VLSI ARCHITECTURE OF PARALLEL MULTIPLIER– ACCUMULATOR BASED ON RADIX-2 MODIFIED BOOTH ALGORITHM

Mr.M.V.Sathish, Mrs Sailaja
P.G student in ECE Department, Narsaraopet Engg College,
Narsaraopet, Guntur Associate Prof , Dept of ECE,
Narsaraopet Engg College,
Narsaraopet, Guntur
E-mail - **ksatish.maddula@gmail.com**

ABSTRACT **- A new architecture of multiplier-and-accumulator (MAC) for high-speed arithmetic. By combining multiplication with accumulation and devising a hybrid type of carry save adder (CSA), the performance was improved. Since the accumulator that has the largest delay in MAC was merged into CSA, the overall performance was elevated. The proposing method CSA tree uses 1's-complement-based radix-2 modified Booth's algorithm (MBA) and has the modified array for the sign extension in order to increase the bit density of the operands. The proposed MAC showed the superior properties to the standard design in many ways and performance twice as much as the previous research in the similar clock frequency. We expect that the proposed MAC can be adapted to various fields requiring high performance such as the signal processing areas.**

Keywords- Multiplier and accumulator, Carry save adder, Booth algorithm, Standard design

I.INTRODUCTION

Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

A.MULTPLIER AND ACCUMULATOR

A multiplier can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand X and the multiplier Y. The second is adder array or partial product compression to add all partial products. The last is the final addition in which the process to accumulate the multiplied results is included.

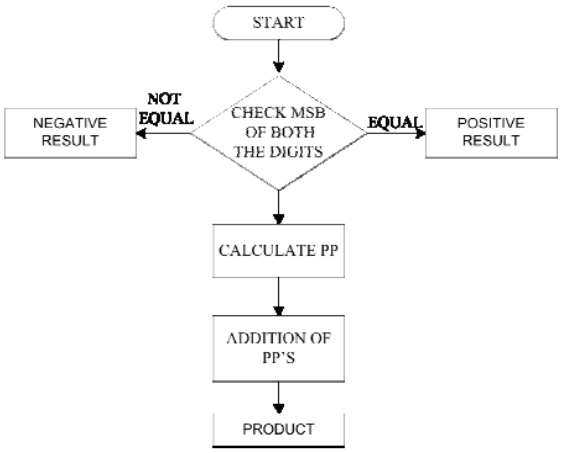

Fig: Signed Multiplication Algorithm

It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y. This is added to the previous multiplication result Z as the accumulation step. If -bit data are multiplied, the number of the generated partial products is proportional to N. In order to add them serially, the execution time is also proportional to N. The architecture of a multiplier, which is the fastest, uses radix-2 Booth encoding that generates partial products. If radix-2 Booth encoding is used, the number of partial products, is reduced to half, resulting in the decrease in Addition of Partial Products step. In addition, the signed multiplication based on 2's complement numbers is also possible. Due to these reasons, most current used multipliers adopt the Booth encoding.

II.PRESENT SCHEMES USED

There are different methods present in this domain such as

- Binary Multiplication
- Array Multiplier
- Multiplier and Accumulator Unit

∅ There are lot of disadvantages in the previous methods, such as low performance when number of bits are increased, there is a chance of mismatch of connection to perform different multiplications, additions with carry.

II.A .BINARY MULTIPLICATION

In the binary number system the digits, called bits, are limited to the set [0, 1]. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial-products simple and efficient. Summing these partial-products is the time consuming task for binary multipliers. One logical approach is to form the partial-products one at a time and sum them as they are generated. Often implemented by software on processors that do not have a hardware multiplier, this technique works fine, but is slow because at least one machine cycle is required to sum each additional partial-product.

				1	1	0	1		Multiplicand
			×	0	1	0	1		Multiplier
	1	1	1	1	1	1	0	1	PP1
	0	0	0	0	0	0	0		PP2
	1	1	1	1	0	1			PP3
+		0	0	0	0	0			PP4
1	1	1	1	1	0	0	0	1 = −15	Product

Fig: Multiplication process

For applications where this approach does not provide enough performance, multipliers can be implemented directly in hardware. The two main categories of binary multiplication include signed and unsigned numbers. Digit multiplication is a series of bit shifts and series of bit additions, where the two numbers, the multiplicand and the multiplier are combined into the result. Considering the bit representation of the multiplicand x = xn-1…..x1 x0 and the multiplier y = yn-1…..y1y0 in order to form the product up to n shifted copies of the multiplicand are to be added for unsigned multiplication. The entire process consists of three steps, partial product generation, partial product reduction and final addition.

II.B.ARRAY MULTIPLIER

The Fast power efficient circuit block switch-off scheme proposes a double switch circuit block switch scheme capable of reducing power dissipation during down time by shortening the settling time after reactivation. The drawbacks of this scheme are the necessity for two independent virtual rails and the necessity for two additional transistors for switching each cell. A power-aware scalable pipelined Booth multiplier design presents a multiplier using Dynamic Range Detection (DRD) unit. is used to select the input operand with a smaller effective dynamic range to yield the Booth codes.

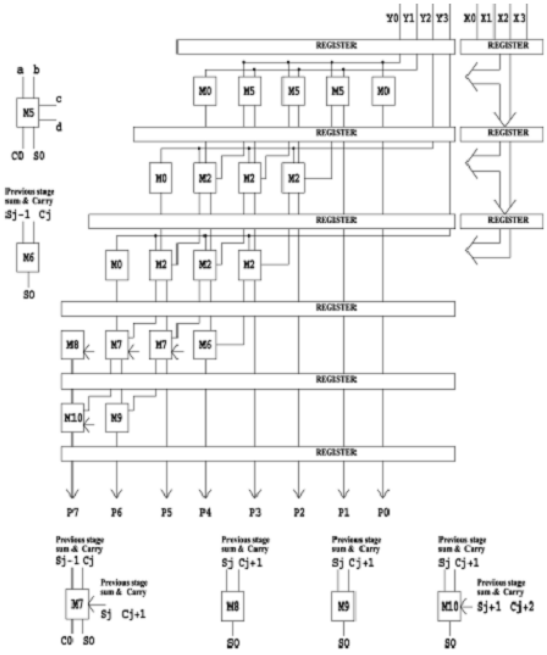

Fig: Array Multiplier

There are three separate Wallace trees for the 4X4, 8X8, and 16X16 multiplications. This will certainly induce area and capacitance penalties. Under a 0.13µm CMOS technology, this design can obtain a 20% power reduction over the conventional multipliers. But there is a 44% n area overhead. Partially guarded computation (PGC) divides the arithmetic units, e.g., adders, and multipliers, into two parts, and turns off the unused part to minimize the power consumption.

II.C MULTIPLIER AND ACCUMULATOR UNIT

The inputs for the MAC are to be fetched from memory location and fed to the multiplier block of the MAC, which will perform multiplication and give the result to adder which will accumulate the result and then will store the result into a memory location. This entire process is to be achieved in a single clock cycle (Weste & Harris, 3rd Ed). The architecture of the MAC unit which had been designed in this work consists of one 16 bit register, one 16-bit Modified Booth Multiplier, 32-bit accumulator. To multiply the values of A and B, Modified Booth multiplier is used instead of conventional multiplier because Modified Booth multiplier can increase the MAC unit design speed and reduce multiplication complexity.

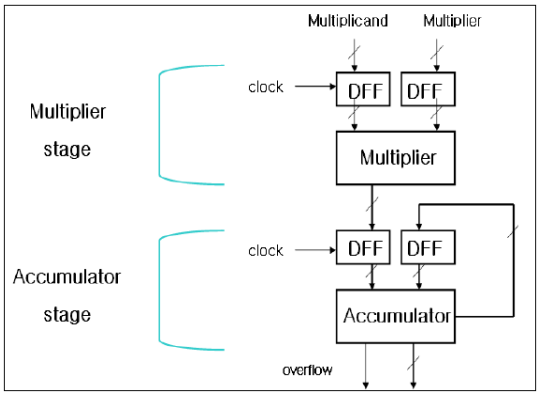

Fig: Simple MAC Architecture

SPST Adder is used for the addition of partial products and a register is used for accumulation.. The operation of the designed MAC unit is as in Equation 2.1. The product of Ai X Bi is always fed back into the 32-bit accumulator and then added again with the next product Ai x Bi. This MAC unit is capable of multiplying and adding with previous product consecutively up to as many as times.

III.PROPOSED MAC

If an operation to multiply two N-bit numbers and accumulate into a 2N-bit number is considered, the critical path is determined by the 2N-bit accumulation operation. if a pipeline scheme is applied for each step in the standard design of Fig ,the delay of last accumulator must be reduced in order to improve the performance of the MAC.

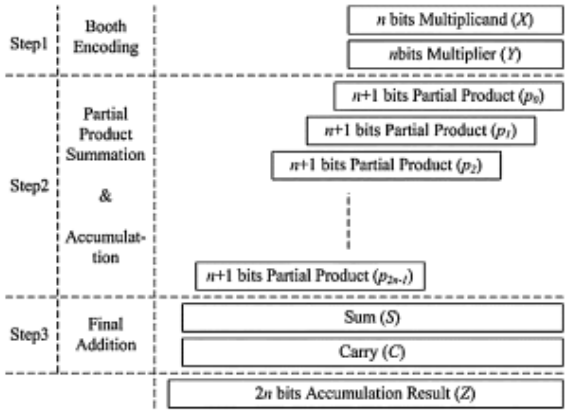

Fig: Arithmetic Operation Of Multiplication And Accumulation

The overall performance of the proposed MAC is improved by eliminating the accumulator itself by combing it with CSA function.if the accumulator has been eliminated, the critical path is then determined by the final adder in the multiplier.The basic method to improve the performance of final adder is to decrease the no of input bits.Inorder to reduce these no input bits,the multiple partial products are compressed into a sum and carry by CSA.the number of bits of sums and carries to be transferred to the final adder is reduced by adding lower bits of sums and carries inadvance within the range in which the overall performance will not be degraded.

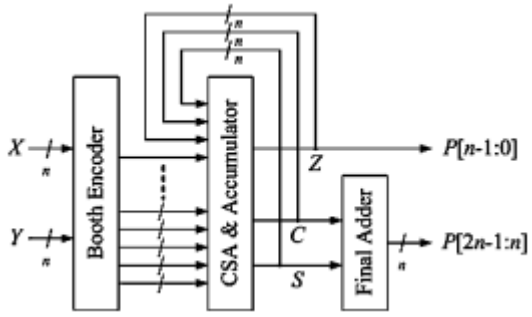

Fig: Hardware Architecture MAC

A 2-bit CLA is used to add the lower bits in CSA.In addition to increase the output rate when pipeling is applied,the sum and carries from the CSA are accumulated instead of outputs from the final adder in the manner that the sum and carry from CSA in the previous cycle are inputted to the CSA.Dut to this feedback of sum and carry,the number inputs to the CSA increases compared to standard design.Inorder to efficiently the increase in the amount of data,a CSA architecture is modified to treat the sign bit.

III.A. CONVENTIONAL HARDWARE - IMPLEMENTATION OF BOOTH'S ALGORITHM

In the circuit for a Booth multiplier the control signals ADD, SUB, ZERO, ADD x 2 and SUB x 2 are generated from the multiplier operand Y using Booth encoding logic. Fig. shows a multiplexer for a single bit position of the multiplicand X. It consists of five trans- mission gate switches, which use the five control signals generated by a Booth encoder, to select the appropriate bit of the multiplicand or its inverse. Subtraction is per- formed using 2's complement addition. This involves adding the inverse of the multiplicand and adding a further '1' at the least significant bit (LSB) position for SUB and at the next most significant bit for SUB x 2.

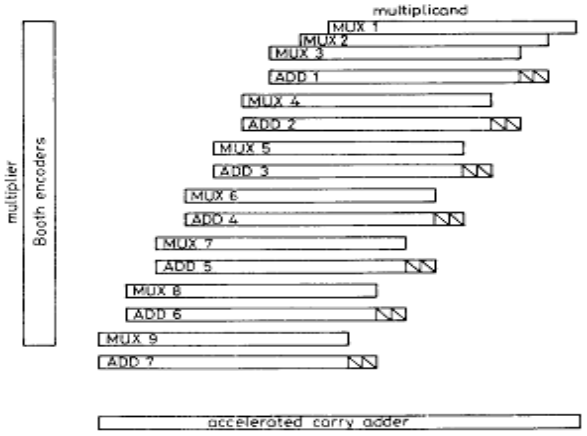

Fig: Standard 16-bit Booth Multiplier Architecture

The standard 16-bit Booth multiplier block diagram, where the product is formed by summing the outputs from the nine multiplexers in sequence. There is one multiplexer for each Booth encoder, and for unsigned arithmetic they are each 18 bits in length. Of the two extra zero-bits at the most significant end, one is required so that the MSB of the multiplicand is zero to indicate a positive number in the internal 2's complement format. The second is to allow for the left-shift in the ADD x 2 and SUB x 2 operations.

The 'add' rows are rows of full adders which produce sum and carry outputs that are then used as inputs to the next row. There is no carry propagation along these rows. The first three multiplexers provide the inputs to the first adder row and thereafter a further multiplexer input, together with the sum and carry from the previous adder row, form the inputs to the next adder row. For the conventional Booth multiplier of Fig. , two extra adders, indicated by diagonal lines, are required at the least significant end of each adder row to add in the subtract flags from the preceding row. It is also necessary to sign extend (replicate the MSB) by two bits at each stage before it is added into the next stage, since each add row is left shifted by two bits with respect to the previous row. These sign extensions at each stage are, in total, equivalent to sign extending the multiplicand to the full width of the result.

III.B.MODIFIED BOOTH 'S ALGORITHM

Multiplication consists of three steps: 1) the first step to generate the partial products; 2) the second step to add the generated partial products until the last two rows are remained; 3) the third step to compute the final multiplication results by adding the last two rows. The modified Booth algorithm reduces the number of partial products by half in the first step. We used the modified Booth encoding (MBE) scheme proposed in. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table I shows the rules to generate the encoded signals by MBE scheme and Fig. 1 (a) shows the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals as shown in Fig.

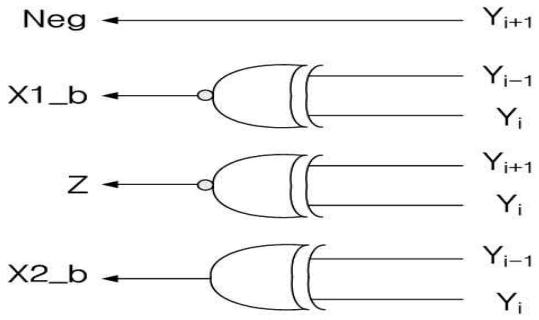

Fig: Booth Encoder

The below fig. shows the generated partial products and sign extension scheme of the 8-bit modified Booth multiplier. The partial products generated by the modified Booth algorithm are added in parallel using the Wallace tree until the last two rows are remained. The final multiplication results are generated by adding the last two rows. The carry propagation adder is usually used in this step.

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier.

The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree

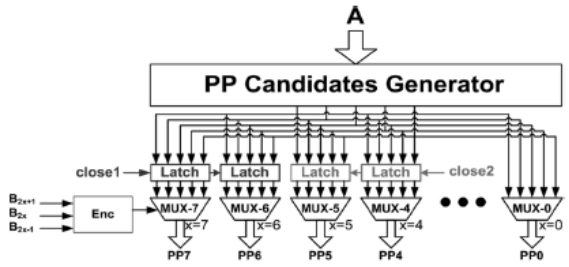

Fig: **SPST equipped modified Booth encoder**

The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2's complement it is negative.

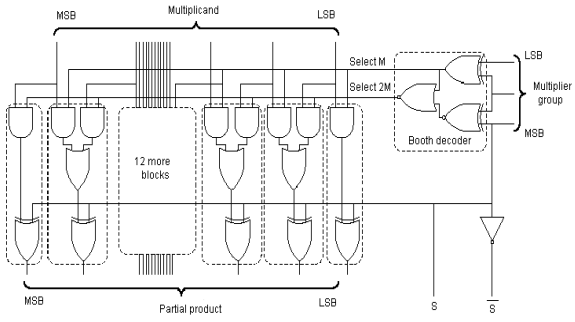

Fig: Booth partial product selector logic

The partial product is generated by doing AND between ‘a’ and ‘b’ which are a 4 bit vectors as shown in fig. If we take, four bit multiplier and 4-bit multiplicand we get sixteen partial products in which the first partial product is stored in ‘q’. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

III.C. SPURIOUS POWER SUPPRESSION TECHNIQUE

The below Figure shows a 16-bit adder/subtractor design example based on the proposed SPST. In this example, the 16-bit adder/subtractor is divided into MSP and LSP at theplace between the 8th bit and the 9th bit. Latches implemented by simple AND gates are used to control the input data of the MSP. When the MSP is necessary, the input data of MSP remain the same as usual, while the MSP is negligible, the input data of the MSP become zeros to avoid switching power consumption. From the derived Boolean equations (1) to (8), the detection logic unit of the SPST is designed as shown in figure which can determine whether the input data of MSP should be latched or not. Moreover, we add three 1-bit to control the assertion of the close, sign, and Carr-ctrl signals in order to further decrease the glitch signals occurred in the cascaded circuits which are usually adopted in VLSI architectures designed for video coding.

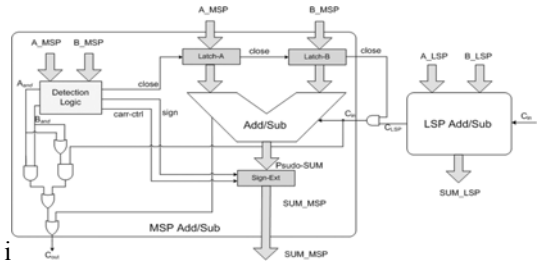

Fig: 16-bit adder/subtractor design

III.D.PROPOSED SPURIOUS POWER SUPPRESSION TECHNIQUE

The SPST uses a detection logic circuit to detect the effective data range of arithmetic units, e.g., adders or multipliers. When a portion of data does not affect the final computing results, the data controlling circuits of the SPST latch this portion to avoid useless data transitions occurring inside the arithmetic units. Besides, there is a data asserting control realized by using registers to further filter out the useless spurious signals of arithmetic unit every time when the latched portion is being turned on. The first implementing

approach of control signal assertion circuit is using registers.

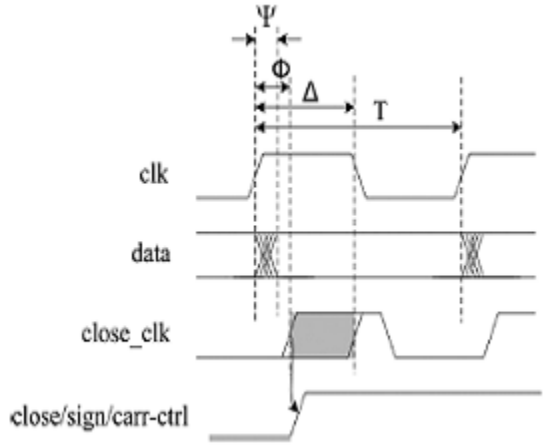

Fig: Timing diagram of the control signals of detection logic circuits after assertion

This is illustrated in figure 4. The three output signals of the detection logic are close,Carr_ctrl, sign. The three output signals the detection logic unit are given a certain amount of delay before they assert. This is demonstrated in the timing diagram shown in figure 5. The delay Φ, used to assert the three output signals, must be set in a range of $\psi < \Phi < \Delta$, where ψ denotes the data transient period and Δdenotes the earliest required time of all the inputs. This will filter out the glitch signals as well as to keep the computation results correct.

Detection logic circuit using an AND gate to assert control signal is shown in figure 6.The timing control of delay Φ in this implementation is slightly different from the one in the first implementation. That is, the range of Φ can be set as $0 < \Phi < \Delta$ to filter out the glitch signals and to keep the computation results correct. This feature allows upper level system to assert the close signal with an arbitrarily short delay closing to the positive edge of the clock signal. This will provides a more flexible controlling space for the delay Φ.

When speed is seriously concerned, this implementing approach enables an extremely high flexibility on adjusting the data asserting time of the SPST equipped multipliers. Therefore the proposed advanced SPST can benefit multipliers on both high speed and low power features.

IV.EXPERIMENT RESULTS

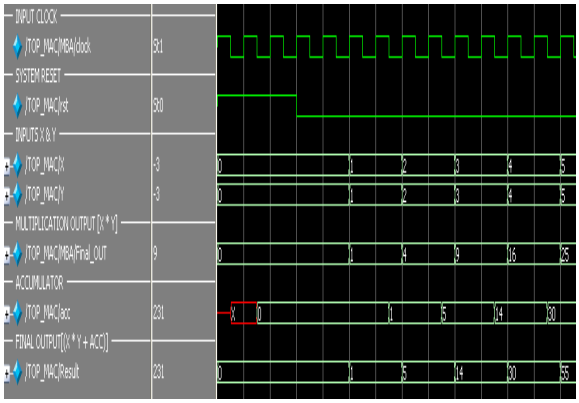

Fig. When we use an Un-signed numbers

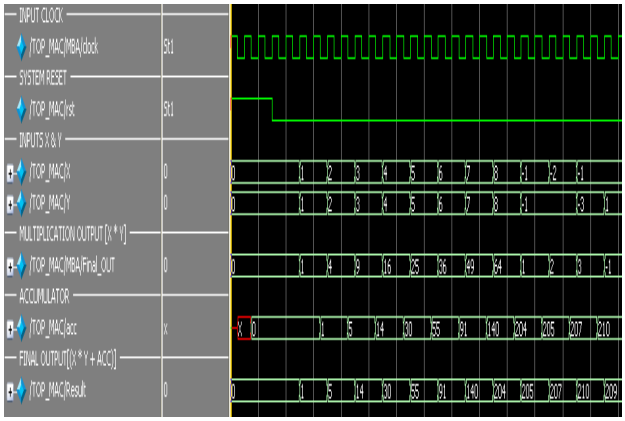

Fig. When we use Signed numbers

ADVANTAGES OF PROPOSED METHOD

The response time is very quick, useful for any no. of bits . Time saving process

CONCLUSION

A 16x16 multiplier-accumulator (MAC) is presented in this work. A RADIX- 4 Modified Booth multiplier circuit is used for MAC architecture. Compared to other circuits, the Booth multiplier has the highest operational speed and less hardware count. The basic building blocks for the MAC unit are identified and each of the blocks is analyzed for its performance. Power and delay is calculated for the blocks. 1-bit MAC unit is designed with enable to reduce the total power consumption based on block enable technique. Using this block, the N-bit MAC unit is constructed and the total power consumption is calculated for the MAC unit.

The power reduction techniques adopted in this work. The MAC unit designed in this work can be used in filter realizations for High speed DSP applications. Table 12 summarizes the results obtained.

REFERENCES

[1]J.J.F.Cavanagh, Digital Computer Arithmetic.

New York: McGraw-Hill,1984.

InformationTechnologyCodingofMovingPictureandAss
ociated Autio, MPEG 2DraftInternationalStandard,
ISO/IEC13818-1,2,3,1994.

[3]JPEG2000PartIFina1119lDraft,ISO/IECJTC1/SC29
WG1.

[4]O.L.MacSorley,“Highspeedarithmeticinbinarycompu
ters,” Proc.IRE,vol.49,pp.6791,Jan.1961.

[5]S.WaserandM.J.Flynn,
IntroductiontoArithmeticforDigital
SystemsDesigners.NewYork:Holt,RinehartandWinston,
1982.

[6]A.R.Omondi,ComputerArithmeticSystems.Englewoo
dCliffs,NJ:Prentice-Hall,1994.

[7]A.D.Booth,“Asignedbinarymultiplicationtechnique,”
Quart.J.Math.,vol.IV,pp.236–240,1952.

[8]C.S.Wallace,“Asuggestionforafastmultiplier,”
IEEETrans.Elec-tronComput.,vol.EC 13,no.1,pp.14–
17,Feb.1964.

[9]A.R.Cooper,“ParallelarchitecturemodifiedBoothmulti
plier,” Proc.Inst.Electr.Eng.G,vol.135,pp.125–128,1988.

[10]N.R.ShanbagandP.Juneja,“Parallelimplementationof
a4 4-bit multiplierusingmodifiedBooth’salgorithm,”
IEEEJ.Solid-StateCir-cuits,vol.23,no.4,pp.1010–
1013,Aug.1988.

[11]G.Goto,T.Sato,M.Nakajima,andT.Sukemura,“A54
54regular structuredtreemultiplier,” IEEEJ.Solid-
StateCircuits,vol.27,no.9,pp.1229–1236,Sep.1992.