Multiplier Design and Performance Estimation with Distributed Arithmetic Algorithm

M. Suhasini, K. Prabhu Kumar & P. Srinivas

Department of Electronics & Comm. Engineering,
Nimra College of Engineering & Technology, Ibrahimpatnam, Vijayawada, India
E-mail : suhasinikrishna14@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 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. Moreover, depending on data switching activity statistically reduce the power consumption

Keywords - *Field-programmable gate array (FPGA), FIR filter; DA algorithm.*

I. INTRODUCTION

In signal processing, a **finite impulse response (FIR)** filter is a filter whose impulse response (or response to any finite length input) is of *finite* duration, because it settles to zero in finite time. The output *y* of a linear time invariant system is determined by convolving its input signal *x* with its impulse response *b*. For a discrete-time FIR filter, the output is a weighted sum of the current and a finite number of previous values of the input.

A new architecture of multiplier-and-accumulator (MAC) for high-speed arithmetic is obtained 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.

Moreover, depending on data switching activity statistically reduce the power consumption. Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications.

It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs.

It can also be extended to other sum functions such as complex multiplies, fourier transforms and so on.

Multiplication is a mathematical operation that at its simplest is an abbreviated process of adding an integer to itself a specified number of times. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result (product). In elementary school, students learn to multiply by placing the multiplicand on top of the multiplier.

The multiplicand is then multiplied by each digit of the multiplier beginning with the rightmost, Least Significant Digit (LSD). Intermediate results (partial products) are placed one atop the other, offset by one digit to align digits of the same weight. The final product is determined by summation of all the partial-products.

Although most people think of multiplication only in base 10, this technique applies equally to any base,

including binary. Fig.1 shows the data flow for the basic multiplication technique just described. Each black dot represents a single digit.

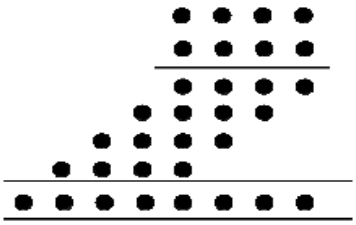

Fig. 1 Basic Multiplication

II. RELATED WORK

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off.

In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. 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.

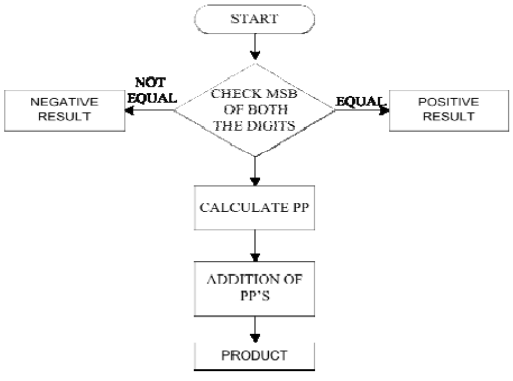

Fig.2 Signed Multiplication algorithm

The first part is dedicated to the generation of partial products, and the second one collects and adds them.

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

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.

III. DISTRIBUTED ARITHMETIC ALGORITHM

The DA algorithm was initially proposed by Crosier in 1973. It attracted attention again after the FPGA LUT (Look-up Table) was invented by Minx in the early 90s of last century and effectively applied in the design of FIR filters [1]. The principle of DA algorithm is as follows [1]. The output of linear time-invariant system is shown as Eq. (1).

$$Y = \sum_{m=1}^{M} A_m X_m$$

Where Am is a fixed factor, Xm is the input data (X*m* <1). Xm can be expressed as Eq. (2) using the binary complement.

$$X_m = -X_{m0} + \sum_{m-1}^{N-1} X_{mn} 2^{-n}$$

$$= \sum_{n=1}^{N-1} \sum_{n=1}^{M} A_m X_{mn} 2^{-2} + \sum_{m=1}^{M} A_m (-X_{m0})$$

In Eq. (3), as the value of xmn is 0 or 1, there are 2^M kinds of different results of $\sum_{m-1}^{M} A_m x_{mn}$

If we construct a LUT which can store all the possible combination of values [2], we can calculate the value of 2^M in advance and store them in the LUT. Using xmn as the LUT address signal, the shifting (2-1 operation) and adding operation are carried out on the output of the LUT. Then $\sum_{m-1}^{M} A_m x_{mn}$ can be realized through N-1 cycles and the result of multiplication accumulation can be achieved directly. So the complicated multiplication-accumulation operation is converted to the shifting and adding operation. The parallel computing is adopted to improve the speed of calculation.The complicated multiplication-accumulation operation is converted to the shifting and adding operation when the DA algorithm is directly applied to realize linear time invariant system. However, the scale of the LUT will increase exponentially with the coefficient. If the coefficient is small, it is very convenient to realize through the rich structure of FPGA LUT; while the coefficient is large, it will take up a lot of storage resources of FPGA and reduce the calculation speed. Meanwhile, the N-1 cycles also result in the too long LUT time and the low computing speed. The paper presents the improvement and optimization of the DA algorithm aiming at the problems of the configuration in the coefficient of FIR filter, the storage resource and the calculating speed, which make the memory size smaller and the operation speed faster to improve the computational performance.

IV. IMPROVED DESIGN

From Eq. (2), Xm can be expressed as Eq. (4).

$$X_m = \frac{1}{2}[X_m - (-X_m)]$$

Where the - X m can be expressed as Eq. (5) according to the binary complement operation [3].

$$-X_m = -X_{m0} + \sum_{n=1}^{N-1} x_{mn} 2^{-n} + 2^{-(N+1)}$$

Put Eq. (5) and Eq. (2) into Eq. (4), Eq. (6) can be achieved.

$$-X_m = \frac{1}{2}[-(-X_{m0} - \overline{x_{m0}} + \sum_{n=1}^{N-1} (x_{mn} - \overline{x_{mn}}) 2^{-n} - 2^{-(N+1)}]$$

For convenience, two variables are defined as follows:

$$\phi_{m0} = -(x_{m0} - \overline{x_{m0}})$$

$$\phi_{mn} = -(x_{mn} - \overline{x_{mn}})$$

In which, as the value of *mn x* is 0 or 1, so the value of *mn* and *m0* is ±1. Then Eq. (6) can be expressed as Eq. 7

$$X_m = \frac{1}{2}[\sum_{n=1}^{N-1} \phi_{mn} 2^{-n} - 2^{-(N-1)}]$$

.

$$X_m = \frac{1}{2}[\sum_{n=1}^{N-1} \sum_{m=1}^{M} A_m \phi_{mn} 2^{-n} - \sum_{m=1}^{M} A_m 2^{-(N-1)}]$$

$$\sum_{m=1}^{M} A_m \phi_{mn}$$

As there are 2M different kinds of results of and the value of *mn* is ±1, so the results show positive and negative symmetry property. If the positive and negative sign are not considered, there are only2M-1 different kind of results and the size of storage will reduce by half. Through the algorithm optimization, Eq. (8) can be simplified as Eq. (9).

Then the size of memory is 2M/4+2M/4=2M/4+1. Compared with the memory size which is 2M-1 before optimizing, its memory scale is only 2-3M/4+2 times of the original. Through the algorithm improvement, the hardware resource is reduced and the operation speed is improved. The simplified hardware circuit structure is shown in Fig. 3.

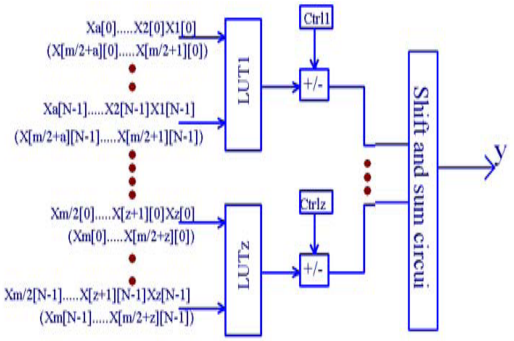

Fig.3 Hardware Circuit

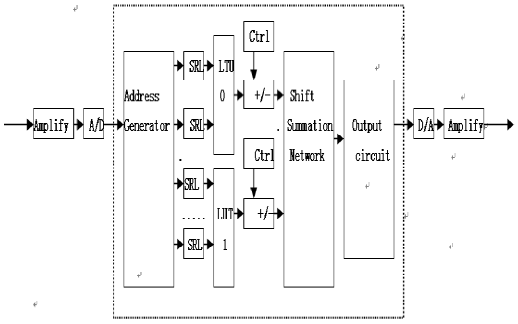

Fig.4 FIR filter Structure

When using the DA algorithm to implement the linear time-invariant system, the algorithm is optimized according the method of section 2. The prestoring value corresponding to the upper half of the memory address of LUT storage will be the negative of the lower half and then the LUT reduces by half using symmetry. The address maker circuit generates the LUT address.

According to result of the improvement and optimization, the LUT is divided into two 4-input LUTs and the address maker circuit divides the input signals into four segments in accordance with the 4-input LUT.

The speed of signal sampling under the control of the FPGA can be adjusted. The data buffer can be established according to the order of the filter. As the designed filter is a 16th-order one,so the sampled serial data can be sent to the 20 bits serial-in parallel-out shift register, and then the data is divided and sent to the LUT in turn. As the coefficient is amplified 216 times, the obtained result is reduced by the output circuit accordingly.

The implementation of filter based on FPGA is realized on EP2C5T144C8 chips by using of IP core, the DA algorithm and the improved DA algorithm separately. The results of compilation and test show that the needed LE is 1522, 1269 and 776 respectively when IP core, the DA algorithm and the improved DA algorithm is used to implement filter. The improved algorithm can greatly reduce the hardware resource and improve the throughput efficiency.

V. CONCLUSION & RESULTS

This paper states FIR filter Developed with DA algorithm is the better of all the multipliers. The complicated multiplication-accumulation operation is converted to the shifting and adding operation when the DA algorithm is directly applied to realize FIR filter. The simulation results of various other multipliers are compared in many factors. Aiming at the problems of the best configuration in the coefficient of FIR filter, the storage resource and the calculating speed, the DA algorithm is optimized and improved in the algorithm structure, the memory size and the look-up table speed.

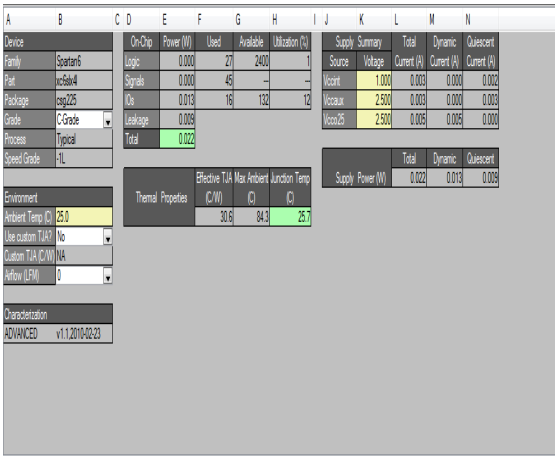

Fig 5.a: Power Report of Booths multiplier

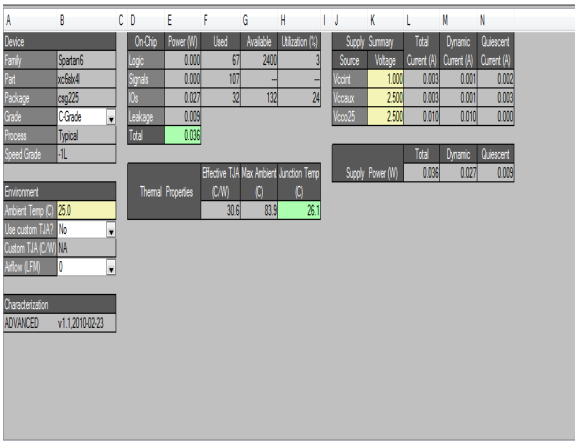

Fig 5.b: Power Report of Wallace tree multiplier

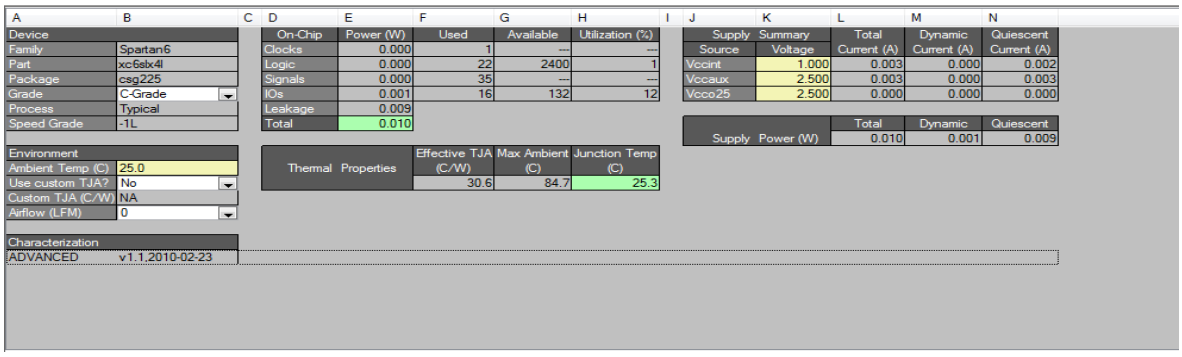

Fig 5.c: Power Report of DA FIR multiplier

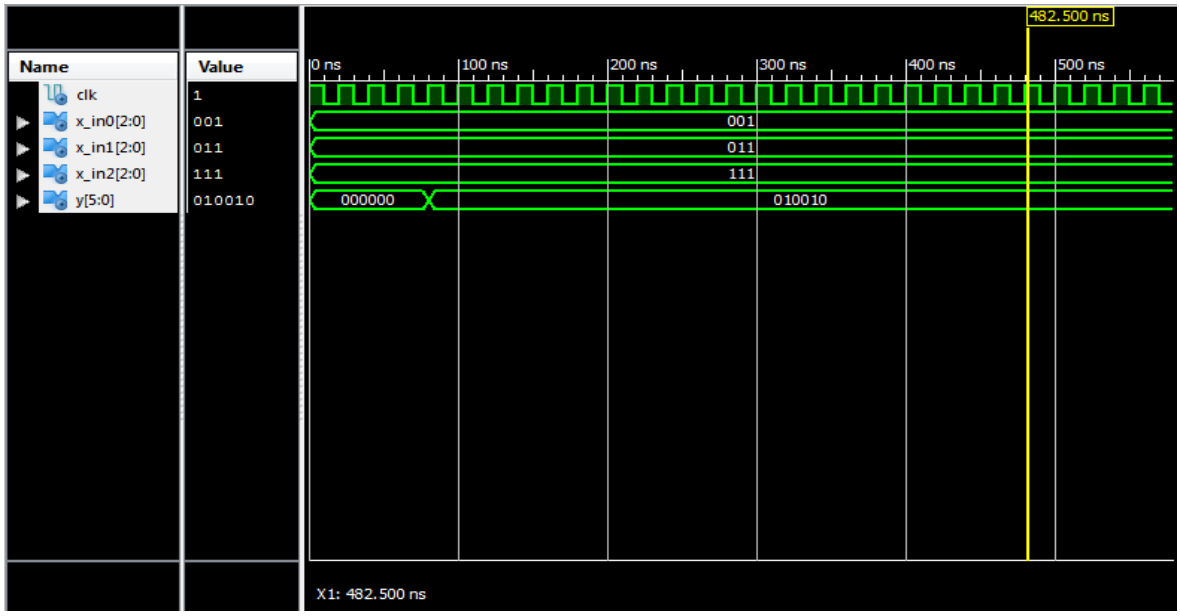

Fig 5.d: Simulation Results of DA FIR multiplier

The arithmetic expression has clear layers of derivation process and the circuit structure is reasonable, which make the memory size smaller and the operation speed faster.

To construct a complete multiplier, a final adder has to be provided to convert the carry-save outputs from the DA multiplier into a single integer. Carry-select adders have been used in [l] to exploit the unimodal input arrival profile.

As this profile is flattened by pipeline registers, using other forms of fast adders, e.g., carry-lookahead or carry-skip seems to make more sense. Moreover, The adder must also be pipelined at the clock rate of the Wallace tree to prevent it from creating a bottleneck at the output of the multiplier. This can be achieved easily at the expense of registers. Fortunately, the required number of pipeline stages is small compared to that required for the Wallace tree. Hence the register cost is not expected to be prohibitive.

The design improves greatly compared to the conventional FPGA realization and it can be flexibility applied to implement high-pass, low-pass and band-stop filters by changing the order and the LUT coefficient. Fig. 5 Shows the simulation results and power reports of the DA multiplier algorithm.

VI. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

REFERENCES:

[1] L. Zhao, W. H. Bi, F. Liu, “Design of digital FIR bandpass filter using distributed algorithm based on FPGA,” Electronic Measurement Technology, 2007,vol. 30, pp.101-104.

[2] P. Girard, O. Héron, S. Pravossoudovitch, and M. Renovell, “Delay Fault Testing of Look-Up Tables in SRAM-Based FPGAs,” Journal of Electronic Testing, 2005, vol. 21, pp. 43-55.

[3] H. Chen, C. H. Xiong, S. N. Zhong, “FPGA-based efficient programmable polyphase FIR filter,” Journal of Beijing lnsititute of Technology, 2005, vol. 14, pp. 4-8.

[4] Y. T. Xu, C. G. Wang, J. L. Wang, “Hardware Implementation of FIR Filter Based on DA Algorithm,” Journal of PLA University of Science and Technology, 2003, vol. 4, pp. 22-25.

[5] D. Wu, Y. H. Wang, H. Z. Lu,“ Distributed Arithmetic and its Implementation in FPGA,” Journal of National University of Defense Technology, 2000, vol. 22, pp.16-19.

[6] L. Wei, R. J. Yang, X. T. Cui, "Design of FIR filter based on distributed arithmetic and its FPGA implementation, " Chinese Journal of Scientific Instrument, 2008, vol. 29, pp. 2100-2104.

[7] W. Zhu, G. M. Zhang, Z. M. Zhang, "Design of FIR Filter Based on Distributed Algorithm with Parallel Structure," Journal of Electronic Measurement and Instrument, 2007, vol. 21, pp. 87-92.

[8] W. Wang, M. N. S. Swamy, M. O. Ahmad, "Novel Design and FPGA Implemention of DA-RNS FIR Filters," Journal of Circuits Systems and Computers, 2004, vol. 13, pp. 1233-1249.

[9] Rainer Dorsch et al., "Accuniulator Based Deterministic BIST," IEEE International Test Conference, 1998, pp.412-421.

[10] M. Nagamatsu et al., "A 15ns 32 x 32-bit CMOS Multiplier with an Improved Parallel Structure," Proc. CICC, pp.10.3.1-4, May 1989.

[11] Y. Harata et al., "A High Speed Multiplier Using Redundant Binary Adder Tree," IEEE JSSC, vol.sc22, no.1, pp.28-34, Feb. 1987.

[12] D. Gizopoulos et al., "Effective Built-In Self-Test for Booth Multipliers," IEEE Design & Test of Computers, July-September 1998, Vo1.15, No.3, pp. 105-1 11.

Authors Profile:

M.Suhasini

M.Suhasini Is pursuing her masters degree in VLSI system design. She is interested in the filed of modern communication system and developments in wireless technology.

K.Prabhu Kumar
M.Tech

K.Prabhu Kumar is an Assistant professor in electronics and communication engineering in Nimra College of Engg & Tech. He got his M.Tech from K.L university

P.Srinivas

P.Srinivas Is A young and dynamic Associate Professor of electronics and communication Engineering.. He Got his M.Tech From Acharya Nagarjuna University. He Worked in various reputed Engineering Colleges as an Assoc Professor and Head Of The Department.

◈◈◈