CHAPTER 1

INTRODUCTION TO COMPUTER ARCHITECTURE
AND MULTIPLIERS

1.1 INTRODUCTION

Digital Signal Processing (DSP) is a technology that is omnipresent in almost every engineering discipline. High speed adders and multipliers are of extreme importance in DSP to perform convolution, discrete Fourier Transforms, Digital filters etc. Public Key Cryptography is another field where the selection of multiplier plays an important role in deciding the performance of the system. It consists of raising the elements of some groups such as GF (2n) or elliptic curves to large powers and reducing the result modulo some given element. This operation is called modular exponentiation.

These different types of applications demand different performance measures from the multipliers such as scalability, reconfigurability, high speed, low power consumption, regularity of layout and less area or a combination of one or more of these features. Despite the variations in size and complexity, the small machines perform their arithmetic and logic operations using the same principles as large ones except that the speed with which algorithms are executed by hardware units differ. In addition special techniques are used to improve the speed by performing several operations in parallel.
Modular arithmetic plays an important role in various fields such as residue number system arithmetic and cryptography. Modular exponentiation is a common operation for scrambling and is used by several public key cryptosystems. The modular exponentiation applies modular multiplication repeatedly. So the performance of public key cryptosystems is primarily determined by the implementation efficiency of the modular multiplication and exponentiation.

As the operands (the plain text or the cipher text or possibly a partially ciphered text) are usually large (ie 1024 bits or more) and in order to improve time requirements of the encryption/decryption operations, it is essential to attempt to minimize the number of modular multiplications performed and to reduce the time required by a single modular multiplication.

The need for high speed processing has been increasing as a result of expanding signal processing and computer applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real time signal and image processing applications. Multiplication operations are quite important and require substantially more hardware resources and processing time than addition and subtraction operations. Existing designs include (i) integrated special purpose arithmetic circuits, which are used by the CPU to execute arithmetic instructions (ii) utilizing external multiplier circuits to reduce the time required to perform multiplication.

Ancient Indian Vedic Mathematics consists of sixteen mathematical formulae reconstructed from the Atharvaveda and has been recognized as a technique for improving the mathematical skills of school students. The multiplication, squaring, cubing and finding square roots and cube roots have always been time consuming processes for man as well as machine. The
Vedic Mathematics provides easy and speedy solutions for these difficult processes.

The proposed architecture using Ancient Indian Vedic Mathematics has the advantage that as the number of bits increases, its gate delay and area increases very slowly as compared to other multiplier architectures. It is estimated that this design is quite sufficient in terms of silicon area/speed. Such a design should enable substantial savings of resources in the FPGA when used for image and video processing applications.

1.2 ALGORITHMS FOR MULTIPLICATION

Algorithms that formalize the operation of multiplication generally consist of two steps: one that generates a partial product and the other that accumulates it with the previous partial products. The common multiplication method uses add and shift operators. The shift operation generates the partial products and the adder units sum them up. The general scheme for unsigned multiplication in base $b$ is shown in Figure 1.1.

![Multiplication Process](image)

**Figure 1.1 Multiplication Process**

Each row or partial product is obtained by multiplying one digit of the multiplier times the multiplicand. The low order digit of a partial product
is determined from just one multiplicand digit, but other digits include the effects of the carry from the digits to the right. In binary, partial products are trivial – either a copy of the multiplicand or zero. The sum of the partial products gives the computed result.

1.2.1 Classification of multipliers

Various ways of performing multiplication, range from, treating one pair of digits at a time, to generating and summing all partial products simultaneously with parallel hardware units. The multipliers are classified as serial and parallel multipliers. Serial multipliers have increased delay while parallel multiplier eliminates this problem by processing the bits simultaneously. Array multipliers can be implemented by directly mapping the manual multiplication process into hardware. In these multipliers, the partial products are accumulated by an array of adder circuits. An n x n array multiplier requires ‘n (n-1)’ adders and ‘n^2’ AND gates, where ‘n’ represents the number of bits.

1.2.1.1 Carry save array multiplier

The carry-save array multiplier (CSA) uses an array of carry-save adders for the accumulation of partial product. It uses a Carry-Propagate Adder (CPA) for the generation of the final product. This reduces the critical path delay of the multiplier as the carry-save adders propagate the carry to the next level of adders rather than to the adjacent ones. The array multipliers are not suited for signed binary multiplication, because of the absence of the sign bit expansion.
1.2.1.2. Booth Multiplier

Booth algorithm is a powerful algorithm for signed-number multiplication and treats both positive and negative numbers uniformly. It is a method that will reduce the number of multiplicand multiples. Booth’s algorithm effectively skips over runs of 1s and 0s that it encounters in the multiplier and thus optimizes the number of add and shift operations. For large multipliers, the modified Booth recoding technique is used. This converts the multiplier into higher radix numbers thereby reducing the number of stages of partial products to be added, by half. The scheme is modified from the original Booth’s recoding to avoid a variable-size partial product array. Generally, array multipliers have larger delay but offer the benefits of regular layout with simpler interconnects.

1.2.1.3. Wallace Multiplier

Wallace C.S. (1964) propounded a fast technique to perform multiplication of large operands. Unlike an array multiplier, the partial product matrix for a tree-multiplier is rearranged in a tree-like format, reducing both the critical path and the number of adder cells needed. The Wallace tree multiplier belongs to a family of multipliers called column compression multipliers. The underlying principle in this family of multipliers is to achieve partial product accumulation by successively reducing the number of bits of information in each column using full adders or half adders. The Wallace tree consists of numerous levels of such column compression structures until finally, only two full-width operands remain. These two operands can then be added using fast carry-propagate adder to obtain the product result. Thus the Wallace tree multiplier uses as much hardware as possible to compress the partial product matrix as quickly as possible into the final product.
1.2.1.4. Dadda Multiplier

Dadda multiplier is a hardware based multiplier design similar to Wallace multiplier. Unlike Wallace multipliers that perform compressions as much as possible on each layer, Dadda multipliers do as few reductions as possible and offers less expensive reduction phase, at the cost of slightly bigger adders. Thus fewer columns are compressed in the first few steps of the column compression tree, and more columns in the later levels of the multiplier. Tree multipliers have the shortest logic delay but has irregular layouts with complicated interconnects. This demands more physical design effort and also introduce significant interconnect delay and introduce noise due to increased number of wiring capacitances.

With all these available multiplier structures and algorithms, it is essential to choose a particular type of multiplier suited for a given application based on certain parameters like delay, power consumption, complexity of the circuit etc.

1.3 VEDIC MATHEMATICS

Vedic Mathematics is an ancient Vedic System of Mathematics, explored recently. This used a unique technique of performing calculations based on simple rules and principles with which a mathematical problem can be solved – be it arithmetic, algebra, geometry or trigonometry. The system is based on 16 Vedic Sutras or aphorisms, which are actually word formulae describing natural ways of solving a whole range of mathematical problems. It also makes multiplication a very simple operation, compared to the conventional methods.
Implementation of Vedic Sutras for processor based multiplication has been tried with various parameters as performance targets. In this work, an attempt has been made to analyse the delay and power requirements of various small and wide bit multipliers implemented using Vedic Sutras. For wide bit multiplication, Karatsuba–Ofman method is selected and implemented using Vedic multiplier and the performance measure speed is obtained and compared with conventional schemes.

1.4 LITERATURE REVIEW

Based on the conventional add and shift algorithm, the multipliers may be classified into serial and parallel multipliers and both form the basis for structured multipliers. For an ‘n’ bit input, Serial multipliers use one adder to add the $n^2$ partial products. Multiplicand and multiplier inputs have to be arranged in a special manner to synchronize with circuit behavior but could be presented at different rates. Hence serial multipliers could be used where area and power is of utmost importance and delay can be tolerated. Serial/parallel multiplier feeds one operand in parallel while the other is serial. N partial products are formed for each cycle.

1.4.1 A Survey on Algorithms for Structured Multipliers

In parallel multipliers, the number of partial products to be added mainly determines the performance of the multiplier. With increasing parallelism, the number of shifts between the partial products and intermediate sum to be added will increase. This may result in increased delay, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand, Rabaey J. et al. (2002) has observed that
serial-parallel multipliers compromise speed to achieve better performance for a specified area and power consumption.

Another category of structural parallel multipliers called tree multipliers reduce the number of sequential adding stages in the array multiplier and hence has increased speed. The trade off is irregularity of structure, complex routing and increased power consumption. However, in high performance DSP systems, multipliers are used to achieve high throughput, but the operand size is not large. In such systems tree multipliers do not have much speed improvement and perform similar to the array multipliers. For example, if the operand size is 16 bits, 6 CSA stages are required in an array multiplier for the generation of a 21 bit final adder input, while a tree multiplier has 5 CSA stages and a 26 bit final adder.

Booth algorithm (Booth A.D., 1951), (Madrid P.E.et al. 1993) is a powerful algorithm for signed number multiplication which treats both positive and negative numbers uniformly. This algorithm is a method that will reduce the number of multiplicand multiples. For a given range of numbers to be represented, a higher representation radix leads to fewer digits. Since a k bit binary number can be interpreted as k/2 digit radix – 4 number, a k/3 digit radix – 8 number and so on it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication.

Most of the multiplication algorithms for high speed implementation are based on Booth encoding algorithm and its modifications. The main disadvantage of the add and shift method is that it is slow and that any temporal correlation between successive data words will be removed due to the time multiplexed nature of the computation which has an adverse effect on the power consumption of the implementation.
Wang Z. et al. (1964) suggests that since with equal size multiplicand and multiplier and only with adder cells as building blocks, the area of a column compression multiplier can greatly be reduced by reducing the number of cross stage interconnections or by allocating approximately the same number of adders to each stage. Also, if the length of the multiplier is short, it is advantageous to shorten the length of the final adder. This improves the area efficiency by approximately 97.5% along with a reported reduction in the length of the final adder from 14 to 10.

The principle for the column compression multipliers was established by the early works of Ofman, Wallace and Dadda. It has been shown that the delay of the column compression multiplier is proportional to $\log_{1.5}n$ and the column compression architecture is widely accepted as time optimal. The irregularity and complicated interconnections of the column compression multiplier do not, however, readily allow efficient VLSI implementations, particularly for large ‘n’. A Complementary Metal Oxide Semiconductor (CMOS) implementation of an 8x8 Dadda multiplier also exists with multipliers using 4-2 compression and (7,3) counters, to improve the speed and/or regularity.

A testable parallel multiplier can be built by breaking the multiplier into summand generator and summand counter. Hong S.J. (1990) proposes a new testable summand generator and testable summand counter which combines to form a parallel multiplier which can be tested using $[3x + 60]$ vectors using only one extra pins.

Booth algorithm (Madrid P.E. et al. 1993), (Demirsoy S.S.et al. 2003) may be implemented on array or tree structure multipliers for sign extension. A combination of Booth array with Wallace tree structure has been
shown to be fastest with a nominal area power product and area delay product.

A method to reduce the number of partial products by a factor of two from the straightforward carry save array multiplier and thus to provide higher speed has been proposed by Aziz S.M. (1995). However, if explicit sign extension is used, this requires extra circuitry and introduces extra delay and requires recoding the sign bits of the partial products as a two’s complement number.

Two basic array implementations of the multiplier are discussed along with a strategy to design for C testability by Boateng K.O et al (1997). These are multipliers based on Modified Booth algorithm with the partial product addition taken care by a skewed array called Iterative Logic Array.

Park B. et al. (1999) propose a new parallel array multiplier using weighted CSA, which reduces area by 27%. Their work aims at by eliminating redundant silicon area by connecting each stage with ‘n’ CSA cells and one Weighted CSA (WCSA) cell for generating the carry output of that stage. This improves the overall performance because the operand size of the final CPA is reduced by eliminating the additional bits.

Area reduction and power reduction in multipliers is a thriving area of research. A new self timed clocking scheme is proposed to reduce the area required for a multiplier based on the Stanford Pipeline Iterative Multiplier architecture by Shin M.C. et al. (2001). This minimizes timing waste and allows the multiplier to be integrated generically independent of the system clock.
VLSI implementation of the wide bit Dadda multiplier has several disadvantages. The first disadvantage is the irregular structure and the associated complicated and long wiring interconnection has the potential to significantly reduce the performance.

The column compression multiplier, using the 4-2 compressor or (7, 3) counter as a building block, may improve the speed and/or structural regularity (Au L.S., 2002).

Recently, minimizing the cost of FPGA hardware had been the goal of many multiplier designers. McPherson K.N. et al. (2004) has presented a new algorithm, which synthesizes multiplier blocks with this objective. They established that minimizing adders does not minimize multiplier block logic depth to reduce FPGA hardware.

To improve the area and increase the speed of a Booth multiplier, an asynchronous counter flow organization is introduced by Hensley J. et al. (2004). In addition, merging the arithmetic and shifter units and performing overlapped execution of multiple iteration of the Booth algorithm provides significant improvement in area, energy and speed.

Once the partial products are generated, the addition process is similar to the array multiplier. Carry Save adders are used with the final sum added using a Carry Propagate Adder. This multiplier is faster and requires less area than other multipliers (Chittibabu A., 2006).

Chittibabu A et al. (2006) propose a versatile new Dadda multiplier architecture for signed multiplication that supports operands in both signed and two’s complement formats. This architecture maintains the pure form of a Dadda multiplier without using Booth Encoding architecture for sign
extension and proves that Dadda multiplier provides better performance in terms of delay and area.

### 1.4.2 A Survey on Multipliers for Modular Multiplication

Tenca A.F. et al. (1999) propose a scalable architecture for the computation of modular multiplication, based on the Montgomery multiplication (MM) algorithm. This is an efficient method for modular multiplication with an arbitrary modulus, particularly suitable for implementation on general purpose computers and embedded microprocessors. The method is based on a representation of the residue class modulo M. This uses simple divisions by M, which are used in a conventional modular operation. This new architecture is capable of manipulating any operand precision while using even a small chip area and it is capable of working at high clock frequencies. It is highly flexible and allows the investigation of several design tradeoffs and hence is suitable for embedded system based implementation.

Analogous to addition, modulo m multiplication can be implemented by means of tables but this approach is limited to small moduli due to exponential growth of the required memory. A modulo (2n+1) multiplier based on Ma’s algorithm is described by Beuchat J.L. et. al. (2003) along with its optimized FPGA implementation.

An appropriate complexity measure for Modular multiplication algorithms is the product of area and time. Buminov V. et. al. (2003) describes a new algorithm for modular multiplication of large integers optimized in hardware complexity as well as latency and finds wide application in the security units of mobile devices. This combines carry save adders and constant time comparison and also a small amount of
precomputing in the loop and so is superior to any reported method with respect to area and time.

Modular multiplication $A \times B \mod M$ can be performed in two different ways: (i) multiplying and then reducing. For eg. computing $P = A \times B$, and then $R = P \mod M$ or (ii) interleave the multiplication and the reduction steps. There are various algorithms that implement modular multiplication. The most prominent are Karatsuba – Ofman’s and the Booth’s methods for multiplying, Barrett’s method for reducing and Montgomery’s algorithm and Brickell’s method for interleaving multiplication and reduction (Nedjah N. et al., 2006).

Karatsuba-Ofman algorithm is considered as one of the fastest ways to multiply long integers. It is based on a divide and conquer strategy. A multiplication of $2n$ digit integer is reduced to two $n$ digit multiplications, one $(n+1)$ digit multiplication, two $n$ digit subtractions, two left shift operations, two $n$ digit additions and two $2n$ digit additions. Here $2n$ bit multiplication is reduced to three $n$ bit multiplications, which makes this multiplier an asymptotically faster one than the standard multiplication algorithm. This algorithm is modified in Adapted Karatsuba algorithm (Nedjah N. et al., 2006) so that the third multiplication is performed efficiently. This is done by using recursive shifting and multiplexing operations. This requires more hardware area but reduces time requirements as well as area x time factor.

One of the widely used algorithms for efficient modular multiplications is the Montgomery’s algorithm. This algorithm computes the product of two integers modulo a third one without performing division by $M$. It yields the reduced product using a series of additions. A modified version of Montgomery’s algorithm called Systolic Montgomery multiplier improves speed. The performance factor is also improved when the operand size is
more than 256 bits. A modular reduction called Barrett’s reduction method can be used to determine the remainder of a division using two simple multiplications. A modular Booth multiplier is proposed by Nadia Nedja et al. (Nedjah N. et al., 2006).

1.4.3 A Survey on Multipliers for Floating Point Arithmetic

In modern computers, the floating point unit is the part of the processor delivering the highest computing power and getting most attention from the design team. Performance of any multiple precision applications will be dramatically enhanced by adequate use of floating point expansions. Three novel multiplication algorithms faster and more integrated than the other in existence, have been proposed and tested on an application which finds the determinant of a matrix by Daumas M. (1999). It has been proved that these algorithms reduce running time from 20% to 50%.

A single precision IEEE 754 floating point multiplier with high speed and low power is proposed by Thapliyal H. and Srinivas M.B. (2004). This is a highly regular, self repairable floating point parallel multiplier architecture which can be directly scaled for larger multiplication. The self repairability is enhanced by introducing 4x4 multiplier modules for 24 x 24 bit modules. This will allow the multiplier to recover from logic faults (stuck at faults) occurring in any module. Reconfigurability at run time is provided for attaining power saving. Use of Urdhva Triyakbhym eliminates the following problems:

- Large delay and irregular layout found in array multipliers
- Complicated interconnect structures
- Dynamic power consumption controlled by bitwidth
1.4.4 A Survey on Evolved Multipliers

Introducing a two-gate delay in the Booth encoder and partial product generator, which eliminates the unnecessary glitches associated with the Booth multiplier was proposed by Fried R. (1996). In addition, a modified signed / unsigned (MSU) and modified sign-generate (MSG) algorithms, suitable especially for signed/unsigned multipliers, was developed by Rafael Fried. This reduces the compression level needed in the Wallace tree and hence reduces the multiplier hardware. Using these features reduces the multiplier array energy dissipation by about 35% and increases speed by about 10%, according to Rafael Fried. The latter is achieved due to the elimination of the unnecessary glitches associated with Booth algorithm and reduction of the number of full adders needed to compress the multiplier array.

A study of scalability of combinational circuits, particularly the evolutionary design of 3 bit multiplier circuits in which the basic building blocks are small sub-circuits was done by Vassilev V.K. et. al. (2000) A major problem in the evolutionary design of electronic circuits is the problem of scaling. This refers to a very fast growth in the number of gates used in the target circuit as the number of inputs of the evolved logic function is increased, resulting in enormous search space and increase in the time required to calculate the fitness value of the circuit. This study shows that, in general, the principles of evolving digital circuits are scalable. Thus to evolve digital circuits using modules is faster, since the building blocks of the circuit are sub circuits rather than two input gates. This can also be a disadvantage, if the number of gates of the evolved designs increases relative to the size of the modules.
Similar studies on power, area and delay of arithmetic circuits have been carried out by Huang Z. et al (2002) by varying different number system representations with special study on two’s complement multipliers.

An accurate area, delay and power comparison for large scale designs comprising of arithmetic blocks was obtained by performing sizing optimization of their gates by Vazquez A.M. et al.(2003). It was demonstrated that improving one constraint results in worsening the other, while maintaining the functionality’s boundaries.

Lin R. et al. (2003) presents an extra regular, complexity reduced, high performance pipelined multiplier architecture, using recently proposed Triple Expansion Schemes. It is based on a parallel counter circuitry, called Borrow Parallel Counter circuits. Conventional architectures have initial partial product bit matrix as triangular or trapezoidal structures that result in low controllability and observability for test and significant layout cost and test cost. This triple expansion scheme with borrow parallel counter circuits allow large multipliers to be generated from smaller multipliers, tripling the size in each expansion. This significantly reduces the complexity of the designs and achieves full self-testability without sacrificing high performance. The advantages include simpler CMOS technology and layout, significant area reduction without sacrificing high performance, low power and full self-testability.

1.4.5 A Survey on Multipliers for Signed Integers

For multiplication with signed integers, three number representation systems are widely used – Sign and Magnitude (SM), Two’s Complement (TC), Signed Digit (SD) or Redundant Binary (RB). Numbers in SM representation have low switching frequency than numbers in TC
representation due to the simpler sign extension in SM. In digital CMOS designs, switching activity is closely connected with the total power consumption. Intuitively, radix-4 multipliers could consume less power than radix-2 counterparts as recoding reduces the number of partial products to half. However, the extra recoding logic and the complicated partial product generation logic are significant overheads. By examining existing radix-4 recoding schemes, two designs with power improvement are proposed for standard cell CMOS technology. With new recoding schemes, the power efficiency of radix-4 and radix-2 representation multipliers are restudied and it is proved that these new conversion schemes consume less than 30% power of the baseline schemes.

Han K. et al. (2004) propose another novel method to reduce the switching frequency in the multipliers by employing word length reduction of multiplicands, using truncation and signed right shift methods. For 32 bit x 32 bit Wallace and Radix -4 modified Booth multipliers, truncation by 16 bit achieves a 4:1 and 2:1 reduction respectively, in switching activity whereas signed right shift gives little or no reduction. But the major contribution is the reduction of power consumption by altering multiplicands in software without requiring any hardware modifications. Data word length reduction by the truncation method decreases the average transition counts and power consumption.

1.4.6. A Survey on Reconfigurable Multipliers

Reconfigurable multiplication structures were implemented and their performance was compared with existing algorithms and parallel multiplier structures by Haynes S.D. et al. (1998). A new methodology for the design of expandable multiplier blocks that leads to designs offering
significant speed advantage was presented. Radix 4 Booth design was shown to hold significantly both the speed advantage and also the benefit of being able to execute multiply and accumulate operations at no extra hardware or speed cost.

Although the above design improves area and time efficiency, it suffers a major limitation on flexibility. If a required multiplier cannot be fitted onto the available on-chip multiplier block resources, it cannot be implemented. Visvakul C. et al. (2000) presents a design for combining reconfigurable array known as Flexible Array Blocks (FABs) and digit–serial techniques to implement arbitrary size multipliers with limited resources. These digits–serial techniques provide the necessary trade off between number of clock cycles needed to perform a multiplication and the number of FAB resources used. This type of design known as DigiFab design allows any size of multiplier to be implemented with any fixed amount of FAB resource.

A reconfigurable and low power multiplier was developed by Margala M. et al (2001) for System-on-Chip applications. This pseudo exhaustive design verification approach greatly reduces the required level of test bench. The multiplier can be configured either as a 8 x 8 bit or 12 x 12 bit multiplier with either radix 2 or radix 4 coding option. Low power is achieved by generating some enable / disable control signals in the recoding stage thereby eliminating successive stage addition modules that do not perform necessary and useful operations.

A set of reconfigurable multiplier blocks that offer significant savings in area over traditional multiplier blocks were proposed by Demirsoy S.S. et al (2003). These are for time–multiplexed digital filters or any other system where only a subset of the coefficients that can be produced by the
multiplier block is needed in a given time. The basic structure comprises a multiplexer connected to at least one input of an adder / subtractor that can generate several partial products, leading to better area utilization.

Happonen A. et al. (2004) proposes a coarse grain reconfigurable processing element for telecommunication equipment which is capable of addition, subtraction, multiplication, division, squaring, square rooting and logarithms with the same hardware processing element.

HP Labs has developed a reconfigurable arithmetic array (RAA) termed “CHESS”, which provides a high computational density, wide internal data bandwidth and sufficient distributed register and memory resource for important multimedia applications. It also offers software flexibility, strong scalability and advance features for dynamic reconfiguration. Marshall A. et al. (1999) describe the architecture and features of this array highlighting its technical advantages over FPGAs.

1.4.7 A Survey on Multipliers using Vedic Mathematics

Vedic Mathematics is the name given to the ancient Vedic System of Mathematics, rediscovered recently. This has a unique technique of calculations based on simple rules and principles with which any mathematical problem can be solved – be it arithmetic algebra, geometry or trigonometry. The system is based on 16 Vedic Sutras or aphorisms, which are actually word formulae describing natural ways of solving a whole range of mathematical problems. Chidgupkar P.D. et al.(2004) highlight the use of multiplication process based on Vedic algorithms and its implementations on 8085 and 8086 microprocessors, resulting in appreciable savings in processing time.
Thapliyal H. and Srinivas M.B. (2004) have developed a NxN bit parallel overlay multiplier architecture for high speed DSP operations. The architecture is based on the Urdhva Tiryakbhyam – the Vertical and Crosswise algorithm of Vedic Mathematics. The whole multiplication operation is decomposed into 4 x 4 bit multiplication modules. The 4 x 4 multiplication modules are implemented using array and booth multipliers and a considerable improvement in the speed is achieved.

A Multiply and Accumulate (MAC) architecture using Vedic Mathematics is proposed by Thapliyal H. and Srinivas M.B. (2004), suited for high speed applications. In the proposed architecture all bits of operands (multiplier and multiplicand) and accumulator are presented in parallel. This multiplier concurrently adds the partial product bits generated with the accumulator bits and improves speed.

A new multiplier and square architecture is proposed by Thapliyal H. and Srinivas M.B. (2004) based on the algorithm of Ancient Indian Vedic Mathematics, for low power and high speed applications. It is based on generating all partial products and their sums in single step. It has been shown that as the number of bits in the multiplier increases the Vedic Multiplier has superior scalability over conventional multiplier methods. Similarly for the square architecture the gate delay and area reduces by 50%.

Liddicoat A.A. et al. (2000) proposes a parallel cubing unit that computes the cube of an operand 25% to 30% faster than can be computed using multipliers. The reduced squaring and cubing units are mathematically modeled and the performance and the area requirements are compared to that of conventional multipliers for 64 bit operands.
Another square and cube architecture based on Vedic Mathematics is presented by Thapliyal H. and Srinivas M.B. (2005). The square architecture is based on the Duplex property of binary numbers and the cube architecture is based on the Anurupya Sutra (algorithm) of ancient Indian Vedic Mathematics. The simulation results show improved speed and scalability.

1.5 INFERENACE AND PRESENT DIRECTION OF STUDY

From the survey of the research papers, it is obvious that the optimization of multipliers for the performance improvement of processors remains explored continuously. Also one can find that optimization of the performance of a multiplier does not depend on any specific parameter, but is the result of a combination of some or all of the parameters. There are still more algorithms and structures that can be evolved and analysed by researchers. Vedic Mathematics remains the least explored topic but the results obtained are encouraging.

The modular multiplication has a wide range of applications like digital signal processing, residue number system, public key cryptography etc. In addition, the modular multiplication provides a challenge for the researchers in that the present technology has strict limitations in chip area and power consumption because of the rather large size of the operands (typically of 1024 bits or more). Hence Urdhva Tiryakbhyam Vedic Multiplication which is a proven method for increasing the speed of calculation using decimal numbers is chosen for studying the performance parameters of the binary numbers in one of the modular multiplication methods - Karatsuba Algorithm.
1.6 OUTLINE OF THE THESIS

The thesis is organized in 6 chapters.

Chapter 2 gives a comprehensive discussion on the implementation of structure based and algorithm based multipliers. The structure-based multipliers presented in this chapter include: Array multiplier, Carry save array multiplier, Wallace tree multiplier and Dadda multiplier. Algorithm based multipliers include Booth multiplier, Booth Wallace multiplier, Column compression multipliers. The delay and power consumption of these multipliers are compared.

Chapter 3 gives details about the Vedic Mathematics which is proposed for use in this work. There are sixteen different aphorisms or formulae which deal with almost all mathematical operations. In this work, the Urdhva Tiryakbhyam Sutra, which elaborates the vertical and crosswise multiplication algorithm, is identified for use with existing multiplier structures.

Chapter 4 explains about the modular multiplication methods, and Karatsuba-Ofman algorithm in particular. Karatsuba-Ofman algorithm is considered as one of the fastest algorithms to multiply long integers. A multiplication of 2n digit integer is reduced to two ‘n’ digit multiplications, one ‘(n+1)’ digit multiplication, two ‘n’ digit subtractions, two left shift operations, two ‘n’ digit additions and two ‘2n’ digit additions.

Chapter 5 explains about VLSI implementation of conventional and modular multiplication with and without Vedic mathematics and summarises their implementation results. The Carry Save Array and the Wallace Tree Structural multipliers along with the Booth multiplier were implemented first.
independently. Then the Dvandva Yoga - Urdhva Tiryakbhyam sutra of Vedic Mathematics was used on the Carry Save Array, Wallace Tree and Booth multipliers separately to obtain the product terms in the multiplication modules. The logic simulation is done using Modelsim Simulator. The synthesis and Field Programmable Gate Array (FPGA) implementation were done using Xilinx navigator. Similar implementation for the wide bit multipliers using the Karatsuba – Ofman algorithm with and without the Urdhva Tiryakbhyam sutra of Vedic Mathematics is also presented.

Chapter 6 presents the conclusion and future work.