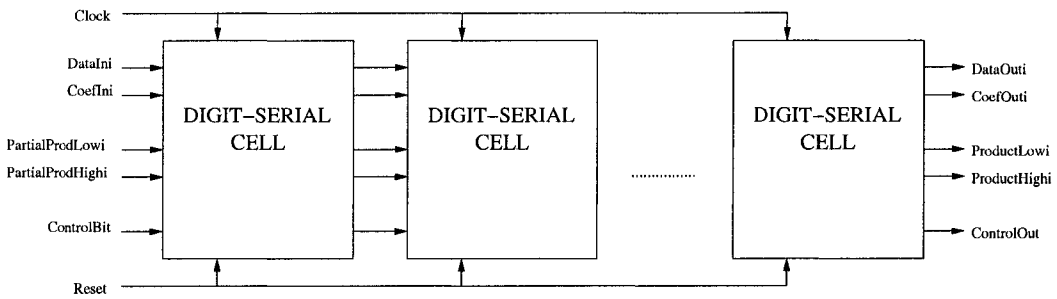

Figure A.2: Block diagram of the digit-serial multiplier

The digit-serial multiplier consists of a set of unit-multiplier cells connected in series. The inputs to the multiplier unit are the data term *DataIn* and the coefficient term *CoefIn*. They are fed into the digit-serial multiplier in the form of their decomposed components: *DataIni* and *CoefIni*. *DataOuti* and *CoefOuti* represent the corresponding decomposed components at the output of the multiplier module. The *PartialProdLowi* and *PartialProdHighi* are the LSW and MSW sections of the decomposed incoming partial products. *ControlBit* is the control signal which has a corresponding output signal *ControlOut*. *PartialProdLowi* and *PartialProdHighi* represent the high and low parts of the decomposed output components. Due to the tandem connection of the unit-cells, the input signals that enter the n^{th} multiplier module are the same output signals that leave the $(n-1)^{th}$ module. The first module however uses initialized signal values. The final products are obtained at the output of the last cell by combining the *PartialProdLowi* and *PartialProdHighi* signals.

A.3 Introduction to digit-serial multiplication

The multiplicand *DataIn* and the multiplier *CoefIn* are decomposed into a unique set of D radix 2^D components *DataIni* and *CoefIni* respectively. Then, the product *ProductWord* is formed as a sum of D radix 2^D product components *ProductWordi*, each of which is determined in terms of the multiplicand components *DataIni* and the multiplier coefficients *CoefIni*. Thus, the individual products *ProductWordi* are formed simultaneously. Incorporating the MBA makes possible a faster design.

A.4 Modified Booth digit-serial multiplier

Consider the multiplication between a pair of two's complement numbers X and Y, where the M bit number $X = X_{M-1}X_{M-2}....X_0$ represents the multiplicand term and the N bit number $Y = Y_{N-1}Y_{N-2}....Y_0$ represents the multiplier coefficient term. The data and coefficient terms in their two's complement format are represented in Equations A.1 and A.2.

$$X = -x_{M-1}2^{M-1} + \sum_{m=0}^{M-2} x_m 2^m \tag{A.1}$$

$$Y = -y_{N-1}2^{N-1} + \sum_{n=0}^{N-2} y_n 2^n \tag{A.2}$$

The product P is given as,

$$P = -y_{N-1}X2^{N-1} + \sum_{n=0}^{N-2} y_n X 2^n \tag{A.3}$$

By employing the MBA, overlapping triplets of multiplier bits Y_{2n+1}, Y_{2n} and Y_{2n-1} are encoded into a single digit z_n. The product P is therefore formed in accordance with Equation A.4.

$$P = \sum_{n \in N} z_n X 4^n \quad where \quad N = 0, 1, ...N/2 - 1 \tag{A.4}$$

Table A.1: Implementation of the MBA in the digit-serial multiplier

Encoded digit Z_n	Sign bit S_n	Magnitude bit B_n	Magnitude bit A_n
0	0	0	0
1	0	0	1
1	0	0	1
2	0	1	0
$\bar{2}$	1	1	0
$\bar{1}$	1	0	1
$\bar{1}$	1	0	1
0	1	0	0

The product P can be computed in $N/2$ successive iterations where the current partial product sum (originally initialized to zero) is updated through the addition of the partial product in the n^{th} iteration. The MBA shown in Table 2.2 can be translated to signals as shown in Table A.1 for incorporation of the same in the digit-serial multiplier.

The encoded digits Z_n can be represented by three bits S_n, A_n and B_n for a corresponding hardware implementation. S_n is the sign bit and the other two bits are used to represent the magnitude of the encoded digit.

A.5 Formation of the final product in terms of the product components

The final product P is formed by multiplying the shifted versions of each decomposed multiplicand component Xi with the encoded bits of the coefficient term. This is represented in Equation A.5.

$$P = \sum_{i \in I} \sum_{n \in N} z_n X_i 4^n 2^i \tag{A.5}$$

Thus the product components P_i can be determined in terms of the partial product sum components. All of the partial product sum components can be simultaneously calculated, independent of each other. The parallel computation of the partial product sum components, facilitates a fast processing speed.

Digit-serial multiplication is illustrated in Equation A.6, by means of a numerical example. This example involves the multiplication between a 4-bit data term $X = x_3x_2x_1x_0$ and a 5-bit coefficient term $Y = y_4y_3y_2y_1y_0$. The digit-size in this example, has been selected as two. The data term is decomposed to its D-radix 2^D components: $X_0 = x_2x_0$ and $X_1 = x_3x_1$, and each of these components is multiplied with the encoded bits of the coefficient term. The product P is obtained by combining these two results: P_0 and P_1.

$$X = 13_{10} = 1101_2 \quad Y = -6_{10} = 11010_2 = \bar{1}\ \bar{2} \quad Digit-size, D = 2$$

$$X_0 = x_2x_0 = 1 \times 2^0 + 1 \times 2^2 = 5_{10} = 101_2$$

$$X_1 = x_3x_1 = 0 \times 2^0 + 1 \times 2^2 = 4_{10} = 100_2$$

$$P_0 = X_0 \times Y = 1\ 0\ 1 \times \bar{1}\ \bar{2}$$
$$= 1\ 1\ 1\ 0\ 1\ 1\ 0$$
$$+1\ 1\ 0\ 1\ 1$$
$$1\ 1\ 0\ 0\ 0\ 1\ 0$$

$$P_1 = X_1 \times Y = 1\ 0\ 0 \times \bar{1}\ \bar{2}$$
$$= 1\ 1\ 1\ 1\ 0\ 0\ 0$$
$$+1\ 1\ 1\ 0\ 0$$
$$1\ 1\ 0\ 1\ 0\ 0\ 0$$

$$P = P_0 \times 2^0 + P_1 \times 2^1$$
$$= 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0$$
$$+1\ 1\ 0\ 1\ 0\ 0\ 0$$
$$1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0$$

$$6 \times -13 = -78_{10} = 110110010_2$$

Appendix B

FPGA synthesis Reports

B.1 Comparison of multipliers

Synthesis tool: Xilinx-ISE

Design: **4-bit-serial multiplier**
Design Statistics:
Number of IOs: 12
Number of Register: 28
Number of Multiplexers: 14

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 18 out of 192
Number of Slice Flip Flops: 32 out of 384
Number of 4 input LUTs: 17 out of 384
Number of bonded IOBs: 10 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 8.472ns
Minimum input arrival time before clock: 4.995ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 358
Additional JTAG gate count for IOBs: 576
==========
Design: **4-bit-parallel multiplier**
Design Statistics
Number of IOs: 16
Number of XORs: 16

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 16 out of 192
Number of 4 input LUTs: 28 out of 384
Number of bonded IOBs: 16 out of 90

Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 21.194ns
Total equivalent gate count for design: 168
Additional JTAG gate count for IOBs: 768
==========
Design: **8-bit-serial multiplier**

Design Statistics
Number of IOs: 13
Number of Registers: 56
Number of Multiplexers: 28

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 31 out of 192
Number of Slice Flip Flops: 57 out of 384
Number of 4 input LUTs: 34 out of 384
Number of bonded IOBs: 7 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 8.472ns
Minimum input arrival time before clock: 5.936ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 660
Additional JTAG gate count for IOBs: 432
==========
Design: **8-bit-parallel multiplier**
Design Statistics
Number of IOs: 32
Number of XORs: 64

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 69 out of 192
Number of 4 input LUTs: 120 out of 384
Number of bonded IOBs: 32 out of 90

Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 720
Additional JTAG gate count for IOBs: 1536
==========
Design: **16-bit-serial multiplier**

Design Statistics
Number of IOs: 12
Number of Registers: 112
Number of Multiplexers: 56

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 65 out of 192
Number of Slice Flip Flops: 112 out of 384
Number of 4 input LUTs: 68 out of 384
Number of bonded IOBs: 11 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 9.102ns
Minimum input arrival time before clock: 4.995ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 1310
==========

Design: **16-bit-parallel multiplier**
Design Statistics
Number of IOs: 64
Number of XORs: 256

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 285(*) out of 192
Number of 4 input LUTs: 496(*) out of 384
Number of bonded IOBs: 64 out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 86.120ns
Total equivalent gate count for design: 2,976
Additional JTAG gate count for IOBs: 3072
==========

B.2 Comparison of filters

Design: **Two-tap bit-serial filter**
Design Statistics:
Number of IOs: 22
Numberof Registers: 84
Number of Multiplexers: 62
Number of Xors: 24

Device utilization summary:

Selected Device : 2s15cs144-6
Number of Slices: 53 out of 192
Number of Slice Flip Flops: 86 out of 384
Number of 4 input LUTs: 66 out of 384
Number of bonded IOBs: 21 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 10.227ns
Minimum input arrival time before clock: 10.692ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 1084
Additional JTAG gate count for IOBs: 1056
==========
Design: **Two-tap bit-parallel filter**
Design Statistics:
Number of IOs: 64
Number of Xors: 128

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 138 out of 192
Number of 4 input LUTs: 240 out of 384
Number of bonded IOBs: 64 out of 90

Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 1440
Additional JTAG gate count for IOBs: 3072
==========
Design: **Three-tap bit-serial filter**
Design Statistics:
Number of IOs: 31
Numberof Registers: 112
Number of Multiplexers: 81
Number of Xors: 32

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 76 out of 192
Number of Slice Flip Flops: 115 out of 384
Number of 4 input LUTs: 103 out of 384
Number of bonded IOBs: 30 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 10.542ns
Minimum input arrival time before clock: 10.593ns

Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 1559
Additional JTAG gate count for IOBs: 1488
==========
Design: **Three-tap bit-parallel filter**
Design Statistics:
Number of IOs: 96
Number of Xors: 192

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 207(*) out of 192
Number of 4 input LUTs: 360 out of 384
Number of bonded IOBs: 96(*) out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 2160
Additional JTAG gate count for IOBs: 4608
==========
Design: **Four-tap bit-serial filter**
Design Statistics:
Number of IOs: 39
Numberof Registers: 140
Number of Multiplexers: 100
Number of Xors: 40

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 95 out of 192
Number of Slice Flip Flops: 144 out of 384
Number of 4 input LUTs: 130 out of 384
Number of bonded IOBs: 38 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 10.542ns
Minimum input arrival time before clock: 10.593ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 1953
Additional JTAG gate count for IOBs: 1872
==========
Design: **Four-tap bit-parallel filter**
Design Statistics:
Number of IOs: 128
Number of Xors: 256

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 276(*) out of 192
Number of 4 input LUTs: 480(*) out of 384
Number of bonded IOBs: 128(*) out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 2880
Additional JTAG gate count for IOBs: 6144
==========
Design: **Five-tap bit-serial filter**
Design Statistics:
Number of IOs: 48
Numberof Registers: 168
Number of Multiplexers: 119
Number of Xors: 48

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 117 out of 192
Number of Slice Flip Flops: 178 out of 384
Number of 4 input LUTs: 157 out of 384
Number of bonded IOBs: 47 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 9.003ns
Minimum input arrival time before clock: 7.299ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 2402
Additional JTAG gate count for IOBs: 2304
==========
Design: **Five-tap bit-parallel filter**
Design Statistics:
Number of IOs: 160
Number of Xors: 320

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 345(*) out of 192
Number of 4 input LUTs: 600(*) out of 384
Number of bonded IOBs: 160(*) out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found

Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 3600
Additional JTAG gate count for IOBs: 7680
==========

Design: **Six-tap bit-serial filter**
Design Statistics:
Number of IOs: 56
Numberof Registers: 196
Number of Multiplexers: 138
Number of Xors: 56

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 136 out of 192
Number of Slice Flip Flops: 207 out of 384
Number of 4 input LUTs: 183 out of 384
Number of bonded IOBs: 55 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 9.003ns
Minimum input arrival time before clock: 7.299ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 2790
Additional JTAG gate count for IOBs: 2688
==========

Design: **Six-tap bit-parallel filter**
Design Statistics:
Number of IOs: 192
Number of Xors: 384

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 414(*) out of 192
Number of 4 input LUTs: 720(*) out of 384
Number of bonded IOBs: 192(*) out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 4320
Additional JTAG gate count for IOBs: 9216
==========

Design: **Seven-tap bit-serial filter**
Design Statistics:
Number of IOs: 64
Numberof Registers: 224
Number of Multiplexers: 157
Number of Xors: 64

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 153 out of 192
Number of Slice Flip Flops: 236 out of 384
Number of 4 input LUTs: 209 out of 384
Number of bonded IOBs: 63 out of 90
Number of GCLKs: 1 out of 4

Timing Summary:
Minimum period: 9.003ns
Minimum input arrival time before clock: 7.299ns
Maximum output required time after clock: 6.788ns
Maximum combinational path delay: No path found
Total equivalent gate count for design: 3184
Additional JTAG gate count for IOBs: 3072
==========

Design: **Seven-tap bit-parallel filter**
Design Statistics:
Number of IOs: 224
Number of Xors: 448

Device utilization summary:
Selected Device : 2s15cs144-6
Number of Slices: 483(*) out of 192
Number of 4 input LUTs: 840(*) out of 384
Number of bonded IOBs: 224(*) out of 90
WARNING:Xst:1336 - (*) More than 100
Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: No path found
Maximum output required time after clock: No path found
Maximum combinational path delay: 42.929ns
Total equivalent gate count for design: 5040
Additional JTAG gate count for IOBs: 10752
==========

Appendix C

ASIC synthesis Reports

C.1 Analysis of the bit-serial adaptive filter

Synthesis tool: Synopsys Design analyzer
Version: 2003.06
Library(s) Used:
vst-n18-sc-tsm-c4-wc(File:/CMC/tools/synopsys/syn-U-2003.06/cmc/cmosp18/syn/vst-n18-sc-tsm-c4-wc.db)

Design: **Top-level filter**
Report: area
Combinational area: 7358.735840
Noncombinational area: 11765.843750
Total cell area: 19124.585938

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 14.8140 mW
Net Switching Power = 4.8303 mW
Total Dynamic Power = 19.6443 mW
Cell Leakage Power = 998.5355 nW

Report: timing
data required time: 1.11
data arrival time: -1.10
slack (MET): 0.01
==========
Design: **Data P/S converter**
Report: area
Combinational area: 150.426987
Noncombinational area: 353.705994
Total cell area: 504.132996

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 2.1353 mW
Net Switching Power = 626.5847 uW
Total Dynamic Power = 2.7619 mW
Cell Leakage Power = 24.0650 nW

Report: timing
data required time: 0.01
data arrival time: 0.00
slack (MET): 0.01
==========
Design: **Coefficient P/S converter**
Report: area
Combinational area: 211.411987
Noncombinational area: 617.968018
Total cell area: 829.380005

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 4.0081 mW
Net Switching Power = 944.9681 uW
Total Dynamic Power = 4.9531 mW
Cell Leakage Power = 38.8589 nW

Report: timing
data required time: 0.01
data arrival time: -0.00
slack (MET): 0.01
==========
Design: **Signal generator**
Report: area
Combinational area: 1191.214111
Noncombinational area: 4004.611572
Total cell area: 5195.825195

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns

Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 421.4829 uW
Net Switching Power = 113.7428 uW
Total Dynamic Power = 535.2257 uW
Cell Leakage Power = 235.0307 nW

Report: timing
data required time: 0.71
data arrival time: -0.71
slack (MET): 0.00
==========
Design: **Generic counter**
Report: area
Combinational area: 463.477997
Noncombinational area: 1455.480835
Total cell area: 1918.958984

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 69.7282 uW
Net Switching Power = 41.9511 uW
Total Dynamic Power = 111.6793 uW
Cell Leakage Power = 91.5842 nW

Report: timing
data required time: 0.72
data arrival time: -0.72
slack (MET): 0.00
==========
Design: **Data counter**
Report: area
Combinational area: 223.608978
Noncombinational area: 670.823975
Total cell area: 894.432983

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 36.4430 uW
Net Switching Power = 12.5606 uW
Total Dynamic Power = 49.0035 uW
Cell Leakage Power = 41.5647 nW

Report: timing
data required time: 0.51
data arrival time: -0.50
slack (MET): 0.00
==========
Design: **Coefficient counter**
Report: area
Combinational area: 260.197998
Noncombinational area: 800.921021
Total cell area: 1061.119019

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 38.0257 uW
Net Switching Power = 32.3635 uW
Total Dynamic Power = 70.3891 uW
Cell Leakage Power = 49.5061 nW

Report: timing
data required time: 0.72
data arrival time: -0.72
slack (MET): 0.00
==========
Design: **Two-tap filter**
Report: area
Combinational area: 4264.825684
Noncombinational area: 5919.522949
Total cell area: 10184.348633

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 5.0521 mW
Net Switching Power = 2.2648 mW
Total Dynamic Power = 7.3169 mW
Cell Leakage Power = 567.1210 nW

Report: timing
data required time: 1.05
data arrival time: -1.05
slack (MET): 0.00
==========

Design: **Four-bit multiplier**
Report: area
Combinational area: 1309.127319
Noncombinational area: 1951.490112
Total cell area: 3260.616943

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 1.8006 mW
Net Switching Power = 788.3537 uW
Total Dynamic Power = 2.5889 mW
Cell Leakage Power = 186.4199 nW

Report: timing
data required time: 1.06
data arrival time: -1.06
slack (MET): 0.00
==========
Design: **Two-bit multiplier**
Report: area
Combinational area: 727.743103
Noncombinational area: 971.679993
Total cell area: 1699.422974

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 1.0726 mW
Net Switching Power = 478.2065 uW
Total Dynamic Power = 1.5508 mW
Cell Leakage Power = 97.0533 nW

Report: timing
data required time: 1.04
data arrival time: -1.04
slack (MET): 0.00
==========
Design: **Bit-serial adder**
Report: area
Combinational area: 467.542999
Noncombinational area: 60.984001
Total cell area: 528.526978

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 485.3752 uW
Net Switching Power = 354.6292 uW
Total Dynamic Power = 840.0044 uW
Cell Leakage Power = 27.3637 nW

Report: timing
data required time: 0.86
data arrival time: -0.86
slack (MET): 0.00
==========
Design: **8:1 Multiplexer**
Report: area
Combinational area: 223.608002
Noncombinational area: 48.786999
Total cell area: 272.394989

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 261.3549 uW
Net Switching Power = 195.2818 uW
Total Dynamic Power = 456.6367 uW
Cell Leakage Power = 9.6240 nW
==========
Design: **4:1 Multiplexer**
Report: area
Combinational area: 121.968002
Noncombinational area: 48.786999
Total cell area: 170.754990

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 136.9265 uW
Net Switching Power = 113.8542 uW
Total Dynamic Power = 250.7807 uW
Cell Leakage Power = 6.6093 nW

==========

Design: **2:1 Multiplexer**
Report: area
Combinational area: 24.393999
Noncombinational area: 0.0000
Total cell area: 24.393999

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 21.3092 uW
Net Switching Power = 19.4835 uW
Total Dynamic Power = 40.7927 uW
Cell Leakage Power = 1.2531 nW

==========

Design: **D-flip flop**
Report: area
Combinational area: 28.459000
Noncombinational area: 60.984001
Total cell area: 89.443001

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 288.5872 uW
Net Switching Power = 164.4337 uW
Total Dynamic Power = 453.0210 uW
Cell Leakage Power = 4.1916 nW

Report: timing
data required time: 0.16
data arrival time: -0.16
slack (MET): 0.00

==========

Design: **Full-adder**
Report: area
Combinational area: 85.376991
Noncombinational area: 0.0000
Total cell area: 85.376999

Report: power
Global Operating Voltage = 1.6
Power-specific unit information:
Voltage Units = 1V

Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
Cell Internal Power = 68.9040 uW
Net Switching Power = 71.8816 uW
Total Dynamic Power = 140.7856 uW
Cell Leakage Power = 2.9335 nW
==========