A HIGH-PERFORMANCE FIR FILTER ARCHITECTURE FOR FIXED AND RECONFIGURABLE APPLICATIONS

E. Mary Priyadarshini, P. Manimegalai, S. Chellaganeshavalli

Karpagam Academy of Higher Education, India. E,mail: darshinipriyabe@gmail.com

ABSTRACT

The efficient distributed arithmetic (DA)-based approaches for high-throughput reconfigurable implementation of finite impulse response (FIR) filters whose filter coefficients change during runtime. Conventionally, for reconfigurable DA-based implementation of FIR filter, the lookup tables (LUTs) are required to be implemented in RAM; and the RAM-based LUT is found to be costly for ASIC implementation. Therefore, a shared-LUT design is proposed to realize the DA computation. Instead of using separate registers to store the possible results of partial inner products for DA processing of different bit positions, registers are shared by the DA units for bit slices of different weightage.The Proposed presents high speed digital Finite Impulse Response (FIR) filter relying on Booth multiplier and Carry Select Adder (CSLA) Using Parallel Pipelining Architecture. Adder has three architectures such as basic CSLA using RCA (Ripple Carry Adder), CSLA using BEC (Binary to Excess-1 Converter) and CSLA using D-latch. In this paper we propose 4-tap FIR Filter architecture using 16-bit CSLA using D-latch and 8-bit Booth tree multiplier. These multipliers and adders are used for high speed operation of digital FIR filter.

Keywords: CSLA, RCA, BEC, D-latch and Booth multiplier.

I.INTRODUCTION

Filters are used in wide range of applications such as multimedia and DSP (Digital Signal Processing) etc. Most of DSP computations involve the use of multi-ply accumulate operations and therefore the design of fast and efficient multiplier and adder units are important. More ever, the demand for portable applications of DSP architectures has dictated the need for low power designs. Digital Finite Impulse Response filter (FIR) has performs lot of arithmetic and logical operations. In general, arithmetic and logic operation modules such as adder and multiplier modules, consume more chip area and delay for each operation is more. Input bit width of the modules is quite important in design para-meter such as less delay and resource efficient filters. The resource utilization and power of digital FIR filter circuit is reduced by optimization of taps and bit width of input signal and filters coefficients. The adders and Booth multipliers are applied for filters to eliminate power consumption due to unwanted data transitions. In they presented a multipliers technique, based upon add and shift operation and common sub expression elimination for low area and high-speed implement-tation of FIR filters. Finite impulse response filters are widely used in various DSP applications.

II. PROPOSED STRUCTURE

MATHEMATICAL RELATION FOR FIR FILTER:

Filters are very important part of digital signal proce-ssing applications. Filters have two uses, one is signal separation and other one is signal restoration. Signal separation is used only when the signal is contaminated with noise or other unwanted signals. Signal restoration is used only when the signal has been distorted. In general filtering is described by simple convolution operation such as

$$Y(n) = x(n) * f(n) = \sum_{k=0} f(k)x(n-k)$$
$$= \sum_{k=0} x(k)f(n-k)$$

The digital filters are commonly linear time invariant filters. The straight forward way of implementing LTI Finite Impulse Response filter is finite convolution of input series x(n) with impulse response coefficients is given by

$$Y(n) = x(n)\ f(n)$$
$$= \sum_{k=0}^{L-1} f(k)x(n-k)$$

Where L is the length of FIR filter, h(n) is filters impulse response coefficients, x(n) is input sequence and y(n) is output of FIR filter. The above equations can also expressed in Z domain as

Y(z) = x(z) H(z)

A. REGULAR CARRY SELECT ADDER

CSLA compromise between Ripple Carry Adder (RCA) and carry look ahead adder (CLA). The main disadvantage of regular CSLA structure is the large area due to the multiple set of ripple carry adder. One set of RCAs for carry as 0 and another set of RCA for carry as 1. The Fig. 1 shows [1] the 16-bit carry select adder. It is divided into five pairs of groups with different bit size RCA. The carry out calculated from the last stage that is least significant bit stage is used to select the calculated values of the output carry and sum. The selection of carry for the next stage is done by using a multiplexer.

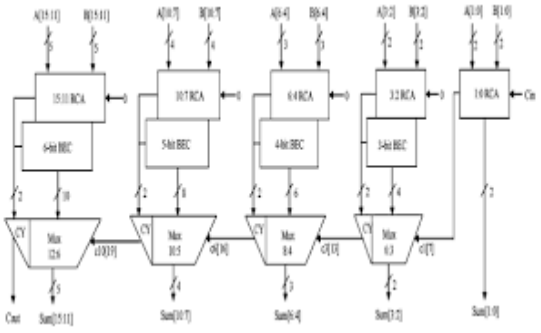


Fig.1. Regular CSLA.

B. CSLA USING BEC

The regular CSLA is not area efficient and delay for the operation is more because it uses multiple set of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input and then the final sum and carry are selected by the multiplexers. To

\

avoid this problem, the regular CSLA structure is modified using n-bit

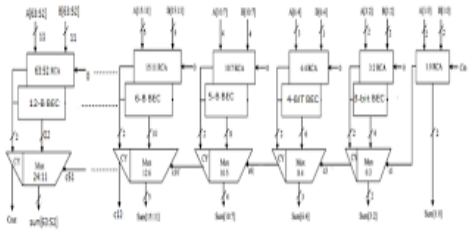


Fig.2. 16-bit CSLA using BEC

Binary to Excess-1 code converter (BEC) to improve the speed of operation [3]. To improve the speed of operation we can use the Binary to Excess-1 Converter (BEC) instead of RCA with Cin=1 in the regular CSLA to achieve less delay. The Fig. 2 shows the structure of 16- bit CSLA using BEC

C.BINARY TO EXCESS-1 CONVERTER

One input of the 8:4 multiplexer 4-bit input such as (B3, B2, B1 and B0) and another input of the multi-plexer is the BEC output. This can produce two possi-ble results in parallel one is direct output and other is BEC output. There is no change in direct output. The multiplexer is used to select either the BEC output or the direct inputs according to the control signal Cin. If BEC input is X then O/P is "X+1". Compared with regular CSLA this can produces less delay for its operation and resource utilization also less

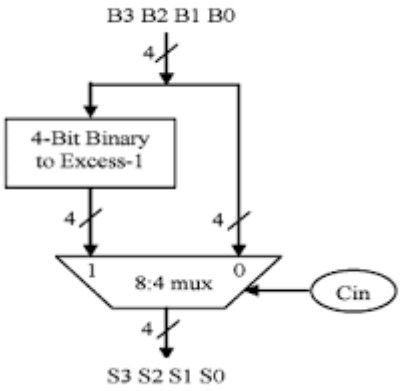
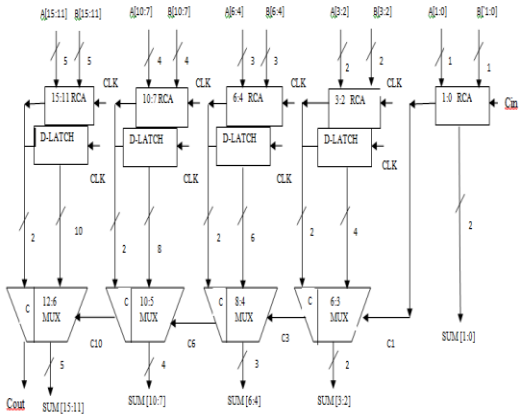


Fig.3. Operation of 4-bit BEC



D. CSLA USING D-LATCH

Here initially when en=1, the output of the RCA is Connected to the input of D-Latch and the output of the Dlatch follows the input and given as an input to the multiplexer [2]. When en=0, there is no change in output which is same as that of previous state therefore the output from the RCA is directly given as an input to the multiplexer It can produce output only when the clock is en=1. Figure 4 shows the architecture of CSLA using D LATCH. This can utilize only less resource for its operation and delay for the operation is less.

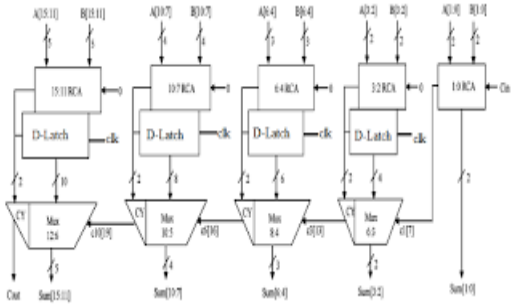
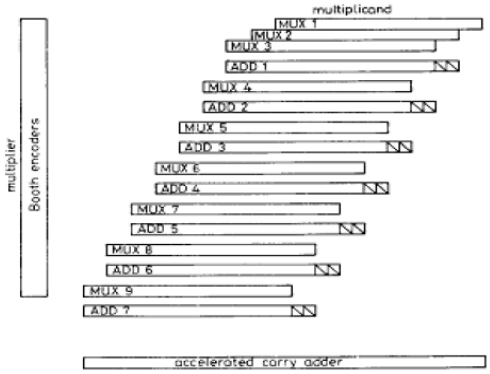


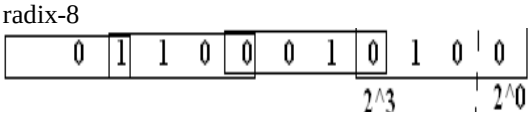
Fig.4. 16-bit CSLA using D-LATCH

III. BOOTH MULTIPLIER

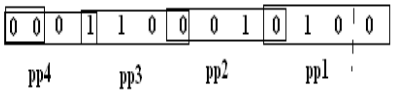
Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invent-ted by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck college in Bloomsbury, London. Booth used desk calculators that were faster at shifting than adding and created the algo-rithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.



Radix-8 encoding applies the similar algorithm to radix-4, but here we take quartets of bits instead of triplets. Each quartet is coded as a signed-digit using the table 1 Consider two inputs of 10 bits each, x=0010010100(148) and y=0110001010(394) Append a 0 to the lsb of the y and group the bits according to radix-8



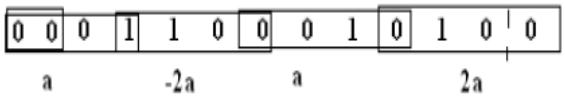
Denote each group as a partial product and add necessary bits to complete group of y i.e.,if msb is 1 add 1's and if msb is 0 add 0's.Denote it as multiplier b



Append two 0's or two 1's to msb of the x by checking msb of x and denote it as Multiplicand

Denote partial products groups of b according to radix-8 encoding table given below

PP bits of b	Partial Products
0 0 0 0	0a
0 0 0 1	+1a
0 0 1 0	+1a
0 0 1 1	+2a
0 1 0 0	+2a
0 1 0 1	+3a
0 1 1 0	+3a
0 1 1 1	+4a
1 0 0 0	-4a
1 0 0 1	-3a
1 0 1 0	-3a
1 0 1 1	-2a
1 1 0 0	-2a
1 1 0 1	-1a
1 1 1 0	-1a
1 1 1 1	0a



Obtain partial products by applying radix-8 encoding on multiplicand a

0 0 0 0 1 0 0 1 0 1 0 0 = a

Obtain 2a by shifting a to left once.4a is obtained by shifting a left twice

0 0 0 1 0 0 1 0 1 0 0 0 = 2a

To obtain –a we need to perform 2's complement on a

1 1 1 1 0 1 1 0 1 1 0 0 = -a(2's complement)

Obtain -2a by shifting –a once to left. -4a is obtained by shifting –a left twice

1 1 1 0 1 1 0 1 1 0 0 0 = -2a

To generate 3a we only have to add 2a and a and similarly -3a is obtained by adding-2a and –a. Here 3a,-3a are denoted as hard multiples.vpp2 is placed under pp1 after leaving three places from lsb of pp1,pp3 is placed after leaving six places from lsb of pp1 and so on.All remaining locations are filled with 0's.Extend sign bits of all the partial products according to corresponding msb's i.e., extend 1's for msb 1 and 0's for msb 0.

MULTIPLICATION

This is the most important stage, product of the mantissa bits is calculated. The multiplication of mantissa bits is performed in the following stages.

Generation of Partial Products

The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by considering certain number of bits of the multiplier at a time, thereby achieving a speed advantage over other multiplier architectures. This algorithm is valid for both signed and unsigned numbers. It can handle signed binary multiplication by using 2"s complement representation. For generating the partial products Radix-8 Modified Booth"s Algorithm is used. Since the multiplier and multiplicand comprises of 24 bits, this algorithm will generate 8 partial products. The shortcoming of Radix 2 Booth algorithm is that it becomes inefficient when there are isolated 1's. For example, 001010101(decimal 85) gets reduced to 01-11- 11-11-1(decimal 85), requiring eight instead of four operations.001010101(0) recoded as 011111111, requiring 8 instead of 4 operations.

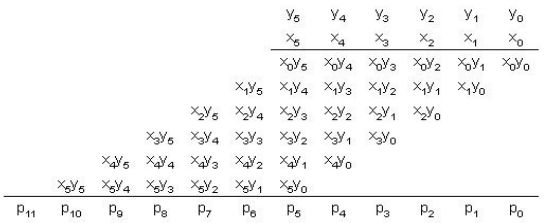


Fig.5. Booth multiplier

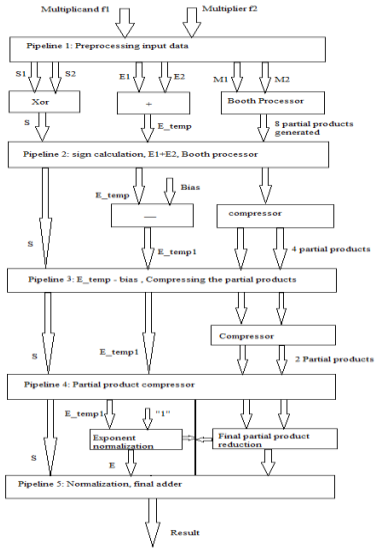
IV. PIPELINING

A pipeline is a set of data processing elements connected in series, so that the output of one element is the input of the next one. It is divided into segments and each segment can execute its operation concurrently with the other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. The final results of each instruction emerge at the end of the pipeline in rapid succession. The pipeline technique is widely used to improve the performance of digital circuits. As the number of pipeline stages is increased, the path delays of each stage are decreased and the overall performance of the circuit is improved In order to enhance the performance of the multiplier, five pipelining stages are used to divide the critical path thus increasing the maximum operating frequency of the multiplier. Five pipelining stages mean that there is latency in the output by five clocks. The pipelining stages are embedded at the following locations:

i. After the Pre-processing of the Multiplicand and Multiplier.
ii. After the Exponent Adder and Generation of 8 Partial Products.

iii. After subtracting the Bias and Compressing the partial Products to 4.
iv. After Compressing the Partial Products to 2.
v. After Normalization and Final Carry Propagate Adder.

Comparing with 3-stage pipelined multiplier, the frequency is increased as the pipeline stages are increased shows the various pipeline stages in the Multiplier



V. DESIGN OF FIR FILTER FIR

FIR filters are used in signal processing applications. Filter structure consists of delay element, adder and multiplier elements. The adder is replaced using CSLA with D-latch architecture and multiplier is replaced using Booth multiplier is shown in fig (6).

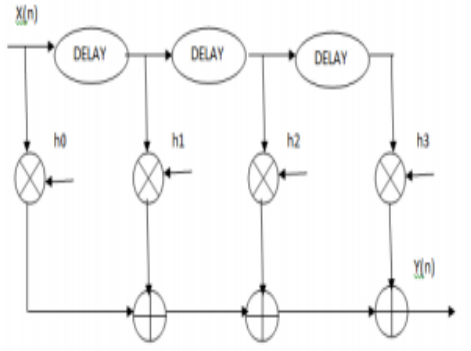


Fig.6. Structure of 4-tap FIR filter

Here X(n) is input filter coefficients and h0, h1, h2 and h3 are transfer function coefficients Y(n) is output filter coefficient.

V. SIMULATION AND SYNTHESIS RESULTS

We perform the simulation and synthesis and summarize the results of all adders and multiplier. Functional verification of all the adders and multiplier are performed and these modified architectures are applied in 4-tap FIR filter finally results are summarized.

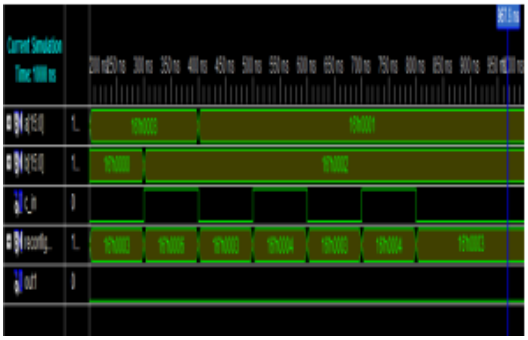


Fig.7. Output for regular CSLA

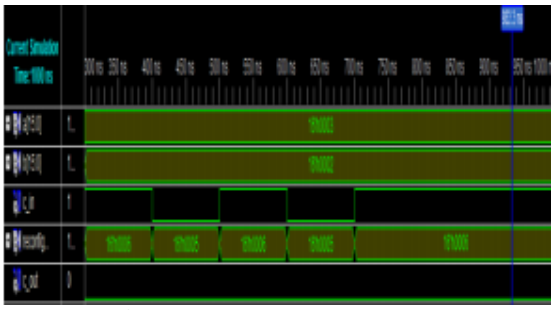


Fig.8. Output for CSLA using BEC

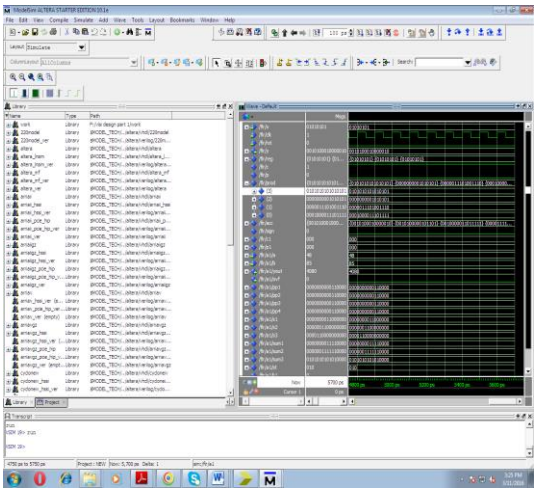


Fig.10. Simulation output for FIR filter using CSLA with D-latch and booth multiplier

Fig. 11 shows the output for 4 tap FIR filter using CSLA with D-latch and booth multiplier. Here X is 8-bit input coefficient that is multiplied with 8-bit multiplier filter coefficients h0, h1, h2 and h3 produces 16-bit output. Both are sum together and produce filter output Y. Here multiplier uses booth multiplier and adder unit uses CSLA with D-latch.

VI. COMPARITIVE RESULTS

After observation of simulation waveforms, synthesis is performed for calculation of delay and area and comparison of adder architectures are made in terms of area and delay and listed in the below table.

Parameters	Regular CSLA	CSLA using BEC	CSLA using D latch
Number of gates used	32	30	28
Destination paths	670	445	380
Delay(ns)	7.132	7.291	5.805

From the above table, it is clear that delay and resource utilization is less in CSLA with D-latch compared with CSLA with BEC and regular CSLA architectures.

VII. CONCLUSION

Thus, the models of CSLA are designed and are Implemented in VHDL using Xilinx 14.5 ISE tool and the results are compared in terms of delay and area. The CSLA with D-Latch proves to be the High Speed and Low area CSLA. And the booth multiplier also occupies less area. This adder and multiplier units are enhanced for FIR filter applications. Thus, a high speed and low power FIR filter can be designed using an Improved CSLA with D-latch and booth multiplier. The Improved FIR filter architecture is therefore, high speed and area efficient for VLSI hardware implementtation.

REFERENCES

[1] Kavita, Jasbir Kaur, Design and Implementation of an Efficient Modified Booth Multiplier using VHDL, International Conference on Emerging Trend in Engineering and Management Vol. 3 (2013).

[2] Yunnan Chang, Janardhan H.K. Parhi, Low power digit-serial multipliers, IEEE International symposium on circuits and systems, June i3-12 (1997)

[3] Bedrij, O. J., Carry-select adder, IRE Trans. Electron. Computer Pp. 340–344 (1962)