COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE

By

P.M. SNEHA ANGELINE

Reg. No. 1020106017

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University, Coimbatore)

COIMBATORE - 641049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

APPLIED ELECTRONICS

APRIL 2012

BONAFIDE CERTIFICATE

Certified that, this project report entitled "**COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE**" is the bonafide work of **Ms.P.M.SNEHA ANGELINE [Reg.No:1020106017]** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

(Ms.M.SHANTHI) (**Dr. RAJESWARI MARIAPPAN**)

Project Guide **Head of the Department**

The candidate with university Register no. 1020106017 is examined by us in the project viva-voce examination held on ……

Internal Examiner External Examiner

ACKNOWLEDGEMENT

I express my profound gratitude to our director **Dr.J.Shanmugham,** for giving this opportunity to pursue this course

At this pleasing moment of having successfully completed the project work, I wish to acknowledge my sincere gratitude and heartfelt thanks to our beloved Principal **Dr.S.Ramachandran,** for having given me the adequate support and opportunity for completing this project work successfully.

I express my sincere thanks to D**r.Rajeswari Mariappan Ph.D.,** the ever active, Head of the Department of Electronics and Communication Engineering, who rendering us all the time by helps throughout this project.

I extend my heartfelt thanks to my internal guide **Mrs.M.Shanthi M.S, Asso. Professor,** for her ideas and suggestion, which have been very helpful for the completion of this project work. Her careful supervision has ensured me in attaining perfection of work.

In particular, I wish to thank and everlasting gratitude to the project coordinator **Mrs.R.Hemlatha M.E., Asst.Professor,** Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

Last, but not the least, I would like to express my gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their encouragement and support throughout the course of this project.

ABSTRACT

The Multiplier and Accumulator (MAC) unit is used as a basic element in most of the digital signal processing application in order to perform repeated multiplication and addition. The conventional MAC architectures uses more shift and add operation at multiplier unit which increases delay in the arithmetic operations.

The main objective is to design a new multiplier and accumulator architecture to perform high speed arithmetic operation. The three cycle MAC (MAC-3C) architecture increase the performance by reducing the critical path delay by inserting an extra pipeline register either inside the partial product (PP) unit or between PP unit and final adder. The two cycle MAC (MAC-2C) architecture performs the carry propagation only in the second stage leads to the similar delay in multiplication and accumulation. The proposed MAC architecture (MAC-NEW) has two stages with the pipeline register inserted after the partial product unit. This unit uses carry-save adder which leads to the reduction of power. Due to the carry propagation in the second stage, multiplier's final adder is eliminated, leading to higher speed and lower energy. The Double Throughput MAC unit (DTMAC) switches between N-bit operations and 2×N/2-bit operations which reduces power and critical path delay on the removal of final adder.

Through the ***"COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE"*** is planned to obtain an efficient performance parameter such as gate count, delay and power for the different MAC architectures. The MAC architecture is designed using MODEL SIM and simulated using Xilinx ISE 9.2i and the parameters is compared to obtain an efficient architecture.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MAC	-------	Multiply- Accumulate Architecture
3-C MAC	-------	Three-Cycle Multiply Accumulate Architecture
2-C MAC	-------	Two-Cycle Multiply Accumulate Architecture
MAC-NEW	-------	Proposed Multiply Accumulate Architecture
DTMAC	-------	Double Throughput Multiply Accumulate Architecture
PP	-------	Partial Product
TP	-------	Twin Precision
TP-PPRT	-------	Twin-Precision Partial-Product
CPA	-------	Carry Propagation Adder
CSA	-------	Carry Save Adder
BW	-------	Baugh-Wooley Algorithm

CHAPTER 1

INTRODUCTION

With the recent rapid advances in multimedia and communication system,real-time signal processings like audio signal processing, video/image processing or large-capacity data processing are intrestingly being demanded.The multiplier and multiplier and accumulator(MAC) are the essentials elements of the digital signal processing such as filtering,convolution and inner products.Most digital signal processing methods use nonlinear functions such as discrete cosine transform(DCT) or discrete wavelet transform Because they are basically accomplished by repetitive application of mulitiplication and addition,the speed of the multiplication and addition arithmetic's determines the execution speed and performance of the entire calculation.As the multiplier requires longest delay among the basic operational blocks in digital system,the critical path is determined by the multiplier.

The multiplier consists of three parts: partial product generation, partial product summation and accumulation. The multiplier is much more complex than the accumulate adder, many design techniques have focused on reducing multiplier delay. In the architecture, the critical path is reduced by inserting an extra pipeline register, either inside the partial product unit or between the partial product unit and final adder. It has a better performance because of the reduction in critical path delay. The most effective way to increase the speed of a multiplier is to reduce the number of partial products using high speed compressors or speed optimized structures because multiplication precedes a series of additions for the partial products. The guard bits are important for avoiding overflow when computing long sequences of multiply accumulate operation.

In order to improve the speed of the MAC unit, there are two major bottlenecks. The first is the partial product reduction network that is used in the multiplication block and the second is the accumulator. Both of these stages require addition of large operands that involve long paths for carry propagation. As the multiplier is more complex than the accumulator, design techniques are proposed on reducing the delay in the multiplier either inside the Partial Product (PP) unit or in the final adder. Inside the PP unit, the partial-product circuitry might be implemented using the modified-Booth algorithm or one of its successors. The partial-product reduction tree of the PP unit can be implemented using high-speed compressors or speed-optimized structures. Mathew *et al.* propose a sparse-tree carry look-ahead adder for fast addition of the PP unit outputs and Liu *et al.* introduce a hybrid adder o reduce delay compared to a design that assumes equal arrival time on all adder inputs.

Here a MAC-NEW architecture is proposed in which the first stage is significantly faster compared to the second stage, leading to a better delay balance between the two stages. The key feature to this architecture is the implementation of product sign extension in the second stage, together with the accumulate adder such as carry save adder and the saturation unit. Guard bits are used for avoiding the overflow on computation of long sequences of multiply-accumulate operation. This MAC-NEW unit is efficient in terms of delay, power and gate count.

1.1 OBJECTIVE OF THE WORK

The performance of the multiply and accumulate unit is improved by either using high speed multipliers or improved fast adder architectures. To obtain a high speed operation, the multiplication unit is combined with accumulation and carry save adder (CSA).The partial product is generated using Baugh Wooley algorithm. The result is sign extended to have the same size as the accumulate adder. The MAC unit is designed using VHDL code and simulated using MODELSIM. The performance parameters such as

power, gate count, and delay are synthesized using XILINX and compared with the conventional MAC architecture.

1.2 INTRODUCTION TO VHDL

VHDL is an acronym which stands for VHSIC Hardware Description Language.VHSIC means Very High Speed Integrated Circuits. It is being used for documentation, verificatoin and synthesis of large digital designs.VHDL is a standard developed by IEEE.The different approaches in VHDL are structural, data flow and behavioral methods of hardware description.

1.2.1 STRUCTURAL DESCRIPTIONS

Building Blocks

Every portion of a VHDL design is considered a block. A VHDL design may be completely described in a single block, or it may be decomposed in several blocks. Each block in VHDL is analogous to an off-the-shelf part and is called an entity. The entity describes the interface to that block and a separate part associated with the entity describes how that block operates. The interface description is like a pin description in a data book, specifying the inputs and outputs to the block. The description of the operation of the part is like a schematic for the block.

The following is an example of an entity declaration in VHDL

```
Entity latch is
  Port (sir: in bit;
        q,nq: out bit);
end latch;
```

The first line indicates a definition of a new entity called latch. The last line is the end of the definition. The lines in between, are called the port clause, which describe the interface to the design. The port clause contains a list of interface declarations. Each interface declaration defines one or more signals that are inputs or outputs to the design. Each interface declaration contains a list of names, mode and type.

The following is an example of an architecture declaration for the latch entity.

```
architecture dataflow of latch is
  signal q0 : bit := '0';
  signal nq0 : bit := '1';
begin
  q0<=r nor nq0;
  nq0<=s nor q0;
  nq<=nq0;
  q<=q0;
end dataflow;
```

The first line of the declaration indicates the definition of a new architecture called dataflow and it belongs to the entity named latch. So this architecture describes the operation of the latch entity. The schematic for the SR latch

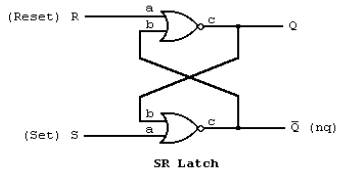

Figure 1.1 Schematic SR Latch

1.2.2 DATA FLOW DESCRIPTIONS

In the data flow approach, circuits are described by indicating how the inputs and outputs of built-in primitive components are connected together.The following SR latch using VHDL is described as in the following schematic.

```
entity latch is
  port (s,r : in bit;
        q,nq : out bit);
```

```
end latch;
architecture dataflow of latch is
begin
  q<=r nor nq;
  nq<=s nor q;
end dataflow;
```

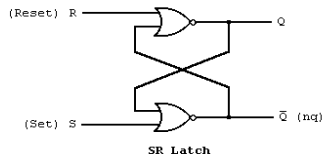

Figure 1.2 Dataflow approach of Schematic SR Latch

The signal assignment operator in VHDL specifies a relationship between signals. The architecture part describes the internal operation of the design. The scheme used to model a VHDL design is called discrete event time simulation. In this the values of signals are only updates when certain events occur and event occurs at discrete instances of time.

The Delay Model

The two models of delay that are used in VHDL. The first is called the inertial delay model. The inertial delay model is specified by adding an after clause to the signal assignment statement. The next is the transport delay model, just delays the change in the output by the time specified.

1.2.3 BEHAVIORAL DESCRIPTIONS

The behavioral approach to modeling hardware components is different from the other two methods in that it does not necessarily in any way reflect how the design is implemented.

The Process Statement

It is basically the black box approach to modeling. It accurately models what happens on the inputs and outputs of the black box, but what is inside the box (how it works) is irrelevant. The behavioral description is usually used in two ways in VHDL. First, it can be used to model complex components.

Behavioral descriptions are supported with the process statement. The process statement can appear in the body of an architecture declaration just as the signal assignment statement does. The process statement can also contain signal assignments in order to specify the outputs of the process.

Using Variables

A variable is kinds of objects used to hold data and also behaves like you would expect in a software programming language, which is much different than the behavior of a signal. Although variables represent data like the signal, they do not have or cause events and are modified differently. Variables are modified with the variable assignment.

Sequential Statements

There are several statements that may only be used in the body of a process. These statements are called sequential statements because they are executed sequentially. The types of statements used here are if, if else, for and loop.

Signals and Processes

This section is short, but contains important information about the use of signals in the process statement. The issue of concern is to avoid confusion about the difference between how a signal assignment and variable assignment behave in the process statement. Remember a signal assignment, if anything, merely schedules an event to occur on a signal and does not have an immediate effect. When a process is resumed, it executes from top to bottom and no events are processed until after the process is complete.

Program Output

In most programming languages there is a mechanism for printing text on the monitor and getting input from the user through the keyboard. Even though the simulator monitors the value of signals and variables in the design, it is able to output certain information during simulation. It is not provided as a language feature in VHDL, but rather as a standard library that comes with every VHDL language system. In VHDL, common code can be put in a separate file to be used by many designs. This common code is called a library. The write statement can also be used to append constant values and the value of variables and signals of the types bit, bit_vector, time, integer, and real.

1.3 SOFTWARE USED

- Modelsim PE5.4E
- Xilinx ISE 9.2i

1.4 ORGANIZATION OF THE REPORT

- **Chapter 2** discusses about the overview of MAC.
- **Chapter 3** discusses the existing architecture of MAC.
- **Chapter 4** discusses the proposed architecture of MAC.
- **Chapter 5** discusses the application of proposed architecture of MAC.
- **Chapter 6** presents the simulation results and discussions.
- **Chapter 7** presents the conclusion and future scope.

CHAPTER 2
OVERVIEW OF MAC

2.1 GENERAL ARCHITECTURE OF MAC

The general construction of the MAC operation is given by the equation

$$Z=A\times B+X$$

Where the multiplier A and multiplicand B are assumed to have n bits each and the addend X has (2n+1) bits. The basic MAC unit is made up of a multiplier and an accumulator as shown in Fig 2.1. The multiplier can also be divided into partial product generator, summation tree and final adder. It executes the multiplication operation by multiplying the input multiplier and multiplicand. This is added to the previous multiplication result as the accumulation step.

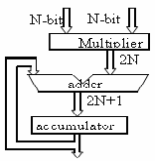

Figure 2.1: General MAC architecture

The summation network represents the core of the MAC unit and occupies most of the area, power and delay. Several algorithms and architectures are developed to optimize the implementation of this block. The addition network reduces the number of partial products into two operands representing a sum and a carry. The final adder is then used to generate the multiplication result out of these two operands. The last block is the accumulator, which is required to perform a double precision addition operation between the multiplication result and the accumulated operand. It involves a very large adder due

to the large operand size. This stage represents a bottleneck in the multiplication process in terms of speed since it involves horizontal carry propagation. The MAC unit is classified into various types such as 2-Cycle MAC unit,3-Cycle MAC unit, MAC-NEW unit and DTMAC unit.

2.2 BLOCK DIAGRAM OF PROJECT

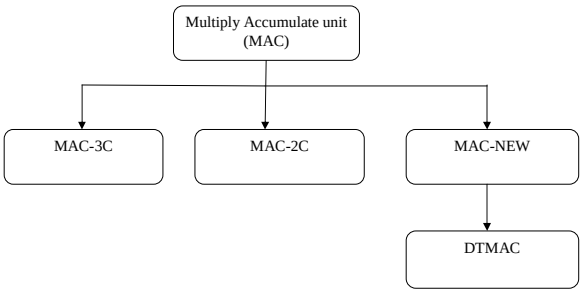

Figure 2.2: Block Diagram of the project

The overall block diagram of the project is shown in Fig2.2.The multiply accumulate unit is broadly classified into three types such as Three-cycle MAC unit(MAC-3C),Two-cycle MAC unit(MAC-2C) and MAC-NEW unit. The three architectures are implemented using BAUGH-WOOLEY algorithm. The proposed MAC unit has the better performance in comparison with the conventional architectures. The MAC-NEW is used to create a versatile MAC unit is called DOUBLE THROUGHPUT MULTIPLIER AND ACCUMULATE UNIT (DTMAC).

2.3 PROCESS FLOW IN MAC

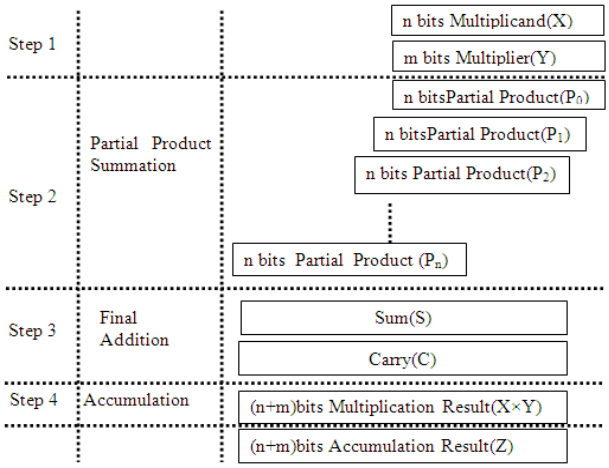

Figure 2.3: Basic Arithmetic steps of multiplication and accumulation

A multiplier can be divided into four operational steps as shown in Fig 2.3. The first step is the multiplication operation with the input multiplier and the multiplicand. The second step is the partial product summation which is used to add all the partial products and convert them into the form of sum and carry. The third step is the final addition in which the final multiplication result is produced by adding the sum and carry. The last step is the accumulation which takes place with the multiplication and the accumulated result.

2.4 BAUGH-WOOLEY ALGORITHM

An algorithm for direct 2's complement array multiplication has been proposed by BAUGH-WOOLEY and this algorithm is used in the design of multiplier and accumulator structures. The primary advantage of this algorithm is that the signs of all the partial products are positive and thus allowing the array to be entirely the same as conventional standard array structures.

The following

- Algorithm for two's-complement multiplication.
- Adjust partial products to maximize regularity of array multiplication.
- Moves partial products with negative signs to the last step also add negation of partial products rather than subtracts.

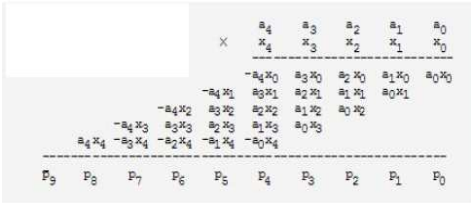

Figure 2.4: Unsigned multiplication for Baugh-Wooley algorithm

The Baugh-Wooley algorithm for the unsigned binary multiplication is based on the concept shown in Fig2.4.The algorithm specifies that all possible AND terms are created first and then sent through an array of half-adders and full-adders with the carry-outs chained to the next most significant bit at each level of addition.

For signed multiplication the Baugh-Wooley algorithm can implement signed multiplication in almost the same way as the unsigned multiplication.

The Baugh-Wooley algorithmic is used to multiply 2's complement numbers using a regular iterative adder structure. For example, for two n-bit numbers and y their product can be defined as:

$$P=2^{2n-2}X_{n-1}Y_{n-1}+\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}2^{i+j}X_iY_j$$
$$+2^{n-1}\left(\sum_{i=0}^{n-2}2^iY_{n-1}X_i+\sum_{j=0}^{n-2}2^jX_{n-1}Y_j\right)$$
$$+2^n+2^{2n-1}$$

Where x and y are in 2's complement format. This algorithm performs the multiplication using only addition of positive bit products. This simplifies the hardware needed to implement the algorithm.

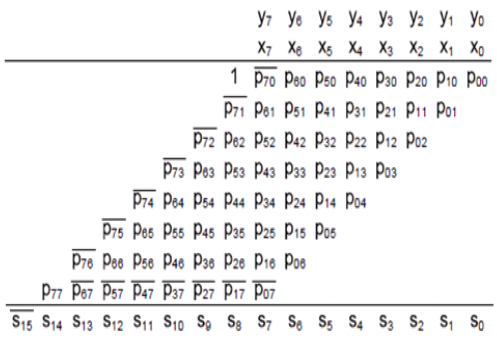

Figure 2.5: Illustration of an 8-bit Baugh-Wooley multiplication

The Baugh-Wooley (BW) algorithm is a relatively straightforward way of doing signed multiplications Fig. 2.5 illustrates the algorithm for an 8-bit case, where the partial-

product bits have been reorganized according to Hatamian's scheme. The creation of the reorganized partial-product array comprises three steps:

i) The most significant bit (MSB) of the first N-1 partial-product rows and all bits of the last partial-product row, except its MSB, are inverted.

ii) A '1' is added to the N^{th} column.

iii) The MSB of the final result is inverted.

Implementing the BW multiplier based on the HPM tree is as straightforward as the basic algorithm itself. The partial-product bits can be generated by using a 2-input AND gate for each pair of operand bits. In the case a partial-product bit should be inverted, we employ a 2-input NAND gate instead. The insertion of '1' in column N is easily accommodated by changing the half adder at top of row N to a full adder with one of the input signals connected to '1'1. Finally, the inversion of the MSB of the result is done by adding an inverter. The final result of the implementation of the BW algorithm is depicted in Fig. 2.6.

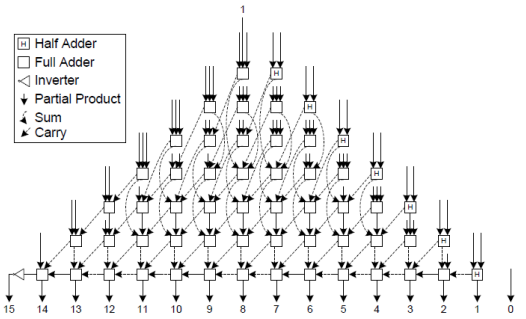

Figure 2.6: Illustration of an 8-bit Baugh-Wooley multiplication using an HPM reduction tree

CHAPTER 3
EXISTING ARCHITECTURE OF MAC

3.1 THREE-CYCLE MULTIPLY ACCUMULATE ARCHITECTURE

The Three-cycle Multiply Accumulate architecture consists of three stages in which the partial product generation is done in the first stage, the partial product addition with carry propagation adder in the second stage and accumulation in the final stage as shown in the Fig 3.1.Multipliers are typically comprised of a partial-product unit (the PP unit) and the final adder. In this unit carry propagation adder is used as the final adder. To increase the to increase MAC performance, we can reduce the critical path delay by inserting an extra pipeline register, either inside the PP unit or between the PP unit and the final adder. This creates three-cycle MAC architecture but increases overhead in terms of delay, power and gate count.

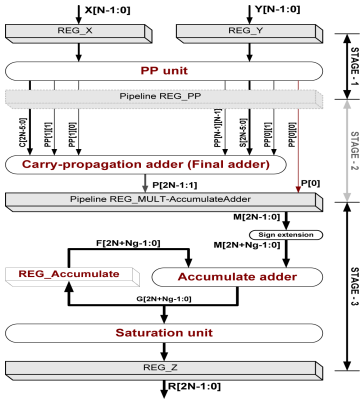

Figure 3.1: Block diagram of the Three-cycle MAC architecture.

3.2 STAGES OF THREE-CYCLE MAC UNIT:

The pipeline register inserted between the PP unit and the final adder forms the first stage as shown in Fig 3.2.Due to the insertion of the pipeline register after the PP unit, the partial products are computed and fed to the next stage through pipeline register. The second stage performs the partial product addition with the carry propagation adder(CPA).The adder adds two n-bit operands and an optional carry-in by performing carry propagation. It performs carry propagation from each bit to higher bit positions and does not occupy a significant area of the chip and less power consumption. The third stage is the accumulation for which each clock cycle the accumulated result is added with the previous result and stored in the register.

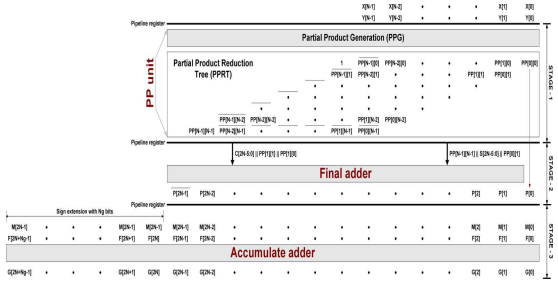

Figure 3.2: Block diagram of the three stages of the Three-cycle MAC architecture.

A multiply -accumulate operation using inputs X and Y, is shown in Fig. 3.2. The multiply-accumulate operation starts with the generation and reduction of partial products. The final adder performs carry propagation of the sums and carries produced by the PP unit. Finally, the accumulate adder sums the pipelined products (M) to the accumulated result (F), producing the new result (G). First we compute the product of the two inputs. Then this result is sign extended to have the same size as the accumulate adder. The accumulate adder is bits wider than the multiplier to allow (2^{Ng}) multiple multiply-accumulate iterations without overflow. Finally, the sign extended product is added to the

stored accumulated value. The disadvantage is that P [2N-1] must be computed and used for sign extension in the accumulating addition. A saturation unit removes the guard bits (Ng) such that the final result is 2N bits wide. The saturation unit takes G [2N+Ng-1:0] as input, where G is the output of the accumulate adder. The three-cycle MAC architecture is used as reference architecture and is compared with the proposed MAC architecture. This unit has increase in power, delay and gate count due to the three stages.

3.3 TWO-CYCLE MULTIPLY ACCUMULATE ARCHITECTURE

The Two-cycle MAC architecture is shown in Fig 3.3. This architecture consists of two stages in which the partial product generation is done in the first stage and the partial product summation and accumulation is done in the second stage. The pipeline register the register between the PP unit and the final adder is removed to obtain a Two-cycle MAC architecture. Our architecture is based on two's complement representation, it uses guarding bits to efficiently support longer MAC loops, and it includes output saturation.

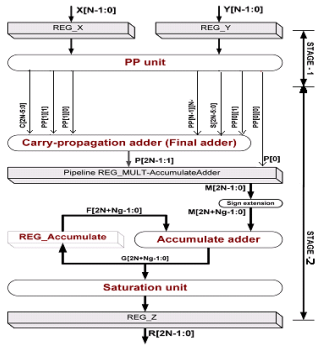

Figure 3.3: Block diagram of the Two-cycle MAC architecture.

In Two-cycle MAC architectures have a first stage that is significantly slower than the second stage. By performing carry propagation only in the second stage of the MAC

pipeline, multiplication and accumulation have similar delays**.** The partial products are generated in the first stage and stored in the pipeline register. In the second stage partial product addition is performed by the carry propagation adder and provides the result in sum and carry. This result is accumulated with the previous result for each consecutive clock cycle in the second stage.

Due to the removal of the pipeline register between the PP unit and the final adder the partial products computed are not fed to the second stage within the stipulated time. The critical path of this unit goes through the PP unit and the final adder. The evaluation results shows that this architecture has better power and gate count when compared with reference architecture. The delay of this unit remains high with the 3-Cycle MAC unit due to the removal of the pipeline register after the PP unit.

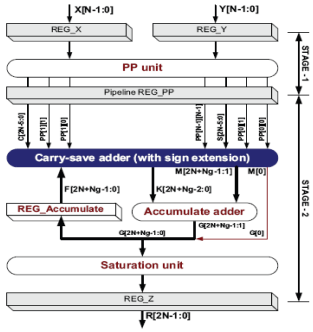

Figure 4.1: Block diagram of the MAC-NEW unit

Carry propagation only takes place in the second stage, which means that the multiplier's final adder is eliminated, leading to higher speed and lower energy. Since accumulation takes place inside the second stage a pipeline register located before the accumulation stage has no impact on functionality. Regardless of pipelining, our MAC unit will produce the correct result in each cycle, and no extra cycles need to be added at the end of the loops– interconnects are localized, which simplifies routing, decreases delay, and reduces energy dissipation.

Because of the above advantages, it supports several guarding bits, making longer loops feasible without any overflow problems. The use of guarding bits in an approach where the accumulated value is fed back to the PPRT's input would most certainly have a negative impact on hardware complexity. The MAC-NEW exploits the fact that the delay of the accumulate adder is shorter than the delay of the PP unit, by at least an amount corresponding to the delay of a full-adder cell.

CHAPTER 4
PROPOSED ARCHITECTURE OF MAC

4.1 PROPOSED MULTIPLY ACCUMULATE ARCHITECTURE

The MAC–NEW architecture is based on two's complement representation, it uses guarding bits to efficiently support longer MAC loops, and it includes output saturation. By performing carry propagation in the second stage of the MAC pipeline, multiplication and accumulation have similar delays. The carry-save adder is used which leads to the reduction of power. With reference to the two cycle MAC architecture, this unit inserts the pipeline register after the partial product unit.

This architecture is based on two conditions such as

- The accumulation should take place in the second stage of a 2-cycle MAC unit.
- The carry should be propagated only once in a MAC pipeline, thus, in the second stage.

The MAC-NEW unit shown in Fig 4.1 consists of two stages: partial product unit in the first stage and the accumulate adder in the second stage. The final adder has been removed, and a carry-save adder has been inserted after the pipeline registers. The maximum delay of the carry-save adder is only that of a single full adder, which means that the MAC's critical path delay still depends on the PP unit. In the carry-save adder there is no need to sign extend the multiplier output instead use a row of '1' to perform the sign extension.

This MAC unit do not require any extra cycles at the end of the loops as the interconnects are localized which simplifies routing, decreases delay and reduces energy dissipation. As the carry propagation and the accumulation takes place in the second stage this architecture uses several guard bits without any overflow problems. The critical path delay of this unit is within the partial product unit.

The critical path is through the PP unit as this architecture uses pipeline registers at the bottom of the PP unit, MAC-NEW obviously can operate at the same speed as MAC-3C, while its performance on average for various operand size such as 16, 32, 48 and 64 is faster than MAC-2C. As far as power dissipation is concerned, the final adder is replaced by the simple carry-save adder,MAC-3C on average dissipates more power than MAC-NEW for the same operating frequency and timing constraint. It requires two cycles for completing the MAC computation, still performs the MAC operation at the same operating frequency as a 3-cycle MAC unit, at lower energy dissipation.

The Evaluation methodology shows that the MAC-NEW unit is efficient in performance parameters such as power, delay and gate count in comparison with the conventional architecture. Due to the efficiency, this architecture is used to create an application architecture called Double Throughput Multiply Accumulate unit [DTMAC].

CHAPTER 5

APPLICATION OF PROPOSED ARCHITECTURE OF MAC

5.1 DOUBLE THROUGHPUT MULTIPLY ACCUMULATE UNIT

A MAC unit that can optionally switch between *N*-bit operation and 2x*N*/2-bit operation is referred as a Double Throughput MAC (DTMAC) is shown in Fig 5.1. This feature would be useful in many DSP-oriented applications, when the dynamic range is lower or when there is a need to simultaneously calculate real and imaginary values. A double throughput 32-bit MAC can be logically implemented by tying together two separate, single 16-bit MACs that support two parallel MAC operations.

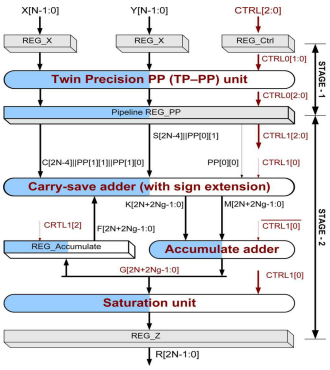

Figure 5.1: Block diagram of the DTMAC unit

Our DTMAC unit in Fig 5.2 is designed to support the efficient execution of several operating modes in a 32-bit data path. The unit employs the Twin-Precision (TP) technique, in terms of a modified 32-bit TP multiplier1 that contains a Twin-Precision Partial-Product Reduction Tree (TP-PPRT) to generate the partial product outputs, which

in conventional schemes are fed to a final adder2. Instead we insert a level of adder cells that combine the outputs of the TPPPRT with the result of the twin-precision accumulate adder; is called"combination unit". In the guarding bit positions of the combination unit, the half adder cells add'1's with the accumulated result, to obtain the correct logical function. The combination unit can be placed after or before the pipeline registers depending on whether the TP-PPRT or the twin-precision accumulate adder represents the dominant delay of the DTMAC unit.

The use of the combination unit makes it possible to build a high-speed, but still flexible DTMAC unit using only two pipeline stages, which limits the clock load and makes for a power-efficient design. The twin-precision accumulate adder is based on the Ladner-Fisher parallel-prefix structure and contains 80 bits, divided in two sections (high and low) each containing 32 data and eight (8) extra guarding bits, as shown in the detailed schematic of Fig. 2(c). Because each of the two sections has eight guarding bits, this DTMAC unit supports loops with 256 iterations without requiring any right shifting of the output to avoid overflow. To control the operating mode, an AND gate is inserted; one control bit (CTRL2[0]) sets the XOR's input at position 40 to either zero or to the carry signal of the 32-bit data part of the low section of the twin-precision accumulate adder.

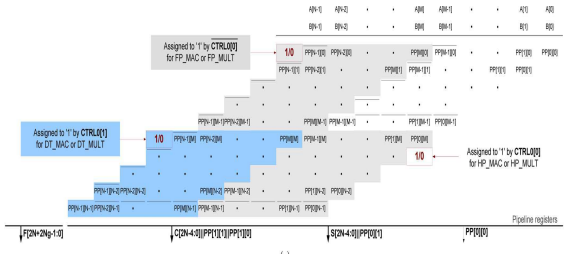

Figure 5.2: Block diagram of the TP-PP unit based on the Baugh–Wooley multiplication algorithm.

5.2 Components of DTMAC unit:

1) TP-PP Unit*:* To support double-throughput operations, the partial-product generation and reduction are based on the twin-precision (TP) technique [24]. Here, the partial products that are not needed during narrow-width operations are forced to zero while some lower-significance partial products are negated4 to provide the correct function for theM -bit multiplication in the lower-significance section. Depending on the operating mode, "1" bits can be set in position N+M, N and M.M=N/2 is assumed as the lower-significance section the "low half."

2) Carry-Save Adder: The carry-save adder (CSA) shown in Fig 5.3 is used for the Partial product addition for the DTMAC unit .In this carry save adder, guard bits and sign extension for the N/2-bit operation in the low half must be accommodated .This is achieved by inserting a row of Ng+1 bits "1" that is summed together with the accumulated value and the most significant bit of the result from the TP-PP unit for the N/2-bit operation in the low half bit position. During N/2-bit operations in the low half, S[N-3] will always be zero, due to the TP technique in which partial products are forced to zero. Since S [N-3] will not carry any useful information during N/2-bit operations in the low half, this signal can be used to add the required "1" at bit positionN-1 . This is easily done by feeding S [N-3] and a control signal through an extra OR gate, whose output may optionally be forced to "1,"

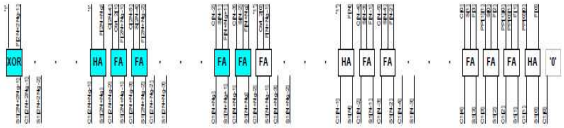

Figure 5.3: Block diagram of the gates of the combination unit in the DTMAC unit.

3) Accumulate Adder: The accumulate adder shown in Fig 5.4 of the DTMAC unit is based on the conditional-sum adder structure, enabling efficient separation into high and low halves, each with Ng guard bits to avoid overflow. To control the operating mode, an AND gate is inserted; one control bit (CTRL1[0]) sets the AND's input at position N+Ng

either to zero or to the carry signal of the N-1 -bit data part of the low half of the accumulate adder. For full precision operations, this effectively by passes the Ng guard bits used for N/2 -bit operations in the low half. Similarly, the accumulator output bits that correspond to unused guard bits (F [N+Ng-1: N]) are discarded during N-bit operation.

4) Saturation Circuit: The saturation unit for the DTMAC not only needs to consider full precision (N) operations but also the N/2 operations in the high and low halves.

- In full-precision mode, 2N+Ng bits in the output of the accumulate adder are processed.
- In half-precision mode, bits N+Ng are processed.
- In double-throughput mode, not only N+Ng bits of the low half are processed, but also N+Ng bits of the high half are processed.

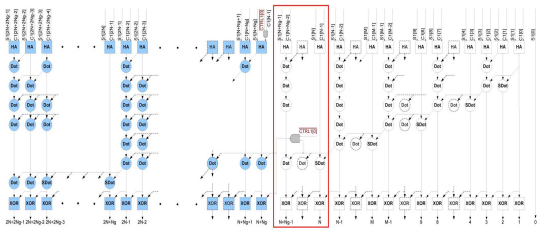

Figure 5.4: Block diagram of the accumulate adder based on the conditional-sum adder architecture

5.3 DTMAC OPERATING MODES

The DTMAC unit operates on two's complement data and supports six operating modes—three for MAC operations and three for multiplications—as determined by the value of the 3-bit control signal (CTRL):

- 000: Full-Precision 32-bit MAC (FP DTMAC).
- 001: Half-Precision 1x16-bit MAC (HP DTMAC).
- 010: Double-Throughput 2x16-bit MAC (DT DTMAC).

- 100: Half-Precision 1x16-bit multiplication (HP MULT).
- 101: Double-Throughput 2x16-bit multiplication (DT MULT).
- 110: Full-Precision 32-bit multiplication (FP MULT).

In the proposed DTMAC unit, there exists no final adder. This makes the critical path delay of the 2-cycle DTMAC dominated by the delay of the TP-PPRT part. The DTMAC actually has the same critical delay as that of a conventional 3-cycle single 32-bit MAC, in which a pipeline register is inserted between the PPRT block and the final adder to several the critical path of the multiplication. The result is that the DTMAC unit, despite the operating mode flexibility, has small area, low power dissipation and short critical path delays. When the DTMAC unit operates in HP DTMAC mode, half of the respective registers are de-activated to isolate the inputs of half of the twin-precision accumulate adder and the MSB input bits of the multiplier are set to zero, to reduce switching activity and dynamic power dissipation.

When the DTMAC unit operates in 1×16-bit MAC mode it dissipates a negligible amount of energy more than the basic, fixed-function, 16-bit MAC unit. The DTMAC unit has a large footprint than MAC32-2C due to extra circuitry to support the multiple operation modes. These comparisons reveal that the implementation of operating-mode flexibility in the DTMAC unit comes at a limited overhead.

The important point is that we can save energy by adjusting the operating mode to the precision of the data:

- When the DTMAC unit operates in the default 32-bitMAC mode (FP_MAC), its energy dissipation is lower than MAC32-2C when performing 32-bit computations.
- When the DTMAC unit operates in 1 16-bit MAC mode (HP_MAC), the 32-bit DTMAC unit performs 16-bit multiply-accumulate operations more energy efficiently than MAC32-2C performs computations on 16-bit operands. This reduction largely stems from avoiding unnecessary switching caused by the 16-bit sign extension of two's complement 32-bit data that carry only 16 bits of information.

- When the DTMAC unit operates in the 2 16-bit MAC mode (DT_MAC), its energy dissipation per 16-bit multiply-accumulate operation is similar to that of MAC16-2C. However, the DTMAC unit uses only half the cycles of MAC16-2C to compute all operations, so the surrounding data path circuits are engaged for a significantly shorter time. This leads to significant energy savings for a system in which the DTMAC unit is integrated.

5.4 MULTIPLICATION THROUGH TWIN PRECISION

The twin-precision technique shown in Fig 5.5 is an efficient way of achieving Double Throughput in a multiplier with low area overhead and delay. The twin- precision technique on signed multipliers based on the regular High Performance Multiplier (HPM) reduction tree. The twin-precision technique can reduce the power dissipation by adapting a multiplier to the bit width of the operands being computed. The technique also enables an increased computational throughput, by allowing several narrow-width operations to be computed in parallel.

Achieving double throughput for a multiplier is not as straightforward as, for example, in an adder, where the carry chain can be cut at the appropriate place to achieve narrow-width additions. It is possible to use several multipliers, where at least two have narrow bit width, and allow them share the same routing, but has several drawbacks: i) The total area of the multipliers would increase, since several multiplier units are used. ii) The use of several multipliers increases the fan out of the signals that drive the inputs of the multipliers. Higher fan out means longer delays and/or higher power dissipation. iii) There would be a need for multiplexers that connect the active multiplier(s) to the result. It is not as easy to deploy the twin-precision technique onto a BW multiplication as it is for the unsigned multiplication, where only parts of the partial products need to be set to zero. To be able to compute two signed multiplications, it is necessary to make a more sophisticated modification of the partial-product array.

For the 4-bit multiplication in the LSP of the array, there is a need for some more modifications. In the active partial-product array of the 4-bit LSP multiplication (shown in

white), the most significant partial product of all rows, except the last, needs to be negated. For the last row it is the opposite, here all partial products, except the most significant, are negated. Also for this multiplication a sign bit '1' is needed, but this time in column. Finally the MSB of the results needs to be negated to get the correct result of the two 4-bit multiplications.

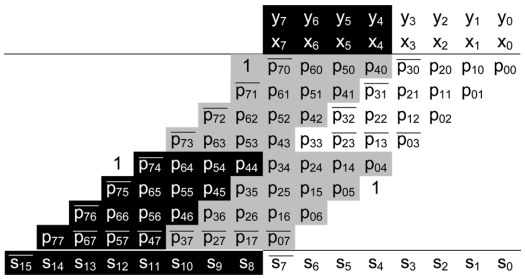

Figure 5.5: Illustration of a unsigned 8-bit multiplication, using the Baugh–Wooley Algorithm

To allow the full-precision multiplication of size to coexist with two multiplications of size in the same multiplier, it is necessary to modify the partial-product generation and the reduction tree. For the -bit multiplication in the MSP of the array all that is needed is to add a control signal that can be set to high, when the N/2-bit multiplication is to be computed and to low, when the full precision multiplication is to be computed. To compute the N/2-bit multiplication in the LSP of the array, certain partial products need to be negated. This can easily be accomplished by changing the two-input AND gate that generates the partial product to a two-input NAND gate followed by an XOR gate. The second input of the XOR gate can then be used to invert the output of the NAND gate. When computing the N/2-bit LSP multiplication, the control input to the XOR gate is set to low making it work as a buffer. When computing a full-precision multiplication the same signal is set to high making the XOR work as an inverter. Finally

the MSB of the result needs to be negated and this can again be achieved by using an XOR gate together with an inverted version of the control signal for the XOR gates used in the partial-product generation. The unwanted partial products to zero can be done by three-input AND gates as for the unsigned multiplication.

5.4.1 HPM IMPLEMENTATION

A twin-precision implementation based on the regular HPM reduction tree is shown in Fig.5.6. For high speed and/or low-power implementations, a reduction tree with logarithmic logic depth, such as TDM [9], Dadda [10], Wallace [11] or HPM [12] is preferred for summation of the partial products. Such a log-depth reduction tree has the benefit of shorter logic depth. Further, a log-depth tree suffers from fewer glitches making it less power dissipating. In fig 5.3, the unsigned multiplication is implemented in Baugh-Wooley algorithm in which 4-bit multiplication, shown in white, can be computed in parallel with a second 4-bit multiplication, shown in black. For simplicity the AND gates for partial-product generation is not shown and a ripple carry is used as final adder.

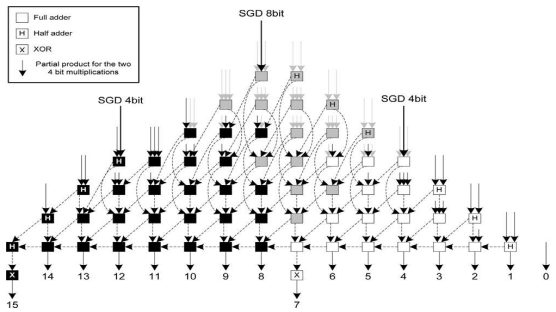

Figure 5.6: Block diagram of an unsigned 8-bit twin-precision multiplier based on the regular HPM reduction tree

5.5 FLOATING POINT MULTIPLIER IN MULTIPLY ACCUMULATE UNIT

Floating Point numbers represented in IEEE 754 format are used in most of the DSP Processors. Floating point arithmetic is useful in applications where a large dynamic range is required or in rapid prototyping applications where the required number range has not been thoroughly investigated. The Floating Point Multiplier IP helps designers to perform floating point Multiplication on FPGA represented in IEEE 754 single precision floating point format.

5.5.1 FUNCTIONAL DESCRIPTION

A Floating point multiplier is the most common element in most digital applications such as digital filters, digital signal processors, data processors and control units. The present Floating Point Multiplier IP has three blocks sign calculator, exponent calculator, mantissa calculator, which works parallel and a normalization unit. The Multiplier is pipelined, so the first result appears after the latency period and then the result can be obtained after every clock cycle.

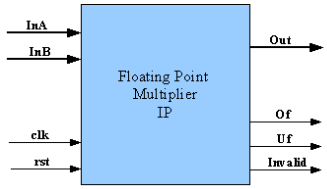

Figure 5.7: Block diagram of the Floating Point Multiplier

The Schematic symbol of Floating Point Multiplier is shown in Fig 5.7. It takes two IEEE 754 format single precision floating point numbers and produces the multiplied output. It also supports the features like underflow, overflow and invalid operations. This

unit consists of two stages, multiplication calculation and normalization. The first stage consists of the following three blocks which work in parallel.

- *Sign Calculator*: The Output Sign is the exor of two sign bit inputs.
- *Exponent Calculator*: The input exponents are added and the bias is removed to produce the exponent of Output.
- *Mantissa Calculator*: Output Mantissa is calculated by multiplying the mantissa's of two inputs. Second stage performs Normalization of the Output obtained from the first stage.
- *Normalization Block*: The normalization is the last and most complicated part. This block is implemented in three pipelined stages.

This block first calculates how much amount the mantissa needs to be left shifted. The mantissa is processed in parallel in a number of modules, each looking at four bits of the mantissa. The first module looks at first four bits of the mantissa and outputs the amount to be shifted assuming a one was found on these four bits. The second module operates on the next four bits of the mantissa treating first four bits are zero and outputs the amount to be shifted left.

This process is repeated for the remaining bits of mantissa. Signals are generated if the four bits of the mantissa are zero. Depending on the signal values the amount of shift is selected. This selection is implemented in three multiplexer stages. Depending on the two leading bits of final mantissa, the final mantissa is shifted left by previously calculated shift amount or shifted right. The final exponent is also corrected accordingly.

CHAPTER 6
SIMULATION RESULTS AND DISCUSSION

All PP units of the MAC architectures are based on the power-efficient Baugh–Wooley algorithm for partial-product generation and the HPM partial-product reduction tree. The accumulate adder is of conditional-sum type and has an extension of eight guard bits (Ng=8). This allows the MAC unit to support loops of up to 256 iterations without requiring the output to be right-shifted to avoid overflow. A final adder based on parallel algorithm of recurrence equation supports fast addition of the PP unit outputs. The Multiply Accumulate architecture is designed using VHDL and simulated using MODEL SIM. The performance parameters are synthesized using Xilinx.

6.1 SIMULATION WAVEFORM OF THREE-CYCLE MAC UNIT

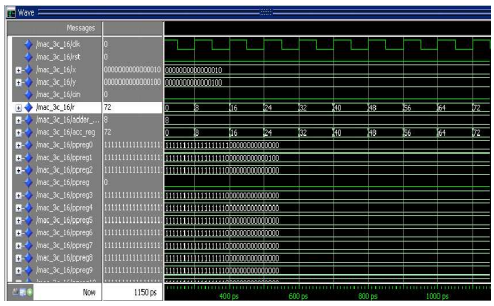

Figure 6.1: Waveform for the three-cycle MAC of operand size 16 -bit

The inputs of MAC-3C unit x and y are of 16 bits. The multiplier output is 16-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32-bit (acc_reg).

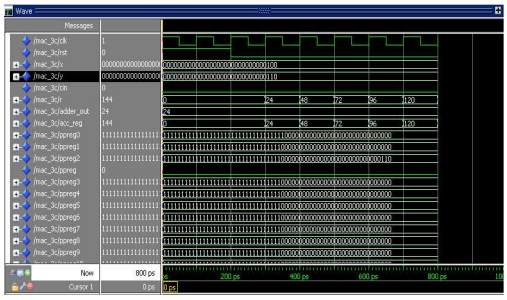

Figure 6.2: Waveform for the three-cycle MAC of operand size 32-bit

The inputs of MAC-3C unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg) .

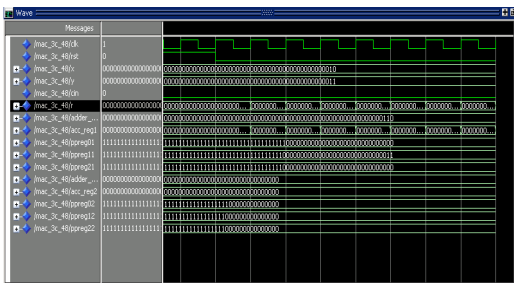

Figure 6.3: Waveform for the three-cycle MAC of operand size 48-bit

The inputs of MAC-3C unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg).

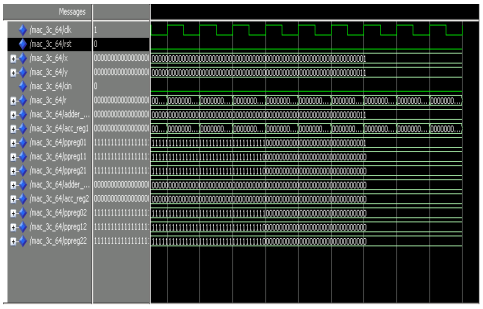

Figure 6.4: Waveform for the three-cycle MAC of operand size 64-bit

The inputs of MAC-3C unit x and y are of 64 bits.The multiplier output is 64 -bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128 -bit (acc_reg).

6.2 SIMULATION WAVEFORM OF TWO-CYCLE MAC UNIT

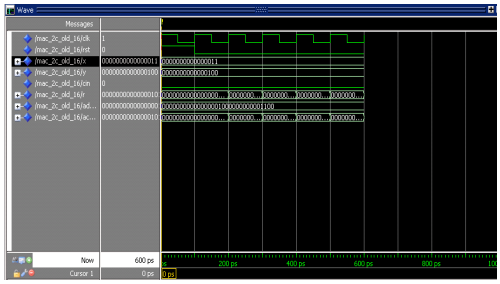

Figure 6.5: Waveform for the two-cycle MAC of operand size 16-bit

The inputs of MAC-2C unit x and y are of 16 bits. The multiplier output is 16 -bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32 -bit (acc_reg).

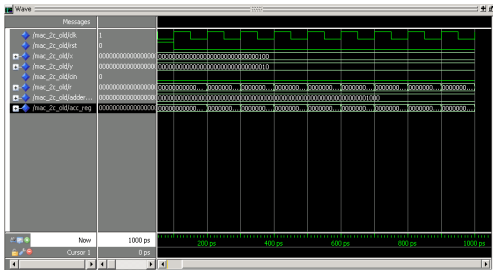

Figure 6.6: Waveform for the two-cycle MAC of operand size 32-bit

The inputs of MAC-2C unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg).

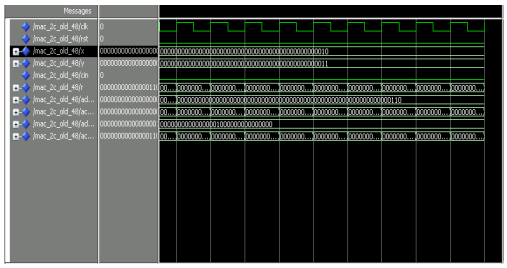

Figure 6.7: Waveform for the two-cycle MAC of operand size 48-bit

The inputs of MAC-2C unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg).

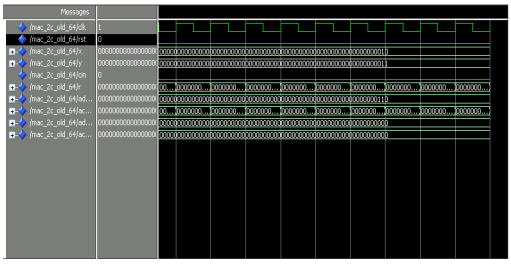

Figure 6.8: Waveform for the two-cycle MAC of operand size 64-bit

The inputs of MAC-2C unit x and y are of 64 bits. The multiplier output is 64-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128-bit (acc_reg).

6.3 SIMULATION WAVEFORM OF MAC-NEW UNIT

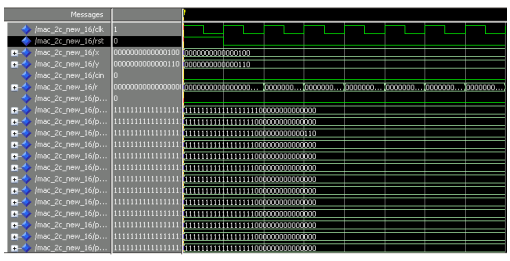

Figure 6.9: Waveform for the MAC-NEW of operand size 16-bit

The inputs of MAC-NEW unit x and y are of 16 bits. The multiplier output is 16-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32-bit (acc_reg).

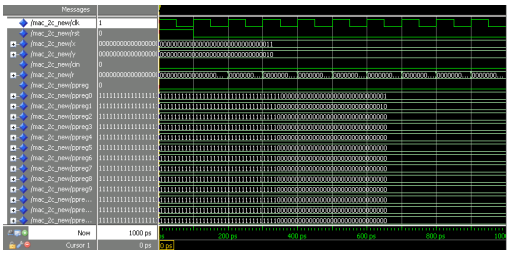

Figure 6.10: Waveform for the MAC-NEW of operand size 32-bit

The inputs of MAC-NEW unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg) .

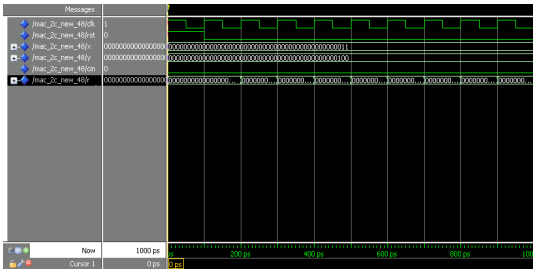

Figure 6.11: Waveform for the MAC-NEW of operand size 48-bit

The inputs of MAC-NEW unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg) .

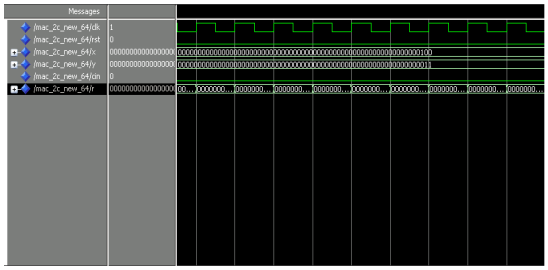

Figure 6.12: Waveform for the MAC-NEW of operand size 64-bit

The inputs of MAC-NEW unit x and y are of 64 bits. The multiplier output is 64-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128-bit (acc_reg).

6.4 SIMULATION WAVEFORM OF DTMAC UNIT

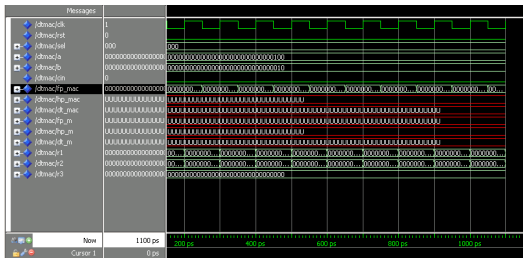

Figure 6.13: Waveform for the Full Precision DTMAC unit

The inputs of the DT_MAC mode is 2×16 bit in which LSB of the a and b-bit is taken as 1×16 bit and MSB of the a and b-bit are taken as 1×16 bit. The selection mode is given 011 and for each consecutive clock cycle the accumulated result is stored in the DT_MAC.

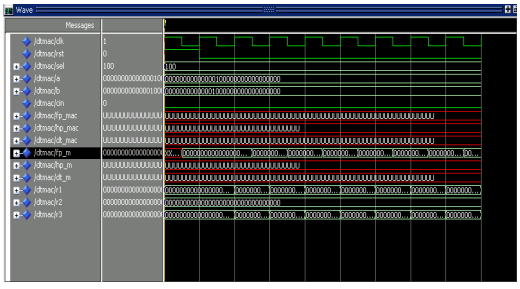

Figure 6.16: Waveform for the Full Precision Multiplication unit

The inputs of the FP_MULT mode is 32- bit in which MSB of the and b-bit is taken as two 1×16 bit. The selection mode is given 100 and for each consecutive clock cycle the multiplication result is stored in the FP_M.

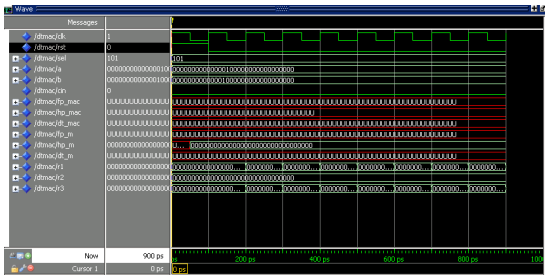

Figure 6.17: Waveform for the Half-Precision Multiplication unit

The inputs of the FP_MAC mode is 32 bit in which LSB of the a-bit and b-bit are taken as two 16-bits.The selection mode is given 000 and for each consecutive clock cycle the accumulated result is stored in the FP_MAC.

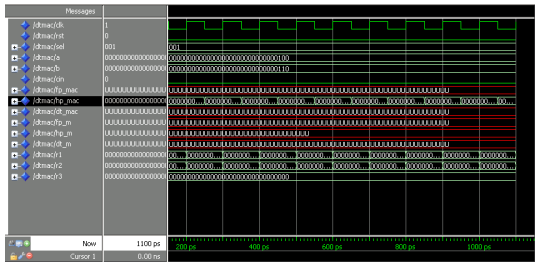

Figure 6.14: Waveform for the Half Precision DTMAC unit

The inputs of the HP_MAC mode is 16 bit in which LSB of the a-bit and b-bit are taken as two 8-bits.The selection mode is given 001 and for each consecutive clock cycle the accumulated result is stored in the HP_MAC.

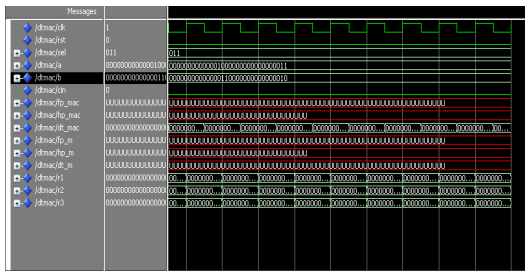

Figure 6.15: Waveform for the DTMAC unit

The inputs of the HP_MULT mode is 1×16 bit in which MSB of the a and b-bit is taken as two 8-bit.The selection mode is given 101 and for each consecutive clock cycle the multiplication result is stored in the HP_M.

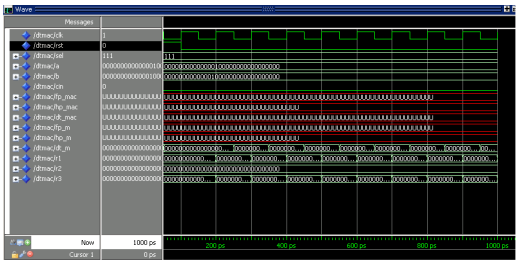

Figure 6.18: Waveform for the Double Throughput Multiplication unit

The inputs of the DT_MULT mode is 2×16 bit in which MSB of the a and b-bit is taken as two 16-bit.The selection mode is given 111 and for each consecutive clock cycle the multiplication result is stored in the DT_M.

6.5 SIMULATION WAVEFORM OF FLOATING POINT MULTIPLIER

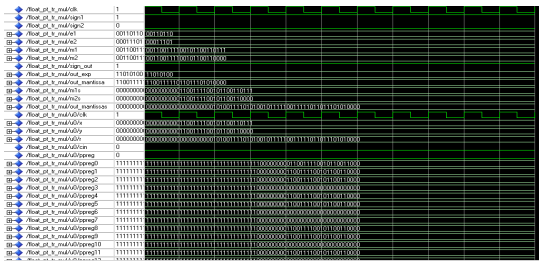

Figure 6.19: Waveform for the Floating Point Multiplier MAC unit

The input of the Floating Point Multiplier is 32- bit in which each of exponents (e1 and e2) is 8 bit. The mantissa bit (m1 and m2) are 23 bit and the sign bit (s1 and s2) is 1-bit.The accumulation is done by the MAC-NEW 32-bit.

6.6 SYNTHESIS REPORT OF THE MAC ARCHITECTURE

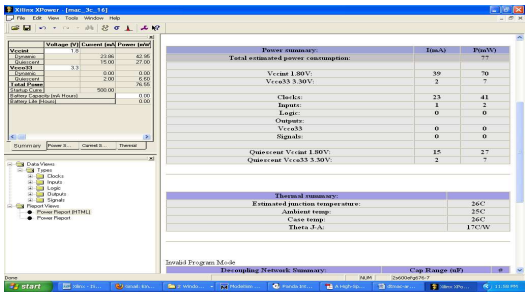

Figure 6.20: Power calculation for 3-C MAC unit of 16-bit

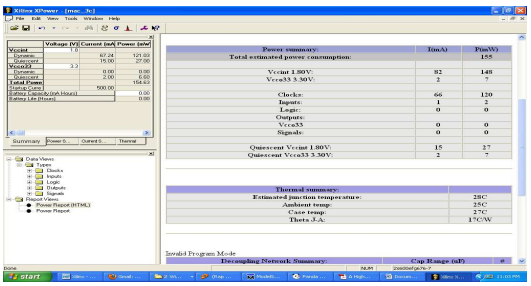

Figure 6.21: Power calculation for 3-C MAC unit of 32-bit

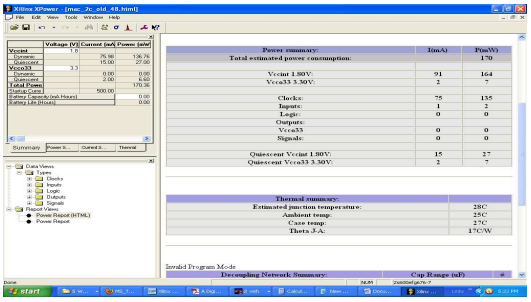

Figure 6.22: Power calculation for 2-C MAC unit of 48-bit

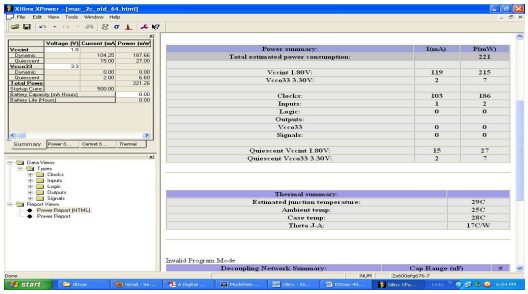

Figure 6.23: Power calculation for 2-C MAC unit of 64-bit

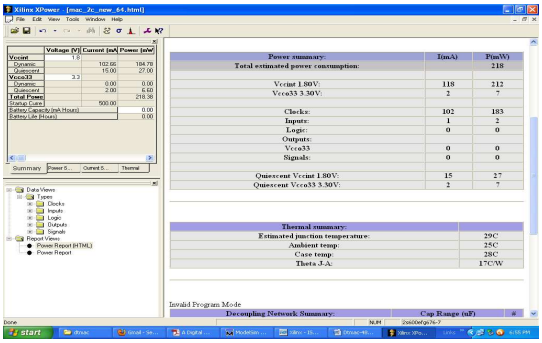

Figure 6.25: Power calculation for MAC-NEW unit of 64-bit

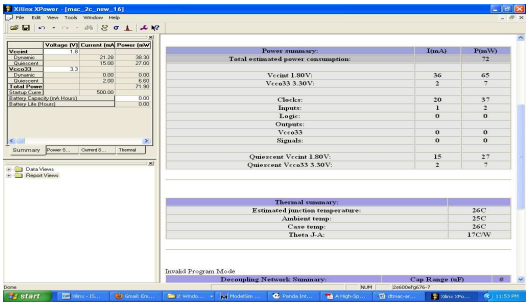

Figure 6.24: Power calculation for MAC-NEW unit of 16-bit

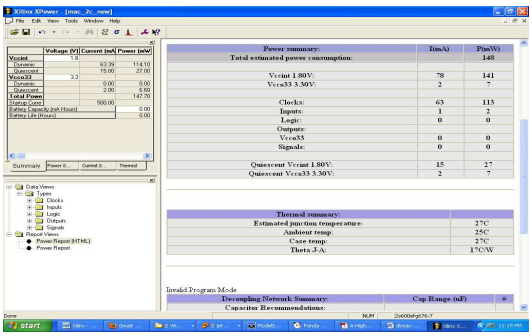

Figure 6.26: Power calculation for Full Precision DTMAC unit

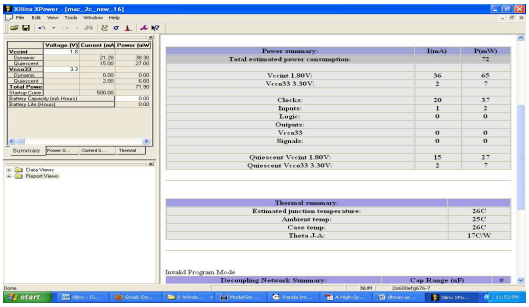

Figure 6.27: Power calculation for Half Precision DTMAC unit

6.7 COMPARISON OF VARIOUS MAC ARCHITECTURES

Table 6.1: Performance Analysis of conventional MAC architectures of the operand size 16 and 32 bit

OPERAND SIZE	16		32		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-2C	MAC-3C	MAC-2C	16-bit	32-bit
POWER(mW)	77	70	155	154	9.09	0.65
DELAY(ns)	79.97	79.24	158.94	162.59	0.92	2.29
GATE COUNT	14,885	13,930	52,439	51,320	6.42	2.13

The performance parameters such as power, delay and gate count are tabulated for the operand size of 16 and 32-bit of the Three-cycle and Two-cycle MAC architecture. The parameters such as power and gate count for the 3-C MAC unit is high in comparison with the 2-C MAC unit but the delay for the 2-C MAC unit remains high. The performance is evaluated for the 16 and 32-bit.

Table 6.2: Performance Analysis of conventional MAC architectures of the operand size 48 and 64 bit

OPERAND SIZE	48		64		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-2C	MAC-3C	MAC-2C	48-bit	64-bit
POWER(mW)	176	170	224	221	3.41	1.34
DELAY(ns)	161.47	163.02	171.75	173.20	0.96	0.84
GATE COUNT	58,974	59.091	88,336	89,132	0.20	0.90

The performance parameters such as power, delay and gate count are tabulated for the operand size of 48 and 64 bit of the Three-cycle and Two-cycle MAC architecture. The parameters such as power and gate count for the 3-C MAC unit is high in comparison with the 2-C MAC unit but the delay for the 2-C MAC unit remains high .The performance is evaluated for the 48 and 64-bit.

Table 6.3: Performance Analysis of 3-C and MAC-NEW architectures of the operand size 16 and 32 bit

OPERAND SIZE	16		32		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-NEW	MAC-3C	MAC-NEW	16-bit	32-bit
POWER(mW)	77	72	155	148	6.50	4.52
DELAY(ns)	79.97	74.21	158.94	153.27	7.20	3.57
GATE COUNT	14,885	13,356	52,439	42,880	10.27	16.44

The performance parameters such as power, delay and gate count are tabulated for the operand size of 16 and 32 bit of the Three-cycle and MAC-NEW architecture. The parameter for the 3-C MAC unit is high in comparison with the MAC-NEW unit. The performance is evaluated for the 16 and 32 bit.

Table 6.4: Performance Analysis of 3-C and MAC-NEW architectures of the operand size 48 and 64 bit

OPERAND SIZE	48		64		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-NEW	MAC-3C	MAC-NEW	48-bit	64-bit
POWER(mW)	176	164	224	218	6.82	2.68
DELAY(ns)	161.47	157.22	171.75	168.96	2.64	1.62
GATE COUNT	58,974	49,180	88,336	77,903	16.61	11.81

The performance parameters such as power, delay and gate count are tabulated for the operand size of 48 and 64 bit of the Three-cycle and MAC-NEW architecture. The parameter for the 3-C MAC unit is high in comparison with the MAC-NEW unit. The performance is evaluated for the 48 and 64 bit.

Table 6.5: Comparison of Operating Modes in DTMAC Architecture

Architecture	FP_MAC	HP_MAC	DT_MAC
Power	148	72	144
Delay	153.27	74.21	148.42
Gate Count	42,880	13,356	26,712

The DTMAC operating modes parameters are tabulated in Table 6.5. The parameters of the FP_MAC are same as 32-bit MAC-NEW architecture. The HP_MAC is same as 16-bit MAC-NEW architecture. The DT_MAC is 2×16-bit MAC-NEW architecture.

Table 6.6 :Parameters of the Floating Point multiplier in MAC unit

Parameters	
Power (mW)	106
Delay(ns)	73.964
Gate Count	3579

6.8 POWER ANALYSIS

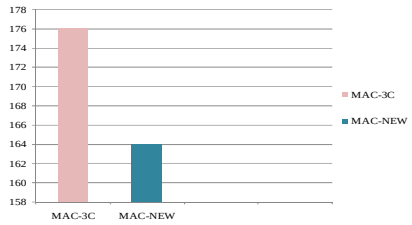

Figure 6.28: Power Analysis of MAC-3C and MAC-NEW of the operand size 32 bit

The Power analysis is performed for the MAC-3C and MAC-NEW architecture. The MAC-3C unit has more power when compared with the MAC-NEW architecture due to the three-pipeline stages.

6.9 DELAY ANALYSIS

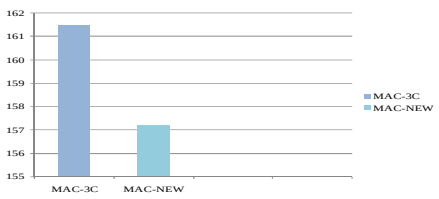

Figure 6.29 :DelayAnalysis of MAC-3C and MAC-NEW of the operand size 32 bit

The Delay analysis is performed for the MAC-3C and MAC-NEW architecture. The MAC-3C unit has more delay when compared with the MAC-NEW architecture due to the three-pipeline stages.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

This project presents the estimation of the efficient performance parameters such as power, gate count, and delay for the different Multiply Accumulate architectures. The architectures are designed using Baugh-Wooley algorithm. The Three-cycle, Two-cycle and MAC-NEW architecture is simulated through MODEL SIM and synthesized using XILINX. The performance parameter of the conventional MAC architecture is compared with the proposed MAC architecture and the results are tabulated.

The comparison is made between the MAC-3C and MAC-2C architecture in which the power and gate count remains high for the MAC-3C but the delay is large for the MAC-2C due to the removal of the pipeline register after the Partial Product (PP) unit. The MAC-NEW is compared with the reference architecture (MAC-3C) and the results are tabulated in which the parameters are efficient for the MAC-NEW architecture. As it is an efficient architecture it is used to create a versatile MAC unit called Double Throughput MAC unit(DTMAC).As a modification to this project, the Floating point multiplier is used in the MAC unit and the parameter are tabulated.

FUTURE SCOPE

The MAC-NEW architecture can be used in the efficient design of digital signal processing circuits such as FIR and IIR filter. As this architecture is efficient in performance parameters it increases the computation of the filter.

BIBLIOGRAPHY

[1] A High-Speed, Energy-Efficient Two-Cycle Multiply-Accumulate (MAC) Architecture and Its Application to a Double-Throughput MAC Unit *IEEE Transaction*, volume 57, NO. 12, Dec 2010.

[2] T. T. Hoang, M. Själander, and P. Larsson-Edefors, "High-speed, energy- efficient 2-cycle multiply-accumulate architecture," in *Proceedings. IEEE International. SOC Conference. (SOC)*, Sep. 2009, pp. 119–122.

[3] M. Själander and P. Larsson-Edefors, "Multiplication acceleration through twin precision," *IEEE Transaction Very Large Scale Integrated. (VLSI).*, volume 17, pp. 1233–1246, Sep. 2009.

[4] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications, "in *Proceedings. IEEE International Symposium Circuits System (ISCAS)*, May 2007, pp. 3199–3202.

[5] T.T. Hoang, M. Själander, and P. Larsson-Edefors, "Double throughput multiply-accumulate unit for Flex Core processor enhancements," presented at the IEEE International Symposium, Parallel Distributed Processing Symposium (IPDPS), Reconfigurable Architecture Workshop (RAW), Rome, Italy, and May 2009.

[6] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications, "in *Proceedings IEEE International Symposium Circuits System (ISCAS)*, May 2007 ,pp. 3199–3202.

[7] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier reduction tree with logarithmic logic depth and regular connectivity," in *Proceedings IEEE International Symposium Circuits System (ISCAS)*, May 2006, pp. 4–8.

[8] M. Själander, H. Eriksson, and P. Larsson-Edefors, "An efficient twin precision multiplier," in *Proceedings IEEE International Conference Computer Design (ICCD)*, Oct. 2004, pp. 30–33.

[9] R. K. Kolagotla, J. Fridman, B. C. Aldrich, M. M. Hoffman, W. C. Anderson, M. S. Allen, D. B. Witt, R. R. Dunton, and L.A. Booth, "High performance dual-MAC DSP architecture," *IEEE Signal Processing Magazine.*, volume 19, no. 4, pp. 42–53, July. 2002.

[10] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proceedings International Symposium High-Performance Computer Architecture*, 1999, pp. 13–22.

[11] P. F. Stelling and V. G. Oklobdzija, "Implementing multiply-accumulate operation in multiplication time," in *Proceedings International Symposium Computer Arithmetic (ARITH)*, July 1997, pp. 99–106.

[12] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Transaction Computer*, volume. C-22, pp.1045–1047, Dec. 1973.

[13] J. Sklansky, "Conditional-sum addition logic," *IRE Transaction Electronic Computer*, volume. EC-9, pp. 226–231, 1960.

COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE

By

P.M. SNEHA ANGELINE

Reg. No. 1020106017

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University, Coimbatore)

COIMBATORE - 641049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

APPLIED ELECTRONICS

APRIL 2012

BONAFIDE CERTIFICATE

Certified that, this project report entitled "**COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE**" is the bonafide work of **Ms.P.M.SNEHA ANGELINE [Reg.No:1020106017]** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

(Ms.M.SHANTHI) (**Dr. RAJESWARI MARIAPPAN**)

Project Guide **Head of the Department**

The candidate with university Register no. 1020106017 is examined by us in the project viva-voce examination held on ……

Internal Examiner External Examiner

ACKNOWLEDGEMENT

I express my profound gratitude to our director **Dr.J.Shanmugham,** for giving this opportunity to pursue this course

At this pleasing moment of having successfully completed the project work, I wish to acknowledge my sincere gratitude and heartfelt thanks to our beloved Principal **Dr.S.Ramachandran,** for having given me the adequate support and opportunity for completing this project work successfully.

I express my sincere thanks to D**r.Rajeswari Mariappan Ph.D.,** the ever active, Head of the Department of Electronics and Communication Engineering, who rendering us all the time by helps throughout this project.

I extend my heartfelt thanks to my internal guide **Mrs.M.Shanthi M.S, Asso. Professor,** for her ideas and suggestion, which have been very helpful for the completion of this project work. Her careful supervision has ensured me in attaining perfection of work.

In particular, I wish to thank and everlasting gratitude to the project coordinator **Mrs.R.Hemlatha M.E., Asst.Professor,** Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

Last, but not the least, I would like to express my gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their encouragement and support throughout the course of this project.

ABSTRACT

The Multiplier and Accumulator (MAC) unit is used as a basic element in most of the digital signal processing application in order to perform repeated multiplication and addition. The conventional MAC architectures uses more shift and add operation at multiplier unit which increases delay in the arithmetic operations.

The main objective is to design a new multiplier and accumulator architecture to perform high speed arithmetic operation. The three cycle MAC (MAC-3C) architecture increase the performance by reducing the critical path delay by inserting an extra pipeline register either inside the partial product (PP) unit or between PP unit and final adder. The two cycle MAC (MAC-2C) architecture performs the carry propagation only in the second stage leads to the similar delay in multiplication and accumulation. The proposed MAC architecture (MAC-NEW) has two stages with the pipeline register inserted after the partial product unit. This unit uses carry-save adder which leads to the reduction of power. Due to the carry propagation in the second stage, multiplier's final adder is eliminated, leading to higher speed and lower energy. The Double Throughput MAC unit (DTMAC) switches between N-bit operations and 2×N/2-bit operations which reduces power and critical path delay on the removal of final adder.

Through the ***"COMPARATIVE ANALYSIS OF DIFFERENT MULTIPLY ACCUMULATE ARCHITECTURE"*** is planned to obtain an efficient performance parameter such as gate count, delay and power for the different MAC architectures. The MAC architecture is designed using MODEL SIM and simulated using Xilinx ISE 9.2i and the parameters is compared to obtain an efficient architecture.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MAC	-------	Multiply- Accumulate Architecture
3-C MAC	-------	Three-Cycle Multiply Accumulate Architecture
2-C MAC	-------	Two-Cycle Multiply Accumulate Architecture
MAC-NEW	-------	Proposed Multiply Accumulate Architecture
DTMAC	-------	Double Throughput Multiply Accumulate Architecture
PP	-------	Partial Product
TP	-------	Twin Precision
TP-PPRT	-------	Twin-Precision Partial-Product
CPA	-------	Carry Propagation Adder
CSA	-------	Carry Save Adder
BW	-------	Baugh-Wooley Algorithm

CHAPTER 1

INTRODUCTION

With the recent rapid advances in multimedia and communication system,real-time signal processings like audio signal processing, video/image processing or large-capacity data processing are intrestingly being demanded.The multiplier and multiplier and accumulator(MAC) are the essentials elements of the digital signal processing such as filtering,convolution and inner products.Most digital signal processing methods use nonlinear functions such as discrete cosine transform(DCT) or discrete wavelet transform Because they are basically accomplished by repetitive application of mulitiplication and addition,the speed of the multiplication and addition arithmetic's determines the execution speed and performance of the entire calculation.As the multiplier requires longest delay among the basic operational blocks in digital system,the critical path is determined by the multiplier.

The multiplier consists of three parts: partial product generation, partial product summation and accumulation. The multiplier is much more complex than the accumulate adder, many design techniques have focused on reducing multiplier delay. In the architecture, the critical path is reduced by inserting an extra pipeline register, either inside the partial product unit or between the partial product unit and final adder. It has a better performance because of the reduction in critical path delay. The most effective way to increase the speed of a multiplier is to reduce the number of partial products using high speed compressors or speed optimized structures because multiplication precedes a series of additions for the partial products. The guard bits are important for avoiding overflow when computing long sequences of multiply accumulate operation.

In order to improve the speed of the MAC unit, there are two major bottlenecks. The first is the partial product reduction network that is used in the multiplication block and the second is the accumulator. Both of these stages require addition of large operands that involve long paths for carry propagation. As the multiplier is more complex than the accumulator, design techniques are proposed on reducing the delay in the multiplier either inside the Partial Product (PP) unit or in the final adder. Inside the PP unit, the partial-product circuitry might be implemented using the modified-Booth algorithm or one of its successors. The partial-product reduction tree of the PP unit can be implemented using high-speed compressors or speed-optimized structures. Mathew *et al.* propose a sparse-tree carry look-ahead adder for fast addition of the PP unit outputs and Liu *et al.* introduce a hybrid adder o reduce delay compared to a design that assumes equal arrival time on all adder inputs.

Here a MAC-NEW architecture is proposed in which the first stage is significantly faster compared to the second stage, leading to a better delay balance between the two stages. The key feature to this architecture is the implementation of product sign extension in the second stage, together with the accumulate adder such as carry save adder and the saturation unit. Guard bits are used for avoiding the overflow on computation of long sequences of multiply-accumulate operation. This MAC-NEW unit is efficient in terms of delay, power and gate count.

1.1 OBJECTIVE OF THE WORK

The performance of the multiply and accumulate unit is improved by either using high speed multipliers or improved fast adder architectures. To obtain a high speed operation, the multiplication unit is combined with accumulation and carry save adder (CSA).The partial product is generated using Baugh Wooley algorithm. The result is sign extended to have the same size as the accumulate adder. The MAC unit is designed using VHDL code and simulated using MODELSIM. The performance parameters such as

power, gate count, and delay are synthesized using XILINX and compared with the conventional MAC architecture.

1.2 INTRODUCTION TO VHDL

VHDL is an acronym which stands for VHSIC Hardware Description Language.VHSIC means Very High Speed Integrated Circuits. It is being used for documentation, verificatoin and synthesis of large digital designs.VHDL is a standard developed by IEEE.The different approaches in VHDL are structural, data flow and behavioral methods of hardware description.

1.2.1 STRUCTURAL DESCRIPTIONS

Building Blocks

Every portion of a VHDL design is considered a block. A VHDL design may be completely described in a single block, or it may be decomposed in several blocks. Each block in VHDL is analogous to an off-the-shelf part and is called an entity. The entity describes the interface to that block and a separate part associated with the entity describes how that block operates. The interface description is like a pin description in a data book, specifying the inputs and outputs to the block. The description of the operation of the part is like a schematic for the block.

The following is an example of an entity declaration in VHDL

```
Entity latch is
  Port (sir: in bit;
        q,nq: out bit);
end latch;
```

The first line indicates a definition of a new entity called latch. The last line is the end of the definition. The lines in between, are called the port clause, which describe the interface to the design. The port clause contains a list of interface declarations. Each interface declaration defines one or more signals that are inputs or outputs to the design. Each interface declaration contains a list of names, mode and type.

The following is an example of an architecture declaration for the latch entity.

```
architecture dataflow of latch is
  signal q0 : bit := '0';
  signal nq0 : bit := '1';
begin
  q0<=r nor nq0;
  nq0<=s nor q0;
  nq<=nq0;
  q<=q0;
end dataflow;
```

The first line of the declaration indicates the definition of a new architecture called dataflow and it belongs to the entity named latch. So this architecture describes the operation of the latch entity. The schematic for the SR latch

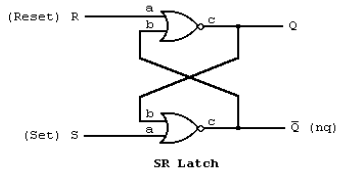

Figure 1.1 Schematic SR Latch

1.2.2 DATA FLOW DESCRIPTIONS

In the data flow approach, circuits are described by indicating how the inputs and outputs of built-in primitive components are connected together.The following SR latch using VHDL is described as in the following schematic.

```
entity latch is
  port (s,r : in bit;
        q,nq : out bit);
```

```
end latch;
architecture dataflow of latch is
begin
  q<=r nor nq;
  nq<=s nor q;
end dataflow;
```

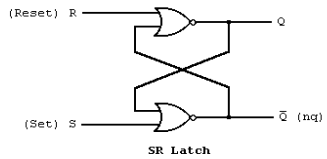

Figure 1.2 Dataflow approach of Schematic SR Latch

The signal assignment operator in VHDL specifies a relationship between signals. The architecture part describes the internal operation of the design. The scheme used to model a VHDL design is called discrete event time simulation. In this the values of signals are only updates when certain events occur and event occurs at discrete instances of time.

The Delay Model

The two models of delay that are used in VHDL. The first is called the inertial delay model. The inertial delay model is specified by adding an after clause to the signal assignment statement. The next is the transport delay model, just delays the change in the output by the time specified.

1.2.3 BEHAVIORAL DESCRIPTIONS

The behavioral approach to modeling hardware components is different from the other two methods in that it does not necessarily in any way reflect how the design is implemented.

The Process Statement

It is basically the black box approach to modeling. It accurately models what happens on the inputs and outputs of the black box, but what is inside the box (how it works) is irrelevant. The behavioral description is usually used in two ways in VHDL. First, it can be used to model complex components.

Behavioral descriptions are supported with the process statement. The process statement can appear in the body of an architecture declaration just as the signal assignment statement does. The process statement can also contain signal assignments in order to specify the outputs of the process.

Using Variables

A variable is kinds of objects used to hold data and also behaves like you would expect in a software programming language, which is much different than the behavior of a signal. Although variables represent data like the signal, they do not have or cause events and are modified differently. Variables are modified with the variable assignment.

Sequential Statements

There are several statements that may only be used in the body of a process. These statements are called sequential statements because they are executed sequentially. The types of statements used here are if, if else, for and loop.

Signals and Processes

This section is short, but contains important information about the use of signals in the process statement. The issue of concern is to avoid confusion about the difference between how a signal assignment and variable assignment behave in the process statement. Remember a signal assignment, if anything, merely schedules an event to occur on a signal and does not have an immediate effect. When a process is resumed, it executes from top to bottom and no events are processed until after the process is complete.

Program Output

In most programming languages there is a mechanism for printing text on the monitor and getting input from the user through the keyboard. Even though the simulator monitors the value of signals and variables in the design, it is able to output certain information during simulation. It is not provided as a language feature in VHDL, but rather as a standard library that comes with every VHDL language system. In VHDL, common code can be put in a separate file to be used by many designs. This common code is called a library. The write statement can also be used to append constant values and the value of variables and signals of the types bit, bit_vector, time, integer, and real.

1.3 SOFTWARE USED

- Modelsim PE5.4E
- Xilinx ISE 9.2i

1.4 ORGANIZATION OF THE REPORT

- **Chapter 2** discusses about the overview of MAC.
- **Chapter 3** discusses the existing architecture of MAC.
- **Chapter 4** discusses the proposed architecture of MAC.
- **Chapter 5** discusses the application of proposed architecture of MAC.
- **Chapter 6** presents the simulation results and discussions.
- **Chapter 7** presents the conclusion and future scope.

CHAPTER 2
OVERVIEW OF MAC

2.1 GENERAL ARCHITECTURE OF MAC

The general construction of the MAC operation is given by the equation

$$Z=A\times B+X$$

Where the multiplier A and multiplicand B are assumed to have n bits each and the addend X has (2n+1) bits. The basic MAC unit is made up of a multiplier and an accumulator as shown in Fig 2.1. The multiplier can also be divided into partial product generator, summation tree and final adder. It executes the multiplication operation by multiplying the input multiplier and multiplicand. This is added to the previous multiplication result as the accumulation step.

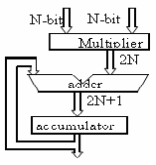

Figure 2.1: General MAC architecture

The summation network represents the core of the MAC unit and occupies most of the area, power and delay. Several algorithms and architectures are developed to optimize the implementation of this block. The addition network reduces the number of partial products into two operands representing a sum and a carry. The final adder is then used to generate the multiplication result out of these two operands. The last block is the accumulator, which is required to perform a double precision addition operation between the multiplication result and the accumulated operand. It involves a very large adder due

to the large operand size. This stage represents a bottleneck in the multiplication process in terms of speed since it involves horizontal carry propagation. The MAC unit is classified into various types such as 2-Cycle MAC unit,3-Cycle MAC unit, MAC-NEW unit and DTMAC unit.

2.2 BLOCK DIAGRAM OF PROJECT

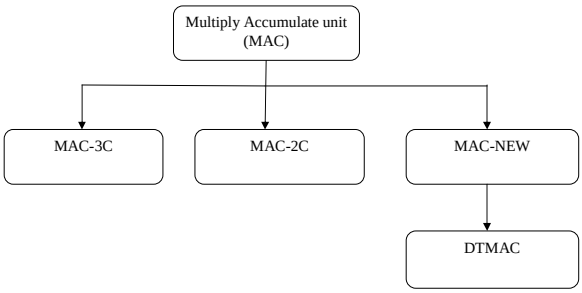

Figure 2.2: Block Diagram of the project

The overall block diagram of the project is shown in Fig2.2.The multiply accumulate unit is broadly classified into three types such as Three-cycle MAC unit(MAC-3C),Two-cycle MAC unit(MAC-2C) and MAC-NEW unit. The three architectures are implemented using BAUGH-WOOLEY algorithm. The proposed MAC unit has the better performance in comparison with the conventional architectures. The MAC-NEW is used to create a versatile MAC unit is called DOUBLE THROUGHPUT MULTIPLIER AND ACCUMULATE UNIT (DTMAC).

2.3 PROCESS FLOW IN MAC

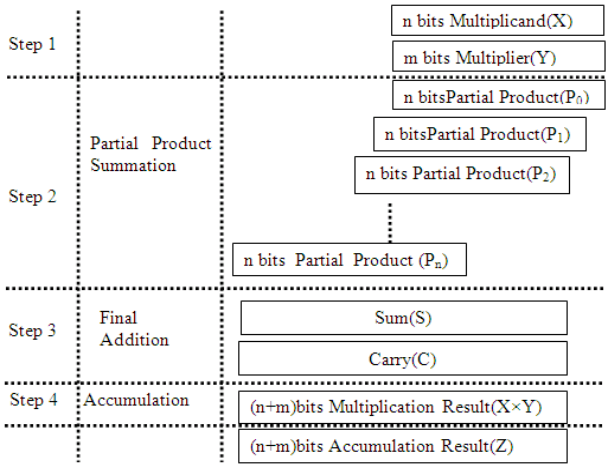

Figure 2.3: Basic Arithmetic steps of multiplication and accumulation

A multiplier can be divided into four operational steps as shown in Fig 2.3. The first step is the multiplication operation with the input multiplier and the multiplicand. The second step is the partial product summation which is used to add all the partial products and convert them into the form of sum and carry. The third step is the final addition in which the final multiplication result is produced by adding the sum and carry. The last step is the accumulation which takes place with the multiplication and the accumulated result.

2.4 BAUGH-WOOLEY ALGORITHM

An algorithm for direct 2's complement array multiplication has been proposed by BAUGH-WOOLEY and this algorithm is used in the design of multiplier and accumulator structures. The primary advantage of this algorithm is that the signs of all the partial products are positive and thus allowing the array to be entirely the same as conventional standard array structures.

The following

- Algorithm for two's-complement multiplication.
- Adjust partial products to maximize regularity of array multiplication.
- Moves partial products with negative signs to the last step also add negation of partial products rather than subtracts.

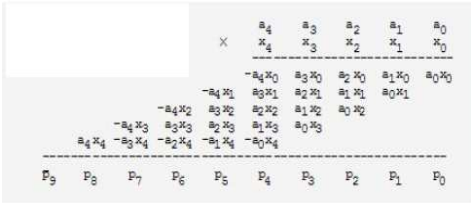

Figure 2.4: Unsigned multiplication for Baugh-Wooley algorithm

The Baugh-Wooley algorithm for the unsigned binary multiplication is based on the concept shown in Fig2.4.The algorithm specifies that all possible AND terms are created first and then sent through an array of half-adders and full-adders with the carry-outs chained to the next most significant bit at each level of addition.

For signed multiplication the Baugh-Wooley algorithm can implement signed multiplication in almost the same way as the unsigned multiplication.

The Baugh-Wooley algorithmic is used to multiply 2's complement numbers using a regular iterative adder structure. For example, for two n-bit numbers and y their product can be defined as:

$$P=2^{2n-2}X_{n-1}Y_{n-1}+\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}2^{i+j}X_iY_j$$
$$+2^{n-1}\left(\sum_{i=0}^{n-2}2^iY_{n-1}X_i+\sum_{j=0}^{n-2}2^jX_{n-1}Y_j\right)$$
$$+2^n+2^{2n-1}$$

Where x and y are in 2's complement format. This algorithm performs the multiplication using only addition of positive bit products. This simplifies the hardware needed to implement the algorithm.

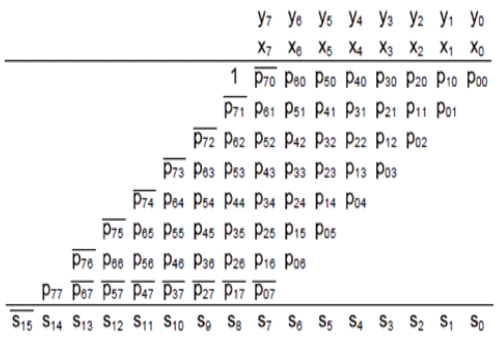

Figure 2.5: Illustration of an 8-bit Baugh-Wooley multiplication

The Baugh-Wooley (BW) algorithm is a relatively straightforward way of doing signed multiplications Fig. 2.5 illustrates the algorithm for an 8-bit case, where the partial-

product bits have been reorganized according to Hatamian's scheme. The creation of the reorganized partial-product array comprises three steps:

i) The most significant bit (MSB) of the first N-1 partial-product rows and all bits of the last partial-product row, except its MSB, are inverted.

ii) A '1' is added to the N^{th} column.

iii) The MSB of the final result is inverted.

Implementing the BW multiplier based on the HPM tree is as straightforward as the basic algorithm itself. The partial-product bits can be generated by using a 2-input AND gate for each pair of operand bits. In the case a partial-product bit should be inverted, we employ a 2-input NAND gate instead. The insertion of '1' in column N is easily accommodated by changing the half adder at top of row N to a full adder with one of the input signals connected to '1'1. Finally, the inversion of the MSB of the result is done by adding an inverter. The final result of the implementation of the BW algorithm is depicted in Fig. 2.6.

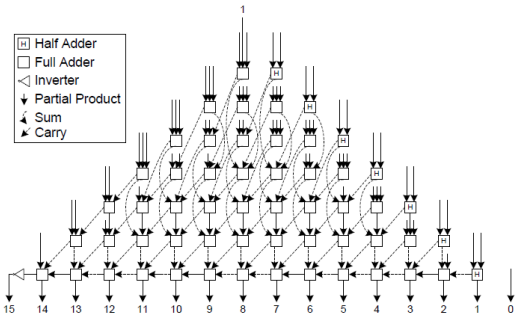

Figure 2.6: Illustration of an 8-bit Baugh-Wooley multiplication using an HPM reduction tree

CHAPTER 3
EXISTING ARCHITECTURE OF MAC

3.1 THREE-CYCLE MULTIPLY ACCUMULATE ARCHITECTURE

The Three-cycle Multiply Accumulate architecture consists of three stages in which the partial product generation is done in the first stage, the partial product addition with carry propagation adder in the second stage and accumulation in the final stage as shown in the Fig 3.1.Multipliers are typically comprised of a partial-product unit (the PP unit) and the final adder. In this unit carry propagation adder is used as the final adder. To increase the to increase MAC performance, we can reduce the critical path delay by inserting an extra pipeline register, either inside the PP unit or between the PP unit and the final adder. This creates three-cycle MAC architecture but increases overhead in terms of delay, power and gate count.

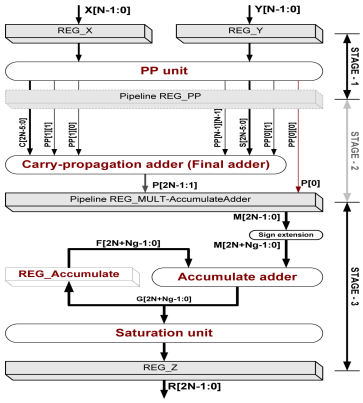

Figure 3.1: Block diagram of the Three-cycle MAC architecture.

3.2 STAGES OF THREE-CYCLE MAC UNIT:

The pipeline register inserted between the PP unit and the final adder forms the first stage as shown in Fig 3.2.Due to the insertion of the pipeline register after the PP unit, the partial products are computed and fed to the next stage through pipeline register. The second stage performs the partial product addition with the carry propagation adder(CPA).The adder adds two n-bit operands and an optional carry-in by performing carry propagation. It performs carry propagation from each bit to higher bit positions and does not occupy a significant area of the chip and less power consumption. The third stage is the accumulation for which each clock cycle the accumulated result is added with the previous result and stored in the register.

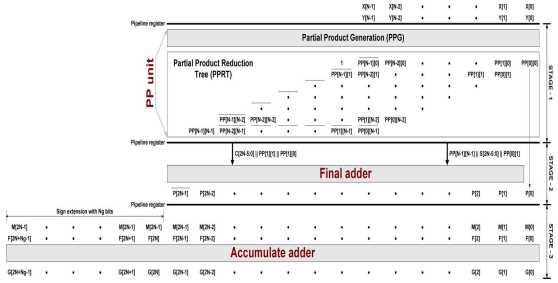

Figure 3.2: Block diagram of the three stages of the Three-cycle MAC architecture.

A multiply -accumulate operation using inputs X and Y, is shown in Fig. 3.2. The multiply-accumulate operation starts with the generation and reduction of partial products. The final adder performs carry propagation of the sums and carries produced by the PP unit. Finally, the accumulate adder sums the pipelined products (M) to the accumulated result (F), producing the new result (G). First we compute the product of the two inputs. Then this result is sign extended to have the same size as the accumulate adder. The accumulate adder is bits wider than the multiplier to allow (2^{Ng}) multiple multiply-accumulate iterations without overflow. Finally, the sign extended product is added to the

stored accumulated value. The disadvantage is that P [2N-1] must be computed and used for sign extension in the accumulating addition. A saturation unit removes the guard bits (Ng) such that the final result is 2N bits wide. The saturation unit takes G [2N+Ng-1:0] as input, where G is the output of the accumulate adder. The three-cycle MAC architecture is used as reference architecture and is compared with the proposed MAC architecture. This unit has increase in power, delay and gate count due to the three stages.

3.3 TWO-CYCLE MULTIPLY ACCUMULATE ARCHITECTURE

The Two-cycle MAC architecture is shown in Fig 3.3. This architecture consists of two stages in which the partial product generation is done in the first stage and the partial product summation and accumulation is done in the second stage. The pipeline register the register between the PP unit and the final adder is removed to obtain a Two-cycle MAC architecture. Our architecture is based on two's complement representation, it uses guarding bits to efficiently support longer MAC loops, and it includes output saturation.

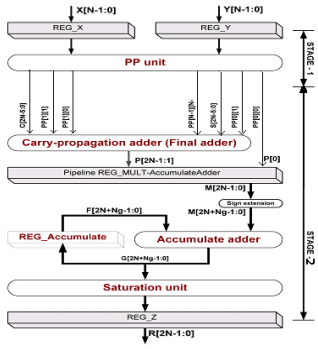

Figure 3.3: Block diagram of the Two-cycle MAC architecture.

In Two-cycle MAC architectures have a first stage that is significantly slower than the second stage. By performing carry propagation only in the second stage of the MAC

pipeline, multiplication and accumulation have similar delays**.** The partial products are generated in the first stage and stored in the pipeline register. In the second stage partial product addition is performed by the carry propagation adder and provides the result in sum and carry. This result is accumulated with the previous result for each consecutive clock cycle in the second stage.

Due to the removal of the pipeline register between the PP unit and the final adder the partial products computed are not fed to the second stage within the stipulated time. The critical path of this unit goes through the PP unit and the final adder. The evaluation results shows that this architecture has better power and gate count when compared with reference architecture. The delay of this unit remains high with the 3-Cycle MAC unit due to the removal of the pipeline register after the PP unit.

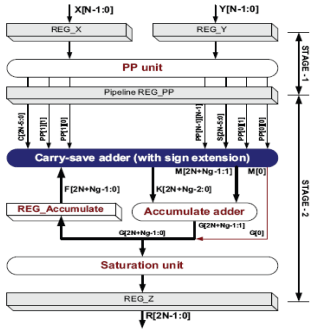

Figure 4.1: Block diagram of the MAC-NEW unit

Carry propagation only takes place in the second stage, which means that the multiplier's final adder is eliminated, leading to higher speed and lower energy. Since accumulation takes place inside the second stage a pipeline register located before the accumulation stage has no impact on functionality. Regardless of pipelining, our MAC unit will produce the correct result in each cycle, and no extra cycles need to be added at the end of the loops– interconnects are localized, which simplifies routing, decreases delay, and reduces energy dissipation.

Because of the above advantages, it supports several guarding bits, making longer loops feasible without any overflow problems. The use of guarding bits in an approach where the accumulated value is fed back to the PPRT's input would most certainly have a negative impact on hardware complexity. The MAC-NEW exploits the fact that the delay of the accumulate adder is shorter than the delay of the PP unit, by at least an amount corresponding to the delay of a full-adder cell.

CHAPTER 4
PROPOSED ARCHITECTURE OF MAC

4.1 PROPOSED MULTIPLY ACCUMULATE ARCHITECTURE

The MAC–NEW architecture is based on two's complement representation, it uses guarding bits to efficiently support longer MAC loops, and it includes output saturation. By performing carry propagation in the second stage of the MAC pipeline, multiplication and accumulation have similar delays. The carry-save adder is used which leads to the reduction of power. With reference to the two cycle MAC architecture, this unit inserts the pipeline register after the partial product unit.

This architecture is based on two conditions such as

- The accumulation should take place in the second stage of a 2-cycle MAC unit.
- The carry should be propagated only once in a MAC pipeline, thus, in the second stage.

The MAC-NEW unit shown in Fig 4.1 consists of two stages: partial product unit in the first stage and the accumulate adder in the second stage. The final adder has been removed, and a carry-save adder has been inserted after the pipeline registers. The maximum delay of the carry-save adder is only that of a single full adder, which means that the MAC's critical path delay still depends on the PP unit. In the carry-save adder there is no need to sign extend the multiplier output instead use a row of '1' to perform the sign extension.

This MAC unit do not require any extra cycles at the end of the loops as the interconnects are localized which simplifies routing, decreases delay and reduces energy dissipation. As the carry propagation and the accumulation takes place in the second stage this architecture uses several guard bits without any overflow problems. The critical path delay of this unit is within the partial product unit.

The critical path is through the PP unit as this architecture uses pipeline registers at the bottom of the PP unit, MAC-NEW obviously can operate at the same speed as MAC-3C, while its performance on average for various operand size such as 16, 32, 48 and 64 is faster than MAC-2C. As far as power dissipation is concerned, the final adder is replaced by the simple carry-save adder,MAC-3C on average dissipates more power than MAC-NEW for the same operating frequency and timing constraint. It requires two cycles for completing the MAC computation, still performs the MAC operation at the same operating frequency as a 3-cycle MAC unit, at lower energy dissipation.

The Evaluation methodology shows that the MAC-NEW unit is efficient in performance parameters such as power, delay and gate count in comparison with the conventional architecture. Due to the efficiency, this architecture is used to create an application architecture called Double Throughput Multiply Accumulate unit [DTMAC].

CHAPTER 5

APPLICATION OF PROPOSED ARCHITECTURE OF MAC

5.1 DOUBLE THROUGHPUT MULTIPLY ACCUMULATE UNIT

A MAC unit that can optionally switch between *N*-bit operation and 2x*N*/2-bit operation is referred as a Double Throughput MAC (DTMAC) is shown in Fig 5.1. This feature would be useful in many DSP-oriented applications, when the dynamic range is lower or when there is a need to simultaneously calculate real and imaginary values. A double throughput 32-bit MAC can be logically implemented by tying together two separate, single 16-bit MACs that support two parallel MAC operations.

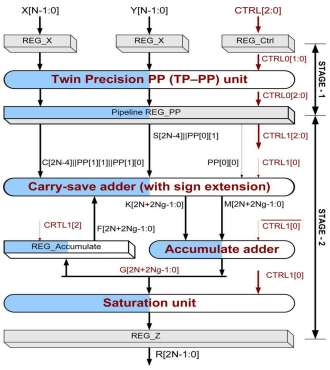

Figure 5.1: Block diagram of the DTMAC unit

Our DTMAC unit in Fig 5.2 is designed to support the efficient execution of several operating modes in a 32-bit data path. The unit employs the Twin-Precision (TP) technique, in terms of a modified 32-bit TP multiplier1 that contains a Twin-Precision Partial-Product Reduction Tree (TP-PPRT) to generate the partial product outputs, which

in conventional schemes are fed to a final adder2. Instead we insert a level of adder cells that combine the outputs of the TPPPRT with the result of the twin-precision accumulate adder; is called"combination unit". In the guarding bit positions of the combination unit, the half adder cells add'1's with the accumulated result, to obtain the correct logical function. The combination unit can be placed after or before the pipeline registers depending on whether the TP-PPRT or the twin-precision accumulate adder represents the dominant delay of the DTMAC unit.

The use of the combination unit makes it possible to build a high-speed, but still flexible DTMAC unit using only two pipeline stages, which limits the clock load and makes for a power-efficient design. The twin-precision accumulate adder is based on the Ladner-Fisher parallel-prefix structure and contains 80 bits, divided in two sections (high and low) each containing 32 data and eight (8) extra guarding bits, as shown in the detailed schematic of Fig. 2(c). Because each of the two sections has eight guarding bits, this DTMAC unit supports loops with 256 iterations without requiring any right shifting of the output to avoid overflow. To control the operating mode, an AND gate is inserted; one control bit (CTRL2[0]) sets the XOR's input at position 40 to either zero or to the carry signal of the 32-bit data part of the low section of the twin-precision accumulate adder.

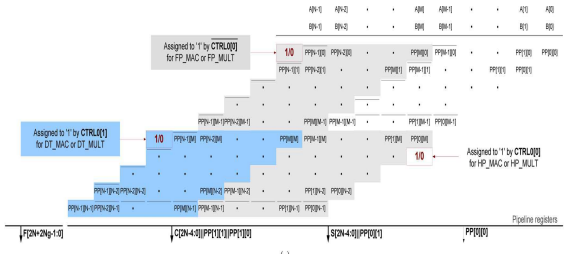

Figure 5.2: Block diagram of the TP-PP unit based on the Baugh–Wooley multiplication algorithm.

5.2 Components of DTMAC unit:

1) TP-PP Unit*:* To support double-throughput operations, the partial-product generation and reduction are based on the twin-precision (TP) technique [24]. Here, the partial products that are not needed during narrow-width operations are forced to zero while some lower-significance partial products are negated4 to provide the correct function for theM -bit multiplication in the lower-significance section. Depending on the operating mode, "1" bits can be set in position N+M, N and M.M=N/2 is assumed as the lower-significance section the "low half."

2) Carry-Save Adder: The carry-save adder (CSA) shown in Fig 5.3 is used for the Partial product addition for the DTMAC unit .In this carry save adder, guard bits and sign extension for the N/2-bit operation in the low half must be accommodated .This is achieved by inserting a row of Ng+1 bits "1" that is summed together with the accumulated value and the most significant bit of the result from the TP-PP unit for the N/2-bit operation in the low half bit position. During N/2-bit operations in the low half, S[N-3] will always be zero, due to the TP technique in which partial products are forced to zero. Since S [N-3] will not carry any useful information during N/2-bit operations in the low half, this signal can be used to add the required "1" at bit positionN-1 . This is easily done by feeding S [N-3] and a control signal through an extra OR gate, whose output may optionally be forced to "1,"

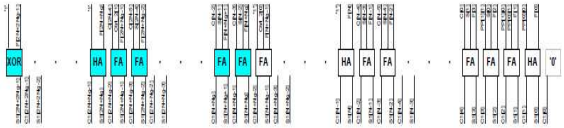

Figure 5.3: Block diagram of the gates of the combination unit in the DTMAC unit.

3) Accumulate Adder: The accumulate adder shown in Fig 5.4 of the DTMAC unit is based on the conditional-sum adder structure, enabling efficient separation into high and low halves, each with Ng guard bits to avoid overflow. To control the operating mode, an AND gate is inserted; one control bit (CTRL1[0]) sets the AND's input at position N+Ng

either to zero or to the carry signal of the N-1 -bit data part of the low half of the accumulate adder. For full precision operations, this effectively by passes the Ng guard bits used for N/2 -bit operations in the low half. Similarly, the accumulator output bits that correspond to unused guard bits (F [N+Ng-1: N]) are discarded during N-bit operation.

4) Saturation Circuit: The saturation unit for the DTMAC not only needs to consider full precision (N) operations but also the N/2 operations in the high and low halves.

- In full-precision mode, 2N+Ng bits in the output of the accumulate adder are processed.
- In half-precision mode, bits N+Ng are processed.
- In double-throughput mode, not only N+Ng bits of the low half are processed, but also N+Ng bits of the high half are processed.

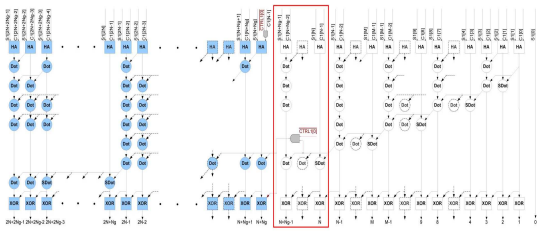

Figure 5.4: Block diagram of the accumulate adder based on the conditional-sum adder architecture

5.3 DTMAC OPERATING MODES

The DTMAC unit operates on two's complement data and supports six operating modes—three for MAC operations and three for multiplications—as determined by the value of the 3-bit control signal (CTRL):

- 000: Full-Precision 32-bit MAC (FP DTMAC).
- 001: Half-Precision 1x16-bit MAC (HP DTMAC).
- 010: Double-Throughput 2x16-bit MAC (DT DTMAC).

- 100: Half-Precision 1x16-bit multiplication (HP MULT).
- 101: Double-Throughput 2x16-bit multiplication (DT MULT).
- 110: Full-Precision 32-bit multiplication (FP MULT).

In the proposed DTMAC unit, there exists no final adder. This makes the critical path delay of the 2-cycle DTMAC dominated by the delay of the TP-PPRT part. The DTMAC actually has the same critical delay as that of a conventional 3-cycle single 32-bit MAC, in which a pipeline register is inserted between the PPRT block and the final adder to several the critical path of the multiplication. The result is that the DTMAC unit, despite the operating mode flexibility, has small area, low power dissipation and short critical path delays. When the DTMAC unit operates in HP DTMAC mode, half of the respective registers are de-activated to isolate the inputs of half of the twin-precision accumulate adder and the MSB input bits of the multiplier are set to zero, to reduce switching activity and dynamic power dissipation.

When the DTMAC unit operates in 1×16-bit MAC mode it dissipates a negligible amount of energy more than the basic, fixed-function, 16-bit MAC unit. The DTMAC unit has a large footprint than MAC32-2C due to extra circuitry to support the multiple operation modes. These comparisons reveal that the implementation of operating-mode flexibility in the DTMAC unit comes at a limited overhead.

The important point is that we can save energy by adjusting the operating mode to the precision of the data:

- When the DTMAC unit operates in the default 32-bitMAC mode (FP_MAC), its energy dissipation is lower than MAC32-2C when performing 32-bit computations.
- When the DTMAC unit operates in 1 16-bit MAC mode (HP_MAC), the 32-bit DTMAC unit performs 16-bit multiply-accumulate operations more energy efficiently than MAC32-2C performs computations on 16-bit operands. This reduction largely stems from avoiding unnecessary switching caused by the 16-bit sign extension of two's complement 32-bit data that carry only 16 bits of information.

- When the DTMAC unit operates in the 2 16-bit MAC mode (DT_MAC), its energy dissipation per 16-bit multiply-accumulate operation is similar to that of MAC16-2C. However, the DTMAC unit uses only half the cycles of MAC16-2C to compute all operations, so the surrounding data path circuits are engaged for a significantly shorter time. This leads to significant energy savings for a system in which the DTMAC unit is integrated.

5.4 MULTIPLICATION THROUGH TWIN PRECISION

The twin-precision technique shown in Fig 5.5 is an efficient way of achieving Double Throughput in a multiplier with low area overhead and delay. The twin- precision technique on signed multipliers based on the regular High Performance Multiplier (HPM) reduction tree. The twin-precision technique can reduce the power dissipation by adapting a multiplier to the bit width of the operands being computed. The technique also enables an increased computational throughput, by allowing several narrow-width operations to be computed in parallel.

Achieving double throughput for a multiplier is not as straightforward as, for example, in an adder, where the carry chain can be cut at the appropriate place to achieve narrow-width additions. It is possible to use several multipliers, where at least two have narrow bit width, and allow them share the same routing, but has several drawbacks: i) The total area of the multipliers would increase, since several multiplier units are used. ii) The use of several multipliers increases the fan out of the signals that drive the inputs of the multipliers. Higher fan out means longer delays and/or higher power dissipation. iii) There would be a need for multiplexers that connect the active multiplier(s) to the result. It is not as easy to deploy the twin-precision technique onto a BW multiplication as it is for the unsigned multiplication, where only parts of the partial products need to be set to zero. To be able to compute two signed multiplications, it is necessary to make a more sophisticated modification of the partial-product array.

For the 4-bit multiplication in the LSP of the array, there is a need for some more modifications. In the active partial-product array of the 4-bit LSP multiplication (shown in

white), the most significant partial product of all rows, except the last, needs to be negated. For the last row it is the opposite, here all partial products, except the most significant, are negated. Also for this multiplication a sign bit '1' is needed, but this time in column. Finally the MSB of the results needs to be negated to get the correct result of the two 4-bit multiplications.

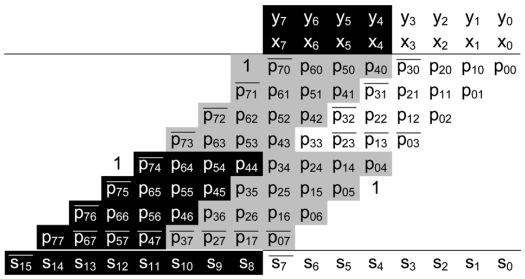

Figure 5.5: Illustration of a unsigned 8-bit multiplication, using the Baugh–Wooley Algorithm

To allow the full-precision multiplication of size to coexist with two multiplications of size in the same multiplier, it is necessary to modify the partial-product generation and the reduction tree. For the -bit multiplication in the MSP of the array all that is needed is to add a control signal that can be set to high, when the N/2-bit multiplication is to be computed and to low, when the full precision multiplication is to be computed. To compute the N/2-bit multiplication in the LSP of the array, certain partial products need to be negated. This can easily be accomplished by changing the two-input AND gate that generates the partial product to a two-input NAND gate followed by an XOR gate. The second input of the XOR gate can then be used to invert the output of the NAND gate. When computing the N/2-bit LSP multiplication, the control input to the XOR gate is set to low making it work as a buffer. When computing a full-precision multiplication the same signal is set to high making the XOR work as an inverter. Finally

the MSB of the result needs to be negated and this can again be achieved by using an XOR gate together with an inverted version of the control signal for the XOR gates used in the partial-product generation. The unwanted partial products to zero can be done by three-input AND gates as for the unsigned multiplication.

5.4.1 HPM IMPLEMENTATION

A twin-precision implementation based on the regular HPM reduction tree is shown in Fig.5.6. For high speed and/or low-power implementations, a reduction tree with logarithmic logic depth, such as TDM [9], Dadda [10], Wallace [11] or HPM [12] is preferred for summation of the partial products. Such a log-depth reduction tree has the benefit of shorter logic depth. Further, a log-depth tree suffers from fewer glitches making it less power dissipating. In fig 5.3, the unsigned multiplication is implemented in Baugh-Wooley algorithm in which 4-bit multiplication, shown in white, can be computed in parallel with a second 4-bit multiplication, shown in black. For simplicity the AND gates for partial-product generation is not shown and a ripple carry is used as final adder.

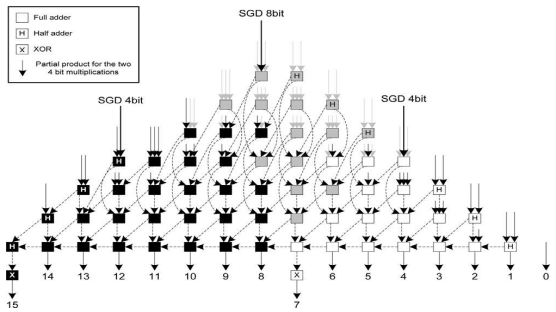

Figure 5.6: Block diagram of an unsigned 8-bit twin-precision multiplier based on the regular HPM reduction tree

5.5 FLOATING POINT MULTIPLIER IN MULTIPLY ACCUMULATE UNIT

Floating Point numbers represented in IEEE 754 format are used in most of the DSP Processors. Floating point arithmetic is useful in applications where a large dynamic range is required or in rapid prototyping applications where the required number range has not been thoroughly investigated. The Floating Point Multiplier IP helps designers to perform floating point Multiplication on FPGA represented in IEEE 754 single precision floating point format.

5.5.1 FUNCTIONAL DESCRIPTION

A Floating point multiplier is the most common element in most digital applications such as digital filters, digital signal processors, data processors and control units. The present Floating Point Multiplier IP has three blocks sign calculator, exponent calculator, mantissa calculator, which works parallel and a normalization unit. The Multiplier is pipelined, so the first result appears after the latency period and then the result can be obtained after every clock cycle.

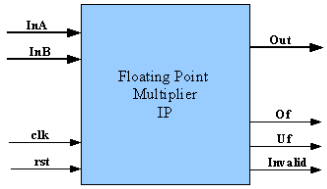

Figure 5.7: Block diagram of the Floating Point Multiplier

The Schematic symbol of Floating Point Multiplier is shown in Fig 5.7. It takes two IEEE 754 format single precision floating point numbers and produces the multiplied output. It also supports the features like underflow, overflow and invalid operations. This

unit consists of two stages, multiplication calculation and normalization. The first stage consists of the following three blocks which work in parallel.

- *Sign Calculator*: The Output Sign is the exor of two sign bit inputs.
- *Exponent Calculator*: The input exponents are added and the bias is removed to produce the exponent of Output.
- *Mantissa Calculator*: Output Mantissa is calculated by multiplying the mantissa's of two inputs. Second stage performs Normalization of the Output obtained from the first stage.
- *Normalization Block*: The normalization is the last and most complicated part. This block is implemented in three pipelined stages.

This block first calculates how much amount the mantissa needs to be left shifted. The mantissa is processed in parallel in a number of modules, each looking at four bits of the mantissa. The first module looks at first four bits of the mantissa and outputs the amount to be shifted assuming a one was found on these four bits. The second module operates on the next four bits of the mantissa treating first four bits are zero and outputs the amount to be shifted left.

This process is repeated for the remaining bits of mantissa. Signals are generated if the four bits of the mantissa are zero. Depending on the signal values the amount of shift is selected. This selection is implemented in three multiplexer stages. Depending on the two leading bits of final mantissa, the final mantissa is shifted left by previously calculated shift amount or shifted right. The final exponent is also corrected accordingly.

CHAPTER 6
SIMULATION RESULTS AND DISCUSSION

All PP units of the MAC architectures are based on the power-efficient Baugh–Wooley algorithm for partial-product generation and the HPM partial-product reduction tree. The accumulate adder is of conditional-sum type and has an extension of eight guard bits (Ng=8). This allows the MAC unit to support loops of up to 256 iterations without requiring the output to be right-shifted to avoid overflow. A final adder based on parallel algorithm of recurrence equation supports fast addition of the PP unit outputs. The Multiply Accumulate architecture is designed using VHDL and simulated using MODEL SIM. The performance parameters are synthesized using Xilinx.

6.1 SIMULATION WAVEFORM OF THREE-CYCLE MAC UNIT

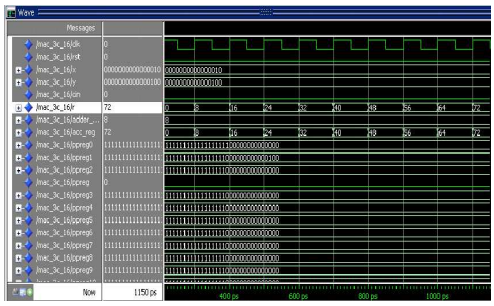

Figure 6.1: Waveform for the three-cycle MAC of operand size 16 -bit

The inputs of MAC-3C unit x and y are of 16 bits. The multiplier output is 16-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32-bit (acc_reg).

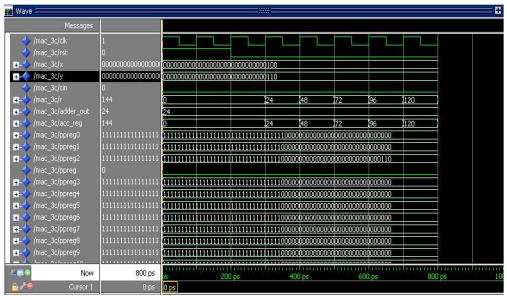

Figure 6.2: Waveform for the three-cycle MAC of operand size 32-bit

The inputs of MAC-3C unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg) .

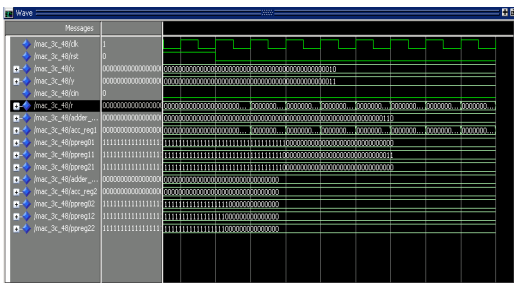

Figure 6.3: Waveform for the three-cycle MAC of operand size 48-bit

The inputs of MAC-3C unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg).

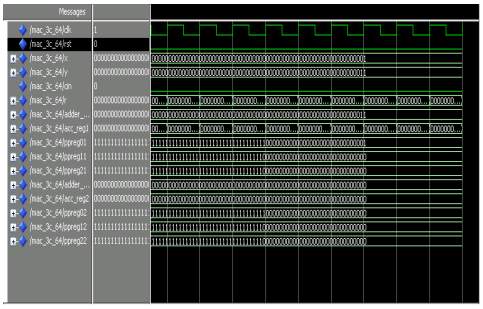

Figure 6.4: Waveform for the three-cycle MAC of operand size 64-bit

The inputs of MAC-3C unit x and y are of 64 bits.The multiplier output is 64 -bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128 -bit (acc_reg).

6.2 SIMULATION WAVEFORM OF TWO-CYCLE MAC UNIT

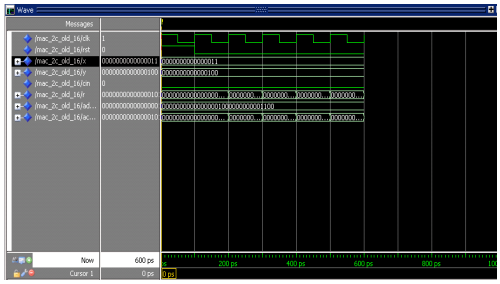

Figure 6.5: Waveform for the two-cycle MAC of operand size 16-bit

The inputs of MAC-2C unit x and y are of 16 bits. The multiplier output is 16 -bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32 -bit (acc_reg).

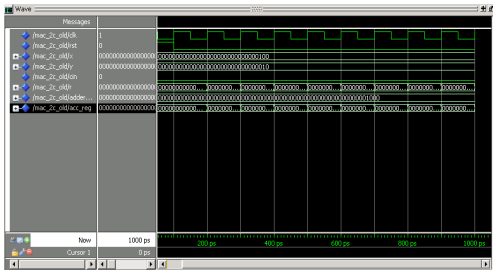

Figure 6.6: Waveform for the two-cycle MAC of operand size 32-bit

The inputs of MAC-2C unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg).

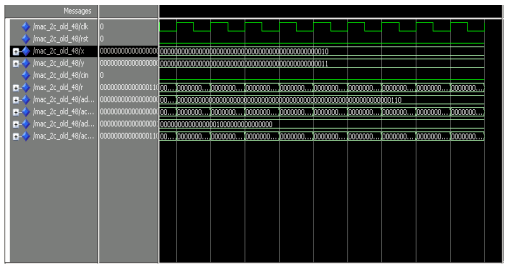

Figure 6.7: Waveform for the two-cycle MAC of operand size 48-bit

The inputs of MAC-2C unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg).

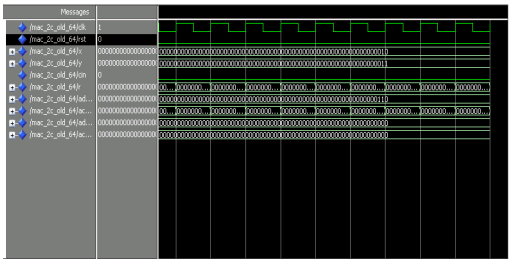

Figure 6.8: Waveform for the two-cycle MAC of operand size 64-bit

The inputs of MAC-2C unit x and y are of 64 bits. The multiplier output is 64-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128-bit (acc_reg).

6.3 SIMULATION WAVEFORM OF MAC-NEW UNIT

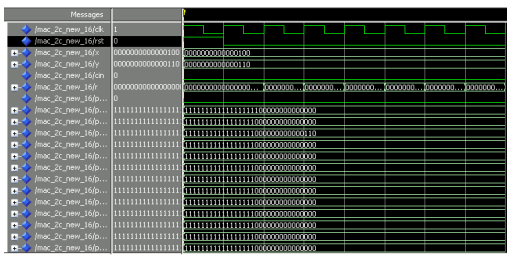

Figure 6.9: Waveform for the MAC-NEW of operand size 16-bit

The inputs of MAC-NEW unit x and y are of 16 bits. The multiplier output is 16-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 32-bit (acc_reg).

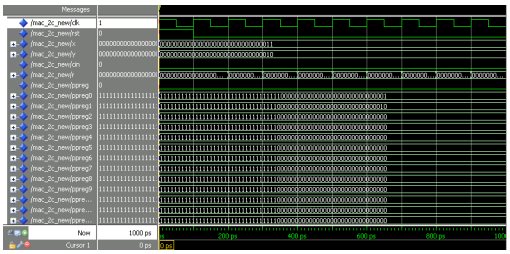

Figure 6.10: Waveform for the MAC-NEW of operand size 32-bit

The inputs of MAC-NEW unit x and y are of 32 bits. The multiplier output is 32-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 64-bit (acc_reg) .

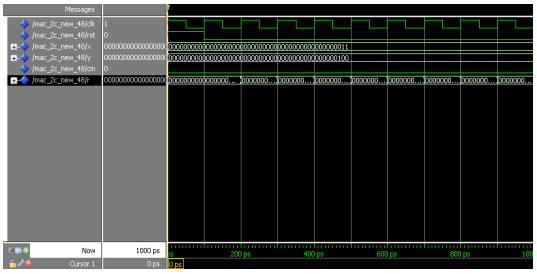

Figure 6.11: Waveform for the MAC-NEW of operand size 48-bit

The inputs of MAC-NEW unit x and y are of 48 bits. The multiplier output is 48-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 96-bit (acc_reg) .

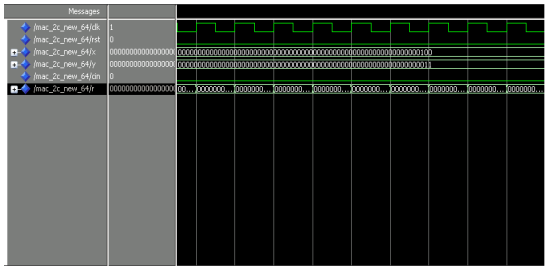

Figure 6.12: Waveform for the MAC-NEW of operand size 64-bit

The inputs of MAC-NEW unit x and y are of 64 bits. The multiplier output is 64-bit stored in the register (r) and the partial product generated is added with the final adder and the result stored in the accumulate register is 128-bit (acc_reg).

6.4 SIMULATION WAVEFORM OF DTMAC UNIT

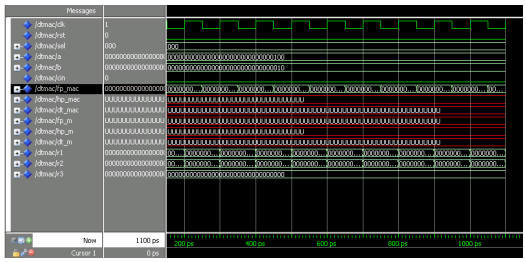

Figure 6.13: Waveform for the Full Precision DTMAC unit

The inputs of the DT_MAC mode is 2×16 bit in which LSB of the a and b-bit is taken as 1×16 bit and MSB of the a and b-bit are taken as 1×16 bit. The selection mode is given 011 and for each consecutive clock cycle the accumulated result is stored in the DT_MAC.

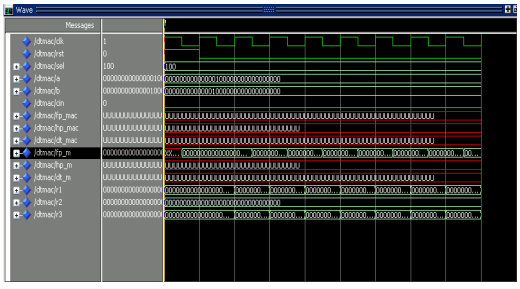

Figure 6.16: Waveform for the Full Precision Multiplication unit

The inputs of the FP_MULT mode is 32- bit in which MSB of the and b-bit is taken as two 1×16 bit. The selection mode is given 100 and for each consecutive clock cycle the multiplication result is stored in the FP_M.

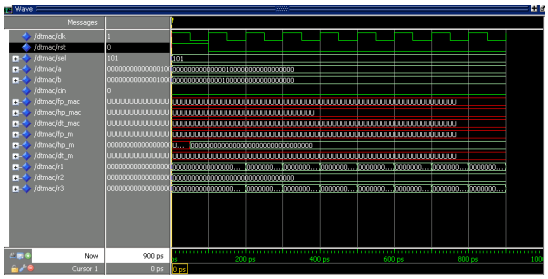

Figure 6.17: Waveform for the Half-Precision Multiplication unit

The inputs of the FP_MAC mode is 32 bit in which LSB of the a-bit and b-bit are taken as two 16-bits.The selection mode is given 000 and for each consecutive clock cycle the accumulated result is stored in the FP_MAC.

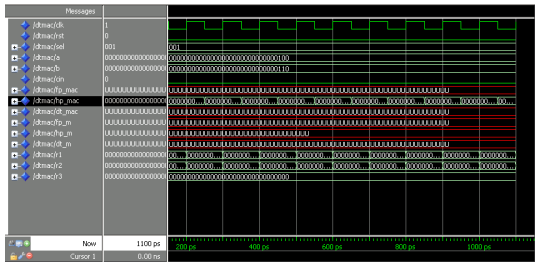

Figure 6.14: Waveform for the Half Precision DTMAC unit

The inputs of the HP_MAC mode is 16 bit in which LSB of the a-bit and b-bit are taken as two 8-bits.The selection mode is given 001 and for each consecutive clock cycle the accumulated result is stored in the HP_MAC.

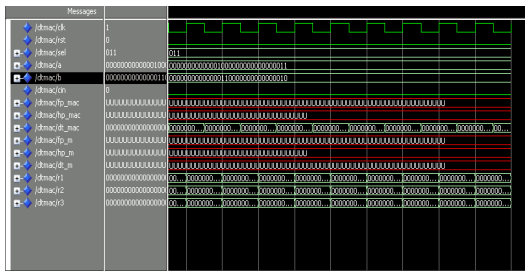

Figure 6.15: Waveform for the DTMAC unit

The inputs of the HP_MULT mode is 1×16 bit in which MSB of the a and b-bit is taken as two 8-bit.The selection mode is given 101 and for each consecutive clock cycle the multiplication result is stored in the HP_M.

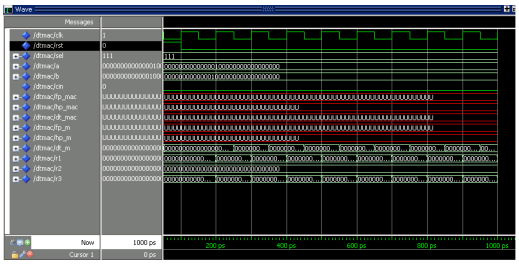

Figure 6.18: Waveform for the Double Throughput Multiplication unit

The inputs of the DT_MULT mode is 2×16 bit in which MSB of the a and b-bit is taken as two 16-bit.The selection mode is given 111 and for each consecutive clock cycle the multiplication result is stored in the DT_M.

6.5 SIMULATION WAVEFORM OF FLOATING POINT MULTIPLIER

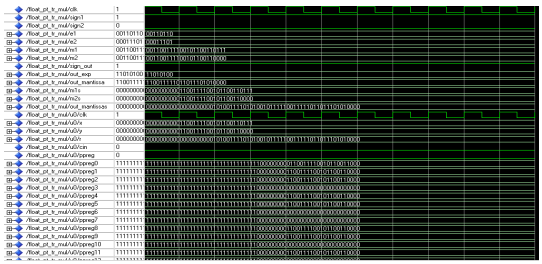

Figure 6.19: Waveform for the Floating Point Multiplier MAC unit

The input of the Floating Point Multiplier is 32- bit in which each of exponents (e1 and e2) is 8 bit. The mantissa bit (m1 and m2) are 23 bit and the sign bit (s1 and s2) is 1-bit.The accumulation is done by the MAC-NEW 32-bit.

6.6 SYNTHESIS REPORT OF THE MAC ARCHITECTURE

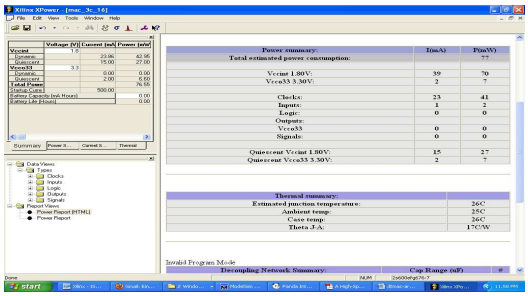

Figure 6.20: Power calculation for 3-C MAC unit of 16-bit

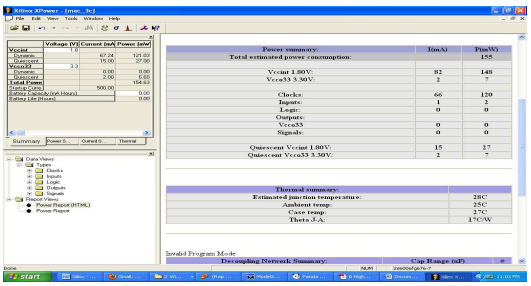

Figure 6.21: Power calculation for 3-C MAC unit of 32-bit

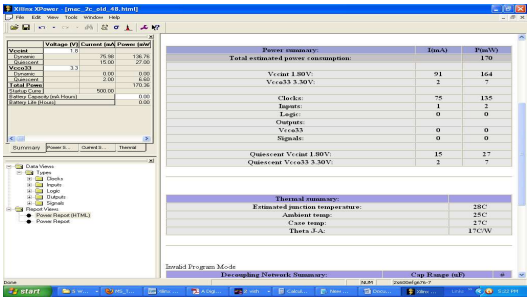

Figure 6.22: Power calculation for 2-C MAC unit of 48-bit

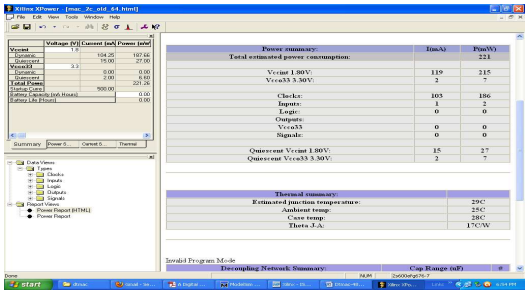

Figure 6.23: Power calculation for 2-C MAC unit of 64-bit

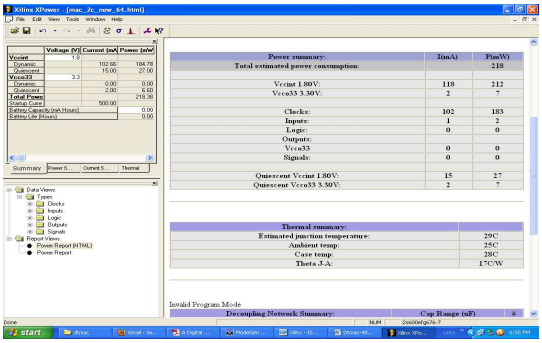

Figure 6.25: Power calculation for MAC-NEW unit of 64-bit

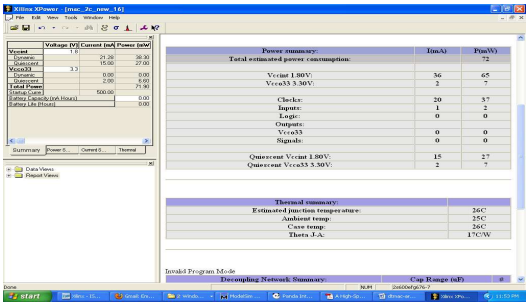

Figure 6.24: Power calculation for MAC-NEW unit of 16-bit

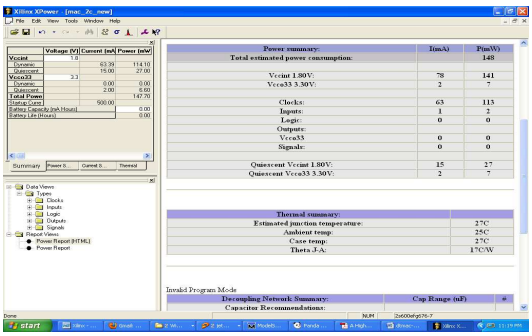

Figure 6.26: Power calculation for Full Precision DTMAC unit

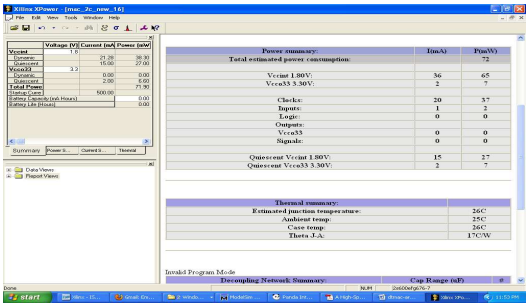

Figure 6.27: Power calculation for Half Precision DTMAC unit

6.7 COMPARISON OF VARIOUS MAC ARCHITECTURES

Table 6.1: Performance Analysis of conventional MAC architectures of the operand size 16 and 32 bit

OPERAND SIZE	16		32		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-2C	MAC-3C	MAC-2C	16-bit	32-bit
POWER(mW)	77	70	155	154	9.09	0.65
DELAY(ns)	79.97	79.24	158.94	162.59	0.92	2.29
GATE COUNT	14,885	13,930	52,439	51,320	6.42	2.13

The performance parameters such as power, delay and gate count are tabulated for the operand size of 16 and 32-bit of the Three-cycle and Two-cycle MAC architecture. The parameters such as power and gate count for the 3-C MAC unit is high in comparison with the 2-C MAC unit but the delay for the 2-C MAC unit remains high. The performance is evaluated for the 16 and 32-bit.

Table 6.2: Performance Analysis of conventional MAC architectures of the operand size 48 and 64 bit

OPERAND SIZE	48		64		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-2C	MAC-3C	MAC-2C	48-bit	64-bit
POWER(mW)	176	170	224	221	3.41	1.34
DELAY(ns)	161.47	163.02	171.75	173.20	0.96	0.84
GATE COUNT	58,974	59.091	88,336	89,132	0.20	0.90

The performance parameters such as power, delay and gate count are tabulated for the operand size of 48 and 64 bit of the Three-cycle and Two-cycle MAC architecture. The parameters such as power and gate count for the 3-C MAC unit is high in comparison with the 2-C MAC unit but the delay for the 2-C MAC unit remains high .The performance is evaluated for the 48 and 64-bit.

Table 6.3: Performance Analysis of 3-C and MAC-NEW architectures of the operand size 16 and 32 bit

OPERAND SIZE	16		32		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-NEW	MAC-3C	MAC-NEW	16-bit	32-bit
POWER(mW)	77	72	155	148	6.50	4.52
DELAY(ns)	79.97	74.21	158.94	153.27	7.20	3.57
GATE COUNT	14,885	13,356	52,439	42,880	10.27	16.44

The performance parameters such as power, delay and gate count are tabulated for the operand size of 16 and 32 bit of the Three-cycle and MAC-NEW architecture. The parameter for the 3-C MAC unit is high in comparison with the MAC-NEW unit. The performance is evaluated for the 16 and 32 bit.

Table 6.4: Performance Analysis of 3-C and MAC-NEW architectures of the operand size 48 and 64 bit

OPERAND SIZE	48		64		Performance Evaluation (%)	
Architecture	MAC-3C	MAC-NEW	MAC-3C	MAC-NEW	48-bit	64-bit
POWER(mW)	176	164	224	218	6.82	2.68
DELAY(ns)	161.47	157.22	171.75	168.96	2.64	1.62
GATE COUNT	58,974	49,180	88,336	77,903	16.61	11.81

The performance parameters such as power, delay and gate count are tabulated for the operand size of 48 and 64 bit of the Three-cycle and MAC-NEW architecture. The parameter for the 3-C MAC unit is high in comparison with the MAC-NEW unit. The performance is evaluated for the 48 and 64 bit.

Table 6.5: Comparison of Operating Modes in DTMAC Architecture

Architecture	FP_MAC	HP_MAC	DT_MAC
Power	148	72	144
Delay	153.27	74.21	148.42
Gate Count	42,880	13,356	26,712

The DTMAC operating modes parameters are tabulated in Table 6.5. The parameters of the FP_MAC are same as 32-bit MAC-NEW architecture. The HP_MAC is same as 16-bit MAC-NEW architecture. The DT_MAC is 2×16-bit MAC-NEW architecture.

Table 6.6 :Parameters of the Floating Point multiplier in MAC unit

Parameters	
Power (mW)	106
Delay(ns)	73.964
Gate Count	3579

6.8 POWER ANALYSIS

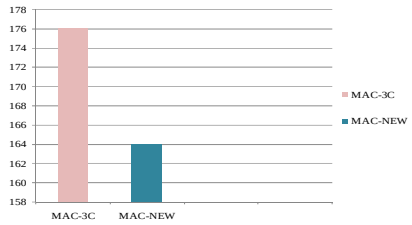

Figure 6.28: Power Analysis of MAC-3C and MAC-NEW of the operand size 32 bit

The Power analysis is performed for the MAC-3C and MAC-NEW architecture. The MAC-3C unit has more power when compared with the MAC-NEW architecture due to the three-pipeline stages.

6.9 DELAY ANALYSIS

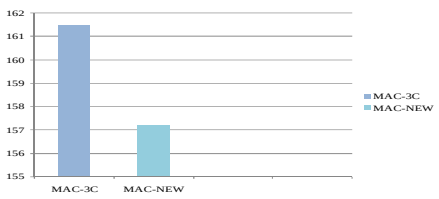

Figure 6.29 :DelayAnalysis of MAC-3C and MAC-NEW of the operand size 32 bit

The Delay analysis is performed for the MAC-3C and MAC-NEW architecture. The MAC-3C unit has more delay when compared with the MAC-NEW architecture due to the three-pipeline stages.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

This project presents the estimation of the efficient performance parameters such as power, gate count, and delay for the different Multiply Accumulate architectures. The architectures are designed using Baugh-Wooley algorithm. The Three-cycle, Two-cycle and MAC-NEW architecture is simulated through MODEL SIM and synthesized using XILINX. The performance parameter of the conventional MAC architecture is compared with the proposed MAC architecture and the results are tabulated.

The comparison is made between the MAC-3C and MAC-2C architecture in which the power and gate count remains high for the MAC-3C but the delay is large for the MAC-2C due to the removal of the pipeline register after the Partial Product (PP) unit. The MAC-NEW is compared with the reference architecture (MAC-3C) and the results are tabulated in which the parameters are efficient for the MAC-NEW architecture. As it is an efficient architecture it is used to create a versatile MAC unit called Double Throughput MAC unit(DTMAC).As a modification to this project, the Floating point multiplier is used in the MAC unit and the parameter are tabulated.

FUTURE SCOPE

The MAC-NEW architecture can be used in the efficient design of digital signal processing circuits such as FIR and IIR filter. As this architecture is efficient in performance parameters it increases the computation of the filter.

BIBLIOGRAPHY

[1] A High-Speed, Energy-Efficient Two-Cycle Multiply-Accumulate (MAC) Architecture and Its Application to a Double-Throughput MAC Unit *IEEE Transaction*, volume 57, NO. 12, Dec 2010.

[2] T. T. Hoang, M. Själander, and P. Larsson-Edefors, "High-speed, energy- efficient 2-cycle multiply-accumulate architecture," in *Proceedings. IEEE International. SOC Conference. (SOC)*, Sep. 2009, pp. 119–122.

[3] M. Själander and P. Larsson-Edefors, "Multiplication acceleration through twin precision," *IEEE Transaction Very Large Scale Integrated. (VLSI).*, volume 17, pp. 1233–1246, Sep. 2009.

[4] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications, "in *Proceedings. IEEE International Symposium Circuits System (ISCAS)*, May 2007, pp. 3199–3202.

[5] T.T. Hoang, M. Själander, and P. Larsson-Edefors, "Double throughput multiply-accumulate unit for Flex Core processor enhancements," presented at the IEEE International Symposium, Parallel Distributed Processing Symposium (IPDPS), Reconfigurable Architecture Workshop (RAW), Rome, Italy, and May 2009.

[6] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications, "in *Proceedings IEEE International Symposium Circuits System (ISCAS)*, May 2007 ,pp. 3199–3202.

[7] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier reduction tree with logarithmic logic depth and regular connectivity," in *Proceedings IEEE International Symposium Circuits System (ISCAS)*, May 2006, pp. 4–8.

[8] M. Själander, H. Eriksson, and P. Larsson-Edefors, "An efficient twin precision multiplier," in *Proceedings IEEE International Conference Computer Design (ICCD)*, Oct. 2004, pp. 30–33.

[9] R. K. Kolagotla, J. Fridman, B. C. Aldrich, M. M. Hoffman, W. C. Anderson, M. S. Allen, D. B. Witt, R. R. Dunton, and L.A. Booth, "High performance dual-MAC DSP architecture," *IEEE Signal Processing Magazine.*, volume 19, no. 4, pp. 42–53, July. 2002.

[10] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," in *Proceedings International Symposium High-Performance Computer Architecture*, 1999, pp. 13–22.

[11] P. F. Stelling and V. G. Oklobdzija, "Implementing multiply-accumulate operation in multiplication time," in *Proceedings International Symposium Computer Arithmetic (ARITH)*, July 1997, pp. 99–106.

[12] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Transaction Computer*, volume. C-22, pp.1045–1047, Dec. 1973.

[13] J. Sklansky, "Conditional-sum addition logic," *IRE Transaction Electronic Computer*, volume. EC-9, pp. 226–231, 1960.