High speed arithmetic Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm

[1]Harilal, M.Tech, [2,]DURGA PRASAD, M.tech,(Ph.D),
[1,]SKTRMCE,K.
[2,]Associate professor, SKTRMCE,

Abstract:
The sustained growth in VLSI technology is fuelled by the continued shrinking of transistor to ever smaller dimension. The benefits of miniaturization are high packing densities, high circuit speed and low power dissipation. Binary multiplier is an electronic circuit used in digital electronics such as a computer to multiply two binary numbers, which is built using a binary adder. A fixed-width multiplier is attractive to many multimedia and digital signal processing systems which are desirable to maintain a fixed format and allow a minimum accuracy loss to output data. This paper presents the design of high-accuracy modified Booth multipliers using Carry Look ahead Adder. The high accuracy fixed width modified booth multiplier is used to satisfy the needs of the applications like digital filtering, arithmetic coding, wavelet transformation, echo cancellation, etc. The high accuracy modified booth multipliers can also be applicable to lossy applications to reduce the area and power consumption of the whole system while maintaining good output quality. This project presents an efficient implementation of high speed multiplier using the shift and add method, Radix_2, Radix_4 modified Booth multiplier algorithm. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does theComputations using lesser adders and lesser iterative steps. As a result of which they occupy lesser space as compared to the serial multiplier. This very important criteria because in the fabrication of chips and high performance system requires components which are as small as possible.

Key words: Booth multiplier, carry save adder (CSA) tree, computer arithmetic, digital signal processing (DSP), ultiplierand- accumulator (MAC).

1. Introduction

In this paper, we propose a high-accuracy fixed width modified booth multiplier. The functional model design consists ofbooth encoder, partial product generator and compression tree which uses Carry Look ahead Adder. The term “high accuracy”implies that the output produced by the normal 8X8 booth multiplication and the proposed 8X8 booth multiplication are equal. Theterm “fixed width” indicates that the partial product bits are adjusted to fixed width for Carry Look ahead.The result and one operand for the new modulo multipliers use weighted representation, while the other uses thediminished - 1. By using the radix-4 Booth recoding, the new multipliers reduce the number of the partial products to n/2 for n evenand (n+1)/2 for n odd except for one correction term. Although one correction term is used, the circuit is very simple. Thearchitecture for the new multipliers consists of an inverted end-around-carry carry save adder tree and one diminished-1 adder. Booth multipliers using generalized probabilistic estimation bias (GPEB) is proposed. The GPEB circuit can be easily builtaccording to the proposed systematic steps. The GPEB fixed-width multipliers with variable-correction outperform the existingcompensation circuits in reducing error. The GPEB circuit has improved absolute average error reduction, area saving, powerefficiency and accuracy. A truncated multiplier is a multiplier with two n bit operands that produces a n bit result. Truncatedmultipliers discard some of the partial products of a complete multiplier to trade off accuracy with hardware cost. This paperpresents a closed form analytical calculation, for every bit width, of the maximum error for a previously proposed family oftruncated multipliers. The considered family of truncated multipliers is particularly important since it is proved to be the design thatgives the lowest mean square error for a given number of discarder partial products. With the contribution of this paper, theconsidered family of truncated multipliers is the only architecture that can be designed, for every bit width, using an analyticalapproach that allows the a priori knowledge of the maximum error. A 2-bit Booth encoder with Josephson Transmission Lines(JTLs) and Passive Transmission Lines (PTLs) by using cell-based techniques and tools was designed. The Booth encoding methodis one of the algorithms to obtain partial products. With this method, the number of partial products decreases down to the halfcompared to the AND array method. a test chip for a multiplier with a 2-bit Booth encoder with JTLs and PTLs was fabricated. Thecircuit area of the multiplier designed with the Booth encoder method is compared to that designed with the AND array method. New fixed-width multiplier topologies, with different accuracy versus hardware complexity trade-off, are obtained by varyingthe quantization scheme. Two topologies are in particular selected as the most effective ones. The first one is based on a uniformcoefficient quantization, while the second topology uses a non-uniform quantization scheme. The novel fixed-width multipliertopologies exhibit better accuracy with respect to previous solutions, close to the theoretical lower bound. The electricalperformances of the proposed fixed-width multipliers are compared with previous architectures. It is found that in most of theinvestigated cases the new topologies are Pareto-optimal regarding the area-accuracy trade-off.This paper focuses on variable-correction truncated multipliers, where some partial-products are discarded, to reduce complexity,and a suitable compensation function is added to partly compensate the introduced error. The optimal compensation function, thatminimizes the mean square error, is obtained in this paper in closed-form for the first time. A sub optimal compensation function,best suited for hardware implementation, is introduced. Efficient multipliers implementation based on sub-optimal function isdiscussed. Proposed truncated multipliers are extensively compared with previously proposed circuits. Power efficient 16times 16 Configurable Booth Multiplier (CBM) supports single 16-b, single 8-b, or twin parallel 8-b multiplication operations isproposed. Dynamic range detector detects the dynamic ranges of two input operands. It deactivates redundant switching activitiesin ineffective ranges.The proposed architecture can be used effectively in the area requiring high throughput such as a real-time digitalsignal processing can be expected.

2. Overview Of Mac

In this section, basic MAC operation is introduced. A multipliercan be divided into three operational steps. The first isradix-2 Booth encoding in which a partial product is generatedfrom the multiplicand and the multiplier. The secondis adder array or partial product compression to add all partialproducts and convert them into the form of sum and carry. Thelast is the final addition in which the final multiplication resultis produced by adding the sum and the carry. If the process toaccumulate the multiplied results is included, a MAC consistsof four steps, as shown in Fig. 1, which shows the operationalsteps explicitly.General hardware architecture of this MAC is shown inFig. 2. It executes the multiplication operation by multiplyingthe input multiplier and the multiplicand. This is added tothe previous multiplication result as the accumulation step.

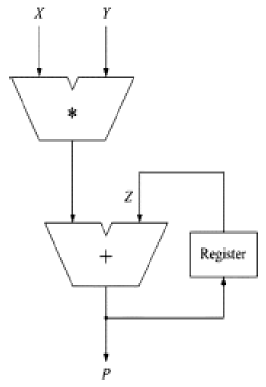


Figure 1. Hardware architecture of general Mac

The -bit 2's complement binary number can be expressedas

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \qquad x_i \in 0,1. \quad (1)$$

If (1) is expressed in base-4 type redundant sign digit form inorder to apply the radix-2 Booth's algorithm, it would be .

$$X = \sum_{i=0}^{N/2-1} d_i 4_i \quad (2)$$

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \quad (3)$$

If (2) is used, multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y. \qquad (4)$$

If these equations are used, the afore-mentioned multiplication–accumulation results can be expressed as

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^i Y + \sum_{j=0}^{2N-1} z_i 2^i. \qquad (5)$$

Each of the two terms on the right-hand side of (5) is calculatedindependently and the final result is produced by addingthe two results. The MAC architecture implemented by (5) is called the standard design.If -bit data are multiplied, the number of the generated partialproducts is proportional to . In order to add them serially,the execution time is also proportional to . The architecture ofa multiplier, which is the fastest, uses radix-2 Booth encodingthat generates partial products and aWallace tree based on CSAas the adder array to add the partial products. If radix-2 Boothencoding is used, the number of partial products, i.e., the inputsto the Wallace tree, is reduced to half, resulting in the decreasein CSA tree step. In addition, the signed multiplication based on2's complement numbers is also possible. Due to these reasons,most current used multipliers adopt the Booth encoding.

3. ProposedMac Architecture

In this section, the expression for the new arithmetic will bederived from equations of the standard design. From this result,VLSI architecture for the new MAC will be proposed. In addition,a hybrid-typed CSA architecture that can satisfy the operationof the proposed MAC will be proposed.

A. Derivation Of Mac Arithmetic

1) Basic Concept: If an operation to multiply two –bitnumbers and accumulate into a 2 -bit number is considered,the critical path is determined by the 2 -bit accumulation operation.If a pipeline scheme is applied for each step in the standarddesign of Fig. 1, the delay of the last accumulator mustbe reduced in order to improve the performance of the MAC.The overall performance of the proposed MAC is improved byeliminating the accumulator itself by combining it with the CSAfunction. If the accumulator has been eliminated, the criticalpath is then determined by the final adder in the multiplier. Thebasic method to improve the performance of the final adder is todecrease the number of input bits. In order to reduce this numberof input bits, the multiple partial products are compressed into asum and a carry by CSA. The number of bits of sums and carriesto be transferred to the final adder is reduced by adding the lowerbits of sums and carries in advance within the range in whichthe overall performance will not be degraded. A 2-bit CLA isused to add the lower bits in the CSA. In order to efficientlysolve the increase in the amount of data, a CSA architecture is modified to treat the sign bit.

2) Equation Derivation: The aforementioned concept is appliedto (5) to express the proposed MAC arithmetic. Then, themultiplication would be transferred to a hardware architecturethat complies with the proposed concept, in which the feedbackvalue for accumulation will be modified and expanded for thenew MAC.First, if the multiplication in (4) is decomposed and rearranged,it becomes

$$X \times Y = d_0 2Y + d_1 2^2 Y + d_2 2^4 Y + \ldots + d_{N/2-1} 2^{N-2} Y. \qquad (6)$$

If (6) is divided into the first partial product, sum of the middlepartial products, and the final partial product, it can be reexpressedas (7). The reason for separating the partial product additionas (7) is that three types of data are fed back for accumulation,which are the sum, the carry, and the preadded

$$X \times Y = d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y. \qquad (7)$$

results ofthe sum and carry from lower bits

Now, the proposed concept is applied to in (5). If is firstdivided into upper and lower bits and rearranged, (8) will bederived. The first term of the right-hand side in (8) correspondsto the upper bits. It is the value that is fed back as the sum andthe carry. The second term corresponds to the lower bits and is

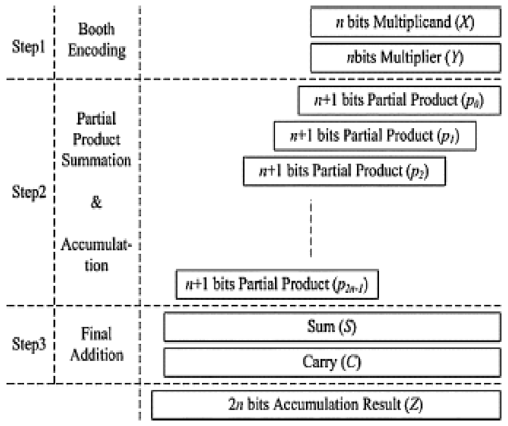

Step1
Booth Encoding
n bits Multiplicand (X)
nbits Multiplier (Y)
Step2
Partial Product Summation & Accumulat-tion
n+1 bits Partial Product (p0)
n+1 bits Partial Product (p1)
n+1 bits Partial Product (p2)
n+1 bits Partial Product (p2n-1)
Step3
Final Addition
Sum (S)
Carry (C)
2n bits Accumulation Result (Z)

Figure2: Proposed arithmetic architecture of MAC

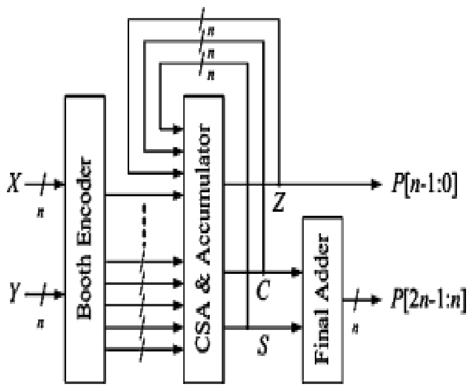


Figure3: Hardware architecture of proposed MAC

the value that is fed back as the addition result for the sum andcarry

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i. \qquad (8)$$

The second term can be separated further into the carry termand sum term as

$$\sum_{i=N}^{2N-1} z_i 2^i = \sum_{i=0}^{N-1} z_{N+i} 2^i 2^N = \sum_{i=0}^{N-2} (c_i + s_i) 2^i 2^N. \qquad (9)$$

Thus, (8) is finally separated into three terms as

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N. \qquad (10)$$

If (7) and (10) are used, the MAC arithmetic in (5) can beexpressed as

$$P = \left(d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y \right) + \left(\sum_{i=0}^{N-1} z_i 2^i 2^N + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N \right). \quad (11)$$

If each term of (11) is matched to the bit position and rearranged,it can be expressed as (12), which is the final equationfor the proposed MAC. The first parenthesis on the right is theoperation to accumulate the first partial product with the addedresult of the sum and the carry. The second parenthesis is theone to accumulate the middle partial products with the sumoftheCSA that was fed back. Finally, the third parenthesis expressesthe operation to accumulate the last partial product withthe carry of the CSA.

B. Proposed Mac Architecture

If the MAC process proposed in the previous section is rearranged,it would be as Fig. 3, in which the MAC is organizedinto three steps. When compared with Fig. 1, it is easy to identifythe difference that the accumulation has been merged intothe process of adding the partial products. Another big differencefrom Fig. 1 is that the final addition process in step 3 is notalways run even though it does not appear explicitly in Fig. 3.

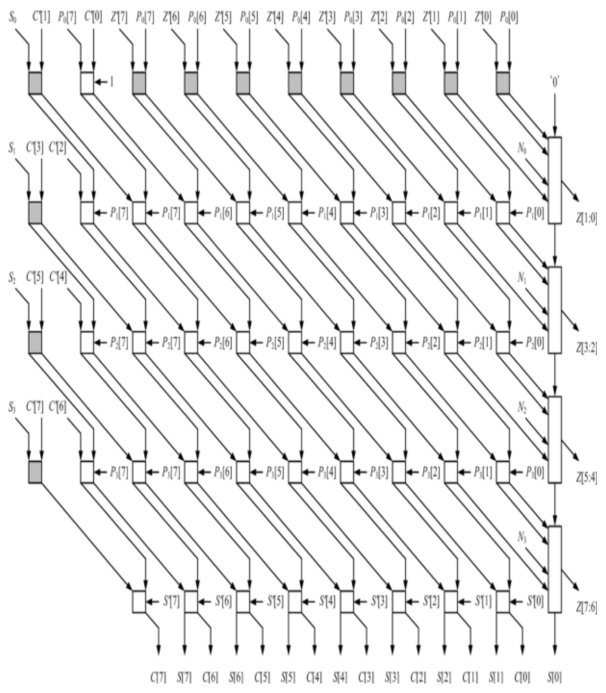

Z[1:0]
Z[3:2]
Z[5:4]
Z[7:6]
C[7] S[7] C[6] S[6] C[5] S[5] C[4] S[4] C[3] S[3] C[2] S[2] C[1] S[1] C[0] S[0]

Figure 4.Architecture of the proposed CSA tree.

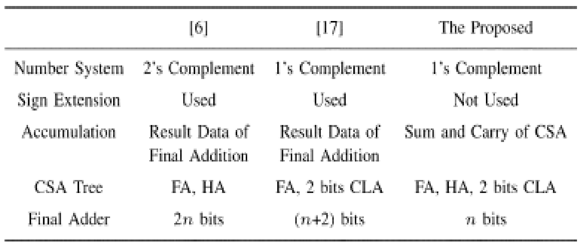
$$P = \left(d_0 2Y + \sum_{i=0}^{N-1} z_i 2^i \right) + \left(\sum_{i=1}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{N-2} c_i 2^i 2^N \right) + \left(d_{N/2-1} 2^{N-2} Y + \sum_{i=0}^{N-2} s_i 2^i 2^N \right). \quad (12)$$

Since accumulation is carried out using the result from step 2 insteadof that from step 3, step 3 does not have to be run until thepoint at which the result for the final accumulation is needed.The hardware architecture of the MAC to satisfy the processin Fig. 3 is shown in Fig. 4. The -bitMAC inputs, and , areconverted into an -bit partial product by passing throughthe Booth encoder. In the CSA and accumulator, accumulationis carried out along with the addition of the partial products. Asa result, -bit , and (the result from adding the lower bitsof the sum and carry) are generated. These three values are fedback and used for the next accumulation. If the final result forthe MAC is needed, is generated by adding andC in the final adder and combined with that wasalready generated.

C. Proposed Csa Architecture

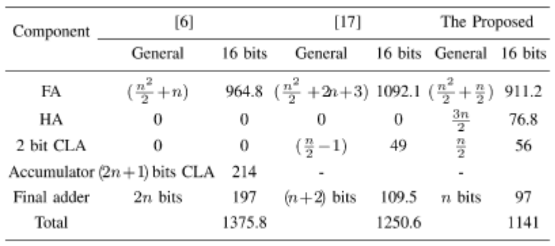
The architecture of the hybrid-type CSA that complies withthe operation of the proposed MAC is shown in Fig. 5, whichperforms 8 8-bit operation. It was formed based on (12). InFig. 5, is to simplify the sign expansion and is to compensate1's complement number into 2's complement number.and correspond to the th bit of the feedback sum andcarry. is the th bit of the sum of the lower bits for eachpartial product that were added in advance and is the previousresult. In addition, corresponds to the th bit of theth partial product. Since the multiplier is for 8 bits, totally fourpartial products are generated from theBooth encoder. In (11), and correspond toand , respectively. This CSA requires at leastfour rows of FAs for the four partial products. Thus, totally fiveFA rows are necessary since one more level of rows are neededfor accumulation. For an -bit MAC operation, the levelof CSA is . The white square in Fig. 5 represents anFA and the gray square is a half adder (HA). The rectangularsymbol with five inputs is a 2-bit CLA with a carry input.The critical path in this CSA is determined by the 2-bit CLA.It is also possible to use FAs to implement the CSA withoutCLA. However, if the lower bits of the previously generatedpartial product are not processed in advance by the CLAs, thenumber of bits for the final adder will increase. When the entiremultiplier or MAC is considered, it degrades the performance.In Table I, the characteristics of the proposed CSA architecturehave been summarized and briefly compared with other architectures.For the number system, the proposed CSA uses 1's complement, but ours uses a modified CSA array without signextension.Thebiggest difference between ours and the othersis the type of values that is fed back for accumulation. Ours hasthe smallest number of inputs to the final adder.

Table1: Characteristics of CSA



	[6]	[17]	The Proposed
Number System	2's Complement	1's Complement	1's Complement
Sign Extension	Used	Used	Not Used
Accumulation	Result Data of Final Addition	Result Data of Final Addition	Sum and Carry of CSA
CSA Tree	FA, HA	FA, 2 bits CLA	FA, HA, 2 bits CLA
Final Adder	$2n$ bits	$(n+2)$ bits	n bits

Table2: Calculation of Hardware Resources



Component	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bits
FA	$(\frac{n^2}{2}+n)$	964.8	$(\frac{n^2}{2}+2n+3)$	1092.1	$(\frac{n^2}{2}+\frac{n}{2})$	911.2
HA	0	0	0	0	$\frac{3n}{2}$	76.8
2 bit CLA	0	0	$(\frac{n}{2}-1)$	49	$\frac{n}{2}$	56
Accumulator	$(2n+1)$ bits CLA	214	-		-	
Final adder	$2n$ bits	197	$(n+2)$ bits	109.5	n bits	97
Total		1375.8		1250.6		1141

4. Implementation And Experiment

In this section, the proposed MAC is implemented andanalyzed. Then it would be compared with some previousresearches. First, the amount of used resources in implementingin hardware is analyzed theoretically and experimentally, thenthe delay of the hardware is analyzed by simplifying Sakurai'salpha power law [20]. Finally, the pipeline stage is defined andthe performance is analyzed based on this pipelining scheme.Implementation result from each section will be compared withthe standard design and Elguibaly's design, each ofwhich has the most representative parallel MBA architecture.

A. Hardware Resource

1) Analysis of Hardware Resource: The three architecturementioned before are analyzed to compare the hardwareresources and the results are given in Table II. In calculatingthe amount of the hardware resources, the resources for Boothencoder is excluded by assuming that the identical ones wereused for all the designs. The hardware resources in Table II arethe results from counting all the logic elements for a general16-bit architecture. The 90 nm CMOS HVT standard cell libraryfrom TSMCwas used as the hardware library for the 16 bits. Thegate count for each designwas obtained by synthesizing the logicelements in an optimal form and the resultwas generated by multiplyingit with the estimated number of hardware resources. Thegate counts for the circuit elements obtained through synthesisare showninTable III, which are based on a two-input NANDgate.Let us examine the gate count for several elements in Table IIIfirst. Since the gate count is 3.2 for HA and 6.7 for FA, FA isabout twice as large as HA. Because the gate count for a 2-bitCLA is 7, it is slightly larger than FA. In other words, even if a2-bit CLA is used to add the lower bits of the partial products inthe proposed CSA architecture, it can be seen that the hardwareresources will not increase significantly.

Table3: Gate size of logic circuit element

Element	Gate Size
Inverter	0.8
2/3/4-NAND	1/1.5/2.5
2/3/4-NOR	1/2/2.2
2/3/4-XOR	2/4/6
2/3/4-AND	1.2/1.5/2
2/3/4-OR	1.2/1.5/2
Half Adder	3.2
Full Adder	6.7
D Flip-Flop	6.2
4×1 MUX	6
8×1 MUX	14.2
2 bits CLA	7
4 bits CLA	20.5

Table4: Estimation of gate size synthesis

	CSA		Booth Encoder	Final Adder		Total (C/L)	
nm	[17]	Proposed		[17]	Proposed	[17]	Proposed
90	1,067	1,009	713	104	97	1,884	1,819
130	1,216	1,158	864	118	110	2,198	2,131
180	1,581	1,484	808	120	114	2,510	2,407
250	2,027	2,001	1,129	141	131	3,297	3,261

As Table II shows, the standard design uses the most hardwareresources and the proposed architecture uses the least. Theproposed architecture has optimized the resources for the CSAby using both FA and HA. By reducing the number of input bitsto the final adder, the gate count of the final adder was reducedfrom 109.5 to 97.*2) Gate Count by Synthesis:* The proposed MAC and [17]were implemented in register-transfer level (RTL) using hardwaredescription language (HDL). The designed circuits weresynthesized using the Design Complier from Synopsys, Inc.,and the gate counts for the resulting netlists were measured andsummarized in Table IV. The circuits in Table IV are for 16-bitMACs. In order to examine the various circuit characteristicsfor different CMOS processes, the most popular four processlibraries (0.25, 0.18, 0.13 m, 90 nm) for manufacturing digitalsemiconductors were used. It can be seen that the finer theprocess is, the smaller the number of gates is.As shown in Table II, the gate count for our architecture isslightly smaller. It must be kept in mind that ifa circuit is implemented as part of a larger circuit, the number ofgates may change depending on the timing for the entire circuitand the electric environments even though identical constraintswere applied in the synthesis. The results in Table IV were forthe combinational circuits without sequential element. The totalgate count is equal to the sum of the Booth encoder, the CSA,and the final adder.

Table 5: Normalized Capacitance and Gate Delay

Gate	Comment	C_i	T_g
Inverter	-	3	t+c
8×1 MUX	4-level logic	4	35.2+t+c
D-F/F	Slave delay	4	16.1+t+c
1 bit FA	input-to-sum	12	39.6+t+c
1 bit FA	input-to-carry	12	38.7+t+c
2 bits CLA	input-to-sum	12	64.9+t+c
2 bits CLA	input-to-carry	16	53.9+t+c
4 bits CLA	input-to-sum	12	96.8+t+c
4 bits CLA	input-to-carry	24	88+t+c

Table 6: Delay Time analysis and comparison

Step	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bits
Step1	Booth Encoding		Booth Encoding		Booth Encoding	
	$52.8n + 59.9$	904.7	$10.6n + 81.1$	250.7	$10.6n + 81.1$	250.7
Step2	CSA		Hybrid CSA		Hybrid CSA	
	$25.95n - 51.9$	363.3	$33.55n - 67.1$	469.7	$33.55n$	536.8
Step3	Final Addition		Final Addition		Final Addition	
	$57.2n$	915.2	$28.6n + 57.2$	514.8	$28.6n$	457.6
Step4	Accumulation		-		-	
	$57.2n$	915.2	-	-	-	-
Critical Path	Accumulation		Final Addition		Hybrid CSA	
	$57.2n$	915.2	$28.6n + 57.2$	514.8	$33.55n$	536.8

B. Delay Model

1) Modeling: In this paper, Sakurai's alpha power law isused to estimate the delay. Because CMOS process is used andthe interconnect delay that is not due to gates related to logicoperation is ignored, was used. The delay by simplifyingthe alpha power law was modeled. Order for easy comparisonswith other architectures, the modeled values identicalto are used in this paper. The normalized input capacitanceand gate delay for the hardware building blocks withthese modeled values are shown in Table V.In Table II, is the ratio of the saturation velocity.Andare the load gate capacitance and gate capacitance of theminimum-area transistor, respectively. is the duration timeand is the falling time of the minimum-area inverter due to. Since delay modeling and its simplification process is notthe focus of this paper, it will not be described in detail here.

2) Delay Analysis: The results of delay modeling for theBooth encoder , the CSA , and the final adderusing Table VI are given in (13)–(16). It represents the select logic delay, buffer delay, andMUX delay, respectively.

$$T_f = \left(\frac{n}{4}\right) T_4(\text{carry}) = 28.6n.$$

$$T_b = T_s + (n+2)T_p + T_m \quad (13)$$
$$T_b = 12.3 + (n+2) \times 10.6 + 47.6 = 10.6n + 81.1 \quad (14)$$
$$T_c = \left(\frac{n}{2}\right) T_2(\text{carry}) = \left(\frac{n}{2}\right) 67.1 + 33.5n \quad (15)$$

(16)

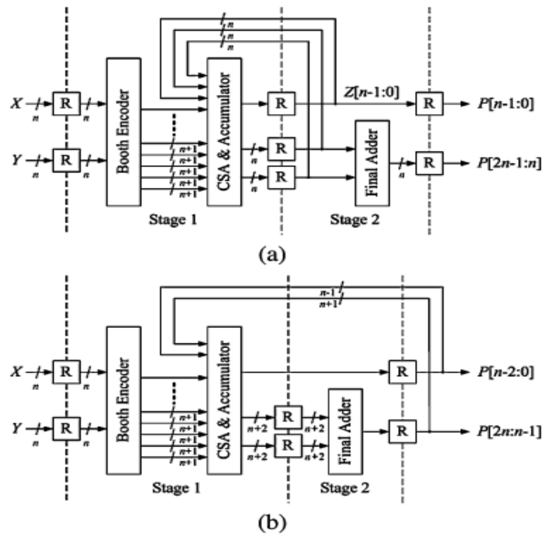


Figure5: pipelined Hardware structre. (a)proposed structure, (b)Elgubaly’s structure

The delays in Table VI were obtained using the hardwareresources in Table II and the gate delays in Table V. FromTable VI, it is easily recognizable that the delay of [6] is considerablylarger than others. The proposed architecture uses thesame Booth encoder and the delay is also identicalto. Because the critical path for the CSA tree isdetermined by the 2-bit CLA, the delay is proportional to it.The proposed architecture has one more 2-bit CLA comparedto [17], as shown in Table II where the delay is greater by 67.1.The number of input bits for the final adder is less by one in ourarchitecture and the delay is also faster by 57.2.If pipelining is applied for each step, the critical path for theproposed architecture is 33.55 and it corresponds to the valueof 536.8 for 16-bit MAC. However, if the performance of the actual outputrate is considered, it can be verified that the proposed architectureis superior. The reason will be explained in detail in the nextsection with the pipelining scheme.Because of the difficulties in comparing other factors,only delay is compared. The sizes of bothMACs were 8 8 bitsand implemented by a 0.35mfabrication process. The delay of ours was 3.94, while it was it 4.26 ns, which meansthat ours improved about 7.5% of the speed performance. Thisimprovement is mainly due to the final adder. The architecture should include a final adder with the size of 2 to performan multiplication. It means that the operational bottleneckis induced in the final adder no matter how much delays are reduced in the multiplication or accumulation step, whichis the general problem in designing a multiplier. However, ourdesign uses -bit final adder, which causes the speed improvement.This improvement is getting bigger as the numberof input bits increases.

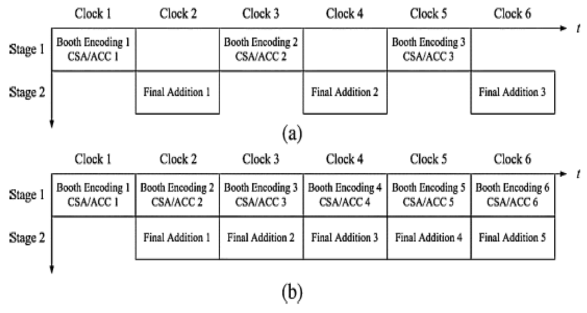
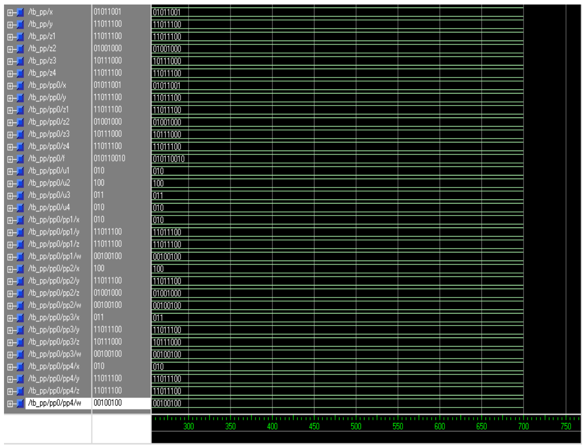
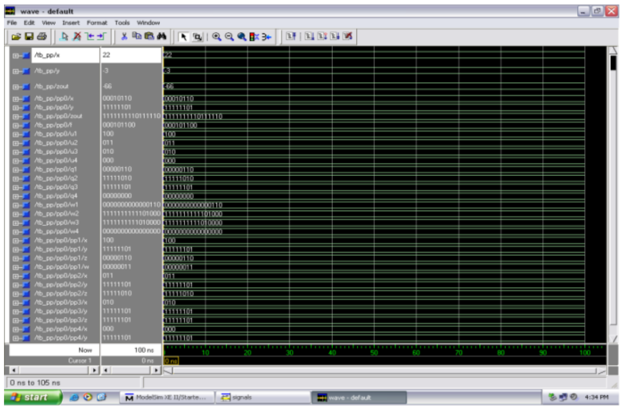

Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
t
Stage 1
Stage 2
Booth Encoding 1 CSA/ACC 1
Booth Encoding 2 CSA/ACC 2
Booth Encoding 3 CSA/ACC 3
Final Addition 1
Final Addition 2
Final Addition 3
(a)
Booth Encoding 4 CSA/ACC 4
Booth Encoding 5 CSA/ACC 5
Booth Encoding 6 CSA/ACC 6
Final Addition 4
Final Addition 5
(b)

Figure 6: Pipelined operational sequence. (a) Elguibaly's operation. (b) Proposed operation.

Experimental Results:



```
/tb_pp/x            01011001     01011001
/tb_pp/y            11011100     11011100
/tb_pp/z1           11011100     11011100
/tb_pp/z2           01001000     01001000
/tb_pp/z3           10111000     10111000
/tb_pp/z4           11011100     11011100
/tb_pp/pp0/x        01011001     01011001
/tb_pp/pp0/y        11011100     11011100
/tb_pp/pp0/z1       11011100     11011100
/tb_pp/pp0/z2       01001000     01001000
/tb_pp/pp0/z3       10111000     10111000
/tb_pp/pp0/z4       11011100     11011100
/tb_pp/pp0/f        010110010    010110010
/tb_pp/pp0/u1       010          010
/tb_pp/pp0/u2       100          100
/tb_pp/pp0/u3       011          011
/tb_pp/pp0/u4       010          010
/tb_pp/pp0/pp1/x    010          010
/tb_pp/pp0/pp1/y    11011100     11011100
/tb_pp/pp0/pp1/z    11011100     11011100
/tb_pp/pp0/pp1/w    00100100     00100100
/tb_pp/pp0/pp2/x    100          100
/tb_pp/pp0/pp2/y    11011100     11011100
/tb_pp/pp0/pp2/z    01001000     01001000
/tb_pp/pp0/pp2/w    00100100     00100100
/tb_pp/pp0/pp3/x    011          011
/tb_pp/pp0/pp3/y    11011100     11011100
/tb_pp/pp0/pp3/z    10111000     10111000
/tb_pp/pp0/pp3/w    00100100     00100100
/tb_pp/pp0/pp4/x    010          010
/tb_pp/pp0/pp4/y    11011100     11011100
/tb_pp/pp0/pp4/z    11011100     11011100
/tb_pp/pp0/pp4/w    00100100     00100100
```



```
/tb_pp/x             22                   22
/tb_pp/y             -3                   -3
/tb_pp/zout          -66                  -66
/tb_pp/pp0/x         00010110             00010110
/tb_pp/pp0/y         11111101             11111101
/tb_pp/pp0/zout      1111111110111110     1111111110111110
/tb_pp/pp0/f         000101100            000101100
/tb_pp/pp0/u1        100                  100
/tb_pp/pp0/u2        011                  011
/tb_pp/pp0/u3        010                  010
/tb_pp/pp0/u4        000                  000
/tb_pp/pp0/q1        00000110             00000110
/tb_pp/pp0/q2        11111010             11111010
/tb_pp/pp0/q3        11111101             11111101
/tb_pp/pp0/q4        00000000             00000000
/tb_pp/pp0/w1        0000000000000110     0000000000000110
/tb_pp/pp0/w2        1111111111101000     1111111111101000
/tb_pp/pp0/w3        1111111111010000     1111111111010000
/tb_pp/pp0/w4        0000000000000000     0000000000000000
/tb_pp/pp0/pp1/x     100                  100
/tb_pp/pp0/pp1/y     11111101             11111101
/tb_pp/pp0/pp1/z     00000110             00000110
/tb_pp/pp0/pp1/w     00000011             00000011
/tb_pp/pp0/pp2/x     011                  011
/tb_pp/pp0/pp2/y     11111101             11111101
/tb_pp/pp0/pp2/z     11111010             11111010
/tb_pp/pp0/pp3/x     010                  010
/tb_pp/pp0/pp3/y     11111101             11111101
/tb_pp/pp0/pp3/z     11111101             11111101
/tb_pp/pp0/pp4/x     000                  000
/tb_pp/pp0/pp4/y     11111101             11111101
             Now     100 ns
        Cursor 1     0 ns                 0 ns
0 ns to 105 ns
```

5. CONCLUSION:

In this paper, a high-accuracy modified Booth multiplier has been proposed. In the proposed multiplier, the booth encoder has reduced the number of partial product array to half the value. The partial product generator has generated the partialproduct array bits. The compression tree has generated the final output product bits. The adder which is used in the implementationof multiplier is Carry Look ahead Adder. The compression tree along with the carry look ahead adder has reduced the hardwareoverhead and power consumption

Future Work:

The current analysis produces high accuracy for the fixed width output product which is of length 2n - bits i.e. n multiplicandand n multiplier produce 2n - bit output product. There is a further need to produce high accuracy for the fixed width of half,quarter, one by eighth and one by sixteenth of the product term bits. The above need is satisfied by means of comparator and asorting network which uses minimum number of logic gates.

REFERENCES :

[1] J.W Chen, R.H Yao, and W.J Wu, "Efficient 2n + 1 Modulo Multipliers," IEEE Transactions on Very Large Scale Integration (VLSI) systems, V. 19, NO. 12, 2011.

[2] Yuan-Ho Chen, Chung-Yi Li, and Tsin-Yuan Chang, *IEEE*, "Area-Effective and Power-Efficient Fixed-Width Booth Multipliers Using GeneralizedProbabilistic Estimation Bias," IEEE Journal on Emerging and Selected topics in Circuits and Systems, V. 1, NO. 3, 2011.

About the Authors:

(1) HARILAL.J,
M.tech, srikottamtulasireddy memorial college of engineering

(2) K. DURGA PRASAD
M.tech,(Phd) Associate professor,
Sri kottamtulasireddy memorial college of engineering