IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 2, Ver. I (Mar-Apr. 2014), PP 29-35
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Analysis of Different Multiplication Algorithms & FPGA Implementation

K.Harika[1], B.V.Swetha2, B.Renuka[3], D.Lakshman Rao[4], S.Sridhar[5]
[1, 2, 3, 4, 5]*(Electronics & Communication, Lendi Institute of Engineering & Technology, India)*

Abstract: *As the scale of integration keeps growing, more and more sophisticated signal processing systems are being implemented on a VLSI chip. These signal processing applications not only demand great computation capacity but also consume considerable amounts of energy. While performance and area remain to be two major design goals, power consumption has become a critical concern in today's VLSI system design. Multiplication is a fundamental operation in most arithmetic computing systems. Multipliers have large area, long latency and consume considerable power. Multiplication is a basic arithmetic operation which is present in any part of the digital computer especially in signal processing systems. Different techniques are used for multiplication. Some of the techniques are CSA, CSD, Booth's, Grid, Lattice, Combinational, Sequential, Array, Vedic, Wallace-tree etc.*

Keywords: *Multiplier, VHDL, FPGA*

I. Introduction

Power is a problem primarily when cooling is a concern. The maximum power at any time, peak power, is often used for power and ground wiring design, signal noise margin and reliability analysis. Energy per operation or task is a better metric of the energy efficiency of a system, especially in the domain of maximizing battery lifetime. In digital CMOS design, the well-known power-delay product is commonly used to assess the merits of designs.[1]

Generally multiplication consists of three steps: generation of partial products or PPs (PPG), reduction of partial products (PPR), and final carry-propagate addition (CPA). Different multiplication algorithms vary in the approaches of PPG, PPR, and CA. For PPG, radix-2 digit-vector multiplication is the simplest form because the digit-vector multiplication is produced by a set of AND gates. To reduce the number of PPs and consequently reduce the area/delay of PP reduction, one operand is usually recoded into high-radix digit sets. The most popular one is the radix-4 digit set $\{-2, -1, 0, 1, 2\}$. For PPR, two alternatives exist: reduction by rows, performed by an array of adders, and reduction by columns, performed by an array of counters. In reduction by rows, there are two extreme classes: linear array and tree array. Linear array has the delay of O (n) while both tree array and column reduction have the delay of O (log n), where n is the number of PPs. The final CPA requires a fast adder scheme because it is on the critical path. Some low-level techniques that has been studied for multipliers include using voltage scaling, layout optimization, transistor reordering and sizing, using pass-transistor logic and swing limited logic, signal polarity optimization, delay balancing and input synchronization. However, these techniques have only achieved moderate improvement on power consumption in multipliers with much design effort or considerable area/delay overhead. The difficulty of low-power multiplier design lies in three aspects. [1]

This paper is outlined as follows: section 2 clearly elaborates all the multiplication techniques proposed in this work in section 3 results of the experimental analysis is thoroughly discussed followed by conclusion in section 4.

II. Different Multiplication Techniques

Multiplication is basically a shift add operation. There are, however, many variations on how to do it. Some are more suitable for FPGA use than others; some of them may be efficient for a system like CPU. This section explores various verities and attracting features of multiplication hardware. The multiplier area is quadratically related to the operand precision. Second, parallel multipliers have many logic levels that introduce spurious transitions or glitches. Third, the structure of parallel multipliers could be very complex in order to achieve high speed, which deteriorates the efficiency of layout and circuit level optimization. As a fundamental arithmetic operation, multiplication has many algorithm-level and bit-level computation features in which it differs from random logic. These features have not been considered well in low-level power optimization. It is also difficult to consider input data characteristics at low levels. Therefore, it is desirable to develop algorithm and architecture level power optimization techniques. We have designed many multipliers and compared all of them that are given below [1].

2.1. Shift-And-Add Multiplier

Shift-and-add multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand *X* to itself *Y* times, where *Y* denotes the multiplier. To multiply two numbers by paper and pencil, the algorithms is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

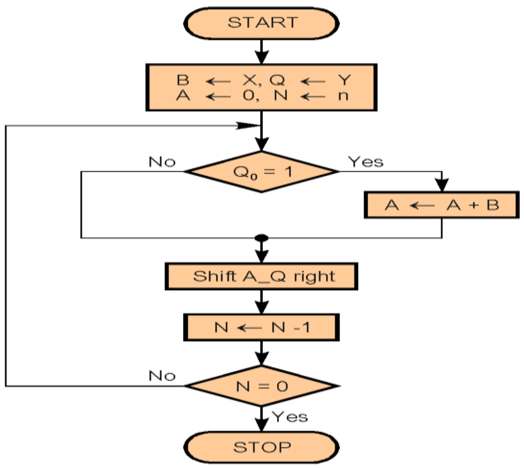

Fig.1.Flowchart for Shift & Add

2.2. Array Multiplier

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added.

The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length

2.2.1. Advantages

First advantage of the array multiplier is that it has a regular structure. Since it is regular, it is easy to layout and has a small size. . A second advantage of the array multiplier is its ease of design for a pipelined architecture.

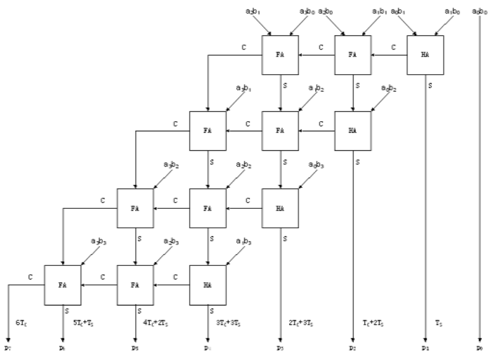

Fig.2.Architecture of 3x3 array multiplier [2]

2.2.2. Limitations

Major limitation of array multiplier is its size. As operand sizes increase, arrays grow in size at a rate equal to the square of the operand size.

2.3. Carry Save Adder

Carry save adder is used to compute sum of three or more n-bit binary numbers. Carry save adder is same as a full adder. A carry-save adder is a type of digital adder, used in computer micro architecture. It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence

of partial sum bits and another which is a sequence of carry bits. The idea of delaying carry resolution until the end, or saving carries, is due to John von Neumann. [3]
If the adder is required to add two numbers and produce a result, carry-save addition is useless, since the result still has to be converted back into binary and this still means that carries have to propagate from right to left. But in large-integer arithmetic, addition is a very rare operation, and adders are mostly used to accumulate partial sums in a multiplication. [3]

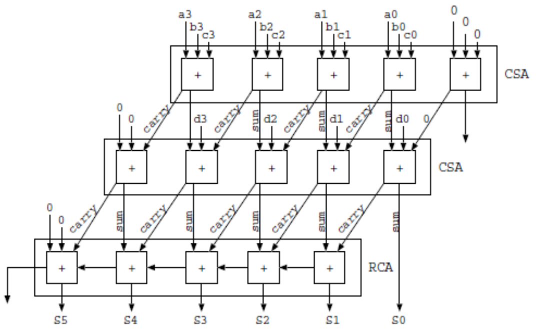

Fig.3. Architecture of Carry save Adder [3]

2.3.1 Drawbacks
At each stage of a carry-save addition,

1. We know the result of the addition at once.
2. We still do not know whether the result of the addition is larger or smaller than a given number (for instance, we do not know whether it is positive or negative)

2.4. Booth Multiplier

Booth‘s Algorithm is a smart move for multiplying signed numbers. It initiate with the ability to both add and subtract there are multiple ways to compute a product. Booth‘s algorithm is a multiplication algorithm that utilizes two‘s complement notation of signed binary numbers for multiplication. [4]
More specific, the following table explains in detail:

Table 1

Bi	Bi-1	Operation
0	0	DO Nothing
0	1	Add a
1	0	Subtract a
1	1	Do nothing

2.5. Modified Booth’s Multiplier

A simplified proof of a modification of Booth's multiplication algorithm by MacSorley to a form which examines three multiplier bits at a time is presented. In comparison with the original Booth’s algorithm, which examines two bits at a time, the modified algorithm requires half the number of iterations at the cost of somewhat increased complexity for each iteration.[5]

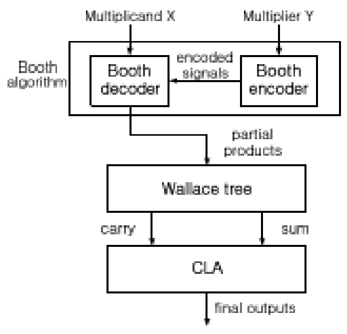

Fig.5.4. Block diagram for modified Booth Multiplier: [5]

III. Experimental Analysis

3.1. Shift And Add Multiplier

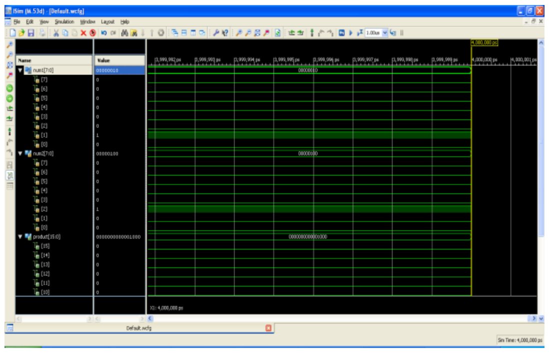

Fig 3.1.Simulation Waveform for shift & add:

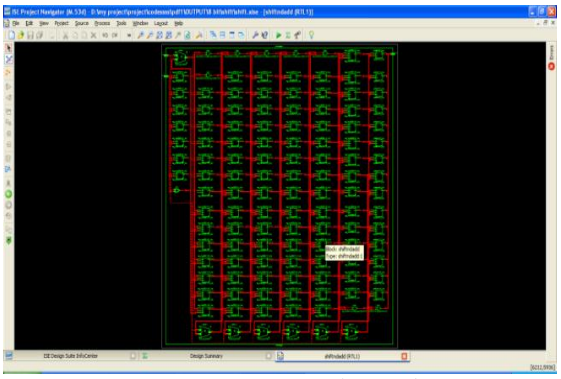

Fig3.2.Synthesized Circuit for Shift & Add :

3.2. Array Multiplier :

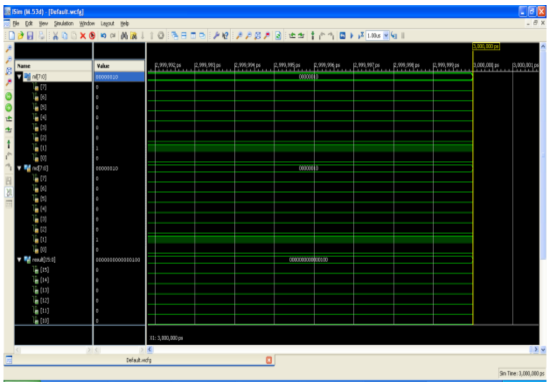

Fig 3.3.Simulation Waveform for Array Multiplier

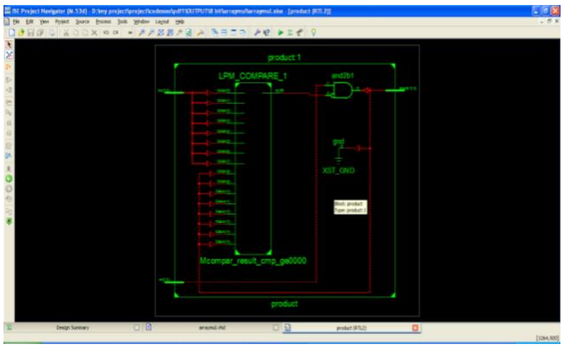

Fig 3.4.Synthesized Circuit for Array Multiplier:

3.3. Carry Save Adder:

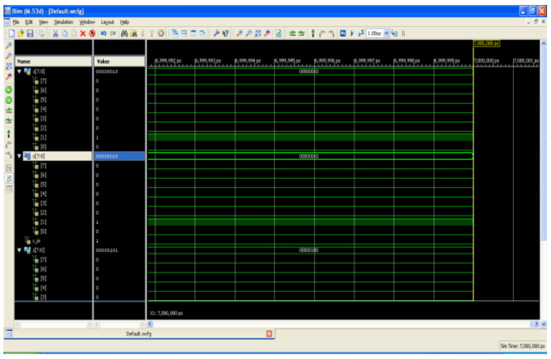
Fig3.5.Simulation Waveform for Carry save Adder:

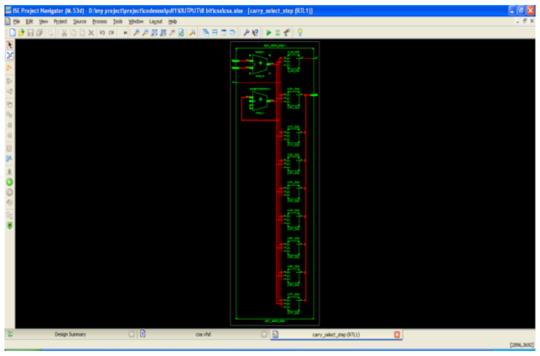
Fig.3.6.Synthesized Circuit for Carry Save Adder:

3.4. Booth's Multiplier:

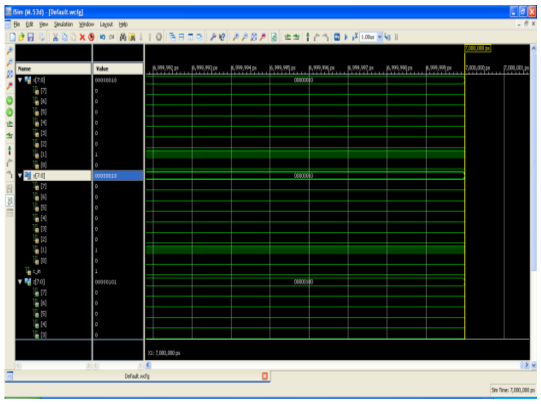
Fig 3.7.Simulation Waveform for Booth Multiplier:

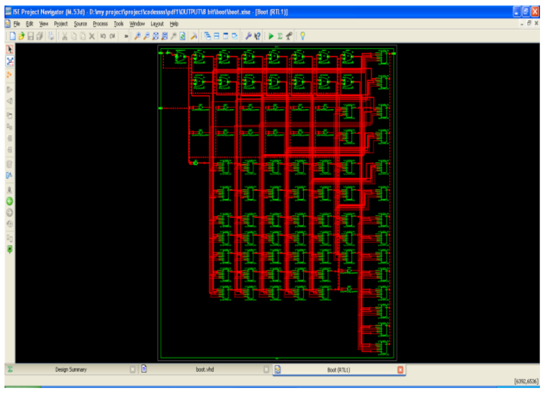
Fig 3.8.Synthesized Circuit for Booth Multiplier

3.5. ModifiedBooth’s Multiplier*:*

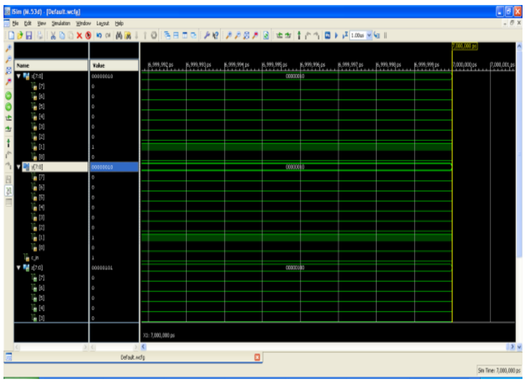
Fig 3.9.Simulation Waveform for Modified Booth Multiplier:

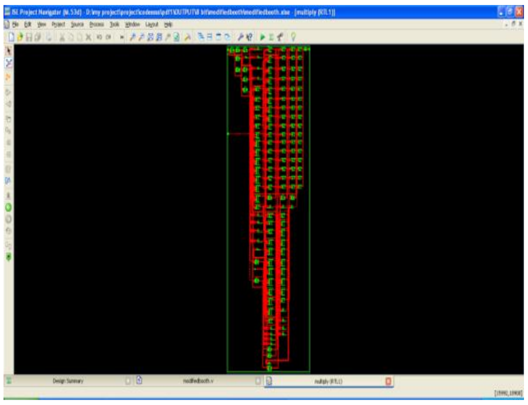
Fig 3.10.Synthesized Circuit for modified Booth Multiplier

Comparision Analysis Clearly Specifying The Time Taken For Execution And Hardware Consumed Are Explained In Table 2 :

Table 2

Technique Name	Booth multiplier	Shift add multiplier	Array multiplier	Csa Multiplier	Modified booth multiplier
Macro cells used	120/144	103/144	17/36	23/36	106/144
Pterms used	640/720	473/720	9/180	113/180	501/720
Registers used	0/144	0/144	0/36	0/36	16/144
Pins used	16/0	32/0	32/34	26/34	33/0

IV. Conclusion:

These days speed of the multiplier has become an asset or constraint due to the importance of multiplier circuit in a wide variety of microelectronic systems. In this paper we analysed different multiplier techniques taking speed as the main criteria. Carry save adder is proved to be more efficient in terms of speed compared to conventional multiplication techniques generated the output in 2.06 sec, whereas the booth multiplier generated the output in 3.09sec while the shift &add multiplier produced the output in 2.31 sec. However the array multiplier generated the output in 2.75sec and modified booth multiplier in around 3.08 sec.The carry save adder on the other hand consumes less hardware than other multiplication techniques.

References:

[1] GarimaTiwari “Analysis, Verification and FPGA Implementation of Low Power Multiplier”.

[2] Kripa Mathew, S.AshaLatha, T.Ravi, E.Logashanmugam “design and analysis of an Array Multiplier using an Area Efficient full adder cell in 32 nm CMOS Technology”.

[3] ChakibAlaoui “Design and Simulation of a Modified Architecture of Carrysave Adder”.

[4] DeepaliChandel,GaganKumawat, PranayLahoty, VidhiVartChandrodaya, Shailendra Sharma.International Journal of Emerging Technology and Advanced Engineering Volume 3, Issue 3, March 2013”Booth Multiplier: Ease of multiplication”.

[5] International Journal of Engineering Science InventionShaik.Kalisha Baba, D.Rajaramesh “Design and Implementation of Advanced Modified Booth Encoding Multiplier”.

[6] G.W. Bewick, “Fast Multiplication: Algorithms and Implementation.“Ph.D. dissertation, Stanford University, Feb. 1994

[7] Shiann-RongKuang, Jiun-Ping Wang, and Cang-Yuan Guo, “Modified Booth multipliers with a Regular Partial Product Array,” IEEE Transactions on circuits and systems-II, vol 56, No 5, May 2009.

[8] 8.M. Zamin Ali Khan1, Hussain Saleem2, Shiraz Afzal3 and Jawed Naseem4, ― An Efficient 16-Bit Multiplier based on Booth Algorithm, international Journal of Advancements in Research & Technology, Volume 1, Issue 6, November-2012 ISSN 2278-7763

[9] Dr. Ravi Shankar Mishra,Prof. PuranGour,BrajBihariSoni, ―Design and Implements of Booth and Robertson‗s multipliers algorithm on FPGA.‖ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622**.**

[10] F.C Cheng, S. H. Unger, "*Self-Timed Carry-Look Ahead Adders*", IEEE Transactions on Computers, Vol. 49, No. 7, July 2000