High Speed Multipler

A dissertation submitted for fulfillment of the

requirement for the award of degree of

Master of Technology

In

VLSI Design

Submitted by

Arun Garg

Roll No. 601161001

Under the supervision of

Dr. Alpana Agarwal

Associate Professor, ECED

Thapar University, Patiala

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)
PATIALA – 147 004 (PUNJAB)

DECLARATION

I hereby declare that the project report entitled "**High Speed Multiplier**" is an authentic record of my work carried out as requirement for the award of degree of M.Tech (VLSI Design) at Thapar University, Patiala, under the supervision of **Dr. Alpana Agarwal**, Associate Professor, ECED and refers other researcher's work which are duly listed in the reference section.

The matter embedded in this dissertation has not been submitted in any other university/institute for the award of any degree.

Date:15/07/2013

Arun Garg

Roll No. 601161001

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Alpana Agarwal
Associate Professor, ECED
Thapar University, Patiala

Coutersigned by:

Head
Electronics & Communication
Engineering Department
Thapar University,Patiala

Dean of Academic Affairs
Thapar University,Patiala

Acknowledgement

First of all, I would like to express my gratitude to **Dr. Alpana Agarwal, Associate Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala for her patient guidance and support throughout my work. I am truly very fortunate to have the opportunity to work with her. I found her guidance to be extremely valuable.

I am also thankful to **Dr. Rajesh Khanna, Professor and Head of the Department**, Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department. I would also like to thank my friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would like to thank my parents for their unconditional support and encouragement.

Date : 15/07/2013 Arun Garg

Place : Patiala Roll no. 601161001

Abstract

Digital multiplication is one of the most basic functions in a wide range of algorithms. The motivation behind the study of multiplier is driven by need of high speed circuits as with the increasing level of device integration and the growth in complexity of microelectronic circuits, reduction in time delay has come to the fore as a primary design goal. This work uses divide and conquer technique for increasing the speed of the booth multiplier. Booth algorithm already increases the speed of the multiplier by using multi bits of the multiplier at a single time, but it is further increased using the above mentioned technique which divide the given task into smaller tasks (subtasks). Using this technique, n bit multiplier can be implemented like implementing n/2 bit multiplier. In this work, 16-bit and 32-bit multiplier are implemented using this technique which reduces time delay quiet significantly.

Table of Contents

List of Figures

Abbreviations

CMOS	Complementary MOSFET
DSP	Digital Signal Processor
PP	Partial Products
PPA	Partial Products Array
FA	Full Adder
HA	Half Adder
DCT	Discrete Cosine Transform
FFT	Fast Fourier Transform
CBM	Configurable Booth Multiplier
LAC	Look Ahead Carry
CS	Carry Select
DRD	Dynamic Range Determination

List of Tables

Chapter 1 Introduction

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, *etc*. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. Furthermore, it is generally the most area consuming.

Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area, speed constraints have been designed with fully parallel Multipliers at one end of the spectrum and fully serial multipliers at the other end.

With the increasing level of device integration and the growth in complexity of microelectronic circuits, reduction of power dissipation has also come to the fore as a primary design goal. While power efficiency has always been desirable in electronic circuits, only recently has it become a limiting factor for a broad range of applications, requiring consideration early on in the design process.

Power dissipation limitations come in two flavors. The first is related to cooling considerations when implementing high performance systems. High speed circuits dissipate large amounts of energy in a short amount of time, generating a great deal of heat as a by-product. This heat needs to be removed by the package on which integrated circuits are mounted. Heat removal may become a limiting factor if the package (PC board, system enclosure, heat sink) cannot adequately dissipate this heat, or if the required thermal components are too expensive for the application.

The second failure mode of high-power circuits relates to the increasing popularity of portable electronic devices. Laptop computers, pagers, portable video players and cellular phones all use batteries as a power source, which by their nature provide a limited time of operation before they require recharging. To extend battery life, low power operation is desirable in integrated circuits. Furthermore, successive generations of applications often

require more computing power, placing greater demands on energy storage elements within the system. Technology improvements in the last few decades have succeeded in reducing power consumption. Trends such as using CMOS instead of bipolar devices, and reduction in feature size of lithographic processes have served to reduce power dissipation, although other objectives, namely high integration and speed, were the primary goals of such improvements.

1.1 **Delay and Power reduction in Integrated Circuits**

Although the above discussion has motivated the ascendancy of power to the attention of the designer, it is important to understand the place that power has relative to other objectives. After guaranteeing correct digital functionality, the primary consideration for system designers has always been, and continues to be speed. A circuit is specified to operate at a particular delay, otherwise the entire system does not work; further reduction in the delay is beneficial but not strictly necessary. The power dissipation characteristics of a system, on the other hand, are often a consequence of the delay specification. Once the delay of the system is achieved, package cooling and/or battery resources will be allocated appropriately (within reason). Other factors may have equal or greater importance than power dissipation. Area of implementation and yield/reliability issues are subjects which the designer must take into account.

A major complication in microelectronic circuits is the fact that many design decisions involve a power-delay trade-off; one cannot be lowered without raising the other. It is important to note, however, that power reduction techniques are not necessarily negatively correlated to delay reduction. For example, one method to reduce delay in a circuit's critical path is to upsize the driving strength of gates, which also results in increased power dissipation.

However, another way to lower delay might be to reduce interconnect capacitance, which reduces both power and delay. Generally, great power savings can be achieved if delay is not an issue, but optimizing power without considering delay is trivial. Power reduction techniques are applied at all levels of the design hierarchy. At the most basic level, the parameters of the lithographic process in which the integrated circuit is manufactured may be

modified. Doping concentrations, minimum geometrical spacing of structures *etc*., all affect power dissipation.

Many of these parameters are beyond the control of the circuit designer. Some of these 'global' constraints, however, may be modified at various stages of the design methodology, although they are by and large left alone. For example, lowering of V_{dd} is a well-accepted method of reducing chip power, and there has been a constant trend towards running even the most high-performance microprocessors at lower supply voltages, despite the delay penalty that low supply voltage incurs. (Lowering V_{dd} goes hand-in-hand with other scaling schemes, such as gate oxide reduction, line-width scaling, etc. Voltage reduction may be seen as a consequence of scaling, and attendant power reduction is a serendipitous result.) Other schemes, such as using multiple supply voltages (high voltage rails for high speed sections, low voltages otherwise) have been proposed, although their use to date has not been large.

Delay and Power reduction techniques applied to the design hierarchy for digital CMOS devices can be subdivided as follows:

- Chip/floorplan level - At this point, the power characteristics for the entire die are planned and power/delay 'budgets' are allocated. V_{dd} and V_{th} are determined based on performance goals. Delay due to the assembly of macroblocks (interconnect and clock nets) is optimized, subject to power and area/routability constraints. As macroblocks are instantiated, this high level information is updated and budgets may be adjusted to increase/decrease specifications on other sub-blocks.

- Macroblock level - This stage comprises the assembly of gates into a basic function such as a block of control logic, or an arithmetic unit. In a standard cell design methodology, it is at this point that the implementer's intellectual property enters the design flow. As such, the various methods of implementing a particular function impact both power and delay. The number of power optimization techniques available at this level are numerous.

- Gate/circuit level - This is lowest level stage visible to the designer, where transistors are assembled into basic gates. In a standard cell methodology, the fundamental gates

used in macroblock assembly are defined here. Power optimizations are very restricted at this level, as delay specification often dictates the power at which a device will operate.

Emphasis is placed on reducing device parasitics and area while maximizing routability. A full custom approach may succeed in achieving lower power than a standard cell approach, as individual transistor sizing may overcome certain obvious inefficiencies. More aggressive circuit families, such as dynamic logic, dual rail logic, or low-swing logic can be implemented at this stage, but these often require complicated design flows (*e.g.*, through the introduction of clocks, noise sensitive nets, the need for level conversion etc.).

1.2 Integrated Circuit Multiplication

Digital multiplication is one of the most basic functions in a wide range of algorithms[3].The ubiquity of this operation in computing has given rise to a large number of multiplier implementations, each with different specifications and goals. Some applications require wide dynamic range, others need high precision, while in some cases, neither of these characteristics are very tightly specified. Digital multiplication is used as opposed to analog when high precision is an issue; it is fairly straightforward to make digital multipliers as accurate as the application requires. Precision required for multiplication varies by function. At the low end, 8 bits are needed, *e.g.*, in image compression algorithms, or 16 bits in more precise DSP tasks. At the high end, we see 53-bit and 64-bit multiplication (IEEE double precision standard [4]) Typically, there are 16-bit multipliers used for digital signal processing and 53/64-bit multipliers used in microprocessor. The basic operation in these designs is integer multiplication; in floating point multipliers, integer multiplication units are sub-blocks of the greater floating point unit. Signed versus unsigned techniques have an impact on the design, and some clever techniques have been suggested for manipulating the bit representation of numbers to generate power savings. However, the primary consideration in multipliers has been and continues to be delay.

1.3 Multipliers

In order to understand delay and power trade-offs in multipliers, basic circuit structure of digital multiplier implementations is described. Some of the techniques which have been developed to reduce multiplier delay, particularly to gain an understanding of their characteristic power dissipation, have been discussed. While some insight can be gained through direct observation of logic structure, power dissipation comes from several sources; techniques which reduce the power due to one of these sources can worsen the power dissipation due to another.

1.3.1 Multiplier Structure

At the most basic level, digital multiplication can be seen as a series of bit shifts and bit additions, where two numbers, the multiplier and the multiplicand are combined into the final result. Consider the multiplication of two numbers: the multiplier P, and multiplicand C, where P is an n-bit number with bit representation {pn-1,pn-2,...,p0}, the most significant bit being pn-1 and the least significant bit being p0; C has a similar bit representation {cn-1,cn-2,...,c0}. For unsigned multiplication, up to n shifted copies of the multiplicand are added to form the result. The entire procedure is divided into three steps: partial product (PP) generation, partial product reduction, and final addition.

1.3.2 Partial Product Generation

The initial step in digital multiplication is to generate n shifted copies of the multiplicand which may then be added in the next stage. Whether a given shifted copy of the multiplicand is added depends on the value of the multiplier bit corresponding to this multiplicand copy. If the i^{th} bit, (i = 0 to n-1) of the multiplier is '1', then the multiplicand is added. If the bit is '0', the multiplicand is not added.

The bit representation of this function can be implemented using a logical AND gate, which performs AND (ci,pj), i = 0 to n-1, j = 0 to n-1. The resulting values are called partial product bits or simply, partial products. The partial product bits are arranged in columns, which are to be added together to form the final value. The resulting trapezoidal structure is called a partial product array or simply PPA. This step is called partial product array generation.

1.3.3 Partial Product Reduction

The heart of an efficient digital multiplier implementation is in the manner in which the PPA bits are added. If conventional carry adders are used to implement these add operations, the delay of all the adders would consume a large amount of time, as each shifted version of the multiplicand would contribute a delay which is proportional to the width of the multiplicand. Instead, the partial product is reduced using a technique called carry-save Addition[5] .

1.3.4 Array Style Reduction

Given the trapezoidal array of partial product bits which must be added using carry-save addition, there exist several ways to implement partial product reduction adder. In this section, most basic method called array partial product reduction is described. For example, in Fig. 1a, there is trapezoidal PPA generated for a 6-bit by 6-bit multiplication. Take the first three bit vectors, and add them using full adders, as in Fig. 1b. Combining the results of the addition with the remaining bits of the PPA, comes a result which appears in Fig. 1c. Note that the outputs of this first set of full adders can now serve as inputs to another row of full adders, along with another bit vector. This is illustrated in Fig. 1d.

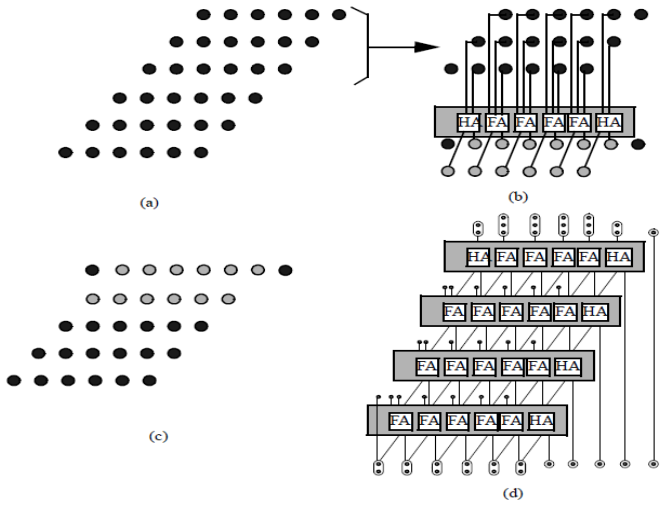

Figure1-Array partial product reduction- (a) the initial partial product (b) using a row of carry-save full adders to reduce 3 bit vectors down to two (c) resulting PPA (d) full array structure

The notable characteristic about the array architecture is its regular structure. This has the advantage that it is very easy to lay out, as a single adder block and associated connections are replicated the width and depth of the array.

1.3.5 Wallace Tree Partial Product Reduction

In 1964, C.S. Wallace [2] observed that the later stages of the array structure must always wait for all the earlier stages to complete before their final values will be established. When performing a series of independent add operations, it is possible to create a structure which has less delay by performing the addition operations in parallel, where possible. For example, in the partial product array for 6-bit x 6-bit multiplication, two carry-save reductions can be done in parallel, resulting in a smaller PPA after just one step (Fig. 2a-c.) This can be repeated, yielding the structure shown in Fig.2.

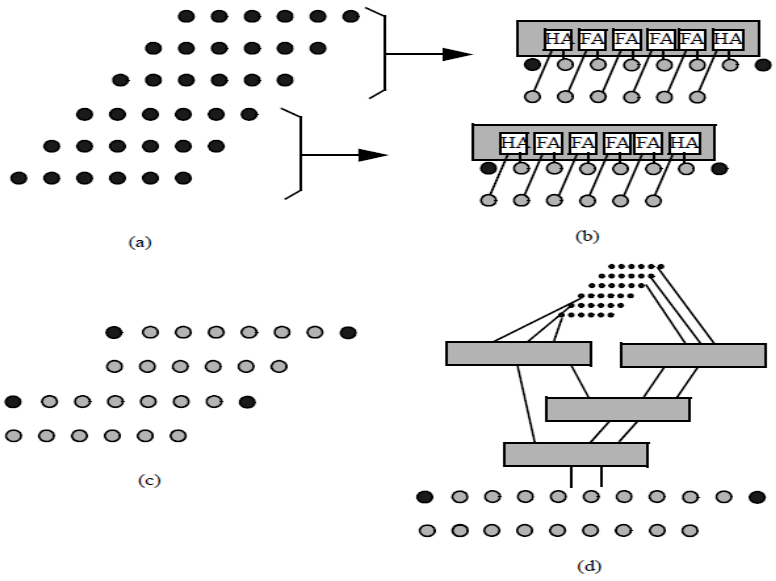

Figure2 - Wallace tree partial product reduction. (a) partial product array (b)parallel carry-save addition (c) resulting PPA (d) complete Wallace tree structure

Such parallel arrangements allow for a large reduction in the delay of the partial product reduction stage. The disadvantage of Wallace trees lies in their irregular layout (especially with respect to array structures), resulting in potentially greater wire loads.

1.3.6 Partial Product Reduction/Generation Using Booth Recoding

The technique of Booth recoding is based on the observation that under certain conditions, namely when a bit in the multiplier is '0', a bank of carry-save adders does not perform a useful function; that is, a '0' is added to the accumulated carry-save result, and the input bits are simply propagated to the output bits. In this case, these carry-save adders could be removed from the multiplier structure, resulting in delay and power savings. Unfortunately, it is not generally possible to know a priori what bits of the multiplier will be '0'. To maintain generality, that case must be provided when all bits of the multiplier are '1'.

It may be possible to reduce circuitry, however, if one considers the largest delay case. In a 4 bit x 4 bit multiplication, circuitry must be provided for the case where the multiplier is '1111'---resulting in a delay of 4 stages. An important observation is that multiplying by'1111' is the same as multiplying by '10000', and subtracting the multiplicand from the result--- multiplying by a power of two is simply a shift, so this costs two stages. Therefore, we have cut down our worst case from 4 stages of delay to 2 stages of delay.

This type of stage reduction can be generalized into the technique known as Booth recoding. Three bits of the multiplicand are used to determine whether a shifted and/or complemented copy of the multiplicand are to be used. Two bits of the multiplicand are used multiplexed to created the actual value. The encoding scheme is shown in Table 1[7].

Table 1: Booth encoding

x_{2i+1}	x_{2i}	x_{2i-1}	d_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

The connections of the Booth multiplexors are shown in Fig. 3. The net result is that the size of the PPA array is reduced, since fewer shifted copies of the multiplicand are necessary in the partial product array. The number of these multiplicand 'copies' needed in the partial product array depends on the degree to which Booth recoding is applied. Generally, each level of recoding cuts the number of partial product bits in half.

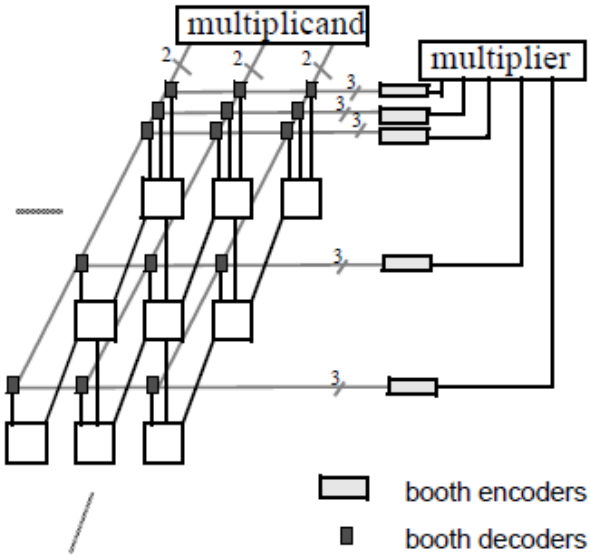

Fig 3. Booth recoding (radix 4)

There is additional circuitry involved in performing the recoding however, so this optimization entails inserting complicated logic which itself adds delay and power consumption. In practice, Booth recoding is applied to one level (called radix-4) or two levels, but hardly ever more than two levels.

1.3.7 Final Adder

A common method for achieving low delay in multipliers is to speed up the final addition stage. Several optimizations exist for performing high-speed addition, as summarized in [5]. The straightforward application of these designs to multipliers has resulted in various designs with high speed or low power .Considering ripple carry, carry skip, and carry select adder structures for this final addition step, adder implementation characteristics are summarized in Table 2. There is an area-delay trade-off between adders which is partly shown in this chart.

Although the asymptotic delays of carry skip adder and carry select adders are similar, the carry select tends to be faster but larger than the carry skip adder, while the carry skip adder tends to be slower but smaller than the carry select adder. This fits the size and delay trends when compared to the ripple adder.

Table 2 Asymptotic time and space characteristics

	Time	Space
Rjpple	O(n)	O(n)
Carry skip	$O(\sqrt{n})$	O(n)
Carry select	$O(\sqrt{n})$	O(n)

1.4 Delay and Power in Multipliers

This work focuses on multiplication performed in digital signal processing (DSP) algorithms. Multiplications in this regime typically require a precision of 8 or16 bits. From a delay perspective, algorithms place two constraints on multiplication: latency and throughput. Latency is the real delay of computing a function, a measure of how long after the inputs to a device are stable, is the final result available on the outputs. Throughput is a measure of how many multiplications can be performed in a given amount of time. For a simple combinational multiplier, throughput is a function of latency. However, various techniques exist which can compute several multiplications in parallel, *e.g.*, through pipelining; in these cases, latency is only loosely correlated to throughput. For digital signal processing, throughput is a major concern. DSP algorithms are often related to perception of audio/visual stimuli, for example, image or voice transmission and recognition. In these tasks, precision requirements are less stringent than for other applications (*e.g.*, numerical algorithms for scientific computing) so small bit-width multipliers may be used—latency is a function of bit width, and small multipliers do not create long delay paths. For many DSP applications, the

relevant limiting specification is throughput. These tasks often require fairly coarse resolution of images but operate on a fairly coarse large amount of data representing different image or sound samples. For example, image rendering requires performing computations on a large number of polygons, whereas the precision involved (bits required for identifying a particular color, bits required for identifying spacial coordinates) is fairly small.

Voice compression requires a large number of 8-bit calculations. One way of processing this large number of computations quickly can be achieved by lowering the latency of the multiplication; in this manner, the multiplier can start performing the next operation sooner.

A more efficient method to increase the number of computations is to increase the throughput. Various schemes are possible; for example, pipelining/interleaving of data allows one functional unit to compute several operations concurrently, while implementing multiple devices on one chip simply increases the throughput by the number of additional units. These techniques tend to be more efficient than latency reduction, because if one tries to lower the delay of a circuit, diminishing returns are quickly encountered (if a circuit's transistors are upsized, at a certain point, the delay does not decrease further.) When optimizing throughput, on the other hand, for each additional functional block that is added, the number of operations which may be computed in a given amount of time increases by one.

Although pipelining is good for throughput, it may be hard to implement in tightly coupled hardware/software systems. While the logic implementation of pipelining is fairly straightforward, getting compilers to build programs to take advantage of pipelining is often difficult. The problem lies in setting up a series of operations to begin execution while other operations have yet to finish. These 'parallel' or 'multiple-issue' modes of system behavior have timing dependencies which complicate the task of writing a compiler that can take advantage of such hardware techniques. Moreover, while some DSP algorithms lend themselves to parallel operation, others require processing to be more sequential in order, rendering the additional pipeline hardware useless. Finally, extremely high speed code is often implemented by hand in assembly language. Understanding all the methods of optimizing a pipelined function can be very tedious if done manually. These reasons argue

for using multiple multiplication functional units on one chip, as opposed to implementing a heavily pipelined multiplier.

The multiplier is a fairly large block of a computing system. The amount of circuitry involved is proportional to the square of its resolution (*i.e.*, a multiplier of size n-bits, has $O(n^2)$ gates.) Not only is the multiplier a high delay block, but it can be a significant source of power dissipation. Based on the argument delineated above, that several multipliers should be present on-chip as more DSP compute power is needed, the power dissipation involved in multiplication will become more dominant. (Even if the pipelined approach is used, to a first order, the pipelined multiplier will dissipate as much power as several multiplier blocks. Although a pipelined version has fewer gates, it will still experience roughly the same amount of switching.) Therefore, digital multipliers have become one of the prime circuits targeted for power reduction.

1.5 Organization of Dissertation Work

The Dissertation is organized as follows:

Chapter 1: This chapter gives an overview of Binary Multiplication.

Chapter 2: The study of different kinds of multiplier techniques is discussed in this chapter.

Chapter 3: The Various Simulation and power consumption Results of Proposed booth multiplier has been discussed in this chapter.

Chapter 4: A brief conclusion and possible improvements have been discussed in this chapter.

Chapter 2 Literature Review

The review, which has been done, provides the information about how the delay in multiplier can be reduced. The areas or methods to reduce the delay (which are reviewed) in the multiplier are reducing the partial products, row/column bypassing, dividing multiplication into smaller task. Following research paper shows that how can the delay be reduced using these methods.

2.1 Power and delay Optimization in Multipliers Using Multi-Precision Combined with Voltage Scaling Techniques[6]

Low-power design is essential for computation intensive systems such as Digital Signal processors (DSP) as well as battery-powered devices. This paper presents a novel low-power multiplier architecture, which exploits the effective dynamic range of the input data and performs a run-time multi-precision multiplication. Block-wise shutdown and voltage scaling techniques are combined to disable unused resources and adjust the supply voltage and clock frequency to reduce power consumption. This results in nearly a cubic reduction in dynamic power dissipation. Furthermore, by using modified Booth encoding scheme, partial products generating algorithm, and compression topology, our multiplier achieves both delay and power reduction.

In this paper, author implemented a multi-precision multiplier, in which the word length can be configured to suit the application at hand. Objective is to obtain both low power and high-performance at the same time. By exploiting the parallelism of our multi-precision architecture, modified partial products generation and compression schemes are used to shorten the critical path and decrease the number of circuit's transitions at the same time. More efficiently, multi-precision approach is combined with the voltage scaling and block-wise shutdown techniques in order to achieve further power savings.

In most applications, the effective dynamic range of the input data often varies, however, the arithmetic computing units such as the multipliers are often designed for a fixed size for maximum possible precision. When small magnitude data are used for multiplication, a big portion of the hardware will be used only to produce redundant sign extension bits, which

leads to large power consumption waste due to unnecessary signal transitions. Based on this observation, a multi-precision multiplier with dynamic input data ranges detection and unused hardware circuitry deactivation is desirable. One possible approach is to detect the precise range of the input data, then disable all the unnecessary operations at the signal level. However, the overhead resulting from the precise input range detection and subsequent sign extension rectification makes it not the most power-efficient scheme. Instead, taking into account the multiplier's precision and the resulting silicon area overhead, author propose a scheme which has run-time precision flexibility in adapting operands to three different working modes – 16bit×16bit, 16bit×8bit and 8bit×8bit. In the later two modes, block-wise shutdown is applied to completely disable all unnecessary blocks. With little area overhead, our multiplier has the ability to switch between three different working modes dynamically. This helps reduce both dynamic power consumption and static power consumption. Another efficient approach is to combine the dynamic block-wise shut-down approach with voltage scaling techniques. As mentioned above, although most multipliers are designed to operate for their maximum possible width, when short dynamic range data are processed, the full maximum operating performance becomes unnecessary. In this case, if the supply voltage can be scaled down, a cubic reduction of power consumption can be achieved as a result of the power's quadratic dependence on the supply voltage and linear dependence on the operating frequency.

2.1.1 Hardware Implementation

Figure 4 gives the architecture of the proposed multiplier. At first, the input dynamic range detection unit will examine the high 8-bit portions of the multiplicand and multiplier. The detection result is sent to the voltage and clock controller for generating appropriate voltage supplies and clock rates outputs to different building blocks of the multiplier. Sharing the same booth encoder and partial products generating unit, the three main building blocks of the multiplier work in different supply/frequency/power modes according to different precision requirements, which leads to significant power savings.

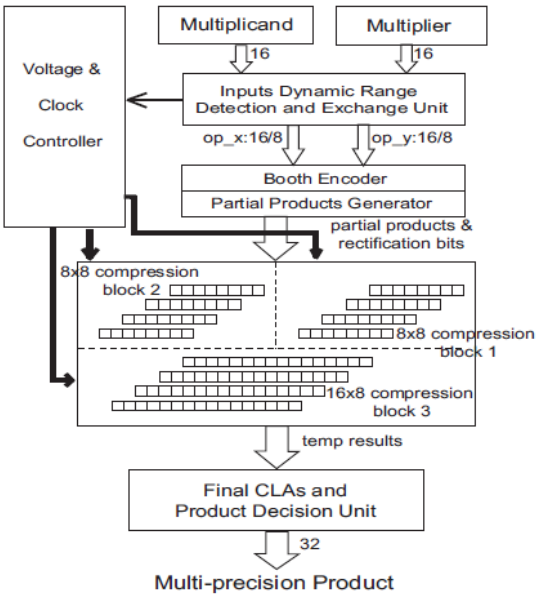

Fig 4. Architecture of the multiplier

The VCC's function is to choose proper supply voltage and clock frequency outputs for different multiplier building blocks based on the detecting signals received from the dynamic input range detection unit. The supply voltage outputs include four values: high voltage, typical voltage, low voltage, and off voltage. The clock frequency outputs include three values: high frequency, typical frequency, and low frequency.

DRD unit detects whether the input multiplicand and the multiplier are in their full 16-bit range of half 8-bit range. The results are sent to the Voltage and Clock Controller to generate proper supply voltages and clock frequencies to be applied to all building blocks of the multiplier. Dynamic range detection and operands exchange unit's circuitry also has another function. If one of the two input operands is found out to be in their half magnitudes, then a 16x8 or 8x16 multiplication will be executed. However, if there is a difference between these two processing, the circuit's complexity will increase much due to extra controlling and results rectification overhead. Here, when the 8x16 multiplication case happens, an inter-exchange will be executed to convert it to the 16x8 multiplication, which requires very little overhead.

2.2 A Low-Power Booth Multiplier Using Novel Data Partition Method [8]

Digital signal processing (DSP) is one of core technologies necessary for the next generation of multimedia and mobile communication systems. Most DSP applications involve addition and multiplication arithmetic operations. For example, DCT, FFT, wavelet transform, and OFDM are essential DSP algorithms used for image and video processing, audio signal processing and mobile communications. Currently, many portable information devices are battery-powered. The multiplication process is complex and dissipates a large amount of power due to the need for summations of the partial products. Therefore, low-power multiplication is a key concern of battery powered multimedia devices. In a CMOS circuit, power consumption can he reduced by using a smaller switching activity in the circuit. The only parameter which can be reduced in an algorithmic level is the switching activity. Therefore, minimizing the switching activity in the algorithmic level during the multiplication process should be considered first before the complex and expensive process of implementing a multiplier is attempted.

The Booth algorithm has a characteristic that the Booth algorithm produces the Booth encoded products with a value of zero when input data stream have sequentially equal values. Therefore, partial products have greater chances of being zero when the one with a smaller dynamic range of two inputs is used as a multiplier. To minimize greater switching activities of partial products, we propose a novel multiplication algorithm and its associated architecture. The proposed algorithm divides a multiplication expression into four multiplication expressions, and each multiplication is computed independently. Finally, the results of each multiplication are added. Therefore, the exchanging rate of two input data calculations can be higher during multiplication.

2.2.1 HARDWARE IMPLEMENTATION

a. Multiplication Input Data Partitioning

In this a low-power multiplication process is built in which the two input data are divided into a large number of terms with smaller bits. For example, in order to increase the chance of data exchanges occur during multiplication, the multiplication process is shown in Fig. 5.

11110101 X 10110110 = (1111 X 1011) X 100000000

= (1111 X 0110) X 10000

= (0101 X 1011) X 10000

= (0101 X 0110) X 1

Fig 5 Example of multiplication with smaller number of bits

The two inputs used for a multiplication are divided into the upper part and the lower pm. With this scheme, the chance of the exchanges can be increased because four terms of a multiplication with a smaller number of bits than those of the original input are compared for Booth encoding. Therefore, such multiplication can increases the chance of partial product becoming zero and reduces the overall power dissipation with little additional hardware. Such multiplier also uses high speed parallel multiplication architecture with smaller bits than the existing Booth multipliers.

b. Architecture of the Multiplier

Figure 6 shows one example of the proposed multiplication architecture where two input data streams are divided into upper and lower parts. The multiplication scheme is composed of four modules; input dividing unit, dynamic range determination unit (DRD), a radix-4 Booth multiplier, and an adder tree used for summing partial products. The input dividing unit divides each input data stream into parts with smaller data bits, i.e., and upper parts and lower parts are used in this example. These smaller-sized data are processed independently for multiplication using Booth encoding.

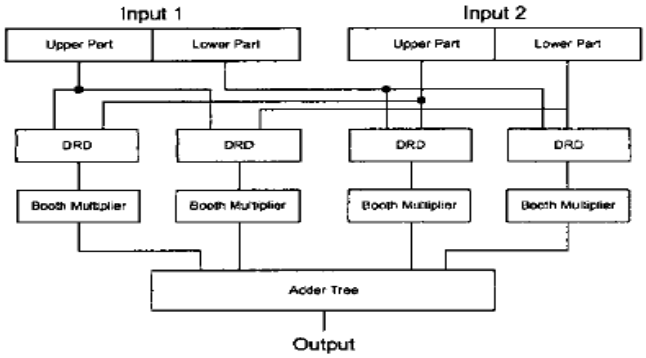

Fig 6 Block diagram of the proposed multiplication

The DRD module detects the effective dynamic range of two inputs and exchanges each so that the following condition is met: the input with a larger effective dynamic range is the multiplicand, and the input with a smaller effective dynamic range is the multiplier. The micro-architecture of the DRD is shown in fig. 7, where two 16-bit multiplication inputs are divided into two 8-bit parts. The first and second groups of the comparator's inputs in the DRD module are the first 4 MSB bits and the next 4 LSB bits of the 8-bit DRD input. The third bit of the DRD input is used commonly io the first and second groups because three hits (i+l, i, and i-1) are needed at once for use in the radix-4 Booth algorithm.

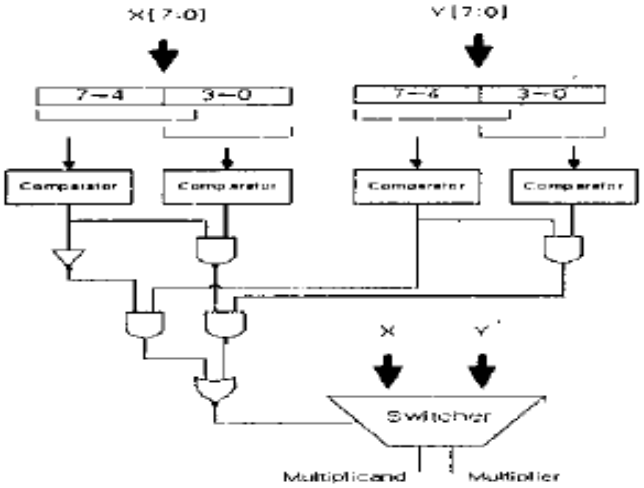

Fig. 7 DRD of the proposed multiplication

Figure 8 shows the comparator that is used in the DRD block. The comparator consists of two AND gates and one OR gate. The comparator output is zero when all input bits are equal value (0 or I).

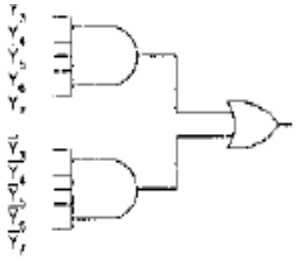

Fig. 8 Comparator

2.3 DESIGN OF LOW POWER AND HIGH SPEED CONFIGURABLE BOOTH MULTIPLIER[9]

The computation of the multipliers manipulates two input data to generate many partial products for subsequent addition operations, which in the CMOS circuit design requires many switching activities. Thus, switching activity within the functional unit requires for majority of power consumption and also increases delay. Therefore, minimizing the switching activities can effectively reduce power dissipation and increase the speed of operation without impacting the circuit's operational performance. Besides, energy-efficient multiplier is greatly desirable for many multimedia applications. This multiplier not only perform single 16-bit, single 8-bit, or twin parallel 8-bit multiplication operations but also offer a flexible tradeoff between output accuracy and power consumption to achieve more power savings.

2.3.1 CONFIGURABLE BOOTH MULTIPLIER DESIGN

In this section, partially guarded computation and the truncation technique are integrated into the configurable multiplication to construct a 16-bit low-power CBM [11]. Figure 9 shows the block diagram of the proposed 16-bit CBM. The configuration signals are utilized to configure the operation of the proposed multiplier into six modes as shown.

When CM[2:1] = 11 or 10, the single 16-bit or single 8-bit multiplication operation is performed. On the other hand, two parallel 8-b multiplication operations that satisfy the high-throughput requirement are carried out if CM[2:1] = 00.The Bit CM[0] decides whether truncation has to be done or not, if it is 0 then truncation will be done through which more power saving and speed is obtained else the output product will not be truncated. Whenever truncation is done error compensation values will be added to maintain output precision.

A. Dynamic Range Detector (DRD)

Given CM[2:0] and input operands A[15:0] and B[15:0] the proposed dynamic-range detector (DRD) in Figure 9 generates switching signals SWLH, SWHH, SWHL and SWLL for each 8-bit Booth multiplication to pick the operand that leads more partial products to zero for booth encoding.

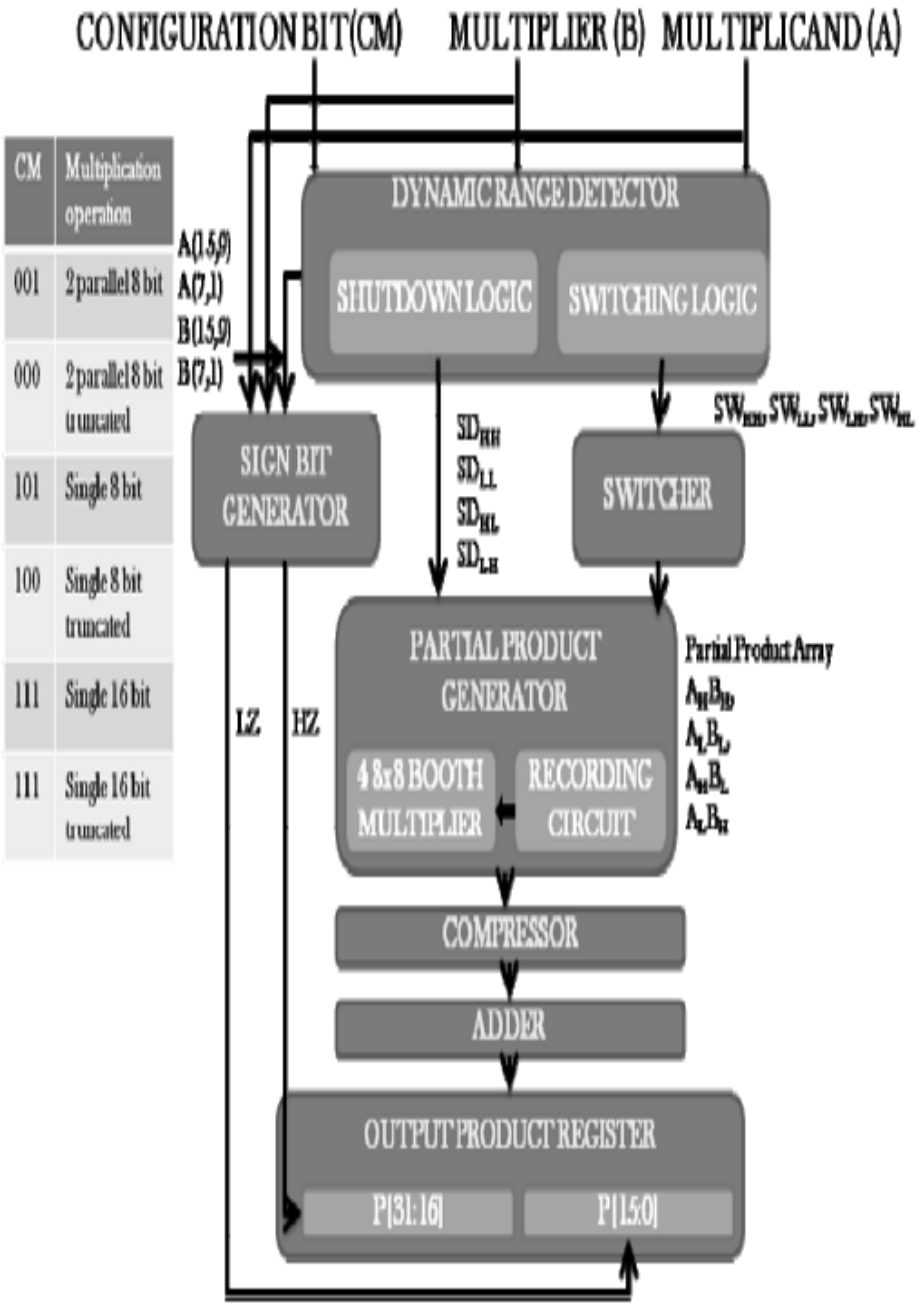

Fig 9. Block diagram of the Configurable Booth Multiplier

In addition to switching signals, DRD produces several extra shutdown signals including SDLH, SDHH, SDHL, and SDLL to dynamically disable the redundant computation of the

multiplier by forcing unnecessary partial-product bits and carry propagations to zero based on the multiplication mode and the effective range of the input operands.

If the output of a comparator is 1, it indicates that the input 3-bit group is successive zeros or ones so that its Booth encoded product will be zero. Finally each operand is compared to generate the switching signal that is used to determine which operand is a multiplier.

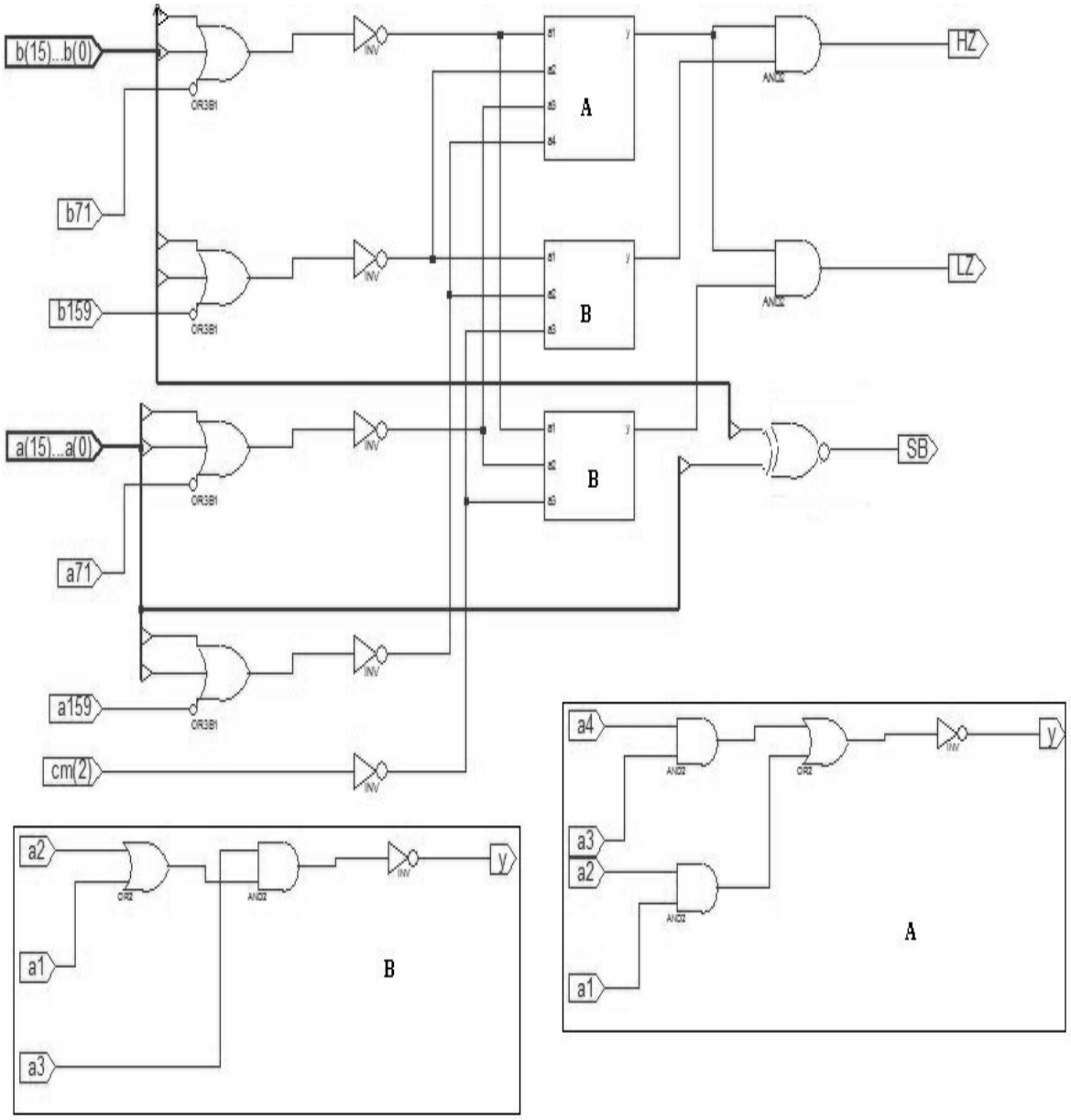

Fig 10. Sign Bit Generator

B. Sign Bit Generator

If one of the input operands is zero, the entire operation of the configurable multiplier can be shut down to obtain more power savings by preventing input registers from loading new data and directly resetting the output registers to zero thereby increasing the speed of operation. Therefore, an SBG is developed as shown in figure 10 to generate an SB, LZ and HZ and shut down the entire multiplier when one of the input operands is zero (clock gating technique [12] and [13]). In partially guarded computation, the sign-extension bits of product are replaced by an SB to avoid unnecessary sign extension computations.

2.4 A Power-Aware Scalable Pipelined Booth Multiplier [10]

The power aware scalable pipelined Booth multiplier consists of a dynamic-range detection unit, a shared radix-4 booth encoder, a shared configurable partial product generation unit,16-bit/8-bit optimized Wallace-trees for partial product summation,4-bit array-based adder-tree and a shared final carry look-ahead adder. The dynamic-range detection unit detects effective dynamic ranges of input data, appropriately selects the multiplier and multiplicand operands, and then generates control signals to deactivate the parts of the Wallace-trees and the array-based adder-tree to match run-time data precision. Depending on the number of multiplier bits, the Booth encoder and partial product generator adjust the number of partial products generated while maintaining the unused partial product generator sections in static condition.

The control and enable signals generated from the dynamic range detection unit select either one of the Wallace-tree or the 4-bit array-based adder-tree for a single multiplication operation. Thus the parts of the unused circuits maintain their static state. In static state, the previous values are, held to avoid any switching in the unused part of the structure. To take advantage of short precision, signal gating can selectively deactivate the parts of Wallace-trees and 4-bit array-based adder-tree not in use, and make it behave like a different size multiplier. In the final stage, the product is generated from the output of the active Wallace-tree and carry look-ahead adder. Thus the Booth encoder, partial product generator and carry look-ahead adder can be shared and reused to process 16 or 8-bit multiplications. For the 4-bit multiplication the Booth encoding and the partial product generator units are unused and

deactivated. This power-aware Booth multiplier is implemented with five pipeline stages where input enable signal is used as a power down switch for each stage.

A. Pipeline Gating Techniques

Latched-based clock gating technique used in each of the five pipeline stages enables the multiplier circuit at the runtime to deactivate the unused part of the logic and avoid excessive power consumption in each multiplication operation. Synopsys Power Compiler feature was utilized for generating a latch-based clock gating circuit. Each pipeline stage in the proposed multiplier is optimized to minimize the amount of switching in the logic between the pipeline stages and also within the pipeline registers. The enable signal 'ENABLE' is used to indicate the validity of the input operands for the multiplication operation. Thus the first stage pipeline registers for the 16-bit multiplicand and multiplier utilizes 16-bit clock gating D-FF with enable signal used as the control signal for the clock gating circuit.

The pipeline registers are divided into two types, Type I and II. The only difference between Type I and II pipeline register units is some extra control unit, which is employed in Type-I to eliminate any dependencies between the three multiplication modes of the proposed multiplier. The 3-bit control bus "CTLBUS' is generated from the dynamic-range detection unit using the dynamic range of the input operands, where the most significant bit is used to indicate 16-bit operand, the second bit is used for 8-bit operand and the least significant bit is used to indicate 4-bit operand.

The second Type-II pipeline register stage is partitioned into three configuring states, that is, the most significant 8-bits of the input operand are selected only if the input range is greater than 8-bits, and the middle 4 bits are selected if the input range is 16-bit or 8-bits, and finally the least 4 bits are selected if the input range is 16-bits, 8-bits or 4-bits. All these control signals for each gated pipeline register are generated if the enable signal is asserted high.

The third pipeline stage after the Booth encoder utilizes Type IV pipeline register unit. As a function of the CTL-BUS, 2-way partitioning is done to generate the control signals for the eight partial products generator units. The first four partial product generator units are split into two modes (16-bit and 8-bit modes), whereas the last four partial product generator units are only activated when 16-bit operands are detected.

2.5 Design of Hybrid Encoded Booth Multiplier with Reduced Switching Activity Technique [11]

Wireless sensor networks are composed of a set of autonomous micro-systems scattered in a specific environment. Each node monitors physical quantities of its close environment and the measured data are stored and then sent through the self organized network to a base station. Main applications of these sensor networks are monitoring of environmental physical quantities such as temperature, humidity or vibrations in different places such as buildings, industries or automotive environments. Low power and low energy VLSI circuits have become an important issue in today's consumer electronics. Digital Signal Processing (DSP) modules are becoming necessary in wireless sensor networks, in which ten to thousands of battery operated micro sensor nodes are deployed remotely and used to relay sensing data to the end user. The DSP functions mostly make use of the Multiply and Accumulate (MAC) operation in which the multiplication function is the most power consuming task. It is essential to implement the power-efficient multipliers for low power DSP modules. The development of multiplier with short critical path and low power consumption has become the important area of investigation.

Multiplier circuits are typically designed for a fixed maximum operand size, such as 64-bit. In practice, however the actual inputs to the multiplier are typically less such as 8-bit. Calculating an 8-bit multiplication on a 64-bit multiplier can lead to serious energy inefficiencies due to unnecessary digital switching on the high bits. The input bit size variation of the multiplication is a source of operational diversity and large monolithic multiplier circuits are insufficiently power aware. An architectural solution to high input bit width diversity is the incorporation of additional smaller multipliers of varying sizes.

The ensemble routes each pair of incoming operands to the smallest multiplier that can compare the result to take advantage of the lower energy consumption of the smaller circuit. Incoming multiplications are routed to the smallest multiplier that can compute the result, reducing the energy overhead of unused bits.

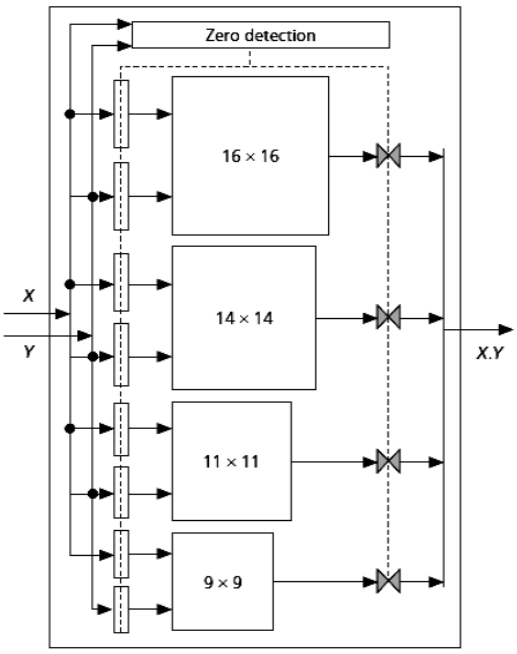

Fig 12. The ensemble of different size multipliers

An ensemble of point systems, each of which is energy efficient for a small range of inputs, takes the place of a single system whose energy consumption does not scale as gracefully with input. The power consumption has been reduced further by using this low power multiplier. This multiplier uses hybrid encoding technique and reduced switching activity technique.

2.5.1 HYBRID ENCODED BOOTH MULTIPLIER

This multiplier consists of three major working units as hybrid encoder, multiplier and controller. Consider a 16-bit encoding if the number of 1's in the multiplier less than or equal to three, consider the entire 16-bit otherwise go for 8-bit encoding. Now the 16-bit is separated in to two 8-bit and again check for three 1's. If the condition is satisfied control goes to the proposed technique otherwise control goes to modified Booth encoding. In the Partial Product (PP) compression the row bypassing has been used when the entire row of the PP is zero. This is done by freeze the adder at that time of the above scene occurs and it will avoid the unwanted switching activity and save power. In the adder unit a column bypassing provision is available to avoid the unwanted addition operation where ever it's possible. The detection logic circuit used to detect the effective data range. When a portion of data does not affect the final computing results, the data controlling circuit latch this portion to avoid

unnecessary data transitions. A glue circuit has been used to control the carry and sign extension unit which will manage the sign.

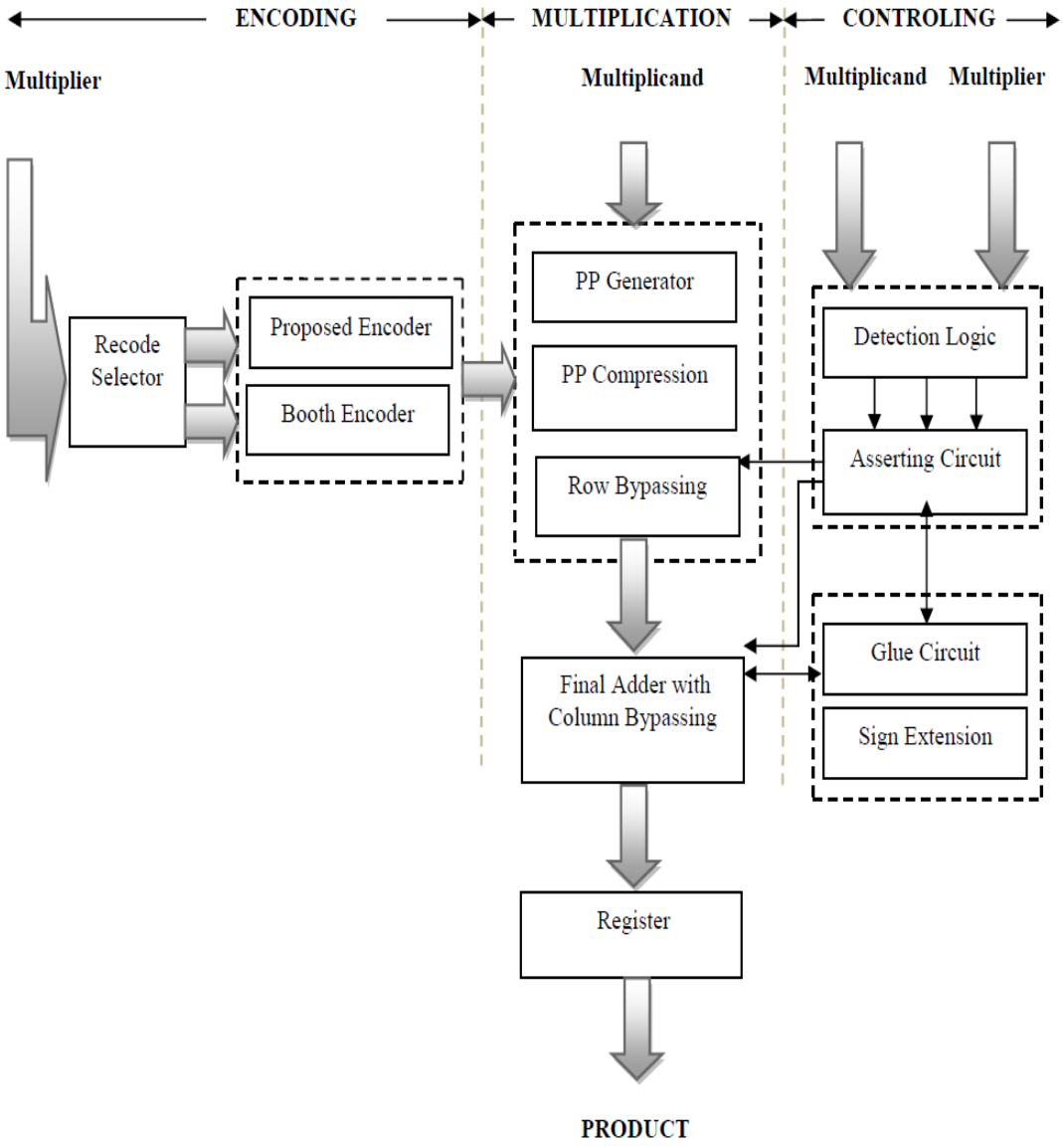

Figure 13. Block diagram of proposed hybrid encoded low power multiplier

Here a new hybrid encoding scheme has been proposed as shown in the following Table 3. According to the number of one's and the position of 1 presented in the multiplier, the operation can be defined.

Table3.Hybrid encoding scheme

Category	Number of 1's in the multiplier	Position of the 1	Operation
A	1	1^{st} bit	Add 0 to Multiplicand (M)
B	1	i^{th} bit	Shift M left by i-1 and add 0
C	2	1^{st} and i^{th} bit	Shift M left by i-1 and add M
D	2	i^{th} and $i+j^{th}$ bit	Shift M left by j , add M and shift the result left by i-1
E	3	$i^{th}=1^{st}$, j^{th} and k^{th} bit	Shift M by k-j , add M and shift the result left by j-i, add M and shift the result left by i-1
F	3	i^{th} , j^{th} and k^{th} bit	Shift M by k-j , add M and shift the result left by j-i, add M and shift the result left by i

For example according to category A, number of 1's in the multiplier has been one and its position is first bit, the result has been arrived by add zero to the multiplicand (M). If the number of 1's has been more than three then the multiplier split in to two and the same above process is continued. The constant value 34 is now multiplied with the neighboring value 85 as shown in figure14. For this multiplication it needs 8 partial products for conventional multiplication and 4 partial products for Booth recoding but only one partial product is enough for this hybrid encoding method. Moreover in this technique no need for 2's complement process and virtual '0' which is to be placed as a first bit of Booth recoding. By this the number of switching activity has been reduced to 12.5% and 25% compared with conventional and Booth encoding respectively.

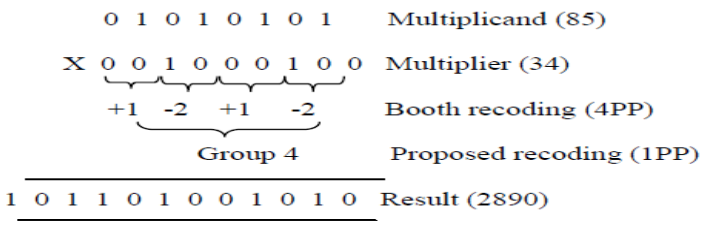

Fig 14. Hybrid encoded multiplication

2.6 Analysis of different multiplier

Table 4 shows the performance of studied multipliers in comparison to non modified multipliers.

Table 4 performance results of multipliers

		Bermak[6]	Kim[8]	Sophia[9]	Lee[10]	Madheswaran[11]
Technology		0.18µm	—	0.09µm	0.13µm	0.13µm
Area	Original	0.1566 mm²	Increases by 9% than Original booth multiplier	1350µm²	Increases by 44.3% from original multiplier	—
	Proposed	0.1425 mm²		3241 µm²		
Power	Proposed	4.02mw	Decrease by 20%	0.486mw	Decrease by 20%	0.04mw
	Original	13.55mw		1.075mw		0.11mw
Delay	Proposed	—	—	29.88ns	—	0.39ns
	Original	—	—	86.18ns	—	1.19 ns

So it is clear from the table-4 that new designed multiplier performs well in reducing the power consumption and delay with the acceptable increase in area.

16*16 bit and 32*32 bit multiplication using radix-2 booth algorithm has been implemented in VHDL using Xilinx ISE tool. At First, Different adders (Carry Select, Ripple Carry, Carry Look-ahead) are used for implementing the Booth multiplier. Out of these adders, Carry look- ahead adder gives the best result in term of speed. So Carry look-ahead adder is chosen for the implementation of booth multiplier.

To further reduce the delay of booth multiplier, Divide and Conquer technique is used. In this technique, both Inputs (multiplicand, multiplier) are divided into two equal halves. With this, four terms are available which are then multiplied according to divide and conquer algorithm. As mentioned in one of the research paper in literature review section, product of these terms are finally added which gives the final result. With this technique, area as well as delay of the multiplier reduces significantly.

Simulation and synthesis details for these booth multiplier (using different adders and div & conq. Technique) is derived with the help of Xilinx ISE tool. Synthesis report provides the details regarding hardware used for implementation of coded multiplier. It is given in terms of the number of slices and LUTs used in FPGA. A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL).

The elementary programmable logic block in Xilinx FPGAs is called slice. Slices are made up of LUTs and flip flops. In digital logic, an n-bit lookup table can be implemented with a multiplexer whose select lines are the inputs of the LUT and whose inputs are constants. An n-bit LUT can encode any n-input Boolean function by modeling such functions as truth tables. This is an efficient way of encoding Boolean logic functions, and LUTs with 4-6 bits of input are in fact the key component of modern field-programmable gate arrays (FPGAs).

Following are the synthesis and power consumption details of booth multiplier implemented using different technique/elements:

3.1 16-bit Booth Multiplier Using Ripple Carry Adder

In this, 16*16 bit multiplier is implemented using ripple carry adder. Synthesis results are shown in figure 15(a).

bm Project Status			
Project File:	booth.xise	**Parser Errors:**	No Errors
Module Name:	bm	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	817	4656	17%
Number of 4 input LUTs	1467	9312	15%
Number of bonded IOBs	64	232	27%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sat Jun 29 12:55:57 2013	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Design Summary

Figure 15(a) Synthesis report

Power consumption report is shown in figure 15(b):

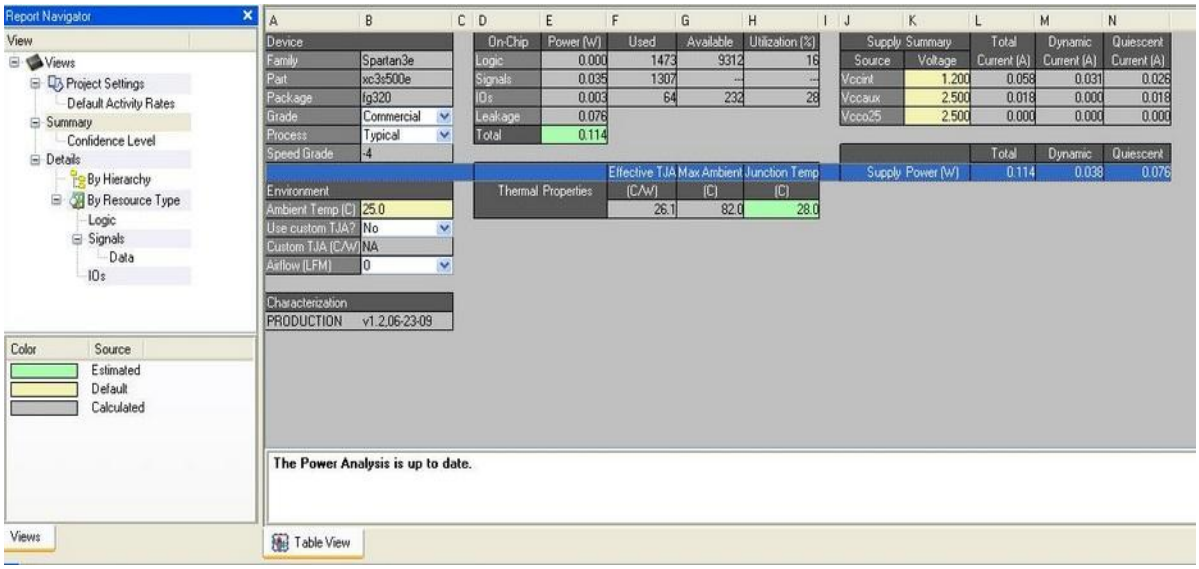

Figure 15(b) Power report

Table 5 shows the Area, power consumed and delay produced in 16-bit booth multiplier which uses ripple carry adder.

Table 5 Performance Report of 16-bit booth multiplier using ripple carry adder

	16-bit booth multiplier using ripple carry adder
No. of slices	817
No. of LUTs	1467
Power Consumed(mw)	1140
Delay(ns)	80.908

3.2 16 bit Booth Multiplier Using Carry Select Adder

In this, 16*16 bit multiplier is implemented using carry select adder. Synthesis results are shown in figure 16(a).

bm_cs Project Status			
Project File:	bmcs.xise	**Parser Errors:**	No Errors
Module Name:	bm_cs	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	964	4656	20%
Number of 4 input LUTs	1706	9312	18%
Number of bonded IOBs	64	232	27%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sat Jun 29 13:34:16 2013	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Figure 16(a) Synthesis report

Power consumption report is shown in figure16(b):

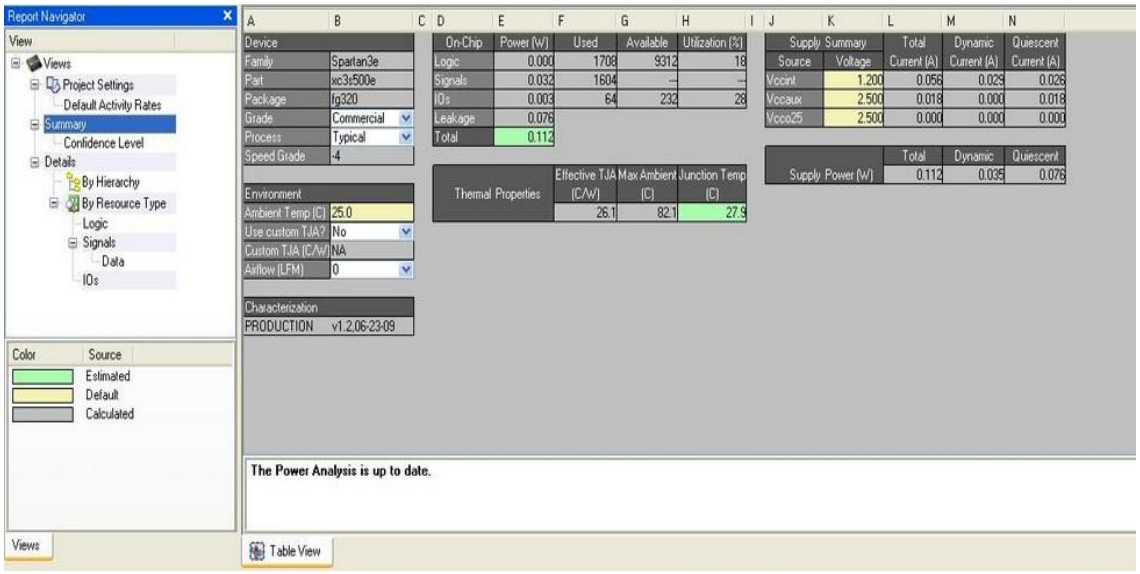

Figure 16(b) Power report

Table 6 shows the Area, power consumed and delay produced in 16-bit booth multiplier which uses carry select adder.

Table 6 Performance Report of 16-bit booth multiplier using carry select adder

	16-bit booth multiplier using carry select adder
No. of slices	964
No. of LUTs	1706
Power Consumed(mw)	1120
Delay(ns)	84.24

3.3 16 bit Booth Multiplier Using Look Ahead Carry Adder

In this, 16*16 bit multiplier is implemented using look ahead carry adder. Synthesis results are shown in figure 17(a).

bm_lac Project Status			
Project File:	bmlac.xise	**Parser Errors:**	No Errors
Module Name:	bm_lac	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	815	4656	17%
Number of 4 input LUTs	1457	9312	15%
Number of bonded IOBs	64	232	27%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sat Jun 29 13:16:29 2013	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Figure 17(a) Synthesis report

Power consumption report is shown in figure 17(b):

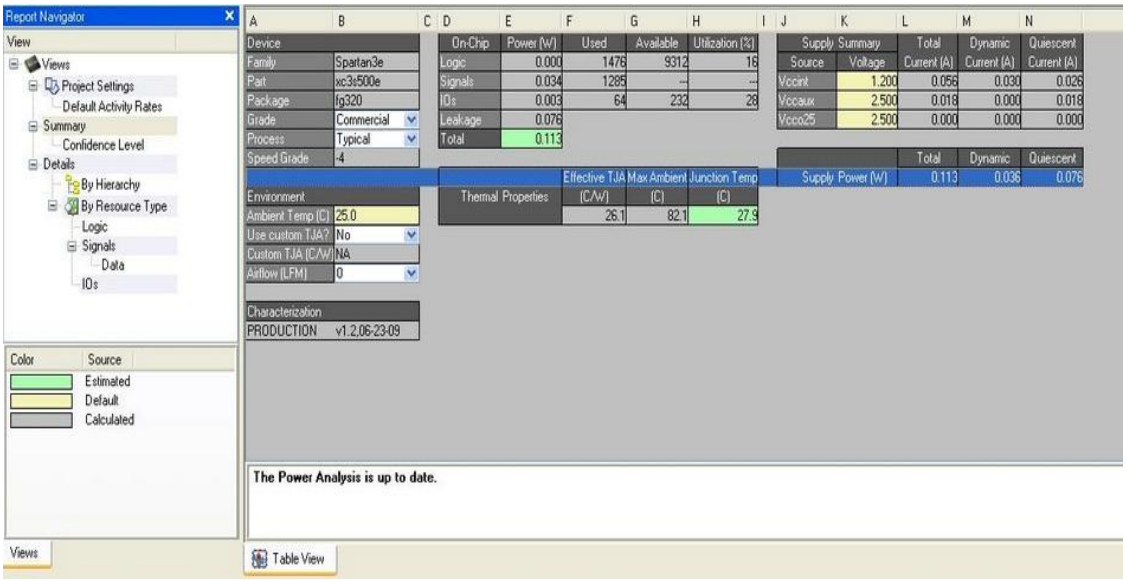

Figure 17(b) Power report

Table 7 shows the Area, power consumed and delay produced in 16-bit booth multiplier which uses look-ahead carry adder.

Table 7 Performance Report of 16-bit booth multiplier using look-ahead carry adder

	16-bit booth multiplier using look ahead carry adder
No. of slices	815
No. of LUTs	1457
Power Consumed(mw)	1130
Delay(ns)	82.829

3.4 16 bit Booth Multiplier Using Div & Con. Tech using Carry Select Adder

In this, 16*16 bit multiplier is implemented using carry select adder with the use of Divide and conquer technique. Synthesis results are shown in figure 18(a).

bm_div Project Status			
Project File:	div.xise	**Parser Errors:**	No Errors
Module Name:	bm_div	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	10 Warnings (10 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	703	4656	15%
Number of 4 input LUTs	1265	9312	13%
Number of bonded IOBs	64	232	27%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sat Jun 29 12:49:03 2013	0	10 Warnings (10 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Figure 18(a) Synthesis report

Power consumption report is shown in figure 18(b):

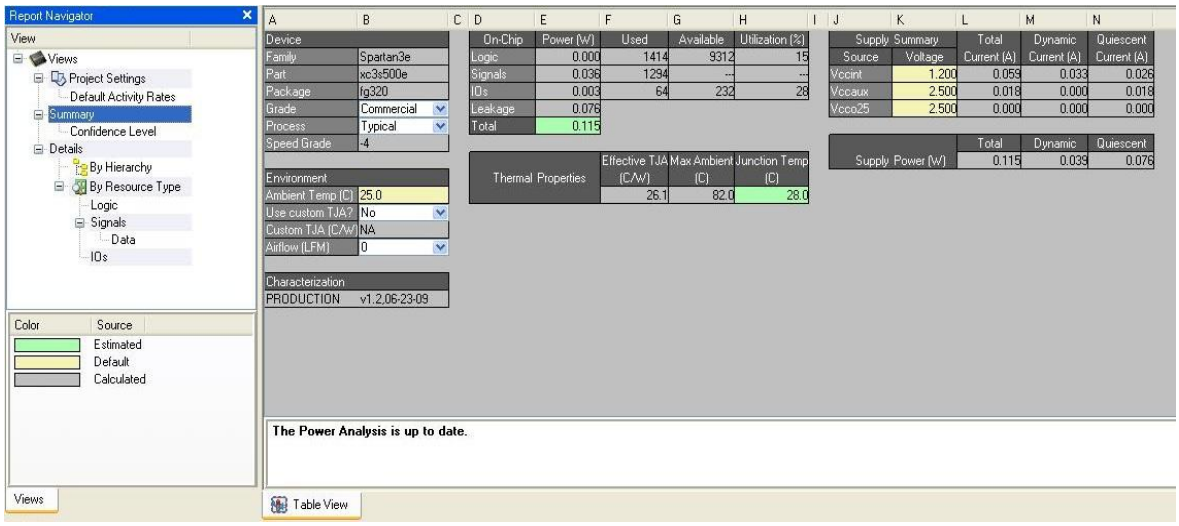

Figure 18(b) Power report

Table 8 shows the Area, power consumed and delay produced in 16-bit booth multiplier which uses Div. and conquer technique and carry select adder.

Table 8 Performance Report of 16-bit booth multiplier using DIV tech. and carry select adder

	16-bit booth multiplier using DIV tech. and carry select adder
No. of slices	908
No. of LUTs	1636
Power Consumed(mw)	1150
Delay(ns)	51.446

3.5 16 bit Booth Multiplier Using Div & Con. Tech using Look Ahead Carry Adder

In this, 16*16 bit multiplier is implemented using look ahead carry adder with the use of Divide and conquer technique. Synthesis results are shown in figure 19(a).

div_d Project Status (07/09/2013 - 06:18:15)			
Project File:	divdrd.xise	**Parser Errors:**	No Errors
Module Name:	div_d	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	10 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	700	4656	15%
Number of 4 input LUTs	1259	9312	13%
Number of bonded IOBs	64	232	27%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Tue Jul 9 06:18:09 2013	0	10 Warnings (0 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Design Summary (Synthesized) div_d.vhd

Figure 19(a) Synthesis report

Power consumption report is shown in figure 19(b):

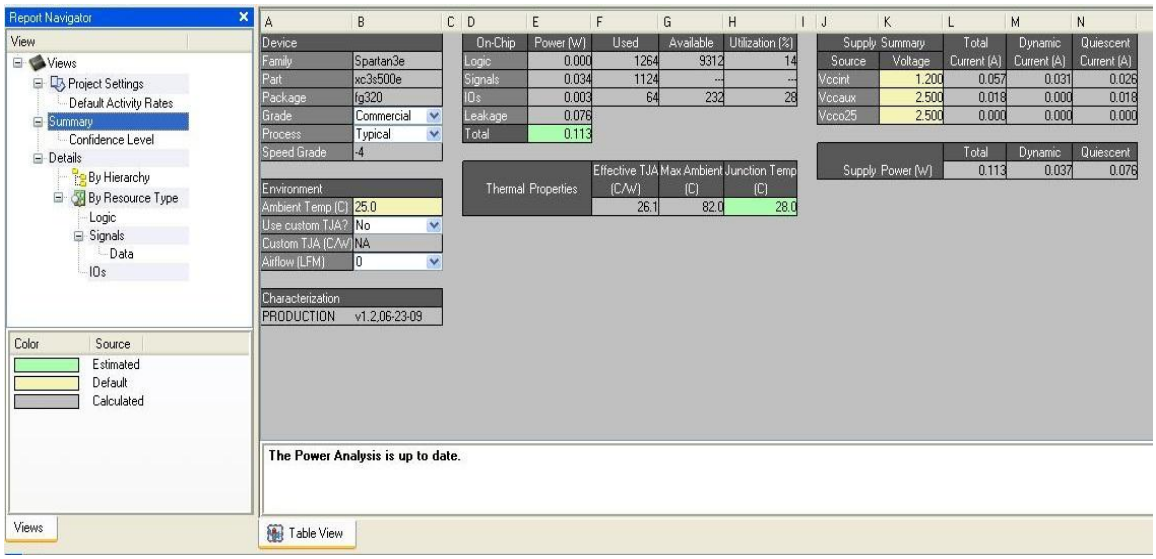

Figure 19(b) Power report

Table 9 shows the Area, power consumed and delay produced in 16-bit booth multiplier which uses Divide and conquer technique and look-ahead carry adder.

Table 9 Performance Report of 16-bit booth multiplier using DIV tech. and LAC adder

	16-bit booth multiplier using DIV tech. and look-ahead carry adder
No. of slices	700
No. of LUTs	1259
Power Consumed(mw)	1130
Delay(ns)	46.94

3.6 32 bit Booth Multiplier Using Ripple Carry Adder

In this, 32*32 bit multiplier is implemented using ripple carry adder. Synthesis results are shown in figure 20(a).

bm32 Project Status (06/30/2013 - 23:19:55)			
Project File:	bm_32.xise	**Parser Errors:**	No Errors
Module Name:	bm32	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**•Errors:**	No Errors
Product Version:	ISE 13.2	**•Warnings:**	No Warnings
Design Goal:	Balanced	**•Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3601	4656	77%
Number of 4 input LUTs	6488	9312	69%
Number of bonded IOBs	128	232	55%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sun Jun 30 23:19:51 2013	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Design Summary (Synthesized)

Figure 20(a) Synthesis report

Power consumption report is shown in figure 20(b):

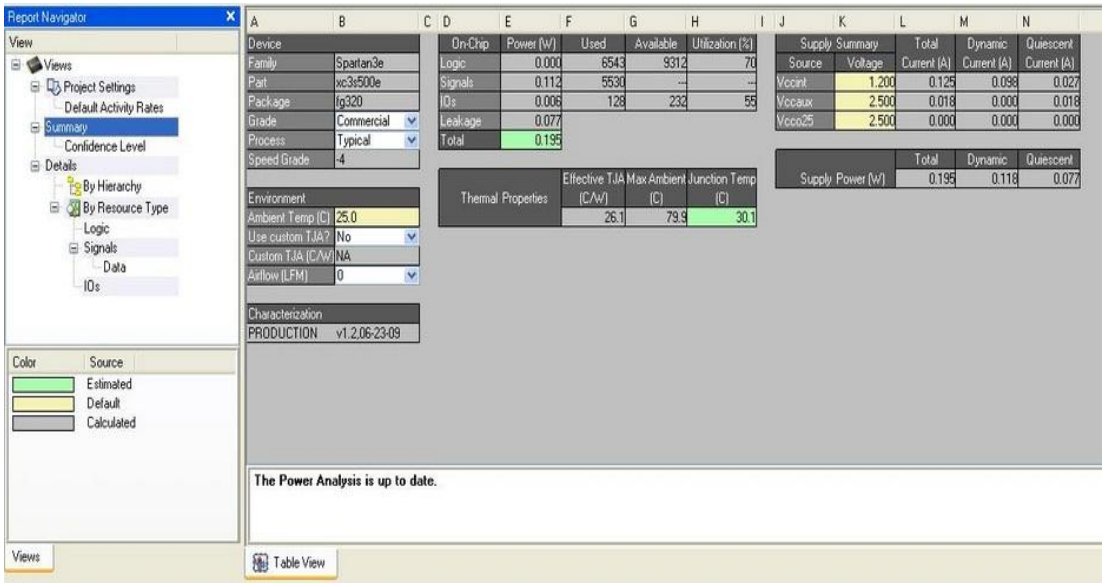

Figure 20(b) Power report

Table 10 shows the Area, power consumed and delay produced in 32-bit booth multiplier which uses ripple carry adder.

Table 10 Performance Report of 32-bit booth multiplier using ripple carry adder

	32-bit booth multiplier using ripple carry adder
No. of slices	3601
No. of LUTs	6488
Power Consumed(mw)	1950
Delay(ns)	165.383

3.7 32 bit Booth Multiplier Using Divide & Conq. Technique

In this, 32*32 bit multiplier is implemented using ripple carry adder with the use of Divide and conquer technique. Synthesis results are shown in figure 21(a).

div_32 Project Status (06/30/2013 - 22:33:15)			
Project File:	div32.xise	**Parser Errors:**	No Errors
Module Name:	div_32	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 13.2	**• Warnings:**	10 Warnings (10 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3353	4656	72%
Number of 4 input LUTs	6050	9312	64%
Number of bonded IOBs	128	232	55%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sun Jun 30 22:32:42 2013	0	10 Warnings (10 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Design Summary (Synthesized)

Figure 21(a) Synthesis report

Power consumption report is shown in figure 21(b):

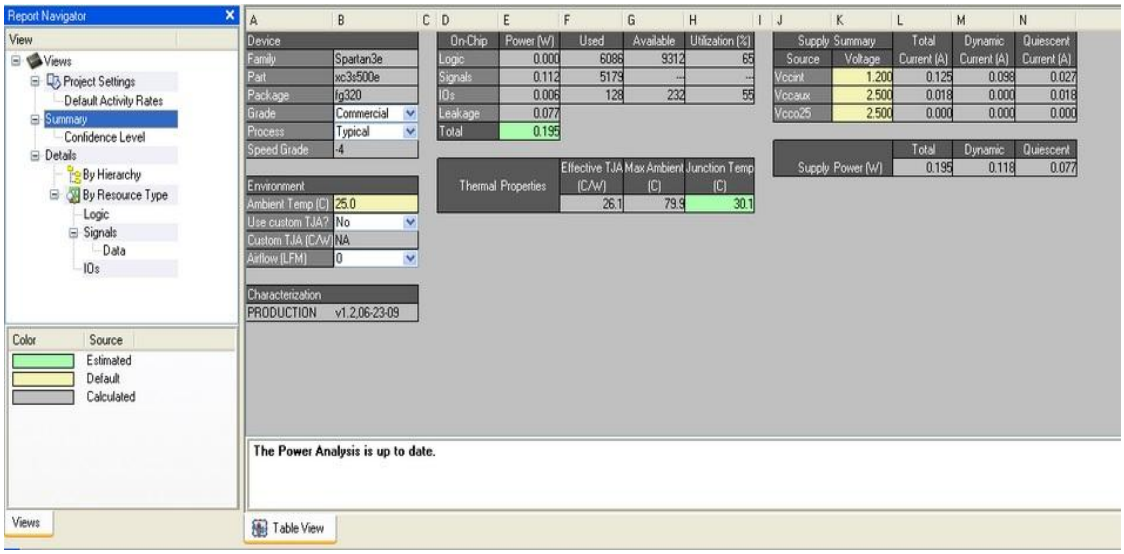

Figure 21(b) Power report

Table 11 shows the Area, power consumed and delay produced in 32-bit booth multiplier which uses Div. and conquer technique.

Table 11 Performance Report of 32-bit booth multiplier using DIV tech.

	326-bit booth multiplier using DIV tech.
No. of slices	3353
No. of LUTs	6050
Power Consumed(mw)	1950
Delay(ns)	98.245

3.8 Simulation Results

Simulation Results for 16 bit multiplier obtained using Model Sim are shown in Table 12.

Table 12 Simulation result table for 16 bit booth multiplier

S.No.	Multiplier	Multiplicand	Result
1.)	0101010101010101	0100101010100100	00011000111000010011110001110100
2.)	0100010010011111	0101010001001101	00010110100110001100111111010011
3.)	0100100100100111	0101010101010101	00011000011000100011110011110011
4.)	0100001001000111	0100000000001100	00010000100101001101101101010100

Simulation Results for 32 bit multiplier obtained using Model Sim is shown in Table 13.

Table 13 Simulation result table for 32 bit booth multiplier

S.No.	Multiplier	Multiplicand	Result
1.)	0101010101010111100001	00100010101011110	0101110001111111101100101101100111110
2.)	010101010101010101 010101010101	10001011010000010100 11100	00101110011010110001100111111111 11010001100101001110011 00
3.)	0101101100110011	0110000101010101	00100010101011001001101011101111
4.)	01010000100111001	0100110001111011	00110000001010100110001001100011

It can be observed from the above results that booth multiplier is working correctly.

3.9 Comparison of different multipliers

On the basis of above shown results(synthesis and power consumption), comparison of these multipliers is done which shows their efficiency in terms of power consumed, area, delay(speed) with respect to each other.

Table 14 Comparison table

	Area			
Module	**Number of slices**	**Number of LUTs**	**Power(mW)**	**Delay(ns)**
16 bit Booth Multiplier	817	1467	114	80.908
16 bit Booth Multiplier with CS Adder	964	1706	112	84.24
16 bit Booth Multiplier with LAC Adder	815	1457	113	82.829
16 bit Booth Multiplier with DIV and CS Adder	908	1636	115	51.446
16 bit Booth Multiplier with DIV and LAC Adder	700	1259	113	46.94
32 bit booth multiplier	3601	6488	195	165.383
32 bit booth multiplier with DIV	3353	6050	195	98.245

It is clear from above comparison that time delay in booth multiplier is minimum when look ahead carry adder is used. It is consuming even minimum area, power when used with carry look ahead adder.

Now booth multiplier is implemented using divide and conquer technique(with carry select as well as look ahead carry adder). Comparison results shows that time delay of booth multiplier is reduced to 46.94 ns, which is lowest among others. So with the help of Div. & Conq. Technique and CLA Adder, speed of the multiplier is increased significantly(almost double) without significant increase in other parameters(power,area).

Similar results are also obtained for 32 bit booth multiplier which shows that time delay of the multiplier is reduced to 98.245 ns from 165.383 ns by the use of divide and conquer technique.

Chapter 4 Conclusion and Future scope

As booth multiplier is synthesized using different Adders, it is observed that carry look ahead adder gives the best response in terms of speed, i.e. time delay is minimum while using carry look ahead adder. Further, divide and conquer technique is used in booth multiplier which divides the multiplication task into sub tasks and thus make the multiplication quiet fast.

Besides this work, there is still more to be done such as designing of configurable booth multiplier, reduction of switching activities in circuit, selection of proper inputs for multiplier, using different architecture of multiplier for reducing the delay.

REFERENCES

1. Pascal Constantin Hans Meier, "Analysis and Design of Low Power Digital Multipliers", A Dissertation, Pittsburgh, Pennsylvania, August, 1999.

2. C.S. Wallace, "Suggestion for a Fast Multiplier", IEEE Trans. Electron. Computers, EC-13, pp.14-17, 1964.

3. T.Cormen, C. Leiserson, R. Rivest, "Introduction to Algorithms", McGraw-Hill Publishers, Massachusetts, 1990.

4. W.J. Cody, "A Proposed Standard for Binary Floating-Point Arithmetic", computer Magazine, 1981.

5. D. Goldberg, J.L. Hennessy and D.A. Patterson, "computer Arithmetic in Computer Architecture A Quantitative Approach", San Mateo, CA: Morgan Kaufmann Publishers, inc., pp A1-A66, 1990.

6. J. Fadavi - Ardekani, "MxN Booth Encoded Multiplier Generator Using optimized Wallace Trees", IEEE Transactions on VLSI Systems, Vol. 1, no. 2, pp. 120-125,June 1993

7. Xiaoxiao Zhang, Amine bermak, Farid Boussaid "Power Optimization in Multipliers using Multi-Precision combined with Voltage Scaling Techniques", Quality Electronics design, pp. 79-82, 2009.

8. Jongsu Park, San kim, Yong-Surk Lee, "A Low-Power booth Multiplier Using Novel Data Partition method", Proceedings of IEEE Asia pacific Conference on Advanced systems integrated circuits, pp. 54-57, 2004.

9. D. Jackuline Moni, P. Eben Sophia, "Design of Low Power and High speed Configurable booth Multiplier", 3rd International conference on Electronics Computer Technology, pp. 338-342, 2011.

10. Hanho Lee, "A Power Aware Scalable Pipelined Booth Multiplier", IEEE International SOC conference, pp. 123-126, 2004.

11. S. Saravanan, M. Madheswaran, “Design of Hybrid Encoded Booth Multiplier with Reduced Switching Activity Technique and Low Power 0.13um Adder for DSP Block in Wireless Sensor Node”, International Conference on Wireless Communication and sensor computing (ICWCSC), pp. 1-6,2010.

12. Richard F. Hobson and Michael W. Fraser, “An Efficient Maximum Redundancy Radix-8 SRT Division and Square Root Method”, IEEE Journal of Solid State Circuits, vol. 30, no. 1, January 1995.

13. Changku Hwang, Akira Hyogo, Mohammed Ismail, Hong-sun Kim and Gyu Moon, “LV CMOS High Speed Multiplier”, Proc. of the Midwest symposium on circuits and system vol. 2, pp. 1189-1192.

14. Young-Ho Seo and Dong-Wook Kim, “New VLSI Architecture of Parallel Multiplier-Accumulator Based on Radix-2 Modified Booth Algorithm”, IEEE Transaction on VLSI systems, vol. 18, no. 2, February 2010.

15. S. Sri Sakthi and N. kayalvizhi, “Power Aware and High Speed reconfigurable Modified Booth Multiplier”, IEEE Recent Advances in Intelligent Computational System, pp. 352-356, Sept. 2011.

16. Jung-Yup Kang and Jean-Luc Gaudiot, “A Simple High Speed Multiplier Design”, IEEE Transactions on Computers, vol. 55, no. 10, October 2006.

17. Wenbing Jin, Bo Zhu and Xuanya Li, “A Novel Pipelined Multiplier Using Divide and Conquer Algorithm”, 2nd International Conference on Industrial technology and Management, 2012.

18. Y. Mareswara Rao, Mr. A. Madhusudan, “Radix-4 Configurable Booth Multiplier for Low Power and High Speed Application”, IOSR Journal of Electronics and Communication Engineering, vol. 4, Issue 2, pp. 31-37, Dec. 2012.

19. R.L. Bhargavi, M.Merlin Moses, V.karthikeyan and C. Karthikeyan, “Design of a High Speed Matrix Multiplier Based on a Balanced Word-Width Decomposition and

karatsuba Multiplication", International Journal of Soft Computing and Engineering, vol. 2, Issue-6, January 2013.

20. Soojin Kim and Kyeongsoon Cho, "Design of High Speed Modified Booth Multipliers Operating at GHz Ranges", World Academy of Science, Engineering and Technology , January 2010.