Implementation and performance evaluation of Synchronous and Asynchronous Modified Booth Multipliers on Artix 7 FPGA

Balaram Murty. S[1], Syamala. M.[2], Srinivas Sabbavarapu[3] and Sadhana. K [4]

[1,2,3]*Anil Neerukonda Institute of Technology and Sciences, Sangivalasa, Visakhapatnam, A. P., India 531162*
{[1]balaramamurty.ece, [2]syamala.ece, [3]ssrinivas.ece, [4] sadhana.14.ece}
@ anits.edu.in

Abstract-- While performance and Area remain to be the two major design tolls, power consumption has also become a critical concern in today's VLSI system design. Multiplication is a fundamental operation in most of the signal processing algorithms which occupies major large area long latency and consume considerable power. Therefore, low-power multiplier design has been an important part in low power VLSI system design. Driving the system with a single clock in modern System on Chip designs degrade the performance of a system that includes multipliers. To improve the speed of a multipliers using asynchronous design techniques has the great impact on the overall speed of a system. In this paper we implemented different multipliers using asynchronous techniques on Artix 7 FPGA which can be integrated to different Signal Processing Modules.

Keywords: Very Large Scale Integration (VLSI), Low Power, Asynchronous design, Booth Multiplier, Field Programmable Gate Array (FPGA).

I. Introduction

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 amount of energy. While performance and Area remain to be the two major design tolls, power consumption has also become a critical concern in today's VLSI system design. Multiplication is a fundamental operation in most of the signal processing algorithms which occupies major large area long latency and consume considerable power. Therefore, low-power multiplier design has been an important part in low power VLSI system design. There has been an extensive work on low-power multipliers at technology, physical, circuit and logic levels. A system's performance is generally determined by the performance of the multiplier as it the slowest element in a system with large area. Hence, optimizing the speed and area of the multiplier is a major design issue. Driving the system with a single clock in modern System on Chip designs degrade the performance of a system that includes multipliers. To improve the speed of a multipliers using asynchronous design techniques has the great impact on the overall speed of a system. Hence, in this paper we implement evaluate different multipliers on *Artix* 7 FPGA. Many works [1]– [10] have been focusing on reducing spurious switching during computation to lower the dynamic power in particular. Power optimization techniques employed at high level in [1]. Multiplier generator proposed in [2] by parallel processing. Further, [3] used Wallace tree algorithm and parallel processing to improve the performance in terms of speed. Spurious switching eliminated in the multiplications process [4] to reduce the power further. Different logic level improvements were proposed [5][6]. Furthermore, [7] used clock-less approach for improving speed and lowering power requirements. Asynchronous logic [7] is often used in low power design due to its time-independent characteristic, which avoids the unwanted switching. Meanwhile, as CMOS process is scaled down to deep sub-micron dimensions, the leakage power starts to dominate in the whole power consumption budget. In this paper, we implanted the asynchronous multipliers on Artix FPGA and showcase the advantages of asynchronous multipliers.

II. LOW-POWER MULTIPLIER DESIGN

A. *Basic principle*

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 digitset {-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.

B. BOOTH'S MULTIPLIER

Though Wallace Tree multipliers were faster than the traditional Carry Save Method, it also was very irregular and hence was complicated while drawing the Layouts. Slowly when multiplier bits gets beyond 32-bits large numbers of logic gates are required and hence also more interconnecting wires which makes chip design large and slows down operating speed.

Booth multiplier can be used in different modes such as radix-2, radix-4, radix-8 etc. But we decided to use Radix-4 Booth's Algorithm because of number of Partial products is reduced to n/2 [1].

One of the solutions realizing high speed multipliers is to enhance parallelism which helps in decreasing the number of subsequent calculation stages. The Original version of Booth's multiplier (Radix – 2) had two drawbacks.

- The number of add / subtract operations became variable and hence became inconvenient while designing Parallel multipliers.
- The Algorithm becomes inefficient when there are isolated 1s .

These problems are overcome by using Radix 4 Booth's Algorithm which can scan strings of three bits with the algorithm given below. The design of Booth's multiplier consists of four Modified Booth Encoded (MBE), four sign extension corrector, four partial product generators (comprises of 5:1 multiplexer) and finally a Wallace Tree Adder. This Booth multiplier technique is to increase speed by reducing the number of partial products by half. Since an 8-bit booth multiplier is used in this project, so there are only four partial products that need to be added instead of eight partial products generated using conventional multiplier. The architecture design for the modified Booths Algorithm used in this project is shown below.

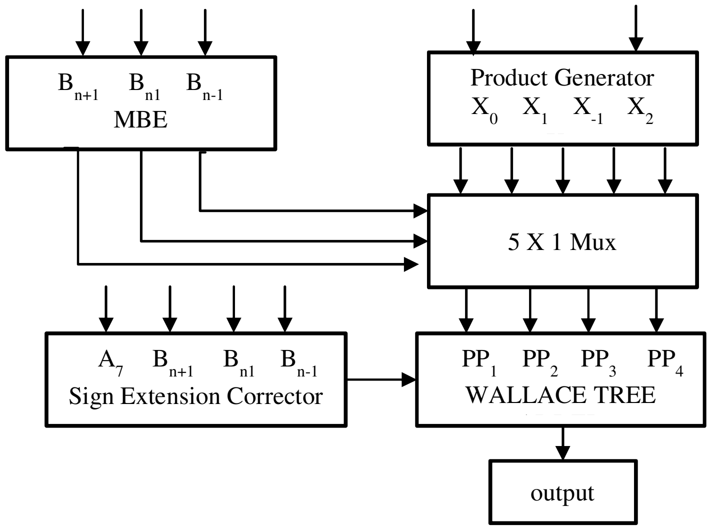

B_{n+1} B_{n1} B_{n-1}
MBE
Product Generator
X_0 X_1 X_{-1} X_2
5 X 1 Mux
A_7 B_{n+1} B_{n1} B_{n-1}
Sign Extension Corrector
PP_1 PP_2 PP_3 PP_4
WALLACE TREE
output

Fig 1. BOOTH multiplier Architecture

B. 1 MODIFIED BOOTH ENCODER (MBE)

Modified Booth encoding is most often used to avoid variable size partial product arrays. Before designing a MBE, the multiplier B has to be converted into a Radix-4 number by dividing them into three digits respectively according to Booth Encoder Table given afterwards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier. The figure above shows that the multiplier has been divided into four partitions and hence that mean four partial products will be generated using booth multiplier approach instead of eight partial products being generated using conventional multiplier.

$$z_n = -2 * B_n + B_{n+1} + B_{n-1} \qquad (1)$$

Let's take an example of converting an 8-bit number into a Radix-4 number.

Let the number be **-36 = 1 1 0 1 1 1 0 0**.

Now we have to append a '**0**' to the LSB. Hence the new number becomes a 9-digit number, that is

1 1 0 1 1 1 0 0 0. This is now further encoded into Radix-4 numbers according to the following given table. Starting from right we have **0*Multiplicand, -1*Multiplicand, 2*Multiplicand, -1*Multiplicand.**

Z_n is representing the Radix-4 number of the3-bit binary multiplier number. For example, if the 3-bit multiplier value is "111", so it means that multiplicand A will be 0.And it's the same for others either to multiply the multiplicand by -1, -2 and so on depending on 3 digit number.

And thing to note is generated numbers are all of 9-bit. From the Table 1, the **M, 2M** and **3M** are the elect control signals for the partial product generator. It will determine whether the multiplicand is multiplied by 0,-1, 2 or -2. **M** and **2M** are designed as an active low circuit which means if let's say the multiplicand should be multiplied by 1 then the **M** select signal will be set to low "0" whereas If the multiplicand should be multiplied by 2 then the **2M** select signal will be set to low "0". The **3M** is representing the sign bit controlsignal and its active high circuit which means if the multiplicand should be multiplied by -1 or -2, then the sign, **3M** will be set to high "1".

TABLE I
MODIFIED BOOTH ENCODER'S TABLE TO GENERATE M, 2M, 3M CONTROL SIGNAL

B_{n+1}	B_n	B_{n-1}	Z_n	Partial Product	1M	2M	3M
0	0	0	0	0	1	1	0
0	0	1	1	1×Multiplicand	0	1	0
0	1	0	1	1×Multiplicand	0	1	0
0	1	1	2	2×Multiplicand	1	0	0
1	0	0	-2	-2×Multiplicand	1	0	1
1	0	1	-1	-1×Multiplicand	0	1	1
1	1	0	-1	-1×Multiplicand	0	1	1
1	1	1	0	0	1	1	0

B.2 PARTIAL PRODUCT GENERATOR (PPG):

Partial product generator is the combination circuit of the product generator and the 5 to 1 MUX circuit. Product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2 or -2. A 5 to 1 MUX is designed to determine which product is chosen depending on the M, 2M, 3M control signal which is generated from the MBE. For product generator, multiply by zero means the multiplicand is multiplied by "0". Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place.

B.3 SIGN EXTENSION CORRECTOR

Sign Extension Corrector is designed to enhance the ability of the booth multiplier to multiply not only the unsigned number but as well as the signed number. As shown in Table 2 when bit 7 of the multiplicand A(A7) is zero (unsigned number) and B_{n+1} is equal to one, then sign E will have one value (become signed number for resulted partial product). It is the same when the bit 7 of the multiplicand A (A7) is one (signed number) and B_{n+1} is equal to zero, the sign E will have a new value. However, when both the value of A7 and B_{n+1} are equal either to zero or one, the sign E will have a value zero (unsigned number). For the case when all three bits of the multiplier value B_{n+1}, B_n and B_{n-1} are equal to zero or one, the sign E will direct have a zero value independent to the A7 value. The table for the Sign Extension Corrector is shown below.

TABLE 2

(A) SIGN E WHEN A IS ZERO

A7	B_{n+1}	B_n	B_{n-1}	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0

(B) SIGN E WHEN A IS ONE

A7	B_{n+1}	B_n	B_{n-1}	E
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

C. ASYNCHRONOUS BOOTH MULTIPLIER

The asynchronous logic offers a more flexible power-management strategy than clocked logic because an asynchronous system only performs processing on demand [8]. We can close the data path, to avoid unnecessary switching, as soon as they finished processing using asynchronous latch controllers. This fine grain control helps in saving dynamic power consumption. For clocked circuits, arithmetic functions always have a fixed operation time independent of their inputs. Otherwise, it is very difficult for the circuit to achieve pipelined operation. On the other hand, an asynchronous logic block co-operates with other asynchronous blocks by sending mutual communication through *request* and *grant* signals. Asynchronous circuits achieve input-dependent computation by their nature, which means the computation time and energy consumption no the same for all the operations but varies with number of inputs and complexity of the operands. for simple operands they take a short computation time and consume little energy whereas for complex operands they take a long computation time and consume more energy.

Analyzing pipeline give necessary clarifications on the advantage of power in asynchronous multiplier. Figure 2 illustrates the different pipeline operations of the asynchronous multiplier and a synchronous multiplier under the conditions of (a) 4 normal cycles and (b) 1 normal cycle. Without early termination as shown in Figure 2(a), the synchronous multiplier has some unnecessary switching activity in the first and last cycles: In the first cycle, only pipeline stage1 needs to do useful work, while stage2 should be idle and wait for the data from stage1. In the last cycle, pipeline stage1 has finished its work, and only stage2 needs to calculate the final result. However, because in the synchronous multiplier both pipeline register and shift register are connected to the global clock signal, both stages operate simultaneously, thus causing some unnecessary switching activity. With 3 early termination cycles, the synchronous multiplier wastes even more energy in unnecessary activity as shown in Figure 2(b). In conclusion, asynchronous design has a more flexible power-management strategy and will stop activity in some or all of the system whenever its operation is unneeded. It is difficult for synchronous logic to achieve this goal because all the subsystems are driven by the same clock signal. The power dissipation only depends on how much work needs to be done.

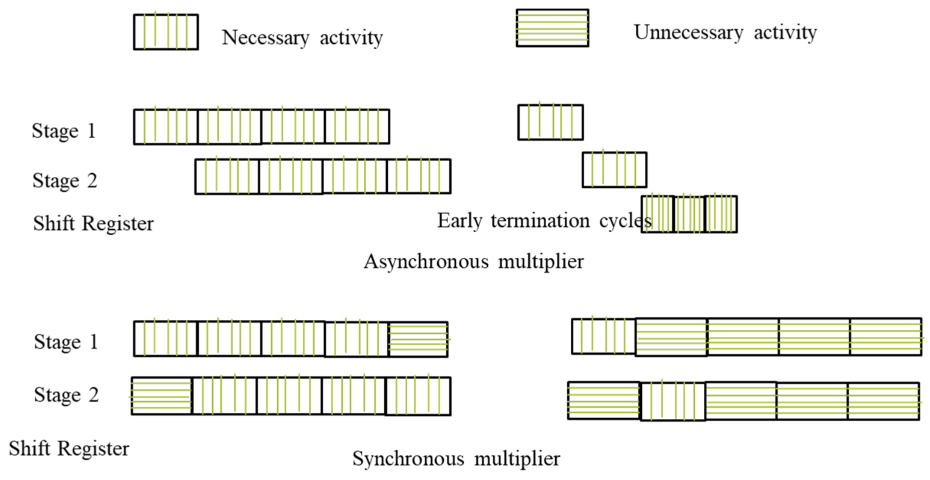


Fig 2. Pipelining

III IMPLEMENTATION DETAILS

We used XILINX VIVADO 16.3 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. We used Xilinx's XPower Estimator (XPE) tool in order to calculate power consumed in any arithmetic circuit.

IV. Results and Analysis

The following figure , Fig 3, gives the output of FPGA implementation for the inputs 5 and 5. Result is displayed through leds.

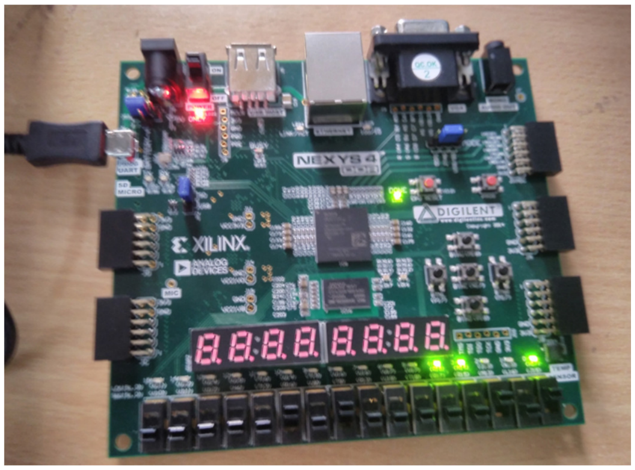
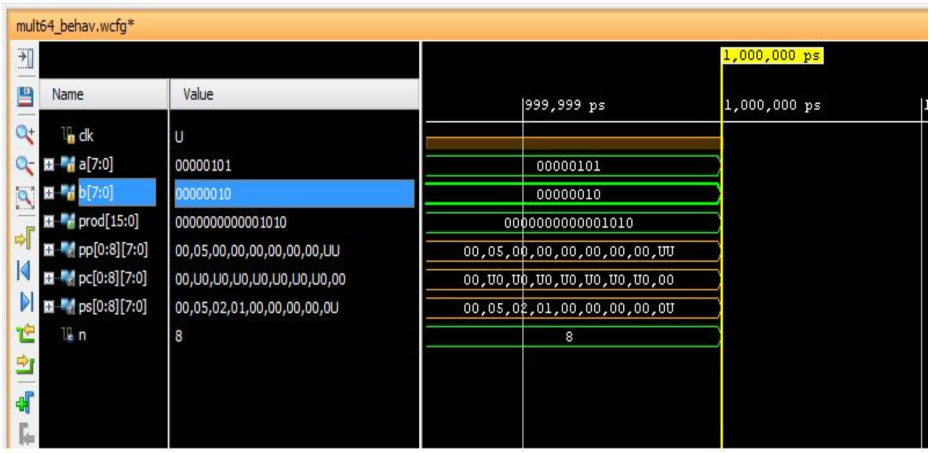


Fig 3. Asynchronous multiplier design.

Fig 4 shows the simulation output for the data 5 and 2.



```
Name            Value
clk             U
a[7:0]          00000101
b[7:0]          00000010
prod[15:0]      0000000000001010
pp[0:8][7:0]    00,05,00,00,00,00,00,00,UU
pc[0:8][7:0]    00,U0,U0,U0,U0,U0,U0,U0,00
ps[0:8][7:0]    00,05,02,01,00,00,00,00,0U
n               8
```

Fig 4. Asynchronous multiplier simulation

The following tables give the synthesis report for synchronous and asynchronous booth multipliers. It is evident from the tables that asynchronous design has more number of latches but improved on speed and power consumption.

TABLE 3
SYNTHESIS REPORT FOR SYNCHRONOUS BOOTH MULTIPLIER

PARAMETERS	**VALUES**
Number Of Slices	**96**
Number Of 4-input LUTs	**178**
Number Of Bonded Input	**32**
Number Of Bonded Output	**32**
Power	**79mW**
Delay	**26.645 ns**

TABLE 4
SYNTHESIS REPORT FOR ASYNCHRONOUS BOOTH MULTIPLIER

PARAMETERS	**VALUES**
Number Of Slices	**110**
Number Of 4-input LUTs	**178**
Number Of Bonded Input	**32**
Number Of Bonded Output	**32**
Power	**41mW**
Delay	**14.45 ns**

V. CONCLUSIONS

The multiplier is one of the most complex design for any signal processing applications. By exploiting the advantages of Asynchronous techniques over synchronous, it is evident the synchronous multipliers got the advantage of speed and low power consumption. Using the asynchronous implementation, almost 45 % improvement in speed and 48% reduction in power consumption is achieved.

REFERENCES

[1] *Z. Huang, "High-Level Optimization Techniques for Low-Power Multiplier Design," PhD dissertation, Univ. of California, Los Angeles, June 2003*

[2] *K.H. Tsoi, P.H.W. Leong, "Mullet - a parallel multiplier generator", pp.691-694, International Conference on Field Programmable Logic and Applications, 2015., 2015*

[3] *M. 0. Lakshmanan, AlauddinMohd Ali, "High Performance Parallel Multiplier Using Wallace-Booth Algorithm," IEEE International Conference on Semiconductor Electronics, pp. 433-436, 2002.*

[4] *Kwen-Siong Chong, Bah-Hwee Gwee, Chang, J.S, "A micro low voltage multiplier with reduced spurious switching," IEEE Transaction on Very Large Scale Integration Systems, vol.13, no.2, pp. 255-265, Feb.2005.*

[5] *M. Mottaghi-Dastjerdi, A. Afzali-Kusha, M.Pedram, "BZ-FAD:A Low- Power Low-Area Multiplier Based on Shift-and-Add Architecture," IEEE Transaction on Very Large Scale Integration Systems,vol.17, no.2, pp. 302-306, Feb.2009.*

[6] *T.Raja, V.D. Agrawal, M.L. Bushnell, "Variable Input Delay CMOS Logic for Low Power Design," IEEE Transaction on Very Large Scale Integration Systems, vol.17, no.10, pp. 1534-1545, Oct..2009.*

[7] *K. Fant, "Logically Determined Design: Clockless System Design with NULL Convertion Logic." Wiley-Interscience, 2005.*

[8] *S. B. Furber, A. Efthymiou, J.D. Garside, M.J.G. Lewis, D.W. Lloyd and S. Temple, "Power Management in the AMULET Microprocessors", IEEE Design and Test of Computers Journal special issue pp. 42-52 (Ed. E. Macii), March-April 2001.*

[9] *Kuhn, K.J. "Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS," Electron Devieces Meeting, 2007, IEDM 2007, IEEE International, 12/2007.*

[10] *C.S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Comput.,vol. 13, no.2, pp.14-17, Feb.1964.*

[11] *V.G. Oklobdzija, D.Villeger, S.S. Liu, "A method for speed optimized patial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput.,vol.45,no.3,pp.294-306, Mar.1996.*

[12] *B. Parhami, Computer Arithmetic, Algorithms, and Hardware Designs. New York: Oxford University. Press,2000.*

[13] *Zhijun Huang, Ercegovac, M.D, "High-performance low-power left-toright array multiplier design," IEEE Transaction on Computers, Vol.54, no.3, pp.272-283, March.2005.*

[14] *Jin-Tai Yan, Zhi-Wei Chen, "Low-cost low-power bypassing-based multiplier," Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp.2338-2341, June.2010*

[15] *S. TahmasbiOskuii, P. G. Kjeldsberg, and O. Gustafsson, "Transition activity aware design of reduction-stages for parallel multipliers," in Proc. 17th Great Lakes Symp. On VLSI, March 2012, pp. 120–125.*

[16] *Ayman A. Fayed, Magdy A. Bayoumi, "A Novel Architecture for Low-Power Design of Parallel Multipliers," wvlsi, pp.0149, IEEE Computer Society Workshop on VLSI 2011, 2011.*