CHAPTER 1

INTRODUCTION

Digital Signal Processing (DSP) is finding its way into more applications, and its fame has materialized into a number of commercial processors. Digital signal processors have different architectures and features than general purpose processors, and the performance gains of these features largely determine the performance of the whole processor. The demand for these special features stems from algorithms that require exhaustive computation, and the hardware is often designed to map these algorithms. Widely used DSP algorithms include the Finite Impulse Response (FIR) filter, Infinite Impulse Response (IIR) filter, and Fast Fourier Transform (FFT). Efficient computation of these algorithms is a direct result of the efficient design of the underlying hardware.

The multiplier and multiplier- accumulator (MAC) are the essential elements of the digital signal processing such as filtering, convolution, and inner products. This unit can calculate the running sum of products, which is at the heart of algorithms such as the FIR and FFT. The ability to compute with a fast MAC unit is essential to achieve high performance in many DSP algorithms, and is why there is at least one fanatical MAC unit in all of the modern commercial DSP processors. Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transform (DWT). For the reason that, they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic's determines the execution speed proceedings and performance of the entire calculation. Since the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined by the multiplier, in general.

1.1 PROBLEM STATEMENT

There is various interpretation of the Moore’s Law that predicts the growth rate of integrated circuits. One estimate states the rate at 2X for every eighteen months. Others claim that the device density increases ten-fold every seven years. Regardless of the exact numbers, everyone agrees that the growth rate is rapid with no signs of slowing down. New generations of processing technology are being developed while present generation devices are at very safe distance from the fundamental physical limits. A need for low power VLSI chips arises from such evolution forces of integrated circuits. The Intel 4004 microprocessor, developed in 1971, had 2300 transistors, dissipated about 1 watt of power and clocked at 1 MHz. Coming to the Pentium in 2001, with 42 million transistors, dissipating around 65 watts of power and clocked at 2.40 GHz as discussed by Chandrakasan et al (1999). While the power dissipation increases linearly as the years go by, the power density increases exponentially because of the ever-shrinking size of the integrated circuits. If this exponential rise in the power density were to increase continuously, a microprocessor designed a few years later, would have the same power as that of the nuclear reactor. Such high power density introduces reliability concerns such as, electro migration, thermal stresses and hot carrier induced device degradation, resulting in the loss of performance. Another factor that fuels the need for low power chips is the increased market demand for portable consumer electronics powered by batteries. The craving for smaller, lighter and more durable electronic products indirectly translates to low power requirements. Battery life is becoming a product differentiator in many portable systems. Being the heaviest and biggest component in many portable systems, batteries have not experienced the similar rapid density growth compared to the electronic circuits. The main source of power dissipation in these high performance battery-portable digital systems running on batteries such as note-book computers, cellular phones and personal digital assistants are gaining prominence. For these systems, low power consumption is a prime concern, because it directly affects the performance by having effects on battery longevity. In this situation, low power VLSI design has assumed great importance as an active and rapidly developing field.

At the circuit design level, considerable potential for power savings exists by means of proper choice of a logic style for implementing combinational circuits. This is because all the important parameters governing power dissipation switching capacitance, transition activity, and short circuit currents are strongly influenced by the chosen logic style.

1.2 POWER USAGE IN CMOS

The power used in CMOS-circuits consists of two parts, dynamic and static power dissipation:

$$P = P_{dynamic} + P_{static} \tag{1.1}$$

The dynamic power consumption is one of the powers used as a function of activity. The static component is power consumed as a function of time.

1.2.1 Static Power Consumption

The static part describes power used even though there is no activity in the circuit. Ideally CMOS components should not have any static power consumption, since there are no direct paths from V_{dd} to ground. In practical applications this is not the case, since MOS transistors are not perfect switches. In MOS transistors, always there will be the leakage in currents and it was discussed by Rabaey & Pedram (1996). Reverse biased currents flows through the source or drain and the substrate, because parasitic diodes in the MOS transistors are one of the static leakage currents. The sub threshold leakages current run through the transistors (from source to drain), because the gate of the transistor is close to the threshold voltage, and therefore some current flow through. These currents used to be negligible, however it seems to become more prominent as transistors become smaller as discussed by Liu & Svensson (1993)and really starts to emerge at 0.13 μm as discussed by Piguet, **(**2007**)**. The static power dissipation is primarily determined by fabrication technology.

1.2.2 Dynamic Power Consumption

The dynamic part of the power consumption in CMOS can be divided into two parts as discussed by Chandrakasan, (1992).

$$P_{dynamic} = P_{short-circuit} + P_{switching} \tag{1.2}$$

The short circuit happens when both the PMOS and the NMOS transistor is open at the same time. This happens in a switch, because the PMOS and NMOS do not switch instantly, but has a switching delay. This makes a short circuit line from V_{dd} to ground through the CMOS component. As seen in Figure 1.1, if the NMOS and PMOS transistors in the inverter are both open at the same time, a short-circuit path is available from $V_{dd\ to}$ ground. This phenomenon is described in equation 1.3, where V_{dd} is the supply voltage and I_{sc} is the current flowing through the short circuit period of the switch. As long as the inputs of the NMOS and PMOS transistors are properly balanced, this power dissipation should be less than 20% of the dynamic power dissipation as discussed by Veendrick, (1984).

$$P_{short-circuit} = V_{dd} I_{sc} \tag{1.3}$$

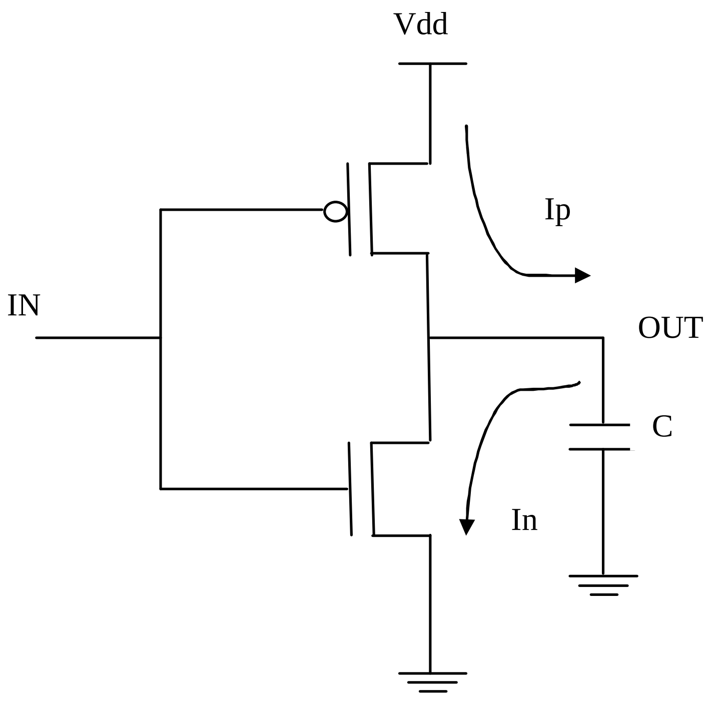

Figure 1.1 Switching power usage in CMOS

The power used in switching the CMOS from one state to another is largely used to charge parasitic capacitance in lines between the CMOS-cells. When the output of a gate is turned from 0 to1, the NMOS part of the CMOS cuts off the connection to ground, and the PMOS part of the CMOS enables a connection from V_{dd} to the output. This causes the capacitance on the output port and line to be charged, with the energy equal to:

$$Energy\ transition = CV^2_{dd} \tag{1.4}$$

where V_{dd} is the power source. Half of this power is dissipated at once in the PMOS transistors, while the other half is stored in the capacitance. When the port is turned from ’1’ to ’0’, the line is connected to ground, and the energy stored in the capacitance is also dissipated (see Figure 1.1).Since an equal amount of energy is used to charge the circuit for each 0 to1 transition, it is possible to get an equation for power used in switching. Considering the frequency f of the circuit, and the probability for a 0 to1 switch at the gate α, and the equation is discussed by Rabaey, (1996):

$$P_{switching} = \alpha f C V^2_{dd} \tag{1.5}$$

Although the other sources of power dissipation have increased their share, switching power consumption is still the largest source for power usage in CMOS today, and is therefore a prime candidate for optimization as discussed by Veendrick, (1984). As seen in the equation, there are three elements to improve power usage: Voltage, physical capacitance and activity. Over the years, lower voltage has been employed in CMOS, causing a reduction in switching power usage. Physical capacitance is strongly correlated to the line length between transistors and the kind of technology being used (size of transistors and lines). Activity is maybe the most system-dependant factor in the equation. By reducing the activity in the design, it is possible to reduce the amount of power used in the design.

1.3 DIGITAL FILTERS

Digital signal processing (DSP) finds innumerable applications in the fields of audio, video, and communications, among others. Such applications are generally based on LTI (linear time invariant) systems, which can be implemented with digital circuitry.

Any LTI system is represented by the following equation:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

where a_k and b_k are the filter coefficients, and x[n-k], y[n-k] are the current (for k=0) and earlier (for k>0) input and output values, respectively. To implement this expression, registers are necessary to store x[n-k] and/or y[n-k] (for k>0),besides multipliers and adders, which are well-known building blocks in the digital domain.

The impulse response of a digital filter can be divided into two categories: IIR (infinite impulse response) and FIR (finite impulse response). The former corresponds to the general case described by the equation above, while the latter occurs when N=0. Only FIR filters can exhibit linear phase, so they are indispensable when linear phase is required, as in many telecom applications. With N=0, the equation above becomes

$$y[n] = \sum_{k=o}^{M} c_k \, x[n-k]$$

where $c_k=b_k/a_0$ are the coefficients of the FIR filter. This equation can be implemented by the system of Figure 1.2, where D (delay) represents a register (flip-flops), atriangle is a multiplier, and a circle means an adder.

An equivalent RTL representation is shown in figure 1.3. As shown, the values of x are stored in a shift register, whose outputs are connected to multipliers and then to adders. The coefficients must also be stored on chip. However, if the coefficients are always the same (that is, if it is a dedicated filter), their values can be implemented by means of logic gates rather than registers (just need to store CONSTANTS). On the other hand, if it is a general purpose filter, then registers are required for the coefficients. In the architecture of figure 1.3, the output vector (y) was also stored, in order to provide a clean, synchronous output.

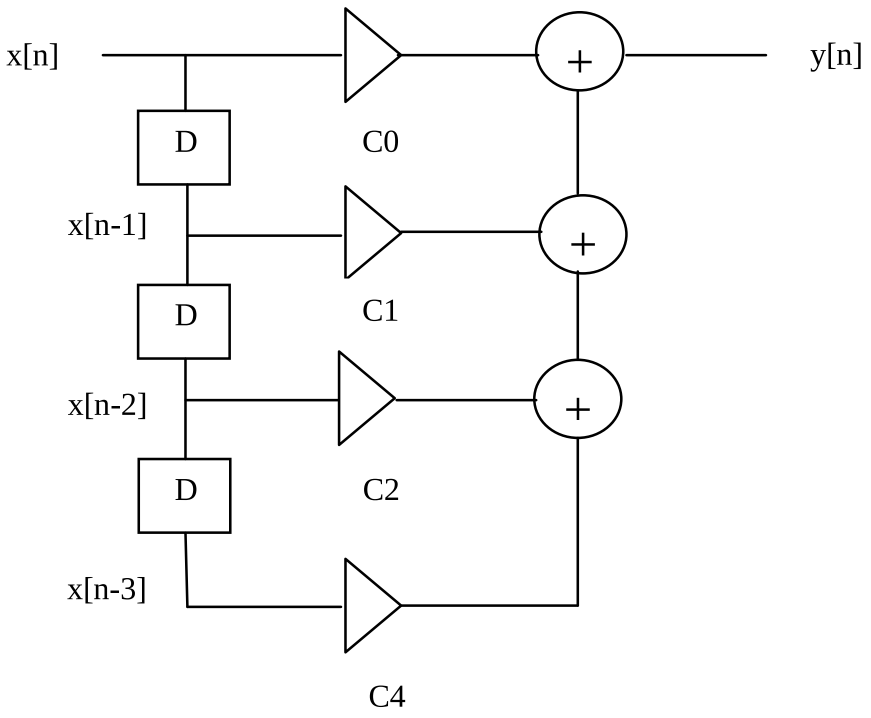

Figure 1.2 FIR filter diagram

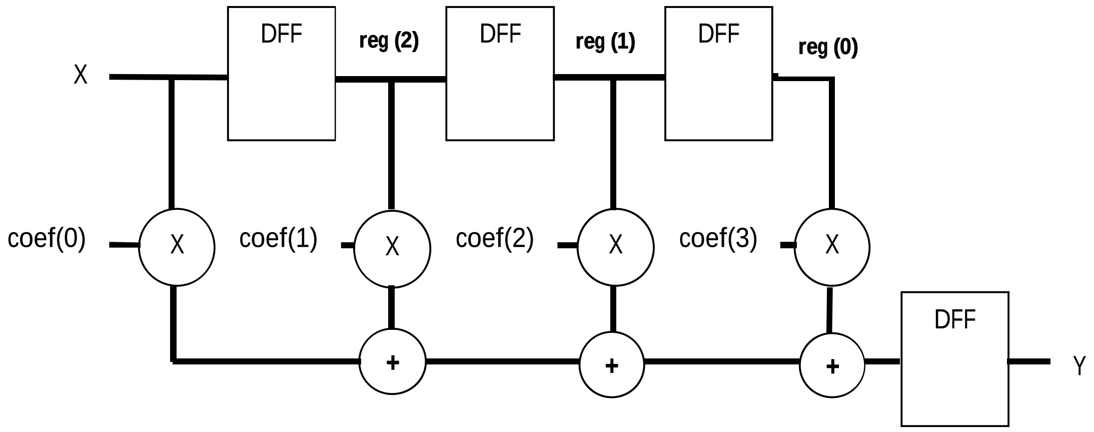

Figure 1.3 RTL representation of FIR filter

The circuit of Figure 1.3 can be constructed in several ways. However, if it is Intended for future reuse or sharing, than it should be as generic as possible.

1.3.1 General Purpose FIR Filter

The design presented above contained fixed coefficients, and is therefore adequate for an ASIC with a dedicated filter. For a general purpose implementation (that is, with programmable coefficients), the architecture of figure 1.4 can be used instead. This structure is modular and allows several chips to be cascaded, which might be helpful in some applications, because FIR filters tend to have many taps (coefficients).

In this structure, there are two shift registers, one is for storing the inputs (x) and the other is for the coefficients. The structure is divided into n equal modules, called TAP1..., TAPn. Each module (TAP) contains a slice of the shift registers, plus a multiplier and an adder.

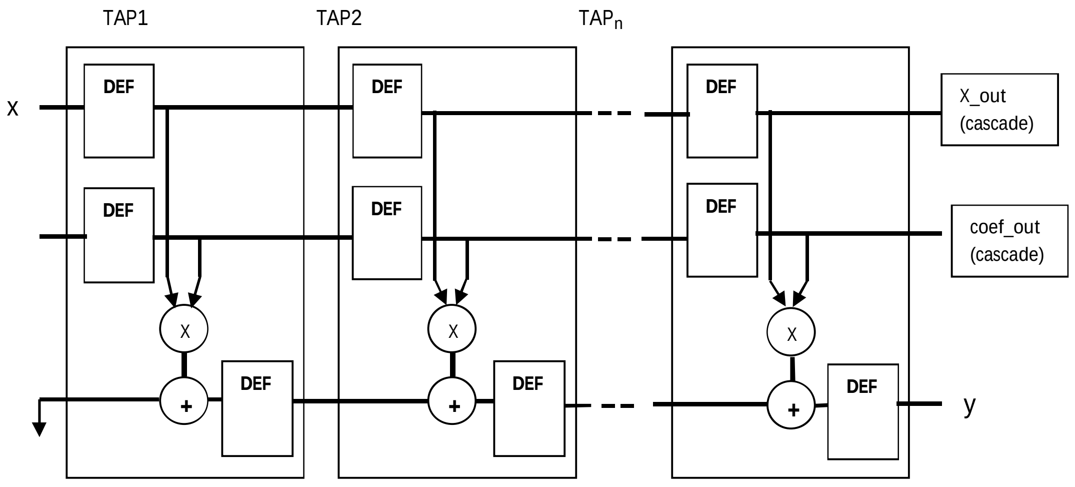

Figure 1.4 General purpose FIR filter

1.4 MAC UNIT

In the core of every microprocessor, DSP and data-processing ASIC is its data path. Statistics showed that more than 70% of the instructions perform additions and multiplications in the data path of RISC machines as discussed by Hsun et al (2000). At the heart of data-path and addressing units in turn are arithmetic units, such as comparators, adders, and multipliers. Digital multipliers are the most commonly used components in any digital circuit design. Multiplication based operations such as Multiply and Accumulate and inner product are among some of the frequently used Computation Intensive Arithmetic Functions, that is currently implemented in many DSP applications such as convolution, fast Fourier transform, filtering and in microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, there is a need of high speed multiplier. Currently, multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip. The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications.

The MAC unit determines the speed of the overall system; it always lies in the critical path. In order to improve the speed of the MAC unit, there are two major bottlenecks that need to be considered. The first one is the partial products reduction network that is used in the multiplication block and the second one is the accumulator. Both of these stages require addition of large operands that involve long paths for carry propagation. To speed up the multiplication process it implements both the multiplication and the accumulation operations within the same functional block by merging the accumulator with the multiplication circuit using tree architectures for the partial products reduction network as discussed by Chan et al (1991).Many researchers have attempted in designing MAC architecture with high speed computational performance and low power consumption. Elguibaly (2000) proposed a fast pipelined implementation to lower the MAC architecture’s critical delay. Murakami et al. adopted the half array implementation to design a high-speed and area-effective MAC architecture. Raghunath (1997) made use of a carry-save multiplier that can simplify sign extension and saturation, and further applies it on MAC architecture to reduce the unit's area and power consumption. Hsun et al (2000) proposed a low-power Multiplication Accumulation Computation (MAC) unit using the radix-4 Booth algorithm, by reducing its architectural complexity and minimizing the switching activities. Kwon et al. developed a merged MAC unit based on fast 5:2 compressors instead of 3:2 and 4:2 compressors. Fayed et al (2002) proposed new data merging architecture for high speed multiply accumulate units. The architecture can be applied on binary trees constructed using 4:2 compressor circuits. Increasing the speed of operation is achieved by taking advantage of the available free input lines of the compressor circuits, which result from the natural parallelogram shape of the generated partial products and using the bits of the accumulated value to fill in these gaps.

The general construction of the MAC operation can be presented by this equation

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

where, the multiplier A and multiplicand B are assumed to have n bits each and the addend Z has (2n+1) bits. The basic MAC Unit is made up of a multiplier and an accumulator as shown in Figure 1.5.

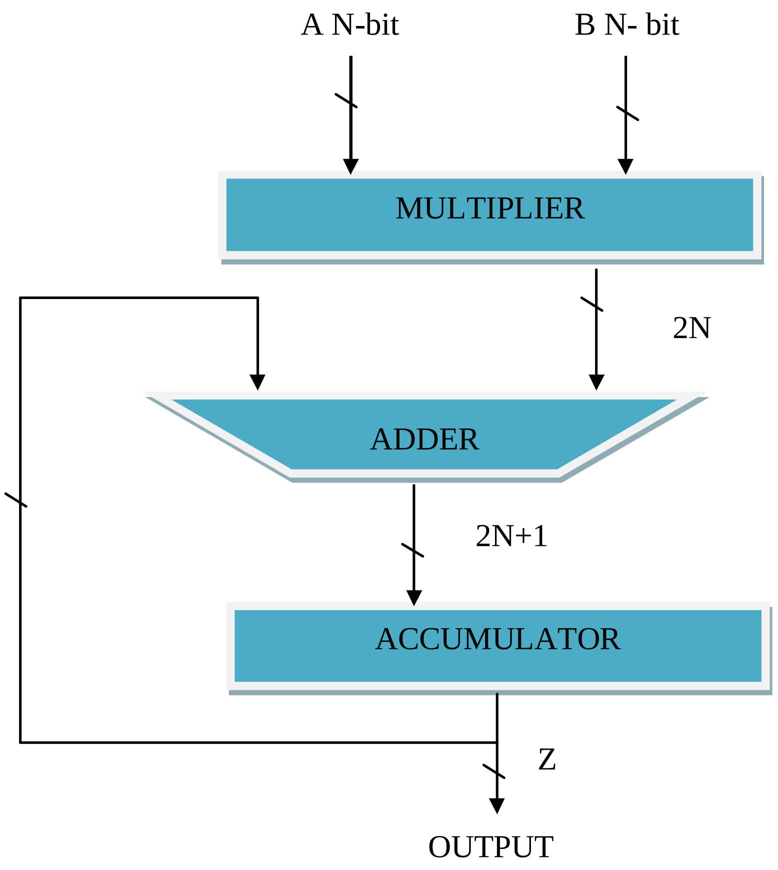

Figure 1.5 MAC unit

The multiplier can also be divided into the partial products generator, summation tree, and final adder. This construction leads to four basic blocks to implement. The summation network represents the core of the MAC unit. This block occupies most of the area and consumes most of the circuit power and delay. Several algorithms and architectures are developed in attempt to optimize the implementation of this block. It executes the multiplication operation by multiplying the input multiplier and the multiplicand. This is added to the previous multiplication result as the accumulation step. The basic MAC operation comprises of a multiplication which can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand and the multiplier. The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. A MAC consists of four steps, as shown in Figure 1.6, which shows the operational steps explicitly.

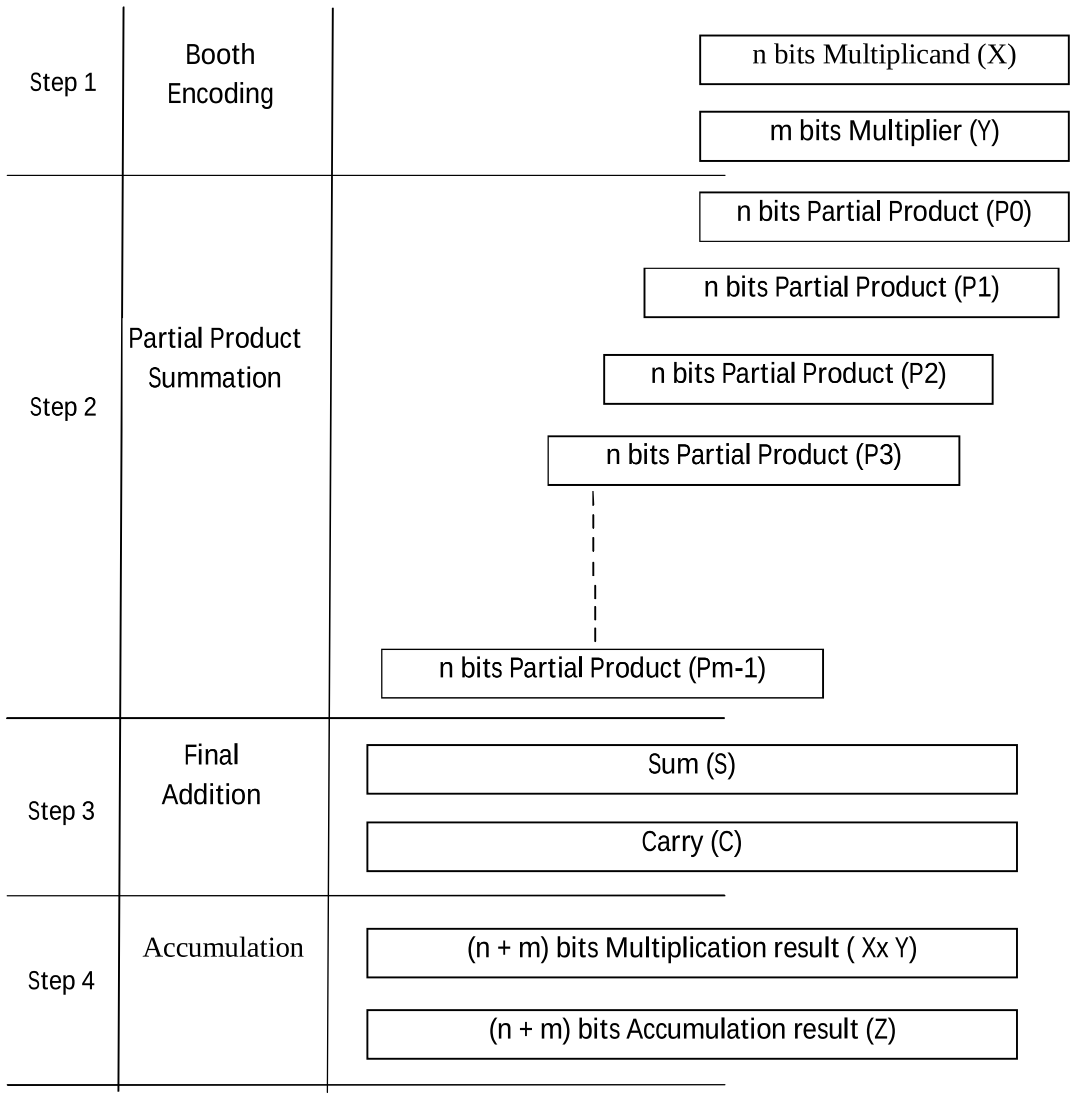
Step 1
Booth Encoding
n bits Multiplicand (X)
m bits Multiplier (Y)
Step 2
Partial Product Summation
n bits Partial Product (P0)
n bits Partial Product (P1)
n bits Partial Product (P2)
n bits Partial Product (P3)
n bits Partial Product (Pm-1)
Step 3
Final Addition
Sum (S)
Carry (C)
Step 4
Accumulation
(n + m) bits Multiplication result (Xx Y)
(n + m) bits Accumulation result (Z)

Figure 1.6 Basic steps for deriving MAC unit

The N-bit 2's complement binary number A can be expressed as-

$$A = -2^{N} - a_{N-1} - \sum_{i=0}^{N-3} a_i 2^i \ : \quad a_i \, \varepsilon 0,1 \qquad (1.6)$$

If Equation 1.6 is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm, it would be.

$$A = \sum_{i=0}^{\frac{N}{2}-1} d_i \, 4^i \qquad (1.7)$$

$$d_i = -2a_{2i+1} + a_{2i} + a_{2i-1} \qquad (1.8)$$

If Equation 1.7 is used, multiplication can be expressed as

$$A \times B = \sum_{i=0}^{\frac{N}{2}-1} d_i 2^{2i} B \qquad (1.9)$$

If these equations are used, therefore mentioned multiplication–accumulation results can be expressed as

$$P = A \times B + Z = \sum_{i=0}^{\frac{N}{2}-1} d_i 2^i B + \sum_{j=0}^{2N-1} z_j 2^j \qquad (1.10)$$

Each of the two terms on the right-hand side of Equation 1.10 is calculated independently and the final result is produced by adding the two results. The MAC architecture implemented by Equation 1.10 is called the standard design as discussed by Abdelgawad & Bayoumi (2007). If N-bit data is multiplied, the number of the generated partial products is proportional to. In order to add them serially, the execution time is also proportional to N. The architecture of a multiplier, which is the fastest, uses radix-2 Booth encoding that generates partial products and a Wallace tree based on CSA as the adder array to add the partial products. If radix-2 Booth encoding is used, the number of partial products, i.e., the inputs to the Wallace tree, is reduced to half, resulting in the decrease in CSA tree step. In addition, the signed multiplication based on 2's complement numbers is also possible. Due to these reasons, most current used multipliers adopt the Booth encoding.

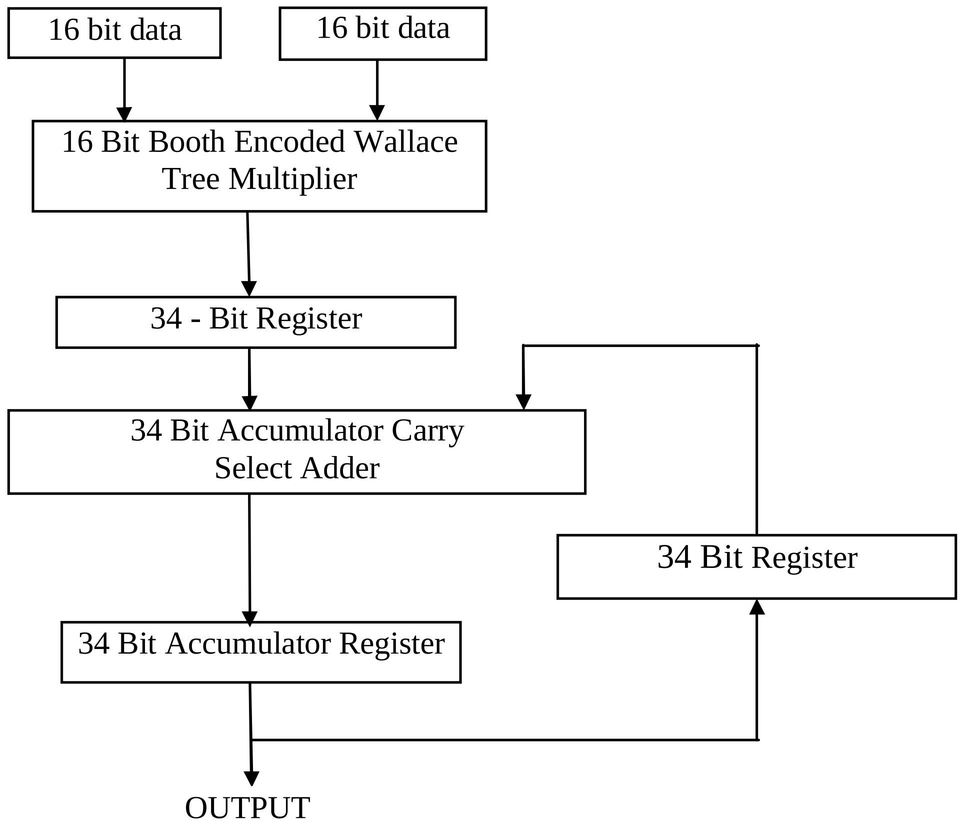
16 bit data
16 bit data
16 Bit Booth Encoded Wallace Tree Multiplier
34 - Bit Register
34 Bit Accumulator Carry Select Adder
34 Bit Register
34 Bit Accumulator Register
OUTPUT

Figure 1.7 MAC unit architecture

Figure 1.7 shows the MAC unit architecture. The inputs for the MAC are fetched from memory location and fed to multiplier block of the MAC, which will perform multiplication and gives the result to adder which will accumulate the result and then will store the result into a memory location. This entire process is to be achieved in a single clock cycle. The design of MAC unit architecture in Figure1.7 shows that the design consists of one 17 bit register, one 8-bit Wallace tree multiplier, 17-bit accumulator using carry look ahead adder (CLA) and two 18-bit accumulator register are used. In this project, Wallace tree multiplier and carry look adder are used for high performance MAC unit design. Wallace tree multiplier is used to multiply the values of A and B. CLA are used in accumulator and carry save adder (CSA) used in the final stage of the given multiplier for reducing power consumption of the MAC unit. The product of Ai X Bi is always fed back into the 17-bit CLA in accumulator and added again with the next product Ai X Bi. This MAC unit is capable of multiplying and adding with the previous product consecutively up to as many as eight times. Output = ΣAi X Bi.

A multiplier design consists of three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand X and the multiplier Y. The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry.

1.5 TECHNICAL GAP

The main motivation of this work is to investigate various multiplier and adder architectures for MAC unit which are suitable for implementing high throughput signal processing algorithms and at the same time achieve low power consumption. A regular MAC unit consists of multipliers and accumulators that contain the sum of the previous consecutive products. Currently more multipliers and adders are available; from the available adders and multipliers in this thesis the adders and multipliers are analyzed and compared. From the comparison a adder and multiplier is chosen and MAC unit is designed. Then the proposed MAC unit is implemented in FIR Filter.

In any multiplication algorithm, the operation is decomposed in three steps. Each partial product shows a multiple of the multiplicand to be summed to the final result. Currently almost all high-speed multipliers use a modified Booth multiplication algorithm. All DSP algorithms would require some form of the Multiplication and Accumulation Operation. This is the most important block in DSP systems which are composed of an adder, multiplier and the accumulator. Usually adders implemented in DSPs are Ripple Carry Adders, Carry-Select or Carry-Save adders, as speed is of utmost importance in a DSP. Basically the multiplier will multiply the inputs and give the results to the adder, which will add the multiplier results to the previously accumulated results. This operation eases the computation of the most important formula i.e., b(n).x(n-k) which is needed in filters, Fourier analyses, etc. The inputs for the MAC are supposed to be fetched from some memory location and fed to the multiplier block of the MAC, which will perform multiplication and give the result to adder which will accumulate the result and then if needed will also store the result into a memory location. This entire process is to be achieved in a single clock cycle.

1.6 ROADMAP OF THE THESIS

The rest of the thesis is organized as follows.

Chapter 2 forms a review of the wide Literature survey and scope of the present study,which deals with the existing method for performance of Low Power MAC Unit.

Chapter 3 offers a detailed study of different adders and Simulation and Analysis of various adders for MAC Unit.

Chapter 4 endow with an exhaustive study of various multipliers and simulation of various multipliers used for MAC unit. .

Chapter 5 discusses the performance of MAC unit designed with Low power adder and multiplier and Pipelining for proposed MAC unit. Then the MAC with FIR filters is discussed.

Finally, a summary of the thesis, its significance and the alternate directions for further extensions are addressed in Chapter 6.