International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 7, September - 2012

Improved Performance of Pipeline Scheme by Using MAC and Modified Booth Algorithm

*L. Sri Harish, **T. Venkata Lakshmi

*(Department Of Electronics And Communication, Gudlavalleru Engineering College, Gudlavalleru

** (Department Of Electronics And Communication, Gudlavalleru Engineering College, Gudlavalleru

Abstract

As per in the previous work, the general booth algorithm and CLA's are used in MAC operation for getting the efficient output results through pipeline scheme. But the problem raises by the general booth algorithm and CLA's in previous work by that pipeline scheme performance will be decreased. To rectify these problems this proposed method has been introduced the modified booth algorithm and CSA. The proposed method introduces a parallel operation by generating the sign bits. The methods to reduce dynamic power consumption of a digital F.F.T for D.S.P applications. The overall performance was elevated as follows. The proposed CSA tree uses 1's-complement-based radix-2 modified Booth's algorithm (MBA) and has the modified array for the sign generation 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.

Keywords - **Carry save adder, DSP, MAC, Modified Booth Algorithm, Pipeline scheme.**

I. INTRODUCTION

The nature of the work is introduced to improve the performance of pipeline scheme by using proposed MAC structure and modified booth algorithm. CLA (carry look ahead adder) is used in MAC structure and general booth algorithm is used in multiplier in existing work. By using general booth algorithm delays will occur and partial products increased by using CLA. So, to reduce the partial products modified booth algorithm is used and to reduce the delays .CSA is used in proposed method for speedup the performance of accumulation [1] in MAC structure.

II. MAC

A. Previous MAC

In MAC structure the multiplier can be divided into four operation steps. First step- booth algorithm, second step – partial product summation, third step – final addition, fourth step – accumulation as shown in Fig.1.

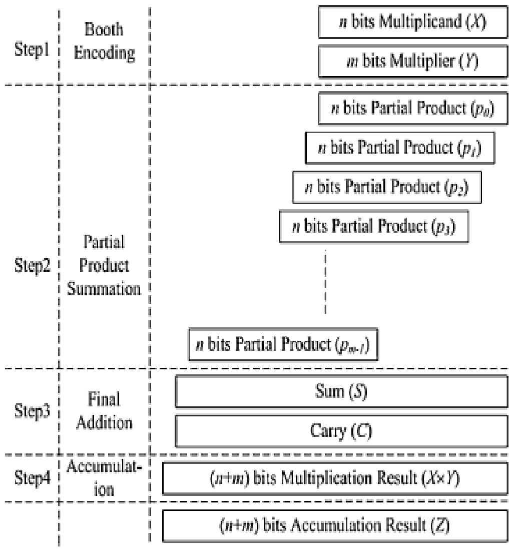
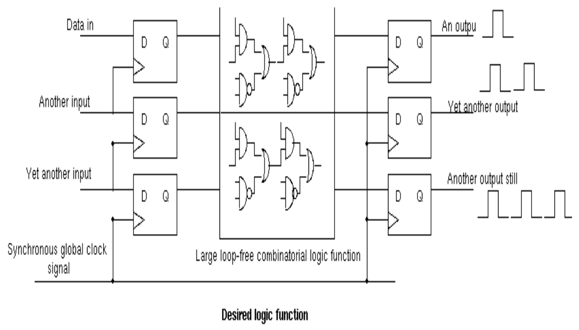

Step1
Booth Encoding
n bits Multiplicand (X)
m bits Multiplier (Y)
Step2
Partial Product Summation
n bits Partial Product (p_0)
n bits Partial Product (p_1)
n bits Partial Product (p_2)
n bits Partial Product (p_3)
n bits Partial Product (p_{m-1})
Step3
Final Addition
Sum (S)
Carry (C)
Step4
Accumulat-ion
(n+m) bits Multiplication Result (X×Y)
(n+m) bits Accumulation Result (Z)

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

Previous internal structure in serial operation



B. General MAC

The general hardware architecture of MAC is shown in Fig.2.

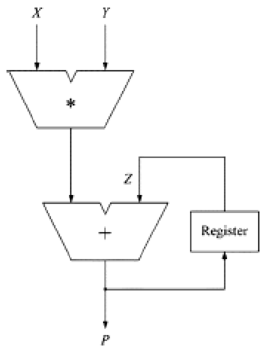


Fig. 2. Hardware architecture of general MAC.

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.

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

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

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].

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

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

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 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

CSA tree is used in proposed MAC Architecture for reducing the delays in accumulation step. 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 [2]. 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.

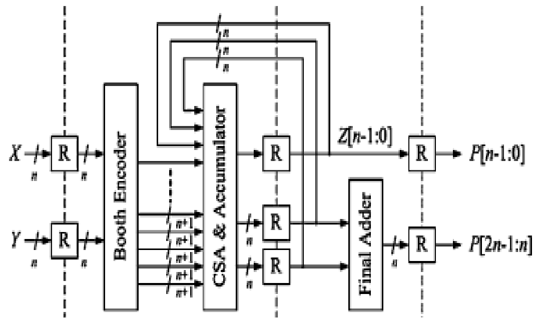


Fig.3 Pipelined hardware architecture of proposed MAC

The number of bits of sums and carries are 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. In addition, to increase the output rate when pipelining is applied, the sums and carries from the CSA are accumulated instead of the outputs from the final adder in the manner that the sum and carry from the CSA previous cycle are inputted to CSA.

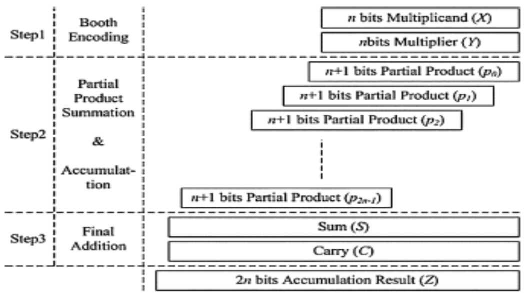
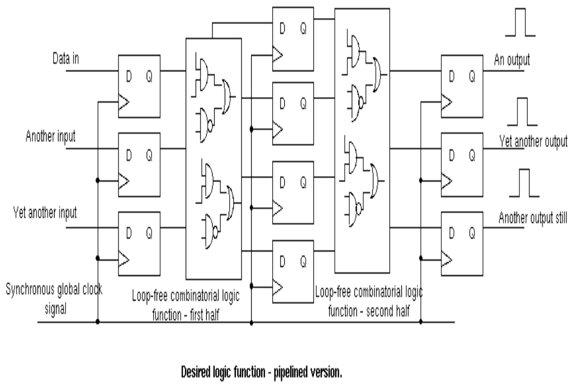

Step1
Booth Encoding
n bits Multiplicand (X)
nbits Multiplier (Y)
Step2
Partial Product Summation & Accumulation
n+1 bits Partial Product (p_0)
n+1 bits Partial Product (p_1)
n+1 bits Partial Product (p_2)
n+1 bits Partial Product (p_{2n-1})
Step3
Final Addition
Sum (S)
Carry (C)
2n bits Accumulation Result (Z)

Fig.4 proposed arithmetic operation of multiplication and accumulation

Proposed internal structure for parallel operation


Data in
Another input
Yet another input
Synchronous global clock signal
D Q
Loop-free combinatorial logic function - first half
Loop-free combinatorial logic function - second half
An output
Yet another output
Another output still
Desired logic function - pipelined version.

IV. 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 x 8-bit operation.

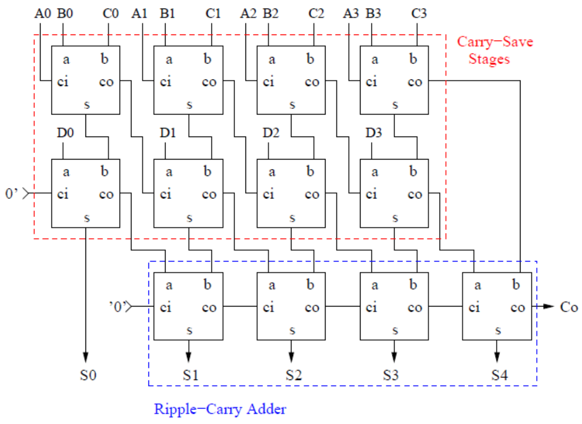


Fig. 5 architecture of proposed CSA tree

In 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 $Z^l[i]$ is the previous result. In addition, $P_j[i]$ corresponds to the i th bit of the j th partial product. Since the multiplier is for 8 bits, totally four partial products ($P_0[7:0]$~$P_3[7:0]$) are generated from the Booth encoder. D_0y 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 [3]. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For an n x 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. The characteristics of the proposed CSA architecture have been summarized and briefly compared with other architectures as shown in table 1. For the number system, the proposed CSA uses 1's complement, but ours uses a modified CSA array without sign extension.

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

A. Hardware resource

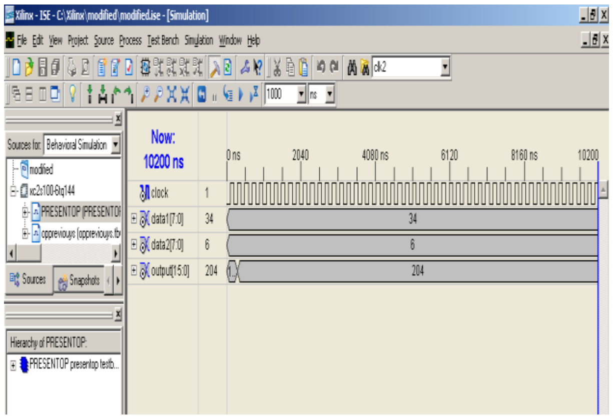
The hardware resources in Table II are the results from counting all the logic elements for a general 16-bit architecture.

TABLE II
CALCULATION OF HADWARE RESOURCE

Component	[6]		[17]		The Proposed	
	General	16 bits	General	16 bits	General	16 bi
FA	$(\frac{n^2}{2}+n)$	964.8	$(\frac{n^2}{2}+2n+3)$	1092.1	$(\frac{n^2}{2}+\frac{n}{2})$	911.
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

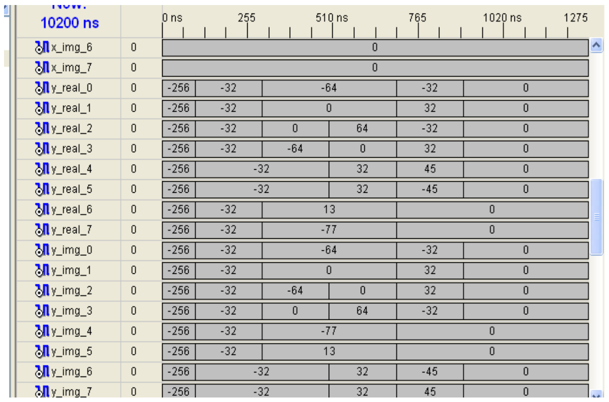
V. SIMULATION RESULTS

Modified booth algorithm


Xilinx - ISE - C:\Xilinx\modified\modified.ise - [Simulation]
File Edit View Project Source Process Test Bench Simulation Window Help
Sources for: Behavioral Simulation
modified
xc2s100-6tq144
PRESENTOP (PRESENTOP
opprevious (opprevious.tf
Sources
Snapshots
Hierarchy of PRESENTOP:
PRESENTOP presentop testb

```
Now:
10200 ns       0 ns   2040   4080 ns   6120   8160 ns   10200
clock          1
data1[7:0]     34    34
data2[7:0]     6     6
output[15:0]   204   204
```

Proposed method implemented in FFT processor



```
10200 ns        0 ns     255     510 ns     765     1020 ns    1275
x_img_6    0    0
x_img_7    0    0
y_real_0   0    -256 | -32 | -64 | -32 | 0
y_real_1   0    -256 | -32 | 0 | 32 | 0
y_real_2   0    -256 | -32 | 0 | 64 | -32 | 0
y_real_3   0    -256 | -32 | -64 | 0 | 32 | 0
y_real_4   0    -256 | -32 | 32 | 45 | 0
y_real_5   0    -256 | -32 | 32 | -45 | 0
y_real_6   0    -256 | -32 | 13 | 0
y_real_7   0    -256 | -32 | -77 | 0
y_img_0    0    -256 | -32 | -64 | -32 | 0
y_img_1    0    -256 | -32 | 0 | 32 | 0
y_img_2    0    -256 | -32 | -64 | 0 | 32 | 0
y_img_3    0    -256 | -32 | 0 | 64 | -32 | 0
y_img_4    0    -256 | -32 | -77 | 0
y_img_5    0    -256 | -32 | 13 | 0
y_img_6    0    -256 | -32 | 32 | -45 | 0
y_img_7    0    -256 | -32 | 32 | 45 | 0
```

VI. CONCLUSION

The proposed method reduces the partial products by using modified booth algorithm and reduces the delays by using proposed parallel CSA Architecture. So, by using this two methods the performance of pipeline scheme has been improved.

REFERENCES

[1] Wen-Chang Yeh and Chein-Wei Jen, “High-speed Booth encoded parallel multiplier design,” IEEE Trans. on Computers, vol. 49, issue 7, pp. 692-701, July 2000.

[2] Jung-Yup Kang and Jean-Luc Gaudiot, “A simple high-speed multiplier design,” IEEE Trans. on Computers, vol. 55, issue 10, Oct. pp. 1253-1258, 2006.

[3] Shiann-Rong Kuang, Jiun-Ping Wang and Cang-Yuan Guo, “Modified Booth multipliers with a regular partial product array,” IEEE Trans. On Circuit and Systems, vol.56, Issue 5, pp. 404-408, May 2009.

[4] Li-rong Wang, Shyh-Jye Jou and Chung-Len Lee, “A well-structured modified Booth multiplier design,” Proc. of IEEE VLSI-DAT, pp. 85-88, April 2008.

[5] A. A. Khatibzadeh, K. Raahemifar and M. Ahmadi, “A 1.8V 1.1GHz Novel Digital Multiplier,” Proc. of IEEE CCECE, pp. 686-689, May 2005.

[6] S. Hus, V. Venkatraman, S. Mathew, H. Kaul, M. Anders, S. Dighe, W. Burleson and R. Krishnamurthy, “A 2GHZ 13.6mW 12x9b mutiplier for energy efficient FFT accelerators,” Proc. of IEEE ESSCIRC, pp. 199-202, Sept. 2005.

[7] Hwang-Cherng Chow and I-Chyn Wey, “A 3.3V 1GHz high speed pipelined Booth multiplier,” Proc. of IEEE ISCAS, vol. 1, pp. 457-460, May 2002.