International Journal of Instrumentation Control and Automation

Volume 1 | Issue 3 Article 10

October 2011

A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm

Sasibala. P
Department of E.C.E,Chaitanya Engineering College,Visakhapatnam, India., sasibala.patnala@gmail.com

S. Raghavendra
Digital Electronics & Communication, Chaitanya Engineering College, Vishakapatnam(A.P), India, raghavendra454@gmail.com

Recommended Citation

P, Sasibala. and Raghavendra, S. (2011) "A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm," *International Journal of Instrumentation Control and Automation*: Vol. 1 : Iss. 3 , Article 10.
DOI: 10.47893/IJICA.2011.1036
Available at: https://www.interscience.in/ijica/vol1/iss3/10

A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm

P.Sasi Bala & S.Raghavendra

Digital Electronics & Communication, Chaitanya Engineering College, Vishakapatnam(A.P), India
E-mail : Sasibala.patnala@gmail.com, raghavendra454@gmail.com

Abstract **-** In this paper, we proposed 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 proposed 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 CSA propagates the carries to the least significant bits of the partial products and generates the least significant bits in advance to decrease the number of the input bits of the final adder. Also, the proposed MAC accumulates the intermediate results in the type of sum and carry bits instead of the output of the final adder, which made it possible to optimize the pipeline scheme to improve the performance. The proposed architecture was synthesized with 250, 180 and 130 m, and 90 nm standard CMOS library. Based on the theoretical and experimental estimation, we analyzed the results such as the amount of hardware resources, delay, and pipelining scheme. We used Sakurai's alpha power law for the delay modeling. 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.

I. INTRODUCTION

With the recent rapid advances in multimedia and communication systems, real-time signal processings like audio signal processing, video/image processing, or large-capacity data processing are increasingly being demanded. The multiplier and multiplier-and-accumulator (MAC) [1] are the essential elements of the digital signal processing such as filtering, convolution, and inner products. Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) [2] or discrete wavelet transform (DWT) [3]. Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic's determines the execution speed proceedings. and performance of the entire calculation. Because the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined by the multiplier, in general. For high-speed multiplication, the modified radix-4 Booth's algorithm (MBA) [4] is commonly used. However, this cannot completely solve the problem due to the long critical path for multiplication [5]

In general, a multiplier uses Booth's algorithm [3] and array of full adders (FAs), or Wallace tree [5] instead of the array of FAs., i.e., this multiplier mainly consists of the three parts: Booth encoder, a tree to compress the partial products such as Wallace tree, and final adder [2], [4]. Because Wallace tree is to add the partial products from encoder as parallel as possible, its operation time is proportional to $O(\log_2 N)$ where N is the number of inputs. It uses the fact that counting the number of 1'samong the inputs reduces the number of outputs into . In real implementation, many (3:2) or (7:3) counters are used to reduce the number of outputs in each pipeline step.

The mos effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication proceeds a series of additions for the partial products. To reduce the number of calculation steps for the partial products, MBA algorithm has been applied mostly whereWallace tree has taken the role of increasing the speed to add the partial products. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been researched [11]–[13]. Among them, the architectures based on the Baugh–Wooley algorithm (BWA) have been developed and they have been applied to various digital filtering calculations [2]–[4].

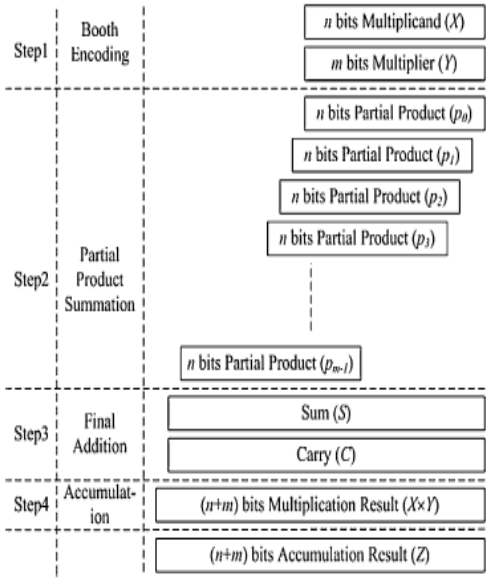


Fig. 1. Basic arithmetic steps of multiplication and accumulation.

In this MAC, the computations of multiplication and accumulation are combined and a hybrid-type CSA structure is proposed to reduce the critical path and improve the output rate. It uses MBA algorithm based on 1's complement number system. A modified array structure for the sign bits is used to increase the density of the operands. A carry look-ahead adder (CLA) is inserted in the CSA tree to reduce the number of bits in the final adder. In addition, in order to increase the output rate by optimizing the pipeline efficiency, intermediate calculation results are accumulated in the form of sum and carry instead of the final adder outputs.

II. OVERVIEW OF MAC

In this section, basic MAC operation is introduced. 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 and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. If the process to accumulate the multiplied results is included, a MAC consists of four steps, as shown in Fig. 1, which shows the operational steps explicitly.

A general hardware architecture of this MAC is shown in Fig. 2. It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y. This is added to the previous multiplication result as the accumulation step.

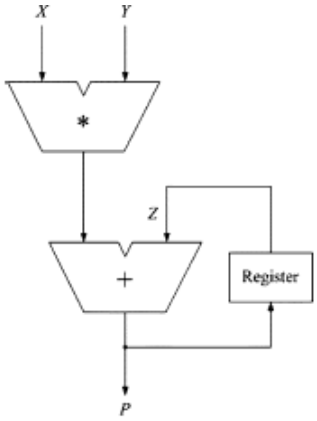


Fig. 2. Hardware architecture of general MAC.

The N-bit 2's complement binary number X can be expressed as

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

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm, it would be [7].

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

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \tag{3}$$

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

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

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

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

Each of the two terms on the right-hand side of (5) is calculated independently and the final result is produced by adding the two results. The MAC architecture implemented by (5) is called the standard design [3].

III. PROPOSED MAC ARCHITECTURE

In this section, the expression for the new arithmetic will be derived 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 operation of the proposed MAC will be proposed.

A. Derivation of MAC Arithmetic

1) Basic Concept:

If an operation to multiply two –bit numbers 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 standard design of Fig. 1, the delay of the last accumulator must be reduced in order to improve the performance of the MAC. The overall performance of the proposed MAC is improved by eliminating the accumulator itself by combining it with the 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 the final adder is to decrease the number of input bits. In order to reduce this number of input bits, the multiple partial products are compressed into a sum and a carry by CSA. The number of bits of sums and carries to be transferred to the final adder is reduced by adding the lower bits of sums and carries in advance within the range in which the overall performance will not be degraded. A 2-bit CLA is used to add the lower bits in the CSA.

2) Equation Derivation*:*

The aforementioned concept is applied to (5) to express the proposed MAC arithmetic. Then, the multiplication would be transferred to a hardware architecture that complies with the proposed concept, in which the feedback value for accumulation will be modified and expanded for the new MAC.

First, if the multiplication in (4) is decomposed and rearranged, it becomes

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

Now, the proposed concept is applied to Z in (5). If Z is first divided into upper and lower bits and rearranged, (5) will be derived. The first term of the right-hand side in (5) corresponds to the upper bits. It is the value that is fed back as the sum and the carry. The second term corresponds to the lower bits and is the value that is fed back as the addition result for the sum and carry

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

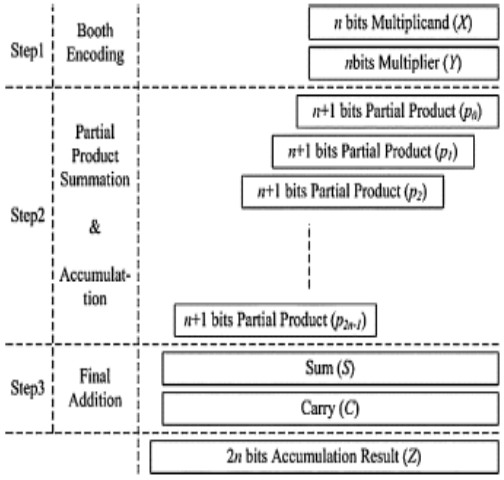


Fig. 3. Proposed arithmetic operation of multiplication and accumulation.

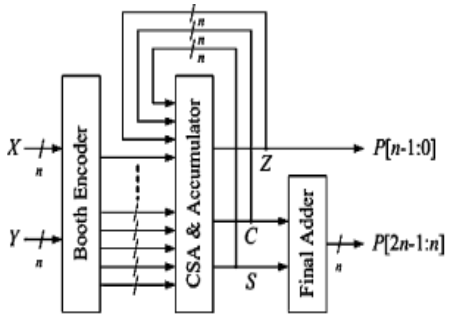


Fig. 4. Hardware architecture of the proposed MAC.

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

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

If (7) and (10) are used, the MAC arithmetic in (5) can be expressed 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 (4), which is the final equation for the proposed MAC. The first parenthesis on the right is the operation to accumulate the first partial product with the added result of the sum and the carry. The second parenthesis is the one to

accumulate the middle partial products with the sum of the CSA that was fed back. Finally, the third parenthesis expresses the operation to accumulate the last partial product with the carry of the CSA

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

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 organized into three steps. When compared with Fig. 1, it is easy to identify the difference that the accumulation has been merged into the process of adding the partial products. Another big difference from Fig. 1 is that the final addition process in step 3 is not always run even though it does not appear explicitly in Fig. 3. Since accumulation is carried out using the result from step 2 instead of that from step 3, step 3 does not have to be run until the point at which the result for the final accumulation is needed.

C. Proposed CSA Architecture

The architecture of the hybrid-type CSA that complies with the operation of the proposed MAC is shown in Fig. 5, which performs 8 8-bit operation. It was formed based on (12). In Fig. 5 S_i, is to simplify the sign expansion and N_i is to compensate 1's complement number into 2's complement number. $S[i]$ and $C[i]$ correspond to the i th bit of the feedback sum and carry. $Z[i]$ is the i th bit of the sum of the lower bits for each partial product that were added in advance and $Z'[i]$ is the previous result. In addition $P_j[i]$, corresponds to the i th bit of the th partial product. Since the multiplier is for 8 bits, totally four partial products $(P_0[7:0] \sim P_3[7:0])$ are generated from the Booth encoder. In (11) $d_0 Y$, and $d_{N/2-1} 2^{N-2} Y$ correspond to $P_0[7:0]$ and $P_3[7:0]$, respectively. This CSA requires at least four rows of FAs for the four partial products. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For an $n \times n$ -bit MAC operation, the level of CSA is $(n/2+1)$. The white square in Fig. 5 represents an FA and the gray square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input.

IV. IMPLEMENTATION AND EXPERIMENT

In this section, the proposed MAC is implemented and analyzed. Then it would be compared with some previous researches. First, the amount of used resources in implementing in hardware is analyzed theoretically and experimentally, then the delay of the hardware is analyzed by simplifying Sakurai's alpha power law [5].

TABLE I
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

TABLE II
CALCULATION OF HADWARE RESOURCE

Component	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bit
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

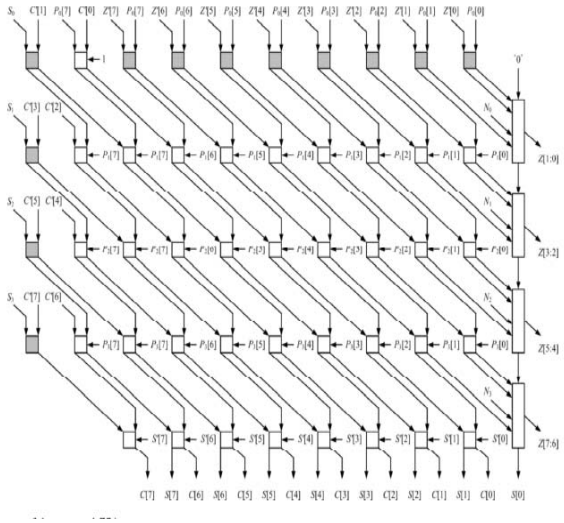


Fig. 5 : Architecture of the proposed CSA tree

A. Hardware Resource

1) Analysis of Hardware Resource:

The three architecture mentioned before are analyzed to compare the hardware resources and the results are given in Table II. In calculating the amount

of the hardware resources, the resources for Booth encoder is excluded by assuming that the identical ones were used for all the designs. The hardware resources in Table II are the results from counting all the logic elements for a general 16-bit architecture. The 90 nm CMOS HVT standard cell library from TSMCwas used as the hardware library for the 16 bits. The gate count for each designwas obtained by synthesizing the logic elements in an optimal form and the resultwas generated by multiplying it with the estimated number of hardware resources.

TABLE III
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

TABLE IV
ESTIMATION OF GATE SIZE BY 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

2) Gate Count by Synthesis:

The proposed MAC and [17] were implemented in register-transfer level (RTL) using hardware description language (HDL). The designed circuits were synthesized using the Design Complier from Synopsys, Inc., and the gate counts for the resulting netlists were measured and summarized in Table IV.

TABLE V
NORMALIZED CAPACITANCE AND GATE DELAY
($\eta = 2, c = C/C_g, t = 0.1 \times T/\gamma$)

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 VI
DELAY TIME ANALYSIS AND COMPARISON

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

B. Delay Model

1) Modeling:

In this paper, Sakurai's alpha power law [20] is used to estimate the delay. Because CMOS process is used and the interconnect delay that is not due to gates related to logic operation is ignored, $\alpha = 1$ was used. The delay by simplifying the alpha power law was modeled in [4]. order for easy comparisons with other architectures, the modeled values identical to [17] are used in this paper. The normalized input capacitance (C_i) and (T_d) gate delay for the hardware building blocks with these modeled values are shown in Table V.

2) Delay Analysis:

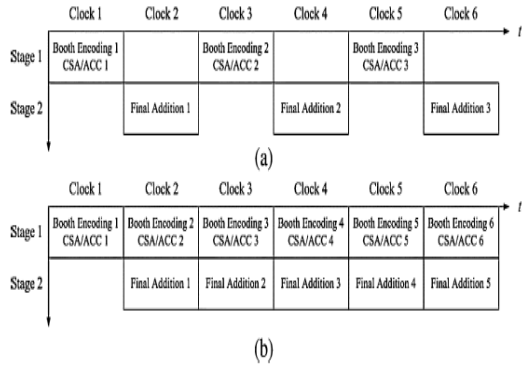
The results of delay modeling for the Booth encoder (T_b), the CSA (T_c), and the final adder (T_f) using Table VI and [17] and [20] are given in (13)–(16). In (13), $T_s, T_p,$ and T_m, and represent the select logic delay, buffer delay, MUX delay, respectively.

$$T_b = T_s + (n+2)T_p + T_m \quad (13)$$

$$T_b = 12.3 + (n+2) \times 10.6 + 47.6 = 10.6n + 81.1 \quad (14)$$

$$T_c = \left(\frac{n}{2}\right) T_2(\text{carry}) = \left(\frac{n}{2}\right) 67.1 + 33.5n \quad (15)$$

$$T_f = \left(\frac{n}{4}\right) T_4(\text{carry}) = 28.6n. \quad (16)$$



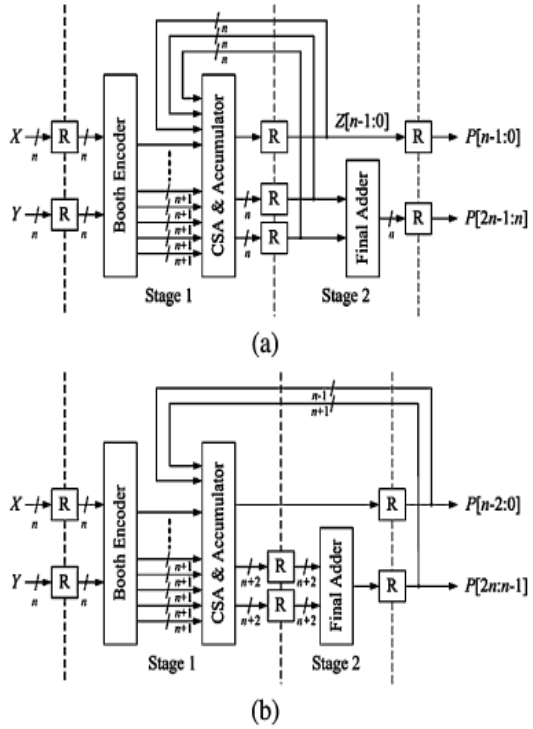


Fig. 6. Pipelined hardware structure. (a) Proposed structure. (b) Elguibaly's structure.

TABLE VII
PIPELINE STAGE

Stage	[6]	[17]	The Proposed
Stage1	$139.95n + 87.1$	$44.15n + 14$	$44.15n + 81.1$
Stage2	$57.2n + 28.5$	$28.6n + 57.2$	$28.6n$

TABLE VIII
PIPELINE AND PERFORMANCE ANALYSIS

Item	[6]	[17]	The Proposed
Output Rate	1 clock	2 clocks	1 clock
Pipeline Delay	$139.95n + 87.1$	$88.3n + 28$	$44.15n + 81.1$

C. Pipelining

1) Stage Analysis:

The pipeline stages were determined based on the delay modeling obtained earlier. step 1 and step 2 in Fig. 3 that correspond to the Booth encoding and CSA operation, respectively, are set to stage 1 and step 3, which correspond to the final adder and are set to stage 2. Such pipeline stage can be organized as shown in Table VII and the clock frequency is determined by this result.

2) Pipeline Structure and Operation:

A hardware incorporates a pipelining scheme to increase the operation speed and ours did too, which is shown in Fig. 6(a), with the one from Elguibaly's scheme [17] in Fig. 6(b) for the purpose of comparison. The difference between the two is because ours carries out the accumulation by feeding back the final CSA outputs rather than the final adder results as in Fig. 6(b).

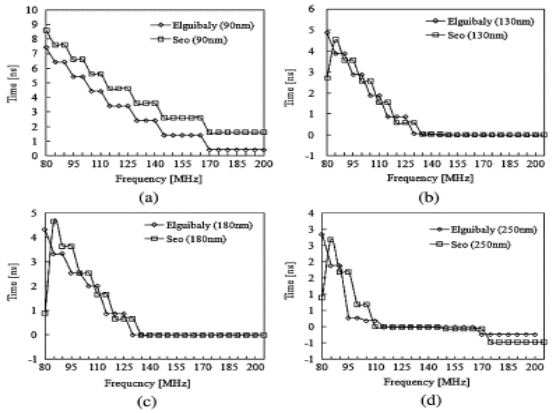


Fig. 8. Timing analysis of the synthesized circuits. (a) 90 nm. (b) 0.13 μm. (c) 0.18 μm. (d) 0.25 μm.

3) Timing Analysis:

After synthesizing using 0.25, 0.18, 0.13, and 0.09 m processes, static timing analyses (STAs) were performed and the results are shown in Fig. 8 graphically. This result is an important result for the physical synthesis and placement and routing (P & R) process in actual chip production. In this figure, the frequencies in x axis mean the target frequencies of the constraints imposed in synthesis and the times in y axis are the timing margins (slacks), i.e., we observed the timing margins increasing target frequency from 80 to 200 MHz.

V. CONCLUSION

In this paper, a new MAC architecture to execute the multiplication-accumulation operation, which is the key operation, for digital signal processing and multimedia information processing efficiently, was proposed. By removing the independent accumulation process that has the largest delay and merging it to the compression process of the partial products, the overall MAC performance has been improved almost twice as much as in the previous work.

REFERENCES

[1] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hill, 1984.

[2] Information Technology-Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard, ISO/IEC 13818-1, 2, 3, 1994.

[3] JPEG 2000 Part I Fina1119l Draft, ISO/IEC JTC1/SC29 WG1.

[4] F. Elguibaly, “A fast parallel multiplier–accumulator using the modified Booth algorithm,” IEEE Trans. Circuits Syst., vol. 27, no. 9, pp. 902–908, Sep. 2000.

[5] T. Sakurai and A. R. Newton, “Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas,” IEEE J.

◈◈◈