Design of Low Power Approximate Radix-8 Booth Multiplier

K. Sindhuja
Department of Electronics and Communication Engineering
SSN College of Engineering
Chennai, India

C. Thiruvenkatesan
Department of Electronics and Communication Engineering
SSN College of Engineering
Chennai, India

***Abstract*---Booth multiplier has been used for high speed signed multiplication. This multiplication is achieved through encoding and reducing the number of partial products generation stage (PPG). The partial products are easily produced by using Radix-4 (Modified Booth) algorithm in the Booth multiplier. Radix-8 algorithm also produces partial products but is slow due to the generation of odd multiples of the multiplicand. This problem could be overcome by some of the application of approximate designs. An approximate 2-bit adder is designed for calculating the sum of 1× and 2× of a binary numbers. This approximate adder requires small area, low power and short critical path delay. Two signed 16 bit and 32 bit approximate radix-8 Booth multipliers are designed using this approximate recoding adder with and without truncation of least significant bit in the partial products. This proposed approximate multiplier will be faster and power efficient than accurate Booth multiplier. This algorithm is implemented using Verilog coding and the synthesis is done with Xilinx software.**

***Keywords*---Radix-8; approximate adder; Partial product generation (PPG); approximate multiplier; FSM**

I. INTRODUCTION

Multipliers were introduced to perform multiplication operation of the arithmetic circuits by using add and shift operation. Generally, a system performance is determined by the performance of the multiplier because the multiplier is considered as the slowest element in the whole system and also it is area consuming. Multiplication is frequently required in digital signal processing. Parallel multipliers provide a high-speed method for multiplication, but require large area for VLSI implementations. Many DSP applications are based on Add-Multiply operations which were designed by adding the bits and giving its output as an input to the multiplier. This increases the area and delay of the circuit. In order to reduce the power consumption of multiplier, the low power Booth recoding methodology is implemented by recoding technique. This Booth decoder will increase number of zeros in multiplicand. Modified Booth algorithm has made multiplication easier. It consists of recoding table which has been used to minimize the partial products of multiplier.

An adder and the multiplier operator of the unit combined to form a single add-multiply unit. In ALU, only add/subtract/shift operations are possible. Multiplication involves two basic operations. They are generation of partial products and their accumulation. There are two ways for speeding up this multiplication which are reducing the number of partial products and accelerating the accumulation. For consecutive 0's and 1's in Booth's algorithm only few partial products are generated. Booth multiplier performs both signed and unsigned operation.

II. PROPOSED SYSTEM

A. *Approximate multiplier*

Approximate multipliers are considered by using the speculative adders to compute the sum of partial products. However, the straightforward application of approximate adders in a multiplier may not be efficient; there exists a trading off accuracy for savings in energy and area. For an approximate multiplier, the main key design aspect is reducing the critical path by adding the partial products. Since multiplication is usually implemented by a cascaded array of adders, some of the least significant bits in the partial products are simply omitted (with some error compensation mechanisms) and also some adders can be removed in array for a faster operation.

The approximate multipliers are classified into four categories i) Approximation in Generating Partial Products. ii) Using simpler structure to generate partial products. iii) Approximation in the Partial Product Tree. iv) Omitting some partial products. Dividing partial products into several sections and applying approximation in the less significant sections.

B. *Algorithm for approximate multiplier*

The algorithm for approximate multiplier is shown in fig.1. First, the input operands are divided into two parts i) accurate part and ii) approximate part. Left part containing the most significant bits is the accurate part and the right part containing least significant bits are called the approximate part. Since least significant bits contribute less when compared to most significant bits, for most-significant bits accurate multiplication is applied. The length of each part need not necessary be equal.

The multiplication process starts from the starting point in two opposite directions simultaneously as shown in fig.1, the two 8-bit input operands, the multiplicand "11100110"(230) and multiplier "00110010"(56) are divided into two equal-sized parts , each part containing 4 input bits. As for the least significant bits of input operands (approximate part), a special mechanism is applied. The carry generation part which is responsible for more power consumption is removed

and partial products are not generated. Every bit position from left to right is checked and if both input bits are 1 or one of two input bit is 1, the corresponding product term is 1 and from this bit onwards all the product bits of right side are 1, if both the input bits are 0, the corresponding product bit is 0 and it has no effect on the next right side bits. Fig.1 shows the operation of approximate multiplier.

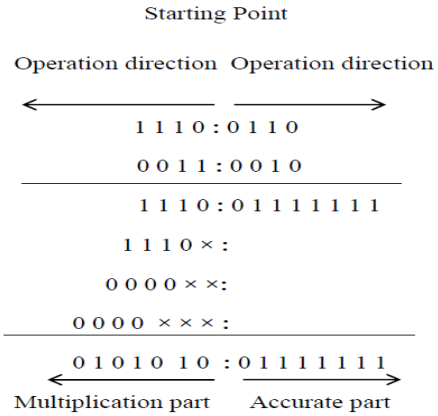

Fig 1: Operation of approximate multiplier

For the most significant bits, input operands fall in to accurate part; the operation is conducted as per the normal multiplication operation from right to left.

C. *Architecture of Approximate Multiplier*

In the proposed design as show in fig.2, the input A and B are divided into two 4-bit blocks each. The control block contains two, 4-input NOR gates. In the first NOR gate input bits A7-A4 inputs are applied and on the second NOR gate input bits B7-B4 inputs are applied and the outputs of these two NOR gates are applied to the input of NAND gate. The control block is first used for detecting the logic "1" in the MSB position of the inputs, (A7-A4) and (B7-B4). When logic "1" is found, the "ctrl" signal will be activated and the input operands are high enough to operate in: (i) approximate part to give the lower order bits of the final output (P7-P0) and (ii) accurate part to generate the higher order bits of the product (P15-P8). If no logic “1” is detected by the control block, all the most significant bits are 0, the multiplexer selects accurate part to generate the lower order bits of the product (P7-P0). In the accurate part, the standard 4-bit parallel multiplier is used to produce higher order output product terms (P15-P8). The approximate part is designed using APGs.

D. *Approximate recoding adder*

Booth multiplication is a technique which allows smaller and faster multiplication in circuits, by recoding the numbers that are multiplied. This standard technique used in chip designing and provides significant improvements over the "long multiplication" technique.

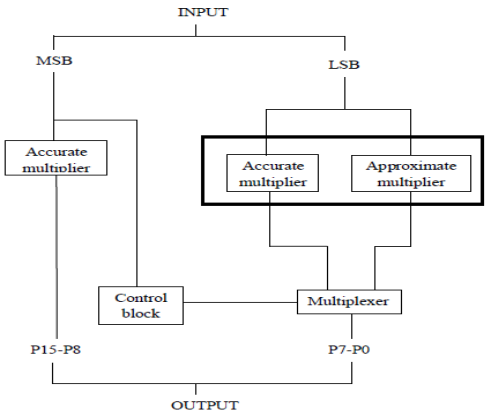

Fig 2: Architecture of approximate multiplier

A standard approach that might be taken by a new conversion method to perform multiplication is to "shift and add", or normal "long multiplication". That is, for each column in the multiplier, the multiplicand is shifted to the appropriate number of columns and multiplied by the value of digit in that column of the multiplier, to obtain a partial product. The partial products are then added to obtain the final result:

0 0 1 0 1 1
0 1 0 0 1 1
0 0 1 0 1 1
0 0 1 0 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 1 0 1 1
0 0 1 1 0 1 0 0 0 1

With this system, the number of partial products is exactly the number of columns in the multiplier.

Approximate designs of a Booth multiplier are proposed. It is based on an approximation scheme which deals not only with the partial product accumulation, but also with the generation of recoded multiplicands. A 2-bit approximate recoding adder is initially designed to reduce the additional delay encountered in previous radix-8 schemes, thereby increasing the speed of the radix-8 Booth algorithm. The partial products in the radix-2 and radix-4 algorithms can be easily generated by shifting or 2’s complementing; 2’s complementing is implemented by inverting each bit and then adding a ‘1’ in the partial product accumulation stage. A preliminary addition is required to calculate 3Y by implementing Y + 2Y which incurs additional delay and power cost. Therefore, a high speed approximate recoding adder is designed for performing Y + 2Y in this section.

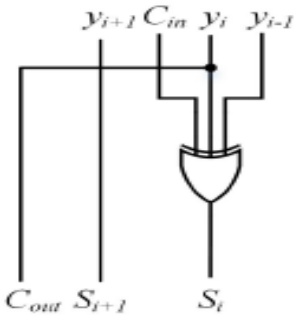

Fig 3: Proposed approximate 2-bit adder

E. *Partial Product Generator*

A product is formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has more number of bits. Partial products are used for calculating larger products and is used in intermediate steps.Radix-8 Booth recoding applies the same algorithm as that of Radix-4, but here quartets of bits are taken instead of triplets. Radix-8 algorithm reduces the number of partial products to n/3, where n is the number of multiplier bits. A time gain in the partial products summation is allowed**.**

Table: 1 Radix-8 Booth multiplier

Multiplier bits	**Recoded operations to be performed on multiplicands**
0000	0
0001	1× Multiplicand
0010	1× Multiplicand
0011	2× Multiplicand
0100	2× Multiplicand
0101	3× Multiplicand
0110	3× Multiplicand
0111	4× Multiplicand
1000	-4× Multiplicand
1001	-3× Multiplicand
1010	-3× Multiplicand
1011	-2× Multiplicand
1100	-2× Multiplicand
1101	-1× Multiplicand
1110	-1× Multiplicand
1111	0

III. RADIX-8 MULTIPLIER

Booth multiplier is known as radix-8 because it perform the 8 different types of operations on the multiplicand that are +Y, +2Y, +3Y, +4Y, –4Y, –3Y, –2Y and –Y where Y denotes the Multiplicand. All multiples are easily obtainable, by simply shifting and complementing except 3Y. The generation of the 3Y (3× multiplicand), cannot be obtained by simple shifting and complementation because it is the hard multiple to generate. It can be produce Y+2Y or 4Y– Y. Here in this project, it is produced by Y+2Y. For example, the 8×8 bit multiplication generates the 8 partial product rows by using simple multiplier , but by radix-8 Booth multiplier, the partial products are reduced to 3 which means that radix-8 Booth multiplier reduces the partial product rows by N/3 where "N" in number of bits in multiplier.

IV. SIMULATION RESULTS

The RTL Schematic and the simulated results of 16-bit Booth multiplier are shown in fig. 4 & 5. Here, the Booth algorithm is implemented using FSM controller. The shifting and addition is performed in the same cycle.FSM controller contains four states. They are wait for go state, initial state, add shift state, done state. In the first state, inputs are checked and if they are ready initial state is activated and if not ready waits till the inputs are given. The second state concentrates on the addition of bit to the LSB of first partial product and padding it to the MSB of the first partial product. If those are satisfied, the add shift state is activated or will remain in the same state. Add shift state is responsible for adding the products and shifting bits. If they are satisfied, the next state is activated. The last state (done state) produces the final partial product term.

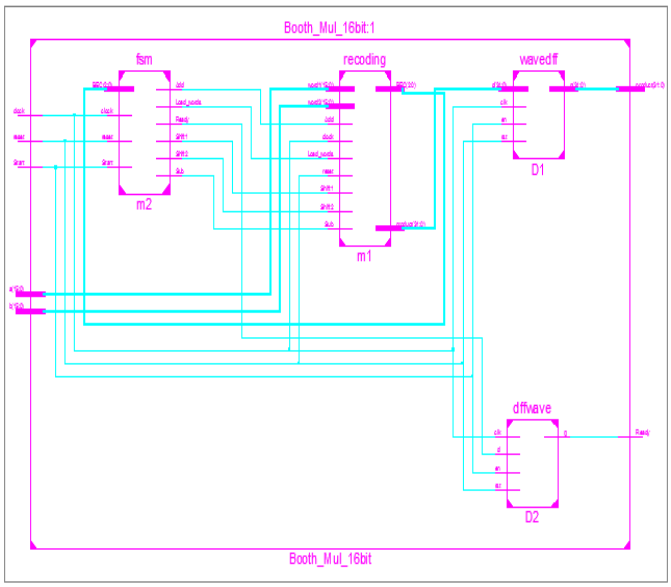

Fig: 4 RTL Schematic of 16 bit Booth multiplier

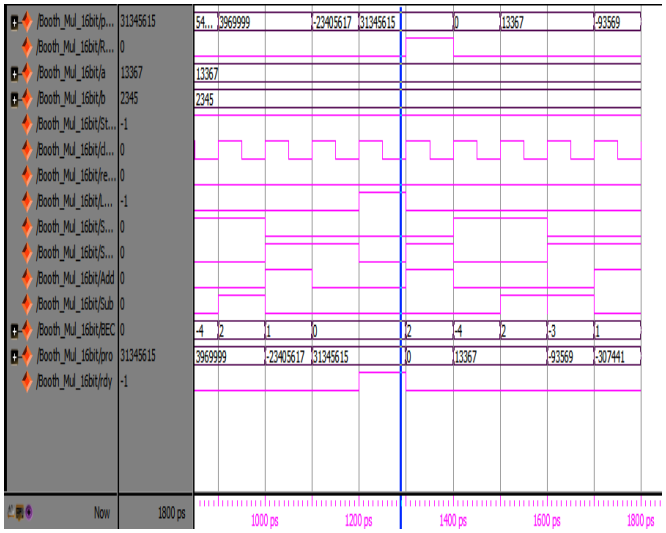

Fig: 5 simulated waveform of 16 bit Booth multiplier

The same FSM operation is used for producing 32 bit signed Booth multiplier. The recoded values are compared from the table1. They are simulated using approximate 2 bit adder, Booth recoding. For storing the next state logic FSM controller is used. Shifting and addition is done in same cycle. These are synthesized using Xilinx ISE 14.5. The RTL Schematic and the waveform are shown in fig 6 & 7.

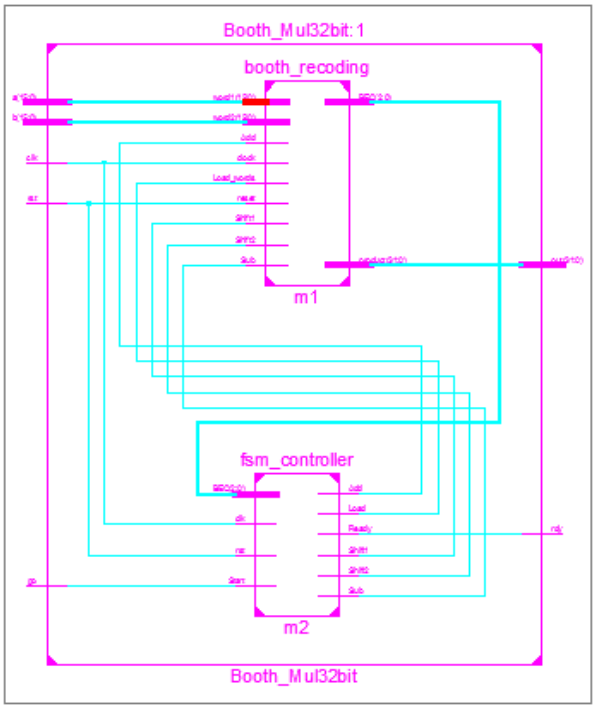

Fig: 6 RTL Schematic of 32 bit Booth multiplier

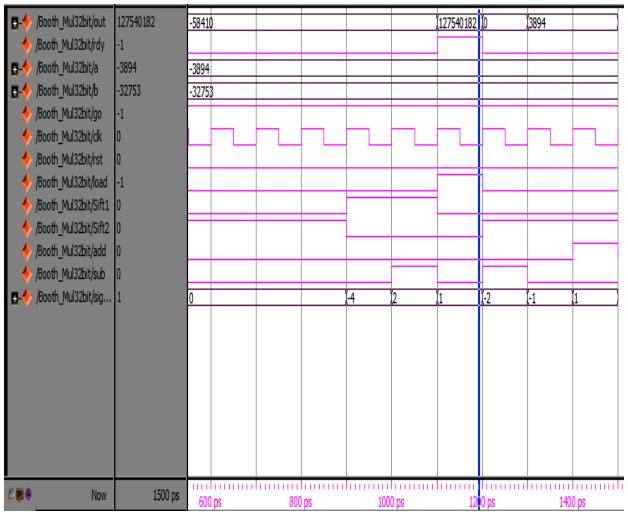

Fig: 7 simulated waveform of 32 bit Booth multiplier

A. Device summary

The Design Summary allows to quickly access design overview information, reports and messages. This tabulation 2&3 results about the number of slices, LUTs, occupied slices, related logic, bonded IOBs, IOB flip flops, buffers used among the available devices. Only 1% of the slices, LUTs related logics are used and the number of bonded IOBs used here is about 15%.The summary in table 2&3 shows the amount of devices used are less from the available devices allotted in the design of 16 and 32-bit Booth multiplier.

Table: 2 Device Summary for 16-bit Booth multiplier

Logic utilization	Used	Available	Utilization
No. of slice Flip flop	87	21,504	1%
No. of 4 I/P LUTs	211	21,504	1%
No. of occupied slices	112	10,752	1%
No. of slices containing only related logic	112	112	100%
No. of slices containing unrelated logic	0	112	0%
Total no. of 4 I/P LUTs	211	21,504	1%
No. of bonded IOBs	68	448	15%
IOB Flip flop	33	-----	-----
No. of BUFG,BUFGRTLs	2	32	6%
No. used as BUFGs	2	-----	-----

Table: 3 Device Summary for 32 bit Booth multiplier

Logic utilization	Used	Available	Utilization
No. of slice Flip flop	87	21,504	1%
No. of 4 I/P LUTs	205	21,504	1%
No. of occupied slices	108	10,752	1%
No. of slices containing only related logic	108	108	100%
No. of slices containing unrelated logic	0	108	0%
Total no. of 4 I/P LUTs	205	21,504	1%
No. of bonded IOBs	68	448	15%
No. of BUFG,BUFGRTLs	1	32	3%
No. used as BUFGs	1	----	----

B.Power Analysis

After synthesis and implementation, use the Power Analyzer to get a detailed view of the power distribution for the design, broken down into individual device elements. The power distribution analysis is done for both 16 and 32-bit which is shown in table 4 & 5. On comparing 16-bit with 32-bit, the 32-bit Booth multiplier consumes less power than 16-bit.

Table: 4 Power analysis of 16-bit Booth multiplier

On-chip	Power (w)	Used	Available	Utilization (%)
Clocks	0.030	2	---	---
Logic	0.005	210	21504	1
Signals	0.025	273	---	---
DCMs	0.000	0	8	0
IOs	0.019	68	448	15
Leakage	0.234	---	---	---
Total	**0.653**	---	---	---

Table: 5 Power analysis of 32-bit Booth multiplier

On-chip	Power (w)	Used	Available	Utilization (%)
Clocks	0.008	1	---	---
Logic	0.008	203	21504	1
Signals	0.025	232	---	---
DCMs	0.000	0	8	0
IOs	0.019	68	448	15
Leakage	0.233	---	---	---
Total	**0.241**	---	---	---

C.Delay Analysis

The Timing Analyzer verifies that the delay along a given path or paths meets your specified timing requirements. It organizes and displays data that allows to analyze the critical paths in your circuit, the cycle time of the circuit, the delay along any specified paths and the paths with the greatest delay. The delay analysis is performed and the results are shown in table 6 by comparing 16 bit with 32 bit. The delay is reduced for 32 bit Booth multiplier which is shown in table 6.

Table: 6 Delay analysis for 16 and 32-bit Booth multiplier

	16-bit	32-bit
Delay	5.187 ns	5.147 ns
Maximum frequency	192.777 MHZ	194.269 MHZ

V.CONCLUSION

This paper has proposed the approximate computing of Booth multiplier for Radix-8 of 16 and 32-bit signed multiplier using approximate 2-bit recoding adder. This adder incurs less delay, power and area. The synthesis is done using verilog coding on Xilinx ISE 14.5. The power and delay analysis had been performed. On comparing 16-bit with 32-bit, the delay and power had been reduced for 32-bit signed Booth multiplier.

REFERENCES

[1] Han, J. and Orshansky, M., (2013) 'Approximate computing: An emerging paradigm for energy-efficient design', IEEE European Test Symposium (ETS), pp. 1-6.

[2] Cho, K.J., Lee, K.C., Chung, J.G. and Parhi, K.K., (2004) 'Design of low-error fixed-width modified Booth multiplier', IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12(5), pp.522-531.

[3] Wang, J.P., Kuang, S.R. and Liang, S.C., (2011) 'High-accuracy fixed-width modified Booth multipliers for lossy applications', IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 19(1), pp.52-60.

[4] Li, C.Y., Chen, Y.H., Chang, T.Y. and Chen, J.N.,(2011) 'A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications'. IEEE Transactions on Circuits and Systems II: Express Briefs, 58(4), pp.215-219.

[5] Chen, Y.H., Li, C.Y. and Chang, T.Y., (2011) 'Area-effective and power-efficient fixed-width Booth multipliers using generalized probabilistic estimation bias'. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 1(3), pp.277-288.

[6] Chen, Y.H. and Chang, T.Y., (2012) 'A high-accuracy adaptive conditional-probability estimator for fixed-width Booth multipliers'. IEEE Transactions on Circuits and Systems I: Regular Papers, 59(3), pp.594-603.

[7] Kulkarni, P., Gupta, P. and Ercegovac, M, (2011), January. 'Trading accuracy for power with an under designed multiplier architecture'. In the 24th International IEEE Conference on VLSI Design pp. 346-351.

[8] Lin, C.H. and Lin, C., (2013), October 'High accuracy approximate multiplier with error correction'. IEEE 31st International Conference on Computer Design (ICCD) pp. 33-38.

[9] Momeni, A., Han, J., Montuschi, P. and Lombardi, F., (2015) 'Design and analysis of approximate compressors for multiplication'. IEEE Transactions on Computers, 64(4), pp.984-994.

[10] Liang, J., Han, J. and Lombardi, F., (2013) 'New metrics for the reliability of approximate and probabilistic adders'. IEEE Transactions on Computers, 62(9), pp.1760-1771.