Energy Efficient Multiplier for High Speed DSP Application

Sumant Mukherjee¹, Dushyant Kumar Soni²

¹Dr. C.V.Raman University, Department of Engineering, Kota, Bilaspur,(C.G.)
²Dr. C.V.Raman University, Department of Engineering, Kota, Bilaspur,(C.G.)

Abstract: A multiplier is one of the important hardware blocks in most digital and high performance systems such as microprocessors. With improving in technology, many researchers have tried and are trying to design multipliers which offer high speed, low power consumption and less area. However area and speed are two most important constraints. In this paper we propose Energy Efficient approximate multiplier using AHA and AFA. Due to this area reduces upto 35% and carry propagation delay also reduces.

Keywords :- Approximate half Adder(AHA), Approximate full adder(AFA), Approximate Multiplier, MAC unit, SPAA(Speed, Power, Area, Accuracy), LUT(Look up Table)

I. Introduction

Multipliers are one the most important component of many systems. In high speed digital signal processing (DSP) and image processing multiplier play a vital role. Multipliers and adders are the key element of the arithmetic units as they lie in the critical path. With the recent advances in technology, many researchers have tried to implement increasingly efficient multiplier. They aim at offering low power consumption, high speed and reduced delay. Digital signal Processing (DSP) is finding its way into more applications [19], and its popularity has materialized into a number of commercial processors [18]. 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. Basic operation found in MAC is the
binary addition. Besides of the simple addition of two numbers, addition forms the basis for many processing operations, from counting to multiplication to filtering. But also simpler operations like incrimination and magnitude comparison based on binary addition. Therefore, binary addition is the most important arithmetic operation. It comparison based on binary addition. It is also a very critical one if implemented in hardware because it involves an expensive carry-propagation step, the evaluation time of which is dependent on the operand word length.

II. Literature Review

In this part we begin with the basic building blocks used for addition and multiplication, then go through different algorithms.

A. Basic Adder blocks

1. Half Adder

The Half Adder (HA) is a combinational circuit with two binary input and two binary outputs such as sum and carryout. The equation (1) and (2) are the Boolean equations for sum and carryout, respectively.

\[
\text{sum} = a \oplus b \\
\text{carryout} = a \land b
\]  

(1)  
(2)

2. Full Adder

The Full Adder (FA) is a combinational circuit that adds two bits and a carry and outputs a sum bit and a carry bit. Equation (3) , (4) and (5) are the Boolean equations for the full adder sum and full adder carryout, respectively. In both those equations cin means carryin.

\[
\text{sum} = a \oplus b \oplus \text{cin} \\
\text{carryout} = a \land b + b \land \text{cin} + a \land \text{cin} \\
\text{cin} = a \land b + (a + b) \land \text{cin}
\]  

(3)  
(4)  
(5)

From the above equations we see that sum and carryout is depends on carry in.
B. Basic Multiplication Schemes

Multiplication hardware often consumes much time and area compared to other arithmetic operations. Digital signal processors use a multiplier/MAC unit as a basic building block [5] and the algorithms they run are often multiply-intensive. A multiplication operation can be broken down into two steps:

1) Generate the partial products.
2) Accumulate (add) the partial products.

1. Array Multiplier

Each multiplicand is multiplied by a bit in the multiplier, generating N partial products. Each of these partial products is either the multiplicand shifted by some amount, or 0. The generation of partial products consists of simple AND'ing of the multiplier and the multiplicand.

2. Tree Multiplier

The tree multiplier reduces the time for the accumulation of partial products by adding all of them in parallel, whereas the array multiplier adds each partial product in series. The tree multiplier commonly uses CSAs to accumulate the partial products.

2.1 Wallace Tree

The reduction of partial products using full adders as carry-save adders (also called 3:2 counters) became generally known as the "Wallace Tree" [14]. Figure shows an example of tree reduction for an 8*8-bit partial product tree.
3. Baugh–Wooley Algorithm

The Baugh–Wooley Multiplication Algorithm is an efficient way to handle the sign bits. This technique has been developed in order to design regular multipliers suited for 2’s complement numbers. The Baugh-Wooley (BW) algorithm is a relatively straightforward way of doing signed multiplications.

4. Vedic Multiplication

Vedic mathematics is part of four Vedas (books of wisdom) of Indian culture. The Vedic multiplier is based on the Vedic multiplication formulae (Sutras). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system.

4.1 Urdhva–Triyagbhyam (Vertically & Crosswise)

Urdhva tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means “Vertically and Crosswise”.

From the adder architecture we understand that the carry propagation is the main issue. In the ripple carry adder the carry out of each stage is connected to the carryin of the next stage. The sum and carryout bits of any stage cannot be produced, until some time after the carryin of that stage occurs. The time for this implementation of the adder is expressed in below Equation, where $t_{RCA\text{carry}}$ is the delay for the carryout of a FA and $t_{RCA\text{sum}}$ is the delay for the sum of a FA.

\[
\text{Propagation Delay (tＲCAprop)} = (N - 1) \cdot t_{RCA\text{carry}} + t_{RCA\text{sum}}
\]
In the multiplier, after partial product we again have to add that partial product by using adders. So if we want to speed up MAC unit we have to minimize carry propagation delay.

IV. Proposed Architecture of 8 Bit approximate adder

Here we proposed a new architecture of half adder and full adder as we know for 8 bit addition there is total 7 full adder and 1 half adder is require. But in proposed approach we propose a new novel 8 bit architecture where we can put some error on lsb bit of adder. Here in approximate half and full adder there is no any carry generation unit. So on first LSB bit we are using proposed approximate half adder and on second LSB bit we use one approximate full adder for next third bit there is no any carry generate so there is no need to use one full adder so at the place of full adder we are using one half adder and after that we use 5 full adder. So as we can see with small error generation we can reduce the hardware requirement and we can make justice with SPAA matrices.

Figure 6 Critical Path for an N-bit Ripple Carry Adder

Figure 7 Proposed Approximate Half Adder

Figure 8 Proposed Approximate Full Adder
In this section we present the proposed 8-bit approximate multiplier. This multiplier is a combination of accurate and approximate 4-bit multiplier. For generation of this multiplier we are using the divide and concrete approach in which we design one 4-bit approximate multiplier where we are using normal multiplication approach but at the time of final addition we are using our own approximate half and full adder logic. Due to this approach there is reduction in hardware structure of 4-bit multiplier.
VI. Result & Hardware Analysis

Approximate Multiplier Accuracy Level = 87%

The FPGA comparison analysis of proposed and accurate are shown below, here hardware analysis is done on Vertix 6 FPGA which is 45nm based technology.
From the above graphs we can see that 35% reduction in logic block is achieved.
VII. Conclusion

In this paper, the implementation and analysis of a Approximate multiplier architecture is proposed. The comparison result also shows that a significant reduction in the area is achieved. The overall area, delay and frequency analysis are presented and compared. From the results we can depict that approximately up to 25% to 35% of reduction at all levels are achieved. So due to this we use approximation, which will minimize delay. The results obtained prove that the proposed architecture is more efficient than the conventional one in terms of area and delay. This design is particularly useful in computation-intensive applications which are robust to small errors in computation. The potential applications of this approximate Multiplier fall mainly in areas where there is no strict requirement on accuracy or where super-low power consumption and high speed performance are more important than the accuracy. One example of such applications is in the DSP application for portable devices such as cell phones and laptops.

References