

Comparison of Conventional Multiplier with Bypass Zero Multiplier

¹Kalyani Chetan Kumar, ²Shrikant Deshmukh, ³Prashant Gupta.

M.tech VLSI Student
SENSE Department,
VIT University, Vellore, India. 632014.

Abstract - Low power is one of the most important designing factors in today's VLSI design market because we need to meet the Moore's law and as per customers requirements so we can either reduce static or dynamic power. So basically here in this paper dynamic power reduction is performed on the multiplication circuit because multiplier is one of the most important circuits of many important digital circuits so power reduction of this is one of the important parameter to concern these days. In this work another low power architecture is implemented which is called as bypass zero feed a directly (BZFAD) which usually reduces the switching activity and on the basis of that total dynamic power reduces. Finally comparison is done for both the multiplier and results are shown on the basis of that we can say that BZFAD is much more optimized in terms of power and area.

Keywords –Bypass zero feed a direct (BZFAD), Radix 4 booth multiplier, Conventional Multiplier, Ring Counter, Low power.

I. INTRODUCTION

Multiplier is one of the most important arithmetic circuits used in multimedia, and digital signal processing such as discrete cosine transform, fast Fourier transform. Because of its massive usefulness lots of algorithms are developed to improve constraints like area, power and speed. Multiplication consists of three major steps: 1) recoding and generating partial products; 2) reducing the partial products by partial product reduction schemes (e.g., Wallace tree [1] [2] [3]) to two rows; and 3) adding the remaining two rows of partial products by using a carry-propagate adder (e.g., carry look ahead adder) to obtain the final product.

There are already many techniques developed in the past years for these three steps to improve the performance of multipliers. In this brief, we will focus on the first step (i.e., partial product generation) to the area, delay, and power consumption of multipliers.

Generally, $L \times L$ fixed-width multipliers that generate L most significant result bits (MSBs) and truncate L last significant result bits (LSBs) are frequently utilized to maintain a fixed-length operation in DSP applications. Therefore, reducing truncation error becomes an important issue for designing fixed-width multipliers. When the multiplication width increases, the hardware cost of post truncation (P-T) increases dramatically in the very-large-scale integration (VLSI) design. The architecture for modulo $(2^n + 1)$ multiplication proposed in [4], based on a Wallace tree, is specific for the diminished-1 number representation. It requires the pre-computation of a data-dependent CT before applying the Wallace tree, which introduces an additional delay corresponding to some adders to implement an $(n-1)$ -bit counter.

Recently, Booth encoding is popular in multiplier for this reason, several error-compensation works are presented for fixed-width Booth multipliers design [1] [2] [3] [4] [5] [6]. However, the Truncation error cannot be depressed because the input information is limited in estimating the carry propagation from the truncated part also the high-accuracy fixed-width multipliers would result in large circuit area [7]. Therefore an area efficient high accuracy circuit is required.

This paper contains multiplier which uses modified booth Encoding (MBE) algorithm. This technique reduces number of partial product rows with fast and simple row generation. It allows the reduction of the size of the partial product array by almost half. Two's complement method used here gives further reduction in area.

II. II. METHODOLOGY

A. Modified booth algorithm

Detailed The Booth algorithm forms the base of Signed number multiplication algorithms and the main reason behind that this algorithm is simple in terms of implementing at hardware level, and that will increase the speed of signed multiplication.

Now just because of the number of partial products which are more in number this booth algorithm become little bit slow so to increase the speed of the multiplication modified booth algorithm comes into picture the main reason behind that it reduces the number of partial product by half. Modified boot algorithm (radix-4 MBE) scan the multiplier in term of three bits through three bit window and for each three bit only one partial product will be generated this whole process is done through a table (default) in that table1 for each 3 bit combination we have partial products and this is how we calculate through sliding window method every time. On the basis of this partial products three types of signals will generate i.e. one two and neg and all this signals area implemented through three different logic diagram.

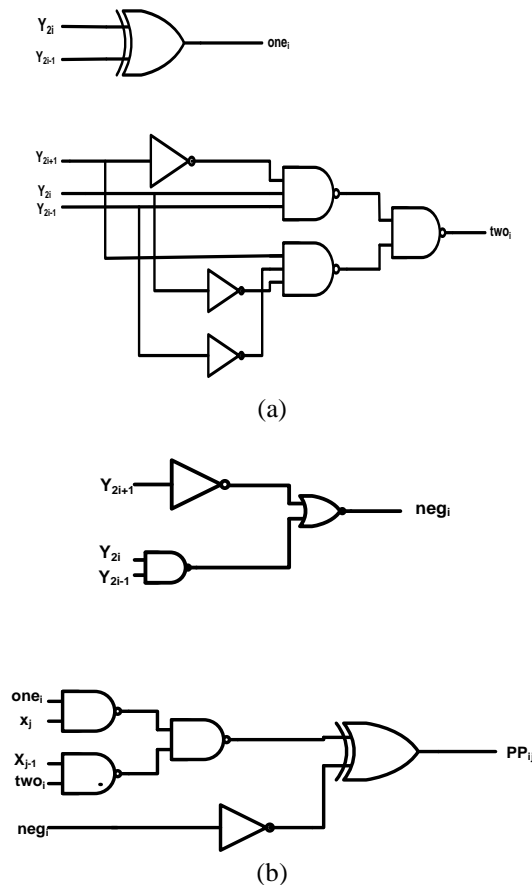


Fig 1 Partial product generator using Gate level diagram. (a) Modified booth encoder signal generator (b) Partial product generator

For $n \times n$ multiplier, there will be $\lfloor n/2 \rfloor + 1$ partial products generated, and the maximum height of which we get for the partial product array is $\lfloor n/2 \rfloor + 1$.

The first row in our multiplication split into two sub rows temporary, the first one contain partial product bits (from right to left) from pp_0 to pp_8 while second one with two bits set at “one” in positions 9 and 8. Then, neg_3 of fourth partial product row is moved to second sub row.

Table 1 RADIX 4 MODIFIED BOOTH ENCODING

Y_{2i+1}	Y_{2i}	Y_{2i-1}	Generated Partial Product
0	0	0	0x
0	0	1	1x
0	1	0	1x
0	1	1	2x
1	0	0	-2x
1	0	1	-1x
1	1	0	-1x
1	1	1	-0x

Aim of doing this is second sub row containing also the bit neg_3 , can now be easily added to the first sub row, with a constant short carry propagation of three positions i.e., independent of the length of the operands, for square multipliers. In fact, with reference to the notation of Fig. we have that $qq_90 \text{ } qq_90 \text{ } qq_80 \text{ } qq_70 \text{ } qq_60 \text{ } \frac{1}{4} \text{ } 0 \text{ } 0 \text{ } pp_80 \text{ } pp_70 \text{ } pp_60 + 0 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } neg_3$.

As introduced above, due to the particular value of the second operand, i.e., $0 \text{ } 1 \text{ } 1 \text{ } 0 \text{ } neg_3$, in [11], we have observed that it requires a carry propagation only across the least-significant three positions, a fact that can also be seen by the implementation shown in Fig.7. To have min delay, it is necessary that the generation of the other rows is done in parallel with the generation of the first row cascaded by the computation of the bits $qq_90 \text{ } qq_90 \text{ } qq_80 \text{ } qq_70 \text{ } qq_60$ in Fig. 6b. Booth encoding for the first row is computed more easily than for the other rows, because the $y-1$ bit of MBE is always zero. For preliminary analysis we refer to the circuits in the following fig. 2

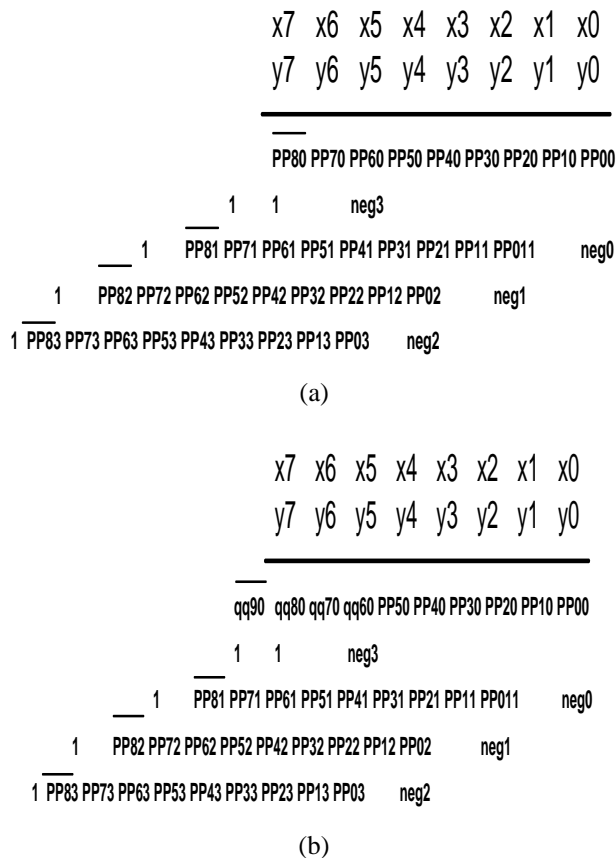


Fig 2 Partial product after adding the neg value to the first row (a) Basic (b) Resultant array.

For 1st partial product row we use this circuit which contain all previous mentioned signal one, two, and neg as shown in the fig 3.

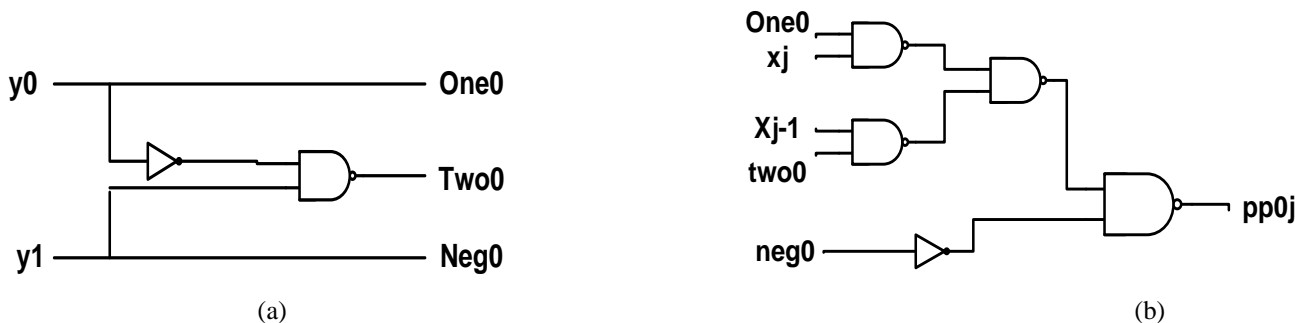


Fig 3: First row partial product generator

B. Bypass zero feed a direct (BZFAD)

This work totally deals with the low power architecture and the architecture which uses this technique of BZFAD the main reason behind that is other architecture like shift and adds multiplier ,carry select adder and tree multiplier in all this multiplier shift and add consider as the best just because of simple deign and area.

So to achieve low power we need to deal with few parameters i.e. according to formula $P = cLv^2f$ so here different parameter that are used to reduce power are capacitance , voltage and frequency and in this work we are reducing the switching activity that is another factor on which power directly proportional.

Two multiplier implemented in this paper first is conventional and second one is BZFAD. First design i.e. conventional multiplier, in that multiplier every time there is a requirement of shifting B bit. For any particular multiplier we have six major sources of switching activity they are (1) shifting value of B register, (2) counter activity, (3) adder activity, (4) '0' and A switching between in the multiplexer, (5) multiplexer select activity, (6) partial product shifting. So this paper can conclude that power can be minimized by reducing any of the above mentioned parameter. Still there is a need for more optimized then one can reduce by switching activity of higher capacitance nodes.

1) Architecture - This paper shows the two architecture first one is conventional architecture and second one is BZFAD [7] [8][9][10] , so second one is main architecture on which this paper is concluded. So this architecture deals with the switching activity of multiplier

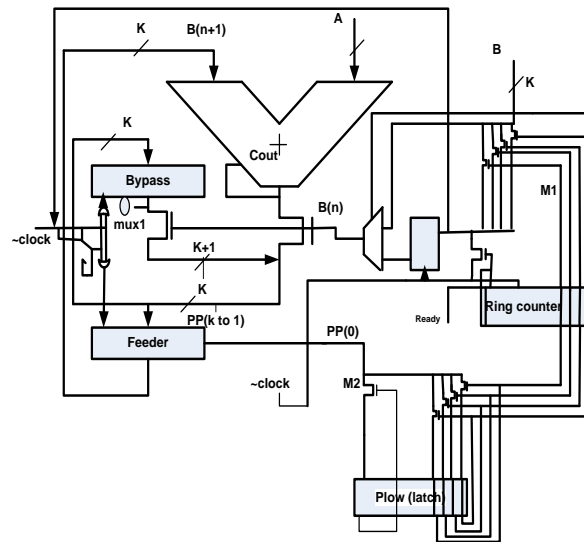


Fig 4 Architecture of Bypass zero feed a direct

In first architecture register B should be shifted to the right in every clock cycle. This depends on the value of right bit of B (0). B (0) (multiplier) bit is used to select particular value of A (multiplicand) and '0' value. If B (0) is equal to one then A should be added to partial products while if B (0) is equal to zero, then '0' should be added to partial product. Afterwards rights shifting of B on the basis of partial product in each cycle give rise to switching activities. So on the basis of the value of the 'B(0)' select the required bit of B(0) without shifting partial products in each clock cycle to avoid this we use low power ring counter. The 2nd architecture (BZFAD) [8] architecture uses a multiplexer to choose the particular bit of B (0).

2) Feeder and bypass register operation - In the conventional multiplier (1st architecture) on the basis of the right most bit of 'B', multiplier do the further process like if bit is '0' then it will add the zeros to the partial product otherwise value of A (multiplicand) by shifting one bit every time and finally we get the output that why it takes more power because no. Of transition increases so we can decrease this by bypass this zero partial product with the help of bypass and feeder register [9]. Both these register are use to bypass the adder in cycles in which the B (0) is equal to zero. In each clock cycle the next bit is checked and these operations are performed:

- Feeder is clocked if B (0) in next cycle is "1"
- Bypass is clocked if B (0) in next cycle is "0"

3) Ring Counter - In the 2nd architecture (BZFAD) we use ring counter of wider width just because of the reason that if we use any other binary counter then number of transition will increase and dynamic power increases so due to this we use chain of d flip flop which act as a ring counter as all of the flops having common clock and this can be further reduce by applying clock gating circuit to the input and check the input or the if the output is one then only enable the clock circuit so this further can reduce the switching activity or power consumption and the output store in the latch but if we apply this clock gating circuit to all the flip flop then area will increase and this will become another concerning factor so that why we are using for single flip flop [10]. So on the basis of that this work one can conclude the low power ring counter have their own advantage in achieving low switching activity and power.

III. RESULTS

TABLE 2 Results showing area and power required

	AREA REPORT (um ²)	POWER REPORT (nW)
Modified booth multiplier	1398	449368.787
BZFAD	10864	166368.040

From this Table 2 we can conclude that power for bypass zero multiplier is very less in compare to conventional multiplier.

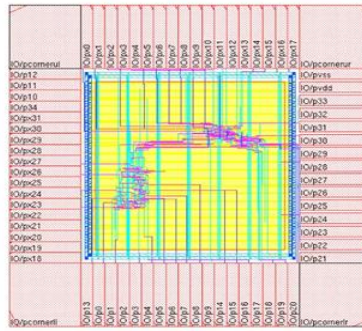


Fig 5 Chip plan and layout information of BZFAD

IV. CONCLUSIONS

This paper and its study show that low power multiplier is implemented which basically deals with the bypassing of zero in the multiplier bit and use of ring counter in spite of conventional one which deal with both the thing i.e. zero bit and binary counter which both leads to more power consumption and area. By comparing both the design we can say that BZFAD consume low power. Beside all this further work can be done in terms of clock gating which can further reduce the power consumption .so on the basis of all the results and lab work we can conclude BZFAD is one of the best choices in terms of low area and power, here final chip plan of this is shown in fig 5.

REFERENCES

- [1] O.L. MacSorley, "High Speed Arithmetic in Binary Computers," Proc. IRE, vol. 49, pp. 67-91, Jan. 1961.
- [2] L. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, May 1965
- [3] J C.S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans. Electronic Computers, vol. EC-13, no. 1, pp. 14-17, Feb. 1964.
- [4] D.E. Shaw, "Anton: A Specialize Machine for Millisecond-Scale Molecular Dynamics Simulations of Proteins," Proc. 19th IEEE Symp. Computer Arithmetic, pg. 3, 2009.
- [5] J.-Y. Kang and J.-L. Gaudiot, "Simple High-Speed Multiplier Design," IEEE Trans. Computers, vol. 55, no. 10, pp. 1253-1258, Oct. 2006.
- [6] J.-Y. Kang and J.-L. Gaudiot, "A Fast and Well-Structured Multiplier," Proc. Euromicro Symp. Digital System Design, pp. 508-515, Sept. 2004.
- [7] F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi, "Speeding-Up Booth Encoded Multipliers by Reducing the Size of Partial Product Array," internal report, http://arith.polito.it/ir_mbe.pdf, pp. 1-14, 2009.
- [8] Reducing the Computation Time in (Short Bit- Width) Two's Complement Multipliers Fabrizio Lamberti, Member, IEEE, Nikos Andrikos, Student Member, IEEE, Elisardo Antelo, Member, IEEE, and Paolo Montuschi, Senior Member, IEEE, IEEE transactions on computers, vol. 60, no. 2, february
- [9] Low power shift and add multiplier design, c. N.MarimuthuDr. P. Thangaraj, Aswathy Ramesan, International Journal of Computer Science and Information Technology, Volume 2, Number 3, June 2010.
- [10] Low power add and shift multiplier design bzfad architecture, prof prasann d.kulkarni, Prof.S.P.DeshpandeDr.G.R.Udupi, Prasann D. Kulkarni, et al International Journal of Computer and Electronics Research [Volume 2, Issue 2, April 2013