International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 6, Nov-Dec-2020, ISSN (Online): 2395-566X

Review of GRPF Strengthen Column

M. Tech. Scholar Farooq Khan, Asst. Prof. Ashwin Hardiya
Department of Elctronics& Communication Engineering
Patel College of Science and Technology, Indore
Khanfarooq371@gmail.com

***Abstract*- Multiplication in hardware can be implemented in two ways either by using more hardware for achieving fast execution or by using less hardware and end up with slow execution. The area and speed of the multiplier is an important issue, increment in speed results in large area consumption and vice versa. Multipliers play vital role in most of the high performance systems. Performances of a system depend to a great extent on the performance of multiplier thus multipliers should be fast and consume less area and hardware. This idea forced us to study and review about the Booth's Algorithm, modified Booth's algorithm and its radix-2, radix-4, radix-8 forms.**

***Keywords*- Booth's Algorithm, Modified Booth's Algorithm, multiplication, multipliers, radix-2, Radix-4, radix-8.**

I. INTRODUCTION

It is a very challenging problem for the hardware designers to develop low power, high speed and area efficient portable electronic design. Mobile phones, smart cards such as hearing aids and PDAs are the example of portable consumer electronic products. It is the main concern for operating hours of the battery and residing in it but also greater computational capacity. At the circuit level voltage scaling, threshold voltage, Transistor sizing, network restructuring power down strategies and logic style are used to achieve low power. In addition to this, this technique also contributes to the reduction of propagation delay and area occupancy as well

.

Digital Signal Processors (DSPs) are used to perform the most common operations such as video processing, filtering and fast flourier transform (FFT). Such modules perform an extensive sequence of multiply and accumulate computations. Multiplication is the most fundamental operation of digital computer systems and digital signal processors.

Multiplication consists of three steps: generation of partial products or (PPG), reduction of partial products (PPR), and finally carry-propagate addition (CPA).In general there are sequential and combinational multiplier implementations. We only consider combinational case here because the scale of integration now is large enough to accept parallel multiplier implementations in digital VLSI systems. Different multiplication algorithms vary in the approaches of PPG, PPR, and CPA. For PPG, radix-2 is the easiest. 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. The final CPA requires a fast adder scheme because it is on the critical path. In some cases, final CPA is postponed if it is advantageous to keep redundant results from PPG for further arithmetic operations[1-2-3].

A large number of transistors with high switching transitions is used to perform a variety of multiplication operations. In 64 point radix-4 pipelined FFT processor the multiplier consumes 30% power and also occupies 46% chip area. Multiplier is most critical, power hungry arithmetic unit that requires more area and Computational time. Array based multipliers consumes low power as compared to Wallace tree multipliers.

In order to improve the performance in tree based multiplier the additional hardware is required, but at the cost of increased layout and parasitic. On the other side, array multiplier has smaller and regular layout. Therefore, array multiplier is a better choice due to its optimized with lesser hardware as small area leads to less switching transitions. An Adder is the fundamental unit of the multiplier and it has significant impact on the overall performance of the system for power dissipation, delay and area occupancy. In this paper, array multiplier is proposed to achieve low power and high speed multiplication operation[4].

II. BOOTH ALGORITHM

Signed multiplication is a vigilant process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Even though in signed multiplication the same procedure cannot be applied forthe reason that the signed number is in a 2„s compliment form which would give in an inaccurate result if multiplied in an analogous manner to unsigned multiplication [5-6].

Thus here Booth's algorithm comes in. Booth,„s algorithm conserves the sign of the end result. While doing multiplication, strings of 0s in the multiplier call for only shifting. While doing multiplication, strings of 1s in the multiplier need an operation only at each end. The multiplier blocks require intensive computations. There are three major steps to any multiplication. In the first step, the partial products are generated. In the second step, the partial products are reduced to one row of final sums and carries. In the third step, the final sums and carries are added to generate the result. A modified booth multiplier should concentrate on the following things.

On reducing the total number of partial products generated. This may include any coding methods or reduction of computation complexity of generation of partial products.

- A significant amount of delay is consumed in finding two‟s complement of multiplicand. So this delay should be reduced.
- The optimization of adder structure. Once partial products generated, they have to be grouped and added in a systematic manner consuming less delay. This may consider the use of parallelism of the process.

Next is focus is on the method included in adding the two operands; the carry propagation should be treated efficiently. Finally for the hardware implementation, suitable hardware descriptive language should be chosen and the code should be well optimized, synthesized and simulated using the optimum tool.

The main focus of recent multiplier papers has been on rapidly reducing the number of partial product rows by using some kind of circuit optimization and identifying the critical paths and signal races and usage of different adder structures to reduce the delay.[7-8-9]

In this paper, we discuss a new approach for 2‟scomplementation and modified carry look ahead adder using Ling‟s equation and their implementation on FPGA chip. In the next section, the booth 2 algorithm and encoding are described in detail. In section 3, modifications done to the booth multiplier is explained. Section 4 presents FPGA implementation. Finally in section 5 Result analyses has been done.

Booth2 algorithm multiplier- generator that creates a smaller number of partial products will allow the partial product summation to be efficient and use less hardware. The simple multiplication generator can be extended to reduce the number of partial products by grouping the bits of the multiplier into pairs, and selecting the partial products from the set of 0, M, 2M or their complements, where M is the multiplicand. This reduces the number of partial products, by a factor two but also generates some extra-bits for the sign extension and the 2‟scomplementation. [10, 11].

All partial products set can be produced using simple shifting and complementing. The multiplier is partitioned into overlapping groups of 3bits, and each group is decoded to select a single partial product as per the selection table 3.1 shown below. Each partial product is shifted 2 bit positions with respect to its neighbors. The number of partial products has been reduced to half of total number of multiplier bits. In general there will be n/2 products, where n is the operand length. The multiply by 2 can be obtained by a simple left shift of the multiplicand and negative of number obtained from its two‟s complement form. Following table shows booth encoding table. According to that partial products are generated and added to get final result.

Table 1 booth encoding table.

Bits of operand	Selection
000	0
001	+ Multiplicand
010	+ Multiplicand
011	+ 2 * multiplicand
100	- 2 * multiplicand
101	- Multiplicand
110	- Multiplicand
111	0

III. LANGUAGE AND TOOLS USED

We used XILINX ISE v 10.2 for our programming. We considered VHDL as our primary language. For test bench waveforms also we also used Xilinx to write our own test benches. Model Synthesis Map report all features in Xilinx helped us a lot. We used Xilinx‟s X Power Estimator (XPE) tool in order to calculate power consumed in any arithmetic circuit.

For calculation of power using Xilinx‟s XPE we need to generate the map report file in XILINX which will be saved in the same directory with an extension ".mrp". But in the later part of the project we used SYNOPSIS tool for finding out Power and delay and Area calculations.

IV. IMPLEMENTATION

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).

- A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
- S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
- P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.

2. Determine the two least significant (rightmost) bits of P.

- If they are 01, find the value of P + A. Ignore any overflow.
- If they are 10, find the value of P + S. Ignore any overflow.
- If they are 00, do nothing. Use P directly in the next step.
- If they are 11, do nothing. Use P directly in the next step.

3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.

4. Repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. this is the product of m and r.

Flowchart Diagram

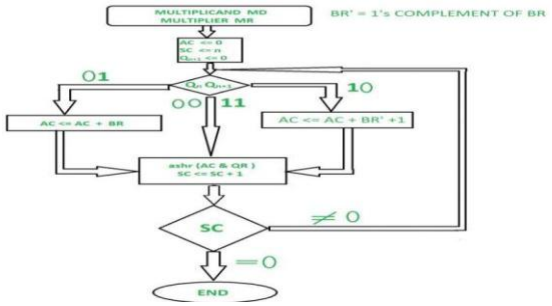

Fig.1 Algorithm flow chart

.

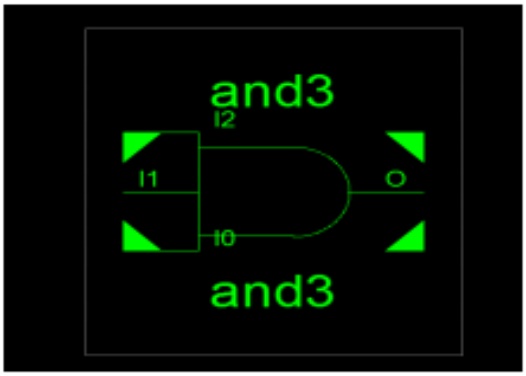

Fig. 2 RTL logic Diagram.

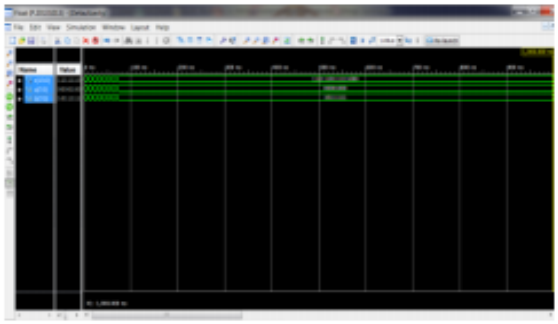

Fig.3 Outcome in Xilinx

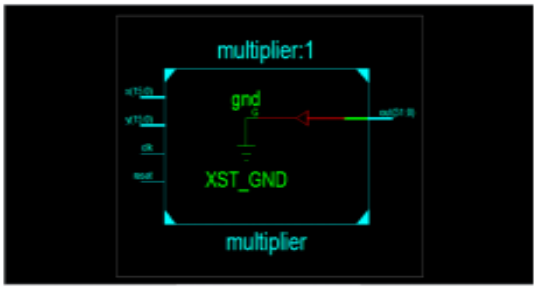

Fig. 4 I/O of multi

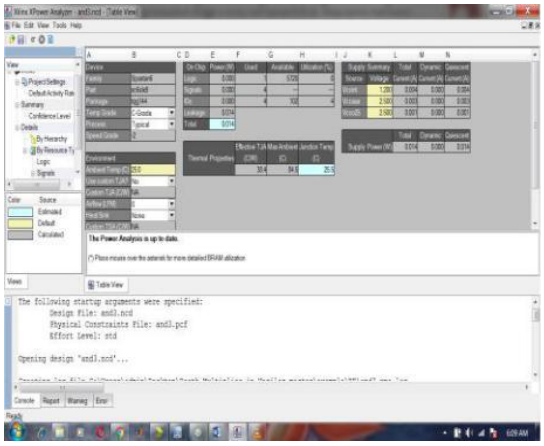

Fig. 5 Power GUI

Data Path: i2 to o Cell:in->out (Net Name)	fanout	Gate Delay	Net Delay	Logical Name
IBUF:I->O (i2_IBUF)	1	1.328	0.910	i2_IBUF
LUT3:I0->O (o_OBUF)	1	0.235	0.681	and2_1/o1
OBUF:I->O		2.912		o_OBUF (o)
Total route)		6.066ns (4.475ns logic, 1.591ns		
		(73.8% logic, 26.2% route)		

V. CONCLUSION AND FUTURE WORK

After going through all the hard work and after facing a lot of problems, we managed to complete the objectives of the project that are to implement Booth‟s Algorithm for the design of a binary multiplier using different adder architectures and carry out power analysis at various levels.. We analyzed the area occupied and the time delay consumed by different adders and found out an appropriate relationship among the time and area complexity the adders which we have taken into consideration.

After comparing all we came to a conclusion that Ripple Carry Adders are best suited for Low Power Applications. Then we turned our focus into the design of Multipliers. First of all we designed a Booth's Radix-2 multiplier and estimated its delay, area and power

.

Then a radix-4 multiplier was designed. A comparison was done between Radix-2 and Radix-4 algorithm. Comparing data between Radix-2 and Radix-4 booth multipliers we found out that radix-4 consumes less power than radix-2, because radix-4 uses almost a half number of iterations than radix-2 As radix-4 seemed more suitable for the design we carried out further analysis on radix-4 multiplier by using different adder architectures like RCA and CLA.

Then we turned our focus into the switching activity based power analysis of the Radix-4 Booth multiplier, and its power estimation. We did power estimation at RTL level using Synopsys Design Compiler. Further work can be carried out on this project in the power estimation section. Power can be estimated at the gate-level by generating gate-level netlist and also the post layout analysis can be done for this design. Another possible direction can be pursued for higher radix encoding.

REFERENCES

[1] S.-W. Lee and I.-C. Park, "Low-power hybrid structure of digital matched filters for direct sequence spread spectrum systems," in International Conference on Multimedia and Expo, ICME, 2003.
[2] N. Srisakthi, C. Rao and M. Vidya, "Implementation of code synchronization for CDMA applications using recursive digital matched filters," in International Conference on Communication and Signal Processing (ICCSP), 2014.
[3] O. M. V.E. Bychkov and V. Pravda, "Modern Digital Matched Filters and Correlators for Active Radar," in International Conference on Telecommunications, and Computer Science (TCSET) ,Lviv, Ukraine, 2006. [4] Z. Deng, Y. Yu, D. Zou, W. Guan and L. yang, "Optimization and implementation of digital matched filters based on FPGA," in IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), 2011.
[4] P. Woodward, "Probability and information theory with application to radar.," in Pregamon Press, 1953. [6] G. Turin, "An introduction to matched filters.," in IRE Transaction on Information Theory, 1960. [7] A. Kumar and C. V. R. Rao, "Low-power structures of Digital Matched Filter for direct sequence spread spectrum," in International Conference on Wireless Communication, Signal Processing and Networking (WiSPNET), 2016
[5] G. Kumar and S. Sahoo, "Implementation of high speed multipliers for high performance and low power applications," in VLSI Design and Test (VDAT), Ahmadabad, 2015.
[6] Yeh and W.-C. a. C.-W. Jen, "High speed Booth encoded parallel multiplier design," in IEEE Transactions on Computers, 2000.
[7] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 2000.
[8] B. R and P. E, "Design of high speed multiplier using Modified Booth Algorithm with hybrid carry lookahead adder," in International Conference on Circuit, Power and Computing Technologies, 2016.