VLSI Architecture of Pipelined Booth Wallace MAC Unit

Naveen Kumar
M.TECH(VLSI)
Thapar University, Patiala

Manu Bansal
M.TECH(VLSI)
Thapar University, Patiala

Navnish Kumar
M.TECH(VLSI)
Thapar University, Patiala

ABSTRACT
This paper describes the pipelined architecture of high-speed modified Booth Wallace Multiply and Accumulator. The proposed multiply and accumulate circuits are based on the Booth algorithm and the pipelining techniques, which are most widely used to accelerate the multiplication speed.
A 32-bit MAC Unit is designed in which the multiplication is done using the Modified Booth Wallace Multiplier and in the final stage addition of multiplier and in accumulator the Carry Select Adder is used and the pipelining is done in the Booth Multiplier and Wallace Tree. This MAC is described in VHDL and synthesized the circuit using 90 nm standard cell library on FPGA and Synopsys Design Compiler. This MAC has higher speed than conventional Booth Wallace MAC Unit.

Keywords
Multiplier, Adder, Pipelining, High-speed, modified Booth algorithm, Synopsys Design Compiler, FPGA.

1. INTRODUCTION
The core of every microprocessor, DSP, and data-processing ASIC is its data path. Statistics showed that more than 70% of the instructions perform additions and multiplications in the data path of RISC machines[1]. At the heart of data-path and addressing units in turn are arithmetic units, such as adders, and multipliers. Multiplication based operations such as Multiply and Accumulate and inner product are among some of the frequently used Computation-Intensive Arithmetic Functions, currently implemented in many DSP applications such as convolution, fast fourier transform, filtering and in microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications.
Pipelined MAC based on Booth Wallace with Carry Select Adder is one of the fastest MAC Unit. Because modified Booth algorithm reduces the number of partial products to be generated and is known as the fastest multiplication algorithm and many researches on the multiplier architectures including array, parallel and pipelined multipliers have been pursued which shows that pipelining is the most widely used technique to reduce the propagation delays of digital circuits.
Finally, the basic operation found in MAC is the binary addition. Therefore, binary addition is the most important arithmetic operation. 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.

2. CONVENTIONAL BOOTH WALLACE MULTIPLY AND ACCUMULATOR
In order to improve the speed of the MAC unit, there are two major bottlenecks that need to be considered. The first one is the fast multiplication network and the second one is the accumulator. Both of these stages require addition of large operands that involve long paths for carry propagation.
The MAC unit basically do the multiplication of two numbers multiplier and multiplicand and add that product in result stored in the accumulator. The general construction of the MAC operation can be represented by this equation:

$$Z = A * B + Z$$

Where the multiplier A and multiplicand B are assumed to have n bits each and the addend Z has (2n+1) bits. A basic MAC unit as shown in Figure1 can be divided into two main blocks [2].

- Multiplier
- Accumulator

2.1 Multiplier
Multiplication is a mathematical operation that at its simplest is an abbreviated process of adding an integer to itself a specified number of times. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result. A Fast Multiplication process consists of three steps [3]:

- Partial Product Generation.
- Partial Product Reduction.
- Final stage Carry Propagate Adder.

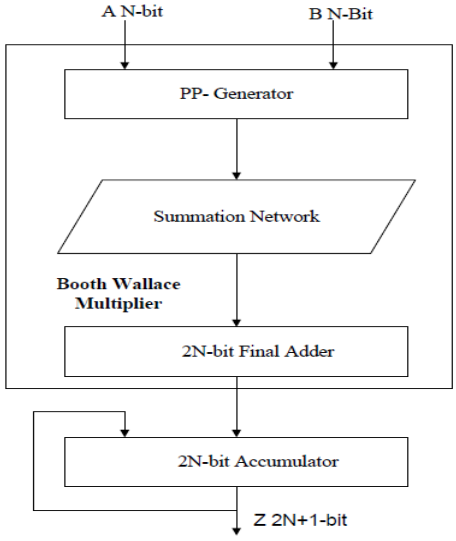

Figure 1: Basic Architecture of MAC Unit [3].

2.1.1 Partial Product Generation

To generate the number of partial product Radix-4 Modified booth encoding techniques have been used [4]. The Modified Booth Encoding (MBE) or Modified Booth's Algorithm (MBA) was proposed by O. L. Macsorley in 1961 [5]. Booth's radix-4 algorithm is widely used to reduce the area of multiplier and to increase the speed. The booth encoding algorithm is a bit-pair encoding algorithm that generates partial products which are multiples of the multiplicand. The booth algorithm shifts and/or complements the multiplicand (X operand) based on the bit patterns of the multiplier (Y operand). Essentially, three multiplier bits [$Y_{(i+1)}$,$Y_{(i)}$ and $Y_{(i-1)}$] are encoded into eight bits that are used to select multiples of the multiplicand [-2X,-X,0,+X,+2X]. The three multiplier bits consist of a new bit pair [$Y_{(i+1)}$ and $Y_{(i)}$] and the leftmost bit from the previously encoded bit pair [$Y_{(i-1)}$]. Grouping the three bits of multiplier with overlapping has half partial products which improve the system speed.

Radix-4 Modified Booth's algorithm [6] is:

- Y (i-1) = 0; Insert 0 on the right side of LSB of multiplier.
- Start grouping each three bits with overlapping from Y (i-1).
- If the number of multiplier bits is odd, add a extra 1 bit on left side of MSB and generate partial product from Table 1.
- When new partial product is generated, each partial product is added two bit left shifting in regular sequence.
- It is then sign extended.

Table 1: Radix-4 Modified Booth Algorithm [6].

Yi+1	Yi	Yi-1	Recoded Digit	Operand Multiplication
0	0	0	0	0 x multiplicand
0	0	1	+1	+1 x multiplicand
0	1	0	+1	+1 x multiplicand
0	1	1	+2	+2 x multiplicand
1	0	0	-2	-2 x multiplicand
1	0	1	-1	-1 x multiplicand
1	1	0	-1	-1 x multiplicand
1	1	1	0	0 x multiplicand

2.1.2 Partial Product Compression

Multiplier require high amount of power and delay during the partial products addition. At this stage, most of the multipliers are designed with different kind of multi operands adders that are capable to add more than two input operands and results in two outputs, sum and carry. The number of adders will be minimized by Wallace Tree.

2.1.3 4:2 Compressor

It has 4 input lines X0, X1, X2, X3 that must be summed and has two output lines C and S which are so called result of compression. A 4:2 compressor can be implemented with two stages of full adder (FA) connected in series as shown in Figure 2. Indeed a 4:2 structure is not a counter, since two output bits cannot represent five possible sums of four bits. Thus a carry out is necessary and subsequently carry in.

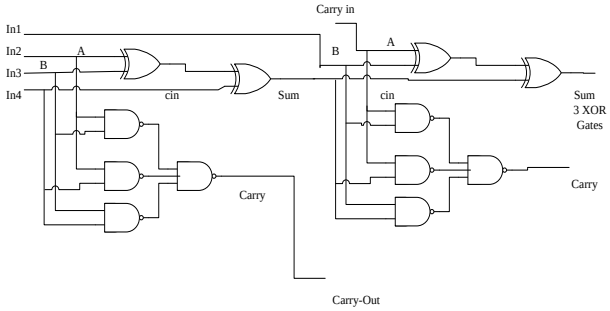

Figure 2: A 4:2 compressor logic diagram [7].

The 4:2 compressor structure actually compress five input bit into three output. The output of 4:2 compressor consists of one bit (sum) in position j and two bits in final carry and intermediate carry in position (j+1). However a carry out is independent on carry in as shown in figure.

2.1.4 Wallace Tree

Wallace Tree is a reduction technique that uses the carry save adder to add the partial products. A block diagram for the data distribution among a tree architecture that employs 4:2 compressors is shown in Figure 3.

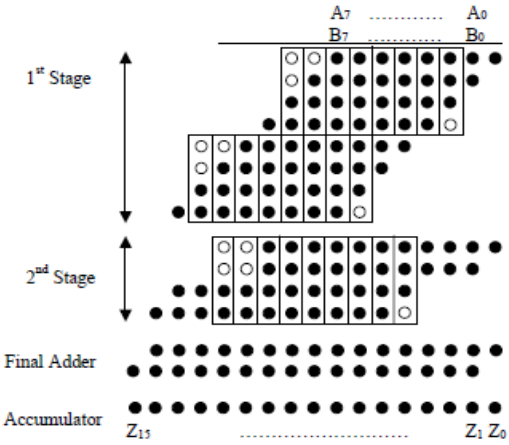

Figure 3: Data distribution among a tree architecture [3].

Each box contains the bits that are fed into a 4:2 compressor cell. Two stages of 4:2 compressors are used to reduce the number of partial products by a ratio of 2:1. This figure presents the reduction tree of 8 partial products to form two operands, which are then added together to form a final product by using a fast carry propagate adder.

Using 4:2 compressor, there is a simple and more regular wiring of multiplier tree .

2.1.5 Final stage Carry Propagate Adder

This stage is also crucial for any multiplier because in this stage addition of large size operands is performed so in this stage fast carry propagate adders like Carry-look Ahead Adder or Carry Skip Adder or Carry Select Adder can be used as per our requirement.

2.2 Accumulator

Accumulator basically consists of register and adder. Register hold the output of previous clock from adder. Holding outputs

in accumulation register can reduce additional add instruction. An accumulator should be fast in response so it can be implemented with one of fastest adder like Carry-look Ahead Adder or Carry Skip Adder or Carry Select Adder .

3. PROPOSED ARCHITECTURE OF MAC UNIT

The pipeline technique is widely used to improve the performance of digital circuits. As the number of pipeline stages is increased, the path delays of each stage are decreased and the overall performance of the circuit is improved [6]. The various pipeline schemes have been investigated to find the optimum number of pipeline stages and the positions for the pipeline registers to be inserted in modified Booth multiplier in order to obtain the high-speed.

At first, modified Booth multiplier is partitioned into three pipeline stages according to the functionality of the circuit as shown in Figure 4. The critical path of the pipelined Booth multiplier is in the Wallace tree because it requires the most intensive computation. It means that delays can be further reduced by adding more pipeline registers within the Wallace Tree as shown in Figure 5.

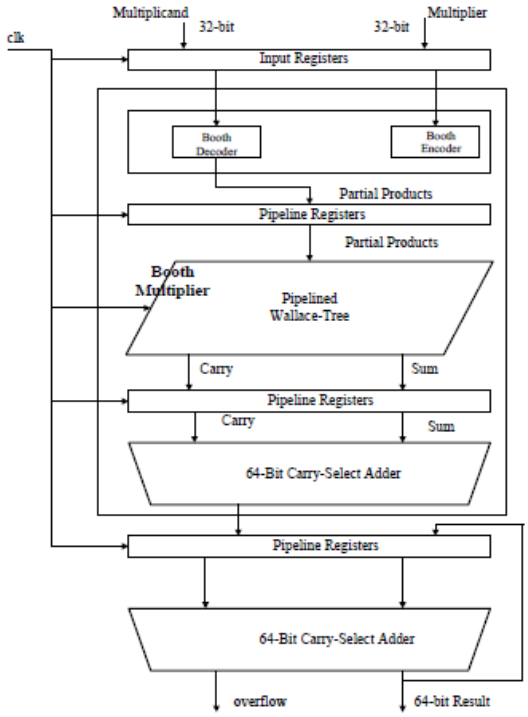

Figure 4: Block Diagram of 32-bit Pipelined Booth Wallace MAC Unit.

3.1 Pipelined Wallace Tree

It is, however, still the most critical part of the multiplier because it is responsible for the largest amount of computation. Naturally the pipeline registers should be inserted within the Wallace tree to improve the performance as shown in Figure 5.

The number of partial products of the N-bit modified Booth multiplier is N/2[4][5][6]. In case of the 32-bit modified Booth multiplier, the number of partial products is sixteen. In this three stage of pipelining is used, in 1st stage sum and carry of first four row of compressors are stored in two pipeline registers. In 2nd stage these sum and carry are fed into the fifth and sixth row of compressors and their otput is stored in pipeline registers. In 3rd stage the sum and carry of fifth and sixth row are added in seventh row of compressors. The Wallace tree adds four rows and generates two rows using 4:2 compressors. One row represents the carries and the other row represents the sums.

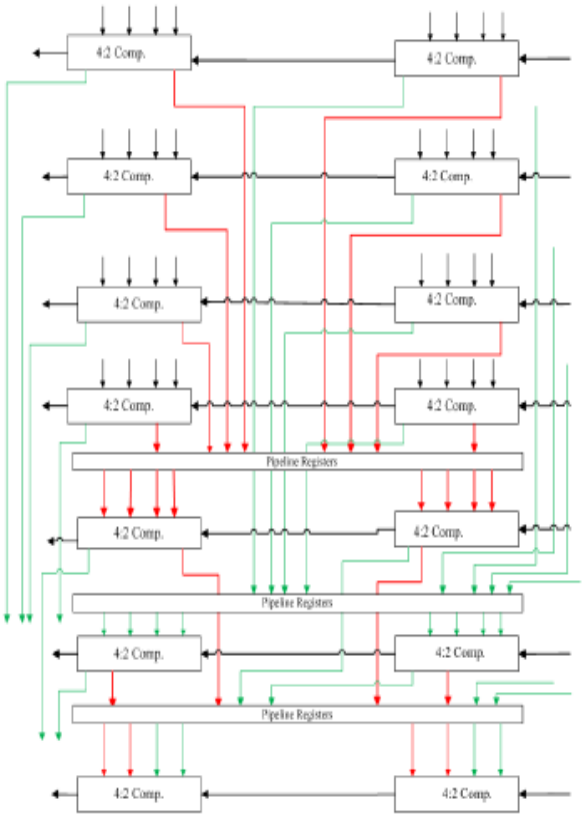

Figure 5: Pipelining in Wallace Tree.

3.2 Carry Select Adder

The result from Wallace tree is the sum and carry vector hence, it needs to be added to get final result. If the ripple carry adder is used to add the sum and the carry vector, then the carry propagation delay has a bad effect on the performance of MAC.

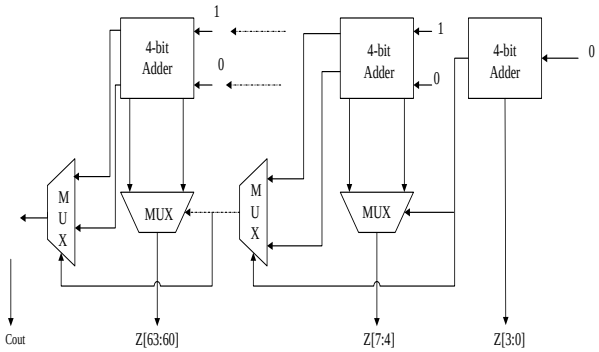

Figure 6: 64-bit Carry-Select Adder

Carry Select Adder computes Sum for carry is 0 and 1. In spite of large implementation, it can increase the speed of execution. Both vectors are added in 64-bit carry select adder block and the final result is generated.

4. EXPERIMENTAL RESULTS

A 32-bit Proposed Pipelined Booth Wallace MAC has been implemented on Spartan XC3S500-4FG320 device and synthesized using 90 nm technology using Synopsys Design Compiler and its results compared with the conventional MAC unit

Table 2: FPGA Synthesis Results of Conventional and Proposed Pipelined 32-bit Booth Wallace MAC Unit

MAC	Conventional MAC	Proposed MAC
Size	32 bits	32 bits
Slice Flip-Flops	237	1399
LUTs	3646	3298
Delay(ns)	24.1	10.5
Frequency (MHz)	41.5	95.23 (x2.29 improvement)
Power (mW)	279.2	531.42

Table 2 shows that Proposed Pipelined Booth Wallace MAC has double speed than a conventional Booth Wallace MAC.
The 32-bit Proposed Pipelined Booth Wallace MAC also synthesized on Design Compiler Table 4 and its results compared with the 32-bit Conventional Booth Wallace MAC unit as shown in Table 3.

Table 3: Design Compiler Synthesis Results of Conventional 32-bit MAC Unit .

	Conventional Booth Wallace MAC		
	Worst Case	Best Case	Avg. Case
Clock Period(ns)	3.97	1.61	2.35
Design Area	101999.96	91334.43	94504.14
Dynamic Power(mW)	10.085	33.379	18.821
Leakage Power(uW)	74.829	179.578	61.434
Frequency (MHz)	251.88	621.11	425.53

Case	***Operating Conditions***		
	Voltage	Temperature	Process
Worst Case	1.62V	125^0C	1
Best Case	1.98V	-40^0C	1
Avg.Case	1.80V	25^0C	1

These are the different operating conditions provided in the library.

Table 4: Design Compiler Synthesis Results of Proposed Pipelined 32-bit MAC Unit

	Proposed Booth Wallace MAC		
	Worst Case	Best Case	Avg Case
Clock Period(ns)	2.43	1.1	1.45
Design Area	79970.35	79425.47	79890.38
Dynamic Power(mW)	17.3468	60.9256	36.5846
Leakage Power(uW)	37.5806	124.1783	39.7328
Frequency (MHz)	411.52	909.09	689.65
Speed Improvement	x1.63	x1.46	x1.62

Table 5: Comparison with other 16-bit Pipelined Multiplier references[6]

Type	Case	# of stages	Gate count	Area increase	Delay(ns)	Speed improvement
A	-	-	3,629	-	2.40	-
B	-	3	4,017	x1.11	1.66	x1.45
C	1	5	4,526	x1.25	0.82	x2.93
	2	5	4,528	x1.25	0.91	x2.64
	3	5	4,528	x1.25	0.91	x2.64
	4	4	4,452	x1.23	1.05	x2.29
	5	4	4,268	x1.17	1.10	x2.18
	6	9	6,069	x1.67	0.47	x5.11

Its clear from the Table 3 and 4 that Proposed MAC has speed from 411 MHz to 909 MHz which is much higher than conventional MAC and also Table 2 shows that proposed MAC has Frequency more than double than conventional MAC Unit .

5. CONCLUSION

The pipelining is the most widely used technique to improvethe performance of digital circuits. We proposed the high-speed modified Booth Wallace MAC with pipeline. The proposed MAC consist of three modules: 1) the modified Booth encoder and decoder module to generate N/2 partial products; 2) the Wallace tree module to add the partial products in parallel; 3) the carry select adder module for the final addition.

Using 90 nm standard cell library, we obtained the 32-bit MAC with the operating frequencies of 0.909 GHz. Since the speed improvement ratio is much greater than the area increase ratio, they can be used in the pplication systems requiring very high performance while the area penalty is tolerable.

6. REFERENCES

[1] Koc, C.K., "RSA Hardware Implementation",RSA Laboratories, RSA Data Security, Inc. 1996.

[2] Abdelgawad, A., Bayoumi, M., "High Speed and Area-Efficient Multiply Accumulate (MAC) Unit for Digital Signal Processing Applications", IEEE International Symposium on Circuits and Systems, pp . 3199 – 3202, 2007.

[3] Fayed, Ayman A., Bayoumi, Magdy A., "A Merged Multiplier-Accumulator for high speed signal processing applications", IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp 3212 -3215, 2002.

[4] A.D.Booth, "A Signed Binary Multiplication Technique", Quarterly J.Mechan Appl.Math., vol.IV, 1951.

[5] Macsorley, O.L., "High-Speed Arithmetic in Binary Computers", Proceedings of the IRE ,vol. 49, pp 67 – 91, 1961.

[6] Kim Soojin; Cho Kyeongsoon; "Design of High speed Modified Booth Multipliers Operating at GHz Ranges", World Academy of Science, Engineering and Technology, Issue 61, January 2010.

[7] Oklobdzija, V.G.; Villeger, D.; Liu, S.S.; "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach" IEEE Transactions on Computers, vol.45, pp 294 – 306, 1996.

[8] P. Devi and A. Girdher, "Improved Carry Select Adder with Reduced Area and Low Power Consumption", International Journal of Computer Applications (0975– 8887) vol. 3 – No.4, June 2010.

[9] R.P.P.,Singh; B. Singh and P. Kumar, "Performance Analysis Of Fast Adders Using VHDL", International Conference on Advances in Recent Technologies in Communication and Computing, 2009.

[10] A. A. A., Gutub and H. A., Tahhan, "Efficent Adders To Speedup Modular Multiplication for Cryptography", Computer Engineering Department, KFUPM, Dhahran, SAUDI ARABIA, 2001.

[11] B. Parhami, "Computer Arithmetic, Algorithm and Hardware Design", Oxford University Press, New York, pp.73-137, 2000.

[12] K.C. Chang, "Digital system design with VHDL and Synthesis" An integrated Approach IEE Computer Society, pp 408-437, 1999.

[13] Design Compiler User Guide v1999.10.

[14] Himanshu Bhatnagar; "Advanced ASIC Chip Synthesis" using Synopsys Design Compiler, Physical Compiler and Prime Time, 2nd ed..Kluwer Academic Publishers, 2002.

[15] Weng Fook Lee, "VHDL Coding and Logic Synthesis with Synopsys" Academic Press pp.147-227, 2000.

[16] Hima Bindu Kommuru Hamid Mahmoodi, "ASIC Design Flow Tutorial Using Synopsys Tools", Nano-Electronics & Computing Research Lab School of Engineering San Francisco State University San Francisco, CA Spring 2009.

[17] Deschamps, J.P.; Bioul, G.J.A; Sutter,G.D.; "Synthesis of Arithmetic Circuits"; FPGA, ASIC and Embedded Systems, John Wily & Sons Inc., Publication, 2006.