Design and Implementation of enhanced BZ-FAD Multipliers for DSP Applications

P.Sadhasivam[1], Dr.M.Manikandan[2],
Research Scholar[1], Associate Professor[2],
Department of Electronics,
St.Peter's University[1], Anna University[2], TN, India.

Abstract: This paper presents a comparative study of Field Programmable Gate Array (FPGA) implementation of standard multipliers using Verilog HDL. Multiplier is a good candidate for digital signal processing (DSP) applications such as finite impulse response (FIR) and discrete cosine transforms (DCT) etc. This paper proposed a new gate into the Bz-fad (Bypass Zero Feed A directly) multiplier to get low area, delay, and power than the conventional all other multipliers.

Keywords: Fast Multiplier, High Speed Adder, Power-Delay Product, Field Programmable Gate Array

1. Introduction

A binary multiplier is an electronic circuit used in digital electronics, such as computer, to multiply two binary numbers. It is built using binary adders. A variety of computer arithmetic techniques can be used to implement a digital multiplier [1].Most techniques involve computing a set of partial products, and then summing the partial products together. For high speed multiplications, a huge number of adders or compressors are to be used to perform the partial product addition 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. 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.

ICGPC 2014
St.Peter's University, TN, India.

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. The major speed limitation in any adder is in the production of carries. Basically, carry save adder is used to compute sum of three or more n-bit binary numbers [2].

2. Low Power Multiplier: BZ-FAD

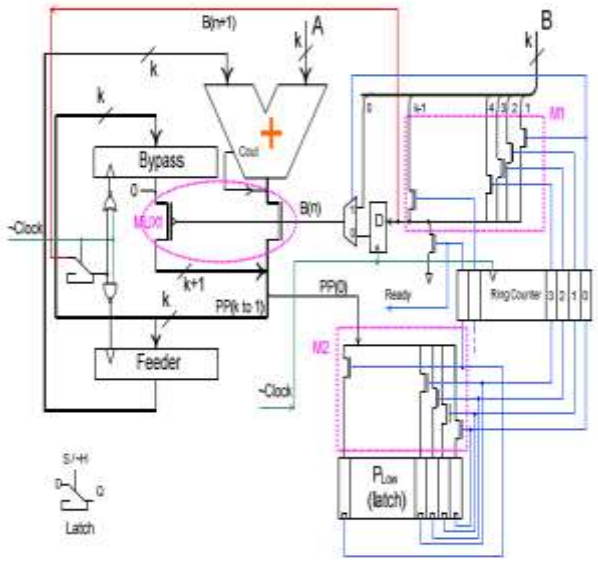

Figure.1 The architecture of BZ-FAD multiplier

1) Shift of the B Register In the traditional architecture (see Fig 1), to generate the Partial product, B (0) is used to decide between A and 0. If the bit is '1', as should be added to the previous partial product, whereas if it is '0', no addition operation is needed to generate the partial product, Hence, I n each cycle, register B should be

shifted to the right so that its right bit appears at B(0); this operation gives rise to some switching activity [7]. To avoid this, in the proposed architecture (Fig1) a multiplexer (M1) with to appear in IEEE Trans. on VLSI Systems, 2008 2one-hot encoded bus selector chooses the hot bit of Bin each cycle. A ring counter is used to select B (n) in the nth cycle. As will be seen later, the same counter can be used for block M2as well. The ring counter used in the proposed multiplier is noticeably wider (32 bits vs. 5 bits for a 32-bit multiplier) than the binary counter used in the conventional architecture; therefore an ordinary ring counter, if used in BZ-FAD, would raise more transitions than its binary counterpart in the conventional architecture. To minimize the switching activity of the counter, we utilize the low-power ring counter, which is described in the next section.

2) Reducing Switching Activity of the Adder In the conventional multiplier architecture (Fig 1), in each cycle, the current partial product is added to A (when B (0) is one) or to 0(when B (0) is zero). This leads to unnecessary transitions in the adder when B (0) is zero. In these cases, the adder can be bypassed and the partial product should be shifted to the right by one bit. This is what is performed in the proposed architecture which eliminates unnecessary switching activities in the adder. As shown in Fig 1, the Feeder and Bypass registers are used to bypass the adder in the cycles where B (n) is zero. In each cycle, the hot bit of the next cycle (i.e., B (n+ 1)) is checked. If it is 0, i.e., the adder is not needed in the next cycle, the Bypass register is clocked to store the current partial product. If B (n+ 1) is 1, i.e., the adder is really needed in the next cycle, the Feeder register is Clocked to store the current partial product which must be fed to the adder in the next cycle. Note that to select between the Feeder and Bypass registers we have used NAND and NOR gates which are inverting logic, therefore, the inverted clock (~Clocking Fig 2) is fed to them. Finally, in each cycle, B (n)determines if the partial product should come from the Bypass register or from the Adder output. In each cycle, when the hot bit B (n) is zero, there is no transition in the adder since its inputs do not change. The reason is that in the previous cycle, the partial product has been stored in the Bypass register and the value of the Feeder register, which is the input of the adder, remains unchanged. The other input of the adder is A, which is constant during the multiplication. This enables us to remove the multiplexer and feed input A directly to the adder, resulting in a noticeable power saving. Finally, note that the BZ-FAD architecture does not put any constraint on the adder type. In this work, we have used the ripple carry adder which has the least average transition per addition among the look ahead, carry skip, carry-select, and conditional sum adders [8].

3) Shift of the PP Register In the conventional architecture, the partial product is shifted in each cycle giving rise to transitions. Inspecting the multiplication algorithm reveals that the multiplication may be completed by processing the most significant bits of the partial product, and hence, it is not necessary for the least significant bits of the partial product to be shifted. We take advantage of this observation in the BZ-FAD architecture. Notice that in Fig 1 for P Low, the lower half of the partial product, we use klatches (for a k-bit multiplier). These latches are indicated by the dotted rectangle M2in Fig 2. In the first cycle, the least significant bit, PP (0), of the product becomes finalized and is stored in the rightmost latch of P Low. The ring counter output is used to open (unlatch) the proper latch. This is achieved by connecting the S/~H line of the nth latch to the nth bit of the ring counter which is '1' in the nth cycle. In this way, the nth latch samples the value of the nth bit of the final product (Fig 1). In the subsequent cycles, the next least significant bits are finalized and stored in the proper latches. When the last bit is stored in the leftmost latch, the higher and lower halves of the partial product form the final product result. Using this method, no shifting of the lower half of the partial product is required. The higher part of the partial product, however, is still shifted. Comparing the two architectures, BZ-FAD saves power for two reasons: first, the lower half of the partial product is not shifted, and second, this half is implemented with latches instead of flip-flops. Note that in the conventional architecture (Figure 1) the data transparency problem of latches prohibits us from using latches instead of flip-flops for forming the lower half of the partial product. This problem does not exist in BZ-FAD since the lower half is not formed by shifting the bits in a shift register.

3. Dadda Multiplier Architecture

The Dadda Tree multiplier has the same general stages as the Wallace Tree. However, unlike the Wallace Tree, Dadda multipliers do not attempt to reduce as many partial products in each layer but rather, perform as few reductions as possible. This makes the Dadda multiplier less costly in the reduction stage but contain longer numbers in each stage, requiring larger carry-save adders (CSA).For this design, formation of partial products was done using the same method as in the Wallace Tree multipliers.

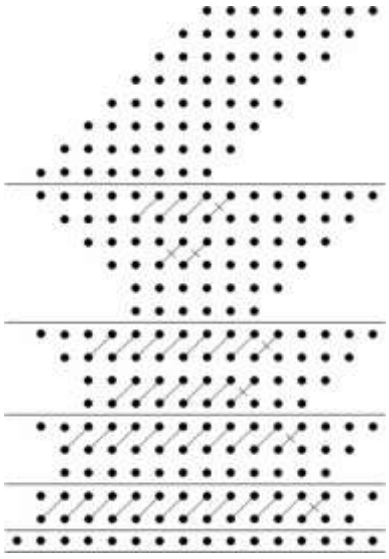

Figure.2 Dot Diagram of an 8x8-bit Dadda Multiplier

As for the reduction stage, a series of generalized recursive steps were used to determine the heights of each stage and the number of additions required to achieve each stage height as described in the following:

1. Let *d1= 2* and *dj+1 = [1.5 _ dj]*. *D* is the height of the matrix for the *jth* stage. Repeat until the largest *j* stage is reached in which the original N height matrix contains at least one column that has more than *d* dots.

2. In the *jth* stage from the end, place (3, 2) and (2, 2) counters as required to achieve a reduced matrix.Only columns with more than d*j* dots as they receive carries from less significant (3, 2) and (2, 2) counters are reduced.

3. Let *j = j – 1* and repeat step 2 until a matrix with a height of two is generated. This should occur when j= *1*.

4. Binary Multiplier Using High Speed Compressors

For higher order multiplications, a huge number of adders or compressors are to be used to perform the partial product addition. We have reduced the number of adders by introducing special kind of adders that is capable to add five/six/seven bits per decade. These adders are called compressors. Binary counter property has been merged with the compressor property to develop high order compressors. Uses of these compressors permit the reduction of the vertical critical paths. A 8×8bit multiplier has been developed using these compressors [4].

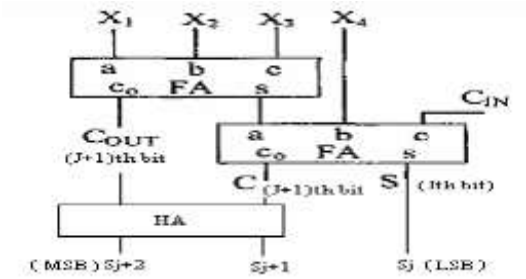

Figure.3 Modified 4-2 compressor

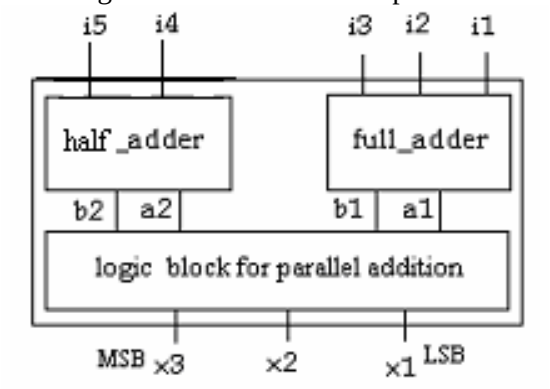

Figure.4 Block diagram of 5-3 compressor

5. Wallace multiplier

The partial products are formed by N2 AND gates. For the conventional Wallace reduction method, once the partial product array (N2bits) is formed, adjacent rows are collected into non-overlapping groups of three. Each group of three rows is reduced by

Figure.5 Wallace multiplier.

(1) Applying a full adder to each column that contains three bits,

(2) Applying a half adder to each column that contains two bits, and

(3) Passing any single bit columns to the next stage without processing.

This reduction method is applied to each successive stage until only two rows remain. This process is illustrated by the conventional 8-bit by 8-bit Wallace multiplier shown in Fig. 5.
The reduction is performed in four stages (each with the delay of one full adder). The third phase will require a (2N-1-S) wide adder, where S – number of stages in reduction.

6. Reduced Complexity Wallace Multiplier

It is the modified version of Wallace multiplier. It has less half adders than the normal Wallace multiplier .The partial products are formed by N2 AND gates. The partial products are arranged in an "inverted triangle" order. The modified Wallace reduction method divides the matrix into three row groups [5].

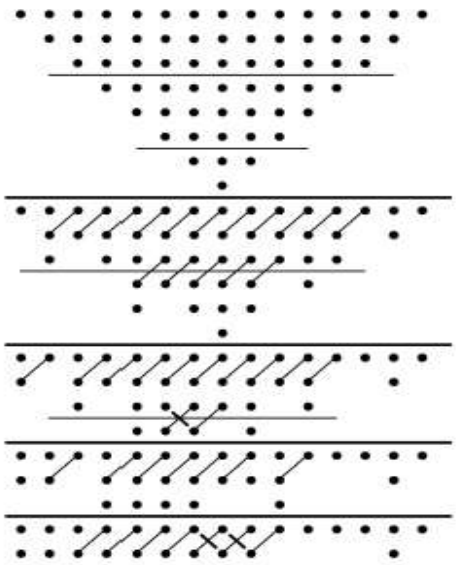

Figure.6. Reduced Complexity Wallace multiplier

1) Use full adders for each group of three bits in a column like the conventional Wallace reduction.

2) A group of two bits in a column is not processed, that is, it is passed on to the next stage (in contrast to conventional method). Single bits are passed on to the next stage as in the conventional Wallace reduction.

3) The only time half adders are used is to ensure that the number of stages does not exceed that of a conventional Wallace multiplier. For some cases, half adders are only used in the final stage of reduction.

7. The Proposed Low Power Multiplier: BZ-FAD

To drive low-power architecture, we concentrate our effort on eliminating or reducing the sources of the switching activity discussed in the previous section. The proposed architecture which is shown in Fig 8 is called BZ-FAD. Instead of using separate gate for nand, nor and mux, we are implemented into single gate is shown in figure.7 [6]. For that getting better Power, Area and Speed compare to all multipliers.

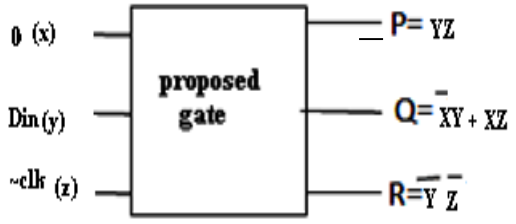

Figure.7.Proposed Gate.

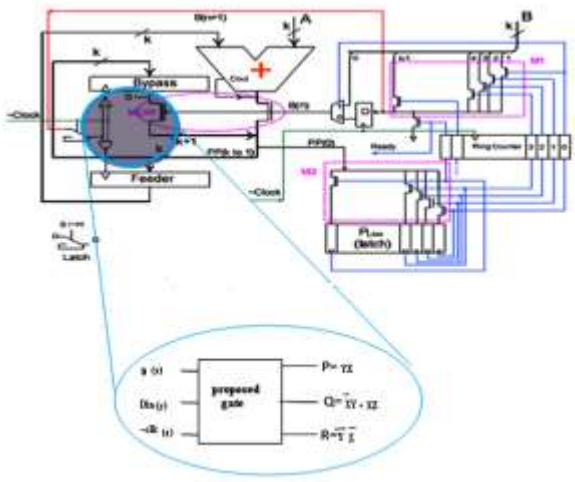

Figure.8.The proposed low power multiplier architecture (BZ-FAD).

8. Modified Carry save Adder

The 16-bit conventional CSA [3] has 17-half adders (H) and 15-full adders (F). Since the Ripple Carry Adder

(RCA) is used in the final stage, this structure yields large carry propagation delay. To reduce the delay, the final stage of CSA is divided into 5 groups as shown in Fig 9.The first group includes n $1+\log_2$ n-bit value and other groups include $\log_2$n-bit value, where n is the bit size of the adder. The divided groups are listed as follows [3]:

1. {c4, s [4:0]}, output s[4:0] is directly assigned as the final output.
2. {c7, x [7:5]} manipulates the partial result by considering c4 is 0.
3. {c10, x [10:8]} manipulates the partial result by considering c7 is 0.
4. {c13, x [13:11]} manipulates the partial result by considering c10 is 0
5. {X [17:14]} manipulates the partial result by considering c13 is0.

Depending on c4 of the first group, the second group mux gives the final result without the carry propagation delay from c4 to c7; depending on c7 of the second group final result, the third group mux gives the final result without the carry propagation delay from c7 to c10; depending on c10 of the third group final result, the fourth group mux gives the final result without the carry propagation delay from c10 to c13 and depending on c13 of the fourth group final result, the fifth group mux gives the final result without the carry propagation delay from c13 to s17.

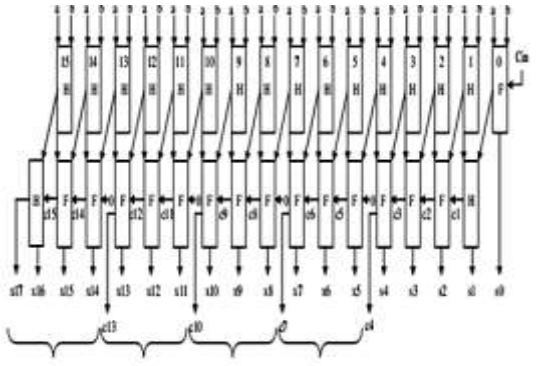

Figure.9. Modified Carry Save Adder

The main advantage of this logic is that each group computes the partial results in parallel and the multiplexers are ready to give the final result "immediately" with the minimum delay of the mux. When the Cin of each group arrives, the final result will be determined "immediately". Thus the maximum delay is reduced in the carry propagation path. This same logic has been used for 32 and 64-bit adder structures to achieve higher speeds. The area indicates the total cell area of the design and the total power is sum of leakage power, internal power, net power and dynamic power. The proposed result shows that the Modified Carry Save Adder (MCSA) [3] has reduced area, delay and consumes lesser power than CSA.

9. Results & Simulation

The design of all multipliers using Hardware Description Language (HDL) for 8-bit unsigned data. To get power and speed report, we use XILINX ISE 10.1 as synthesis tool and Model-Sim XE III 6.3c for simulation. FPGA-Spartan III is used for implementation. The low power multiplier architecture (BZ-FAD) with fixed block size was implemented in 8x8 multiplier architecture. It gives better result than conventional adders in terms of speed and power consumption as shown in the Table-2. The Simulation Result of the proposed multiplier is shown in figure 10.

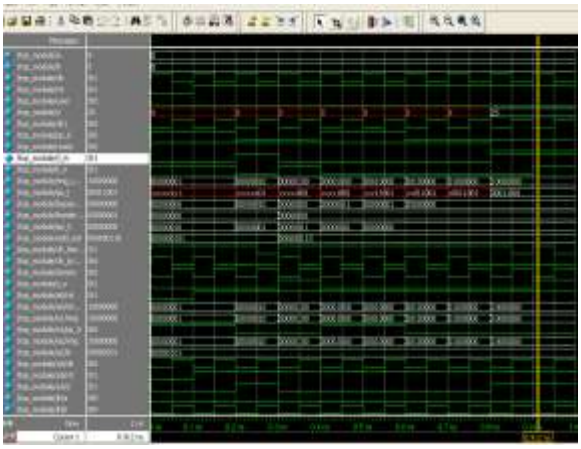

Figure.10. Simulation result of proposed multiplier.

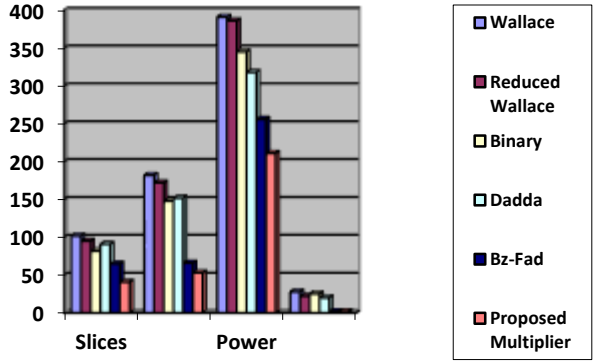

Figure.11. Performance analysis of different Multipliers.

Table 1. Comparison of Area, Delay and Power of multiplier.

Types	Slices	LUT	Power (mw)	Delay (ns)
Wallace	102	182	391	28.11
Reduced Wallace	95	172	386	22.45
Binary	82	148	345	25.491
Dadda	91	152	318	20.21
Bz-Fad	65	66	245	2.010
Proposed Multiplier	41	53	211	1.774

10. Conclusion

This paper proposed an efficient multiplier called modified BZ-fad multiplier. The proposed Novel multiplier provides low area, power and less delay by use of proposed reversible gate .The proposed method reduced 50 % area, power and 95% delay than the existing architecture. The proposed multiplier was implemented on Spartan-III FPGA and the result was analyzed.

REFERENCES

1. Rashidi, B., Rashidi, B., Pourormazd, M., Univ. of Tabriz, Tabriz, Iran , "Design and implementation of low power digital FIR filter based on low power multipliers and adders on Xilinx FPGA, Electronics Computer Technology" (ICECT), 2011 3rd International Conference on 8-10 April 2011 ,
2. Soojin Kim and Kyeongsoon Cho., "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges", World Academy of Science, Engineering and Technology 61 2010.
3. B.Ramkumar, Harish M Kittur, P.Mahesh Kannan, "ASIC Implementation of Modified Faster Carry Save Adder", European Journal of Scientific Research, ISSN 1450-216X Vol.42 No.1 (2010), pp.53-58.
4. Dandapat, P. Bose, Sayan Ghosh, Pikul Sarkar, and D. Mukhopadhyay, "Design of an Application Specific Low-Power High Performance Carry Save 4-2 Compressor" in IEEE VLSI Design and Test Symposium 2007, VDAT-07, pp.360, 2007.
5. Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method", IEEE, 2006.
6. Prashant R. Yelekar ,Prof. Sujata S. Chiwande "Design of sequential circuit using reversible logic" IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012) March 30, 31, 2012
7. M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram "BZ-FAD: A Low-Power Low-Area Multiplier based on Shift-and-Add Architecture", IEEE Transactions on Very Large Integration (VLSI) Systems, vol.17, No.2, February 2009.
8. C. N.Marimuthu, P. Thangaraj , Aswathy Ramesan, "Low Power Shift and Add Multiplier Design" International Journal of Computer Science and Information Technology 2.3 (2010) 12-22