IJCTA, 9(39), 2016, pp. 45-53

Efficient Shift-Add Multiplier Design Using Parallel Prefix Adder

Rohan Pinto* and Kumara Shama*

Abstract : The overall performance of any DSP system depends on the performance of the arithmetic unit. Multiplication is one of the pivotal operation in it. The speed and area of the multiplier is always a matter of concern for the better performance of any processor. In this paper a simple and efficient multiplier for 8-bit and 16-bit have been proposed. The proposed structure is a modified version of Bypass Zero, Feed A Directly (BZ-FAD) multiplier. It has been implemented on FPGA, Spartan 6 device. This structure has low power and area because of its uniqueness in using the proposed parallel prefix structure for addition of partial products in multiplier. The proposed multiplier lowers the switching activity by 55% and area by 64%.

Keywords : BZ-FAD; Parallel prefix adder; Ling adder; FPGA implementation.

1. INTRODUCTION

Multiplication and addition are most common and heavily used arithmetic operations that figure out to be important in all digital signal processing applications. Multiplication is hardware concentrated. Major part of the researcher's interest lies in its high speed, low power and low area. Multiplication is repeated form of shifting and addition operation. The main concern in shift-add multiplication is to speed up the partial product addition. Variety of multiplication algorithms and designs have been discussed in the past. Shen and Chen [1] proposed a low power 16-bit multiplier. It reduced the switching activities taking place inside the multiplier than the conventional multiplier with a reasonable increase in area. Chen *et al.* [2] proposed 16-bit row based, column based and hybrid based multipliers that dissipate less power. Partial product count were reduced by booth encoding which in turn reduced the switching activities. Wang *et al.* [3] developed a fixed width multiplier using left to right algorithm. It reduced the partial product leading to low power. Wang and Sung [4] proposed 8-bit low power multiplier using bypassing technique. Power saving was upto 75% at the cost of large area. Huang and Milos [5] designed different structures of linear array multipliers. The structures led to high performance and low power. Chen and Chu [6] applied spurious power suppression technique (SPST) on multiplier which led to low power performance and high speed as compared to other multipliers. Mottaghi *et al.* [7] proposed a structure, Bypass zero, feed A directly to reduce the switching activities during multiplication. The structure had low power and area. Marimuthu and Thangaraj [8] implemented BZ-FAD structure using latches and flip-flops to reduce the switching activities in multiplier. Vijaykumar and Sumathy [9] designed an error tolerant 8-bit shift-add multiplier. It was a low power and high speed structure. Valan and Baulkani [10] developed a shift-add multiplier structure using modified universal shift register and Johnson counter. It reduced the switching activities in the multiplier as compared to conventional multiplier. Liu *et al.* [11] proposed an approximate multiplier for high performance application. It reduced the power dissipation and critical path delay. D Nikolos *et al.* [12] presented a hybrid prefix adder that combined conventional and ling carry computation. Giorgos and Nikolos [13] proposed a high speed parallel prefix adder. Ling adder equations were modified

* Manipal Institute of Technology, Manipal-576 104, Karnataka, India *E-Mail: rohan.pinto@yahoo.co.in*

and incorporated in the structure. Poornima and Kanchana [14] developed a novel structure by combining two other prefix adders proposed by Ladner-Fischer [15] and Kogge-Stone [16].

In this paper a modified BZ-FAD structure based on shift-add multiplication is proposed. This structure has marginally increased speed with low power and area because of the elimination of some of the components from the conventional BZ-FAD structure [7]. The proposed structure also has an efficient adder being used for addition operation which increases the speed. The proposed 8-bit and 16-bit multipliers have been implemented using XILINX spartan 6 FPGA and it has been noted that the proposed adder outperforms other adders in terms of speed, area and power dissipation.

Rest of the paper is organized as follows: section 2 give the details of shift add multipliers. Section 3 introduces parallel prefix adders used for shift-add multiplication. Section 4 discusses the results, and conclusions are drawn in section 5.

2. SHIFT-ADD MULTIPLIERS

A. Bypass Zero, Feed A Directly (BZ-FAD) Multiplier

Mottaghi *et al.* [7] proposed bypass zero, feed A directly (BZ-FAD) multiplier structure depicted in Fig. 1. Here the switching activities in the multiplier were reduced which led to low power. The conventional shift-add multiplier which multiplies two number X and Y undergoes six switching activities. 1) The switching activity when the multiplier Y is shifted 2) Switching activity in the counter 3) Switching activity in the adder 4) Switching activity in the multiplexer that selects either 0 or X for addition 5) Switching activity in the multiplexer select line which is controlled by 0^{th} bit of Y 6) Switching activity due to shifting of partial product.

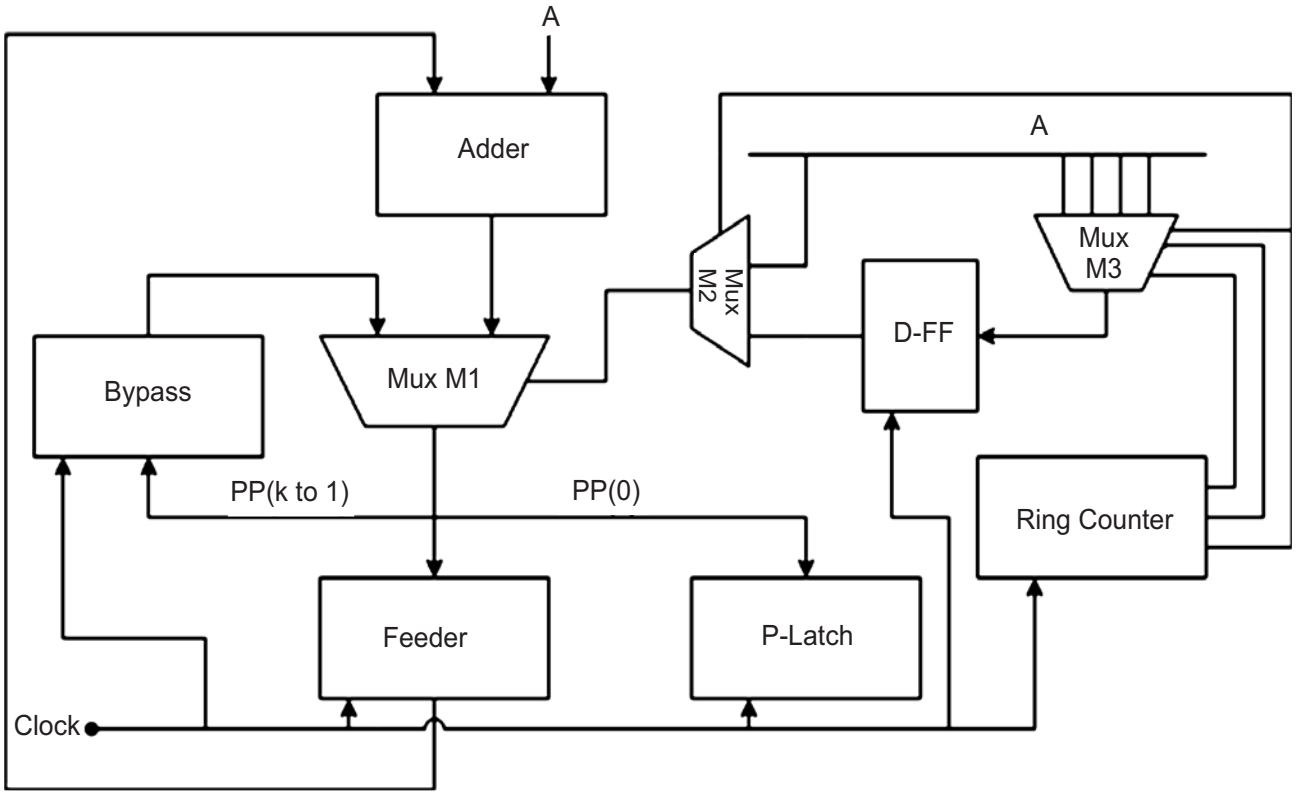

Figure 1: Bypass Zero, Feed A Directly (BZ-FAD) Multiplier [7]

BZ-FAD structure reduced the switching activities in the multiplier. Instead of shifting the multiplier bits to right every time as done in conventional multiplier to check whether the 0^{th} bit of multiplier is 0 or 1, in BZ-FAD one hot encoded bus selector chooses the hot bit of Y in every cycle. A low power ring counter was used to select the required bits in each cycle. This reduced the switching activity in multiplier.

When the 0^{th} bit of the multiplier is zero, then zero is added to the previous partial product and if it is one then multiplicand X is adder to previous partial product in conventional multipliers, this increases the switching activity in the adder, instead in BZ-FAD addition of zero was skipped when multiplier 0^{th} bit was zero using feeder and bypass register.

After the generation of each partial product in every cycle the 0^{th} bit of the partial product is not processed further. It forms the respective final product bit. P-latch was used to store the LSB bits of the product which were obtained in the first few cycles. A low power ring counter was used to open up the respective latch to store the product LSB bits. MSB bits of the product were stored in the feeder register. This eliminated the process of shifting the partial product in every cycle for processing as done in conventional multiplier.

A. Proposed Bypass Zero, Feed A Directly Multiplier

The proposed structure as shown in Fig. 2 further reduces the switching activity taking place in the multiplication operation thereby reducing power. Few of the components are eliminated from the conventional BZ-FAD Multiplier, thereby reducing the delay. The process of reducing the switching activity is explained below.

- **Shifting of multiplier bits :** In the structure proposed by Mottaghi *et al.* [7] two multiplexer, D flip flop and low power ring counter was used to select the 0^{th} bit of the multiplier. These components can be eliminated and instead AND gates can be used to check the content of the multiplier. This reduces the area to a larger extent. Shifting of multiplier bits are eliminated thereby reducing the switching activity.

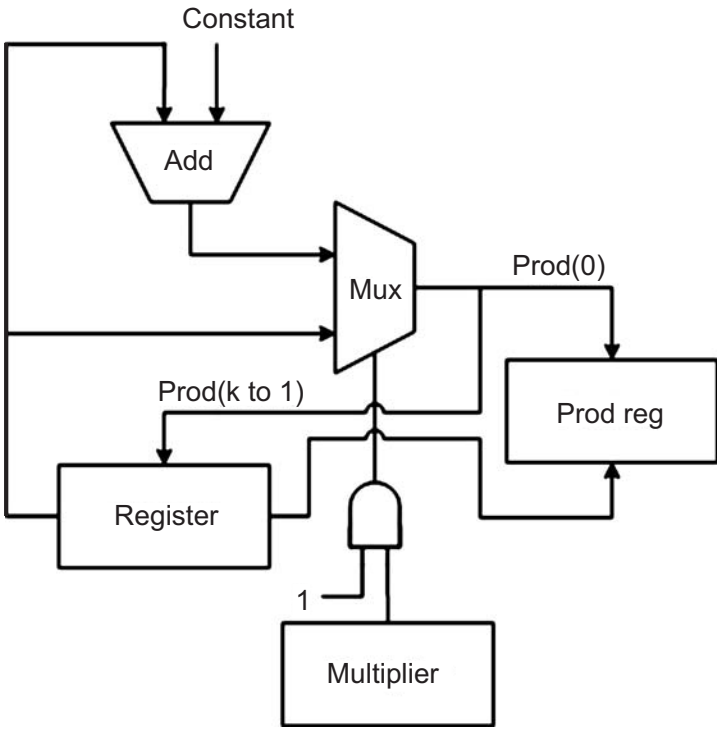

Figure 2: Modified Bypass Zero, Feed A Directly (BZ-FAD) Multiplier

- **Activities of adder :** In BZ-FAD feeder and bypass registers were used to skip the addition of zero to the previous partial product when the multiplier 0^{th} bit was zero. In proposed structure only one register is used to store the intermediate result. The register contents are fed back to the multiplexer when the 0^{th} bit of the multiplier is zero and when the 0^{th} bit of the multiplier is one then the register contents are fed to the adder. This reduces the activity in the adder where addition is done only when the 0^{th} bit of the multiplier is one.

- **Shifting of partial product :** Here in the proposed structure, ring counter existing in Fig. 1 is eliminated that opens up the latch to store the LSB of product register. Instead 0^{th} bit of the partial product moves to the respective position in the product register and remaining bits of the partial product are shifted and moved back to register for further processing. This completely removes the shifting of partial product as done in conventional shift-add multiplier thereby reducing the switching activities in the partial product.

3. PARALLEL PREFIX ADDER

Multiplication can be done by cumulative partial product and successively adding it to properly shifted term. Addition is the fundamental operation in multiplication. A fast and area efficient multiplier is highly influenced by the performance of the adder. Hence an efficient parallel prefix 8-bit and 16-bit adder based on modified ling equation is proposed to be used in the modified BZ-FAD multiplier. This reduces the area and power of the proposed multiplier.

Ling [17] proposed a modified equation of carry look ahead to attain a significant saving in the hardware. The technique depends on calculating pseudo carry H_i instead of conventional carry c_i. This technique saves one logic level at each bit position. Although computation of pseudo carry is simpler compared to conventional carry, but the sum calculation is complicated. Hence ling equation is modified to generate the real carry c_i out of the pseudo carry H_i, so that the final sum calculation be reduced to simple XOR operation.

Consider the pseudo carry equation for 4^{th} bit position.

$$H_4 = g_4 + g_3 + p_3 . g_2 + p_3 . p_2 . g_1 + p_3 . p_2 . p_1 . g_0 \quad (1)$$

Since, $g_i = g_i . p_i$, eq. (1) can be rewritten as

$$H_4 = (g_4 + g_3) + p_3 . p_2 . (g_2 + g_1) + p_3 . p_2 . p_1 . p_0 . g_0$$

Let, G_i^* and P_i^* be the intermediate generate and intermediate propagate bits respectively [13] given as:

$$G_i^* = g_i + g_{i+1} \text{ and } P_i^* = p_i . p_{i-1}, \; 0 \le i \le n-1.$$

With $g_{-1} = p_{-1} = 0$ and $G_k^* = P_k^* = 0$, for $k < 0$;

$$H_4 = G_{4:3} + P_{3:2} . G_{2:1} + P_{3:2} . P_{1:0} . G_{0:-1}$$

The associative operator $\odot$ associates pairs of generate and propagate bits as $(g_i, p_i) \odot (g_i', p_i') = (g_i + p_i . g_i', p_i . p_i')$. Therefore H_4 can now be rewritten using the associative operator $\odot$ as

$$H_4 = (G_{4:3}, P_{3:2}) \odot (G_{2:1}, P_{1:0}) \odot (G_{0:-1}, P_{-1:-2}) \odot (G_4^*, P_3^*) \odot (G_2^*, P_1^*) \odot (G_0^*, P_{-1}^*) \quad (2)$$

The pseudo carries of even H_i and odd H_{i+1} indexed bit position are given as

$$H_i = (G_i^*, P_{i-1}^*) \odot (G_{i-2}^*, P_{i-3}^*) \odot \ldots\ldots \odot (G_0^*, P_{-1}^*)$$

$$H_{i+1} = (G_{i+1}^*, P_i^*) \odot (G_{i-1}^*, P_{i-2}^*) \odot \ldots\ldots \odot (G_1^*, P_0^*)$$

The real carries are now expressed as:

$$c_0 = H_0 . p_0$$
$$c_1 = H_1 . p_1$$
$$c_2 = H_2 . p_2$$
$$c_3 = H_3 . p_3$$
$$c_4 = (G_{4:3} + P_{3:2} . G_{2:1}) p_4$$
$$c_5 = (G_{5:3} + P_{4:3} . G_{3:0}) p_5$$
$$c_6 = (G_{6:3} + P_{5:2} . G_{2:1}) p_6$$
$$c_7 = (G_{7:3} + P_{6:3} . G_{3:0}) p_7$$

The proposed 8-bit parallel prefix adder based on modified ling equation is shown in Fig. 3.

The white square in Fig. 3 calculates generate bit , propagate bit and half sum bit as shown in Fig. 4(*a*). Black square provides intermediate generate and propagate bits shown in Fig. 4(*b*). Group generate and group propagate bits are calculated from intermediate generate and propagate bits as shown in Fig. 5. The proposed prefix cell which calculates the real carry is shown in Fig. 6(*a*). The white circle with an alphabet A in it is the prefix cell which generates carry for the lower order bits from 0 to 3 for 8-bit adder. This is shown in Fig. 6(*b*).

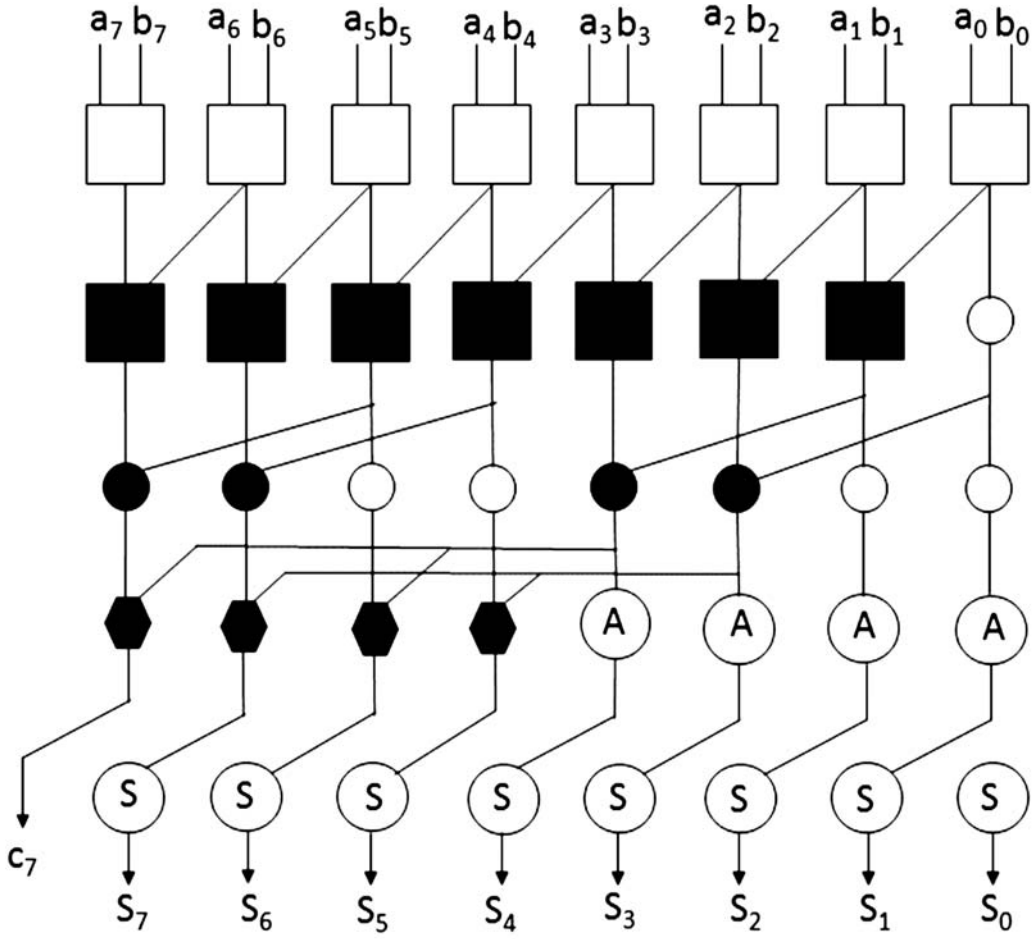

Figure 3: Proposed 8-bit parallel prefix adder

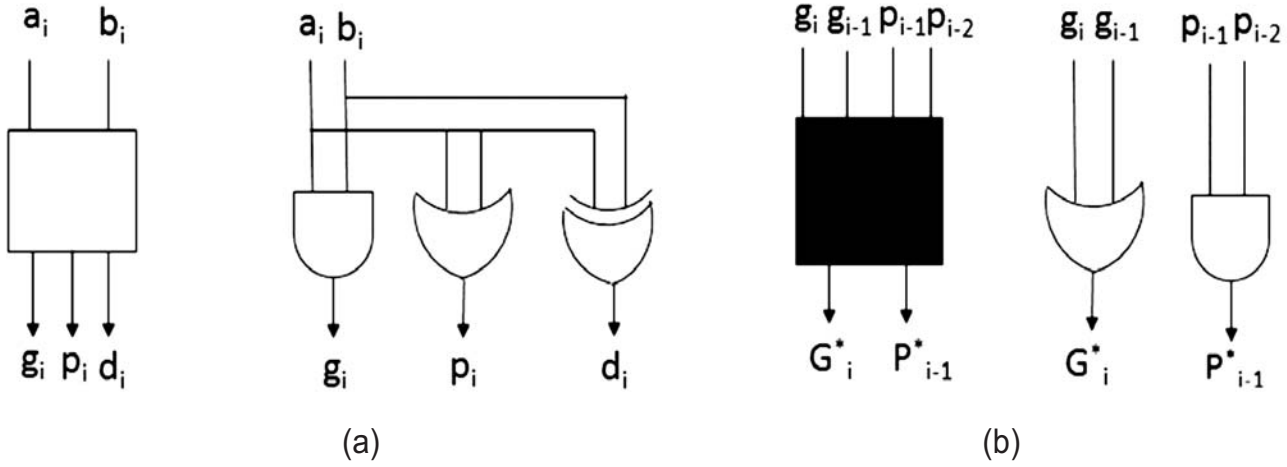

Figure 4: (*a*) Generate, propagate and half sum computing node. (*b*) Intermediate generate and propagate computing node [13]

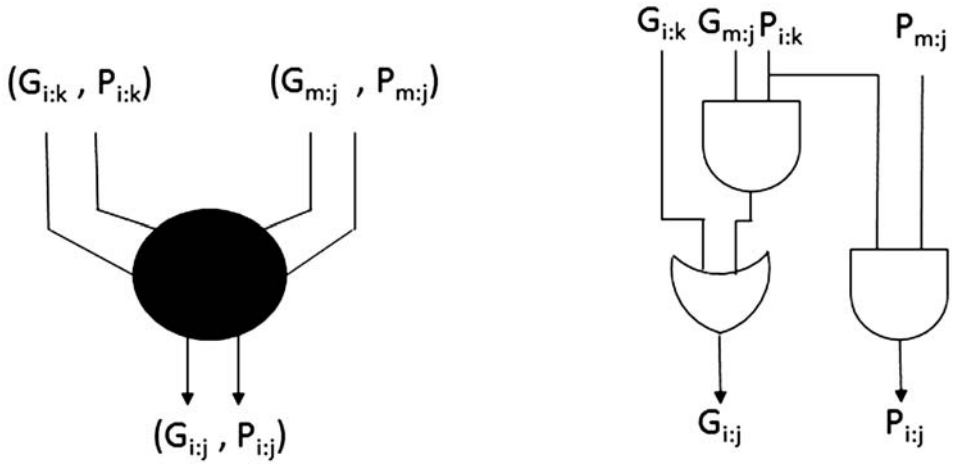

Figure 5: Group generate and propagate computing node [18]

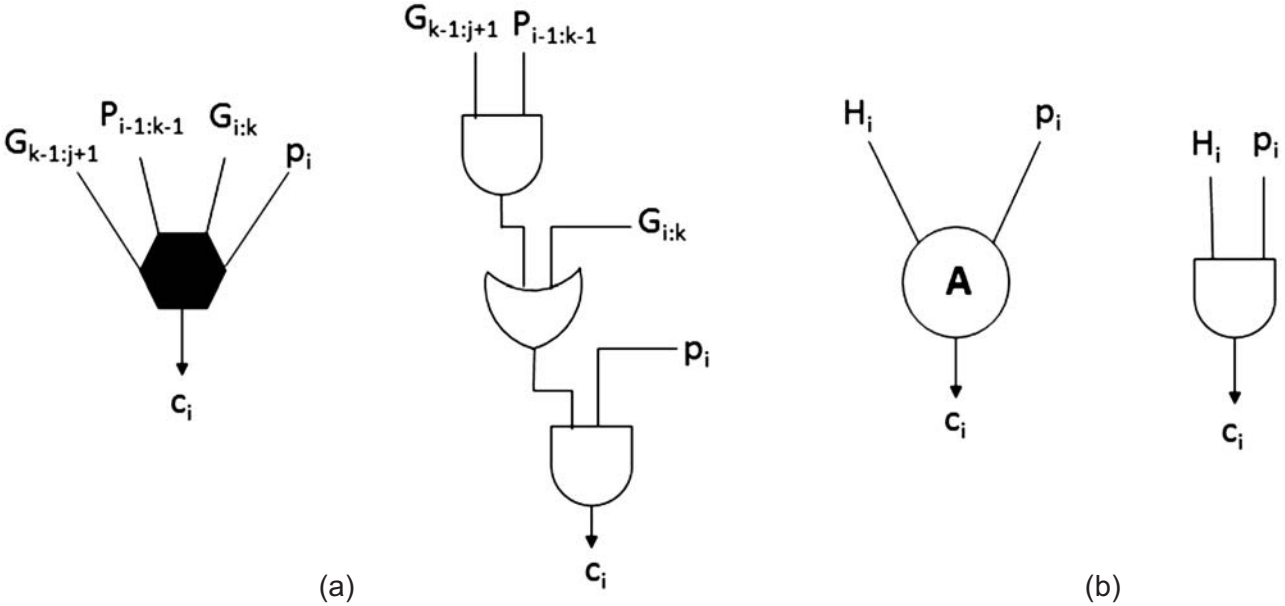

Figure 6: (*a*) Proposed prefix cell for computing real carries. (*b*) Prefix cell for computing real carry [19]

Proposed 16-bit parallel prefix adder based on modified ling equation is shown in Fig. 7. These adders are area and power efficient. This increases the efficiency of the multiplier to further extent.

4. RESULTS

The proposed multiplier was modelled in VHDL, simulated by ISE simulator (ISim) and synthesized using Xilinx Synthesis Technology (XST) tool. Power dissipation was analyzed using Xpower Analyzer. The obtained results were compared with 8-bit BZ-FAD multiplier structure proposed by Marimuthu and Thangaraj [8] and Conventional shift-add multiplier in Table 1. It shows that the proposed structure is efficient in all parameters.

Table 1

Area, power and delay comparison of 8-bit multipliers

Structure	*Delay (ns)*	*Power(mW)*	*Area(slices)*
Proposed Structure 8-bit	21.74	14.0	119
BZ-FAD, 8-bit [8]	48.81	105.0	334
Conventional multiplier, 8-bit [8]	151.1	151.1	662

The proposed structure was also compared with other multipliers proposed by Valan and Baulkani [10], 16-bit BZ-FAD structure implemented in FPGA [9], error tolerant shift-add multiplier proposed by Vijaykumar and Sumathy [9] and high performance error tolerant multiplier proposed by Liu *et al.* [11]. Table 2 and Table 3 give power and delay comparison of 8-bit and 16-bit multipliers respectively. Comparison results show that the proposed structure is area and power efficient.

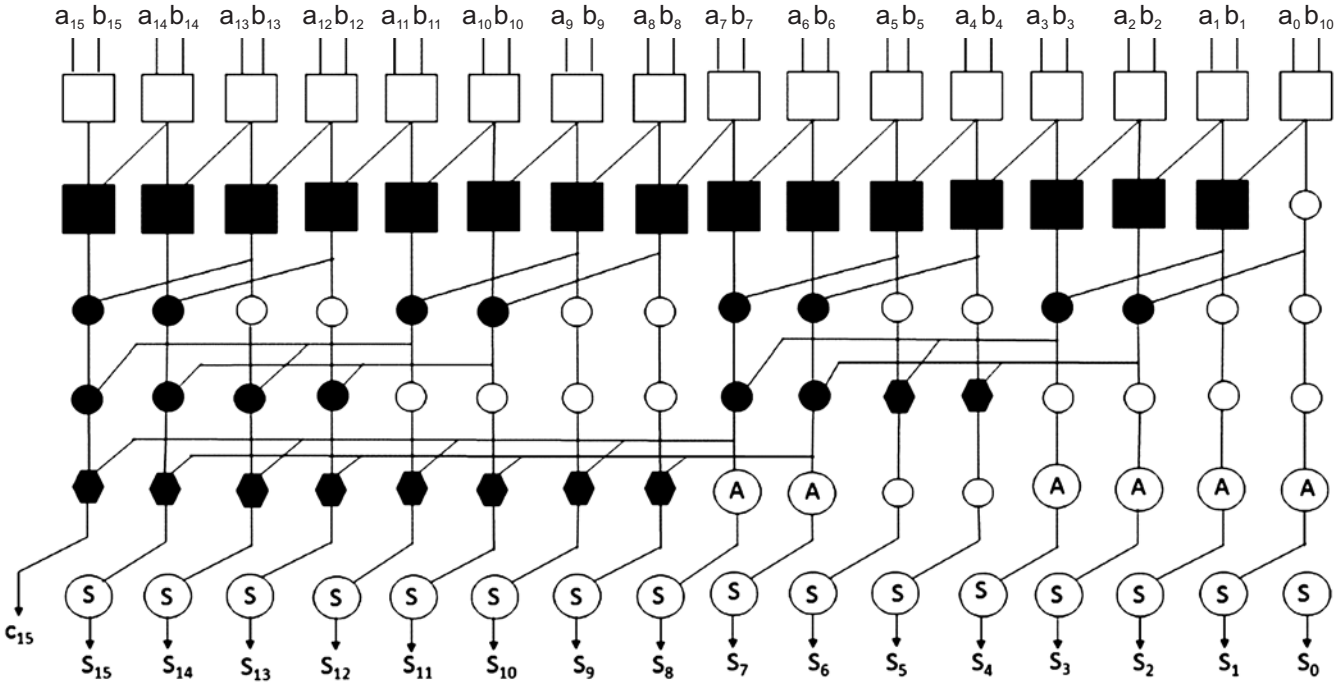

Figure 7: Proposed 16-bit parallel prefix adder

Table 2
Power and delay comparison of 8-bit multipliers

Structure	*Power (mW)*	*Delay (ns)*
Proposed Structure 8-bit	14.0	21.74
Valan [10]	172.0	–
Vijaykumar [9]	228.0	49
BZ-FAD [9]	271.0	61
Conventional Multiplier [9]	295.0	95

Table 3
Power and delay comparison of 16-bit multipliers

Structure	*Power (mW)*	*Delay (ns)*
Proposed Structure 16-bit	14.0	46.5
Valan [10]	177.0	–
BZ-FAD [10]	271.0	–
Liu et al. [11]	150.0	13.9

Table 4 shows the area, delay and power of proposed 8-bit and 16-bit modified BZ-FAD multiplier.

Table 4
Area, power and delay of proposed 8-bit and 16-bit multipliers

Multiplier bits	*Area(slices)*	*Delay(ns)*	*Power(mW)*
8-bit	119	21.7	14.0
16-bit	494	46.5	14.0

5. CONCLUSION

In this paper an area-power efficient modified BZ-FAD multiplier for 8-bit and 16-bit is proposed. The proposed multiplier is fast and efficient because of the parallel prefix adder design which does the carry propagation quickly. Here 8-bit and16-bit adder design based on modified ling equation is also proposed. When compared to conventional BZ-FAD 8-bit multiplier [8] power saving is significant and delay is lowered by 55%.

6. REFERENCES

1. Nan-Ying Shen and Oscal T-C. Chen, "Low-power multipliers by minimizing switching activities of partial products," *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Vol. 4, pp. 93-96, 2002.
2. Oscal T-C Chen, Sandy Wang and Yi-Wen Wu, "Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, Vol. 11, No. 3, pp. 418 – 433, 2003.
3. Jinn-Shyan Wang, Chien-Nan Kuo and Tsung-Han Yang, "Low-power fixed width array multipliers," *Proc. IEEE Int. Symp. Low Power Electronics and Design*, *ISLPED '04*, pp. 307 – 312, 2004.
4. Chua-Chin Wang and Gang-Neng Sung, "Low-Power Multiplier Design Using a Bypassing Technique," *J Sign Process System*, Vol. 57, pp. 331–338, 2009.
5. Zhijun Huang and Milos D. Ercegovac, "High-Performance Low-Power Left-to-Right Array Multiplier Design," *IEEE Trans. Computers*, Vol. 54, No. 3, pp. 272 – 283, 2005.
6. Kuan-Hung Chen and Yuan-Sun Chu, "A Low-Power Multiplier with the Spurious Power Suppression Technique," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 15, No. 7, pp. 846 – 850, 2007.
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 Trans. Very Large Scale Integration (VLSI) Systems*, Vol. 17, No. 2, pp. 302 – 306, 2009.
8. C. N. Marimuthu and P. Thangaraj, "Low Power Multiplier Design Using Latches and Flip-Flops," *J. Computer Science*, Vol. 6, No. 10, pp. 1117-1122, 2010.
9. K. N. Vijaykumar and V. Sumathy, "Design of Low- Power High-Speed Error Tolerant Shift and Add Multiplier," *J. Computer Science*, Vol. 7, No. 12, pp. 1839-1845, 2011.
10. S. P. Valan Arasu and Dr. S. Baulkani, "Modified Universal Shift Register Based Low Power Multiplier Architecture," *J. Theoretical and Applied Information Technology*, Vol. 65, No. 1, 2014.
11. Cong Liu, Jie Han and Fabrizio Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery," *IEEE Conf. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1 – 4, 2014.
12. C. Efstathiou, H.T. Vergos and D. Nikolos, "Ling Adders in Standard CMOS Technologies," *Proc. IEEE Int. Conf. Electronics, Circuits and Systems (ICECS)*, vol. 2, pp. 485-48, 2002.
13. Dimitrakopoulos Giorgos and Dimitris Nikolos, "High-Speed parallel-prefix VLSI ling adders," *IEEE Trans. Computers*, Vol. 54, No. 2, pp. 225-231, 2005.

14. Poornima N and V S Kanchana Bhaaskaran, "Area efficient hybrid parallel prefix adders," *Procedia Materials Science*, vol. 10, pp. 371-380, 2015.
15. R.E. Ladner and M.J. Fisher, "Parallel prefix Computation," *J. ACM*, Vol. 27, No. 4, pp. 831-838, 1980.
16. P.M. Kogge and H.S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Computers*, Vol. 22, No. 8, pp. 786-793, 1973.
17. Huey Ling, "High-speed binary adder," *IBM J. R&D*, Vol. 25, No. 3, pp. 156-166, 1981.
18. R. P. Brent and H.T. Kung, "A regular layout for Parallel adders," *IEEE Trans. Computers*, Vol. 31, No. 3, pp. 260-264, 1982.
19. Tso-Bing Juang, Pramod Kumar Meher and Chung-Chun Kuan, "Area-Efficient parallel-prefix ling adders," *Proc. IEEE Asia Pacific Conf. Circuits and Systems (APCCAS), Kuala Lumpur*, pp.736-39, 2010.