DESIGN OF HIGHSPEED APPROXIMATE REDUNDANT BINARY MULTIPLIER

LAKKAM RAKESH KUMAR REDDY*, DR. PRASAD JANGA, DR.K. NIRANJAN REDDY*****

PG SCHOLAR*, PROFESSOR, HOD & PROFESSOR****

DEPARTMENT OF ECE, CMR INSTITUTE OF TECHNOLOGY KANDLAKOYA VILLAGE, MEDCHAL ROAD, HYDERABAD, TELANGANA ,501401

ABSTRACT: Approximate computing can decrease the design complication with an increase in performance and power efficiency for error tolerant applications like multimedia signal processing and data mining which can tolerate error, exact computing units is not always necessary. They can be replaced with their approximate counterparts. A new design approach for approximation of multipliers based on partial products is altered to introduce varying probability terms. A high speed approximate redundant binary multiplier is designed by using two redundant binary 4:2 compressors. Then comparison in terms of power, delay, area and signal rate of approximate redundant binary multiplier (using conventional modified booth algorithm) with high speed approximate redundant binary multiplier is done using Xilinx tool. In Existing system, Implementation of multiplier comprises three steps generation of partial products, partial products reduction tree, and vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. To reduce power and improve approximate difference, a novel compressor based approximate multiplier is proposed.

Index Terms: 4:2 compressors, Xlinx tool, Area, signal strength, power.

I. INTRODUCTION

The main requirements of the applications for which multipliers are needed, such as computer applications, processing of signals and processing of images, have always been high speed multipliers. In the DSP blocks are primarily computerized by ALU, whose major component is multiplication and addition.[3] The main operation in the element processing are multiplications which generally need high utilization of power and energy Compared with many other arithmetic operations, multiplication is time taking and requires high power. Digital multipliers, in many instances foist a speed limit on the whole design due to the critical paths.[1] Many architectures and numerous high performance algorithms have been proposed to achieve the process of multiplication more aptly [4]. In the first step generation of partial products takes place. Then in the next step summation in pair wise form of partial product rows are performed until the number of left over partial products are two. Thereafter third step the two left over rows are summed to generate the final product by fast carry-propagate adder. So, the conventional multipliers with high speed are not competent of on the condition that the high speed computational needs at low power requirements.[4]The Consequently implementing multipliers with high performance is one of the leading objectives for designers of system to enhance the entire performance of the systems[2]. So here in the beginning step, Modified Booth Encoding (MBE) is used strikingly to optimize the partial product by half. Wallace trees which are some of the Partial product reduction schemes are used for partial product rows summation in the next step. In the step three, the final multiplication product is obtained by the two rows from partial product [5] summing tree which are added by means of fast carry-look ahead or carry save adder or carry select adders can be used [2]. A number of proceedings were put forward in the recent years to with the above three steps to achieve an efficient multiplier.[2] By the module of modified booth encoding the partial product terms are reduced to half and thus credit the design of multiplier by decreasing the size and improving the speed of the summing tree [3]. Hence, the multipliers with high speed, low power and area in VLSI design have always been and are still a research subject. One of the signed digit representations, for fast parallel arithmetic, redundant binary (RB) [1] representation is used.

OVER VIEW:

Binary multipliers are a widely used building block element in the design of microprocessors and embedded systems, and therefore, they are an important target for implementation optimization. Current implementations of binary multiplication follow the steps:

1) Recoding of the multiplier in digits in a certain number system.
2) Digit multiplication of each digit by the multiplicand, resulting in a certain number of partial products.
3) Reduction of the array to two operands using multioperand addition techniques.
4) Carry-propagate addition of the two operands the final result is a key issue, since it determines the number of partial products.

The usual recoding process recodes -r digits by just making groups of m operand into a signed-digit operand with digits in a minimally redundant digit set. As classic Dennard scaling is coming to an end, on chip power consumption has become prohibitively high. Therefore, improvement in the performance of computing systems is encountering significant hurdles at the same power level. Recently, approximate computing has been proposed as a new approach for efficient low power design. In this context, efficiency refers to the generation of approximate results and comparable performance at a lower power consumption. Approximate computing can generate results that are good enough rather than always fully accurate. Approximate computing [1] is driven by applications that are related to human perception and inherent error. Resilience to include digital signal processing (DSP), multimedia, machine learning and pattern recognition [2]. Approximate computing can be applied to these applications due to the large and redundant data sets with significant noise, so numerical exactness can be relaxed. Approximate computing not only reduces power consumption, but also increases performance by reducing the critical path delay. Approximate techniques can be applied at several levels including circuits, architectures and software [3], [4]. At circuit level, the design of approximate arithmetic units has received significant research interest due to its importance in many computing applications. 3 Typical applications, such as DSP and machine learning, require arithmetic computing in the form of addition (or accumulation) and multiplication. Approximate design techniques can be applied in four parts of a multiplier: uses the significant segments of the operands, so the most significant k bits to perform multiplication. Generally, the approximation in operands introduces very large errors compared with other approximation techniques. Approximation of PP generation: Approximate radix-8 Booth multipliers have been proposed.

II RELATED STUDY

In this paper, we focus our attention on 4-2 compressors that are commonly used to build exact multipliers, owing to the simplified wiring and the efficient transistor level implementation. Several approaches have been recently proposed to design approximate 4-2 compressors and to use them to obtain approximate multipliers. Due to the large number of proposed solutions, the designer who wishes to use approximate 4-2 compressors in a multiplier has a difficult time in selecting the right topology. In this paper we present a comprehensive survey and comparison of previously proposed approximate 4-2 compressors, focusing on the architectures designed to be employed in standard treebased multipliers. We show that the stacking circuit technique can be modified to design approximate compressors, we highlight that some of the previously proposed approximate 4-2 compressors can be derived in this way and we present a new approximate 4-2 compressor. Overall, we analyze a total of twelve different approximate 4-2 compressors. The investigated circuits (plus a hybrid solution using two different approximate 4-2 compressors to reduce different parts of the partial product matrix) are employed to design 8 × 8 and 16 × 16 multipliers, implemented in 28nm CMOS technology. For each operand size, we analyze two multiplier configurations, with different levels of approximations, both signed and unsigned. The paper highlights that the error performance of multipliers using approximate 4-2 compressors depends on the specific connection of each partial product to each input of the approximate compressors. This point is overlooked in previous papers and makes challenging the design of the partial product reduction tree. Our analysis shows that approximate compressors are well suited for the implementation of unsigned multipliers, while their use in signed multipliers can cause a significant degradation of precision, especially when used in the left-most columns of the partial products matrix. The comparison between the circuits presented in this paper shows that a significant improvement in electrical performance is provided for certain approximate 4-2 compressors (more than 50% power and delay reduction), while other designs yield a much lower power saving. Considering the power-accuracy tradeoff, our analysis shows that there is no unique winning topology since the best solution depends on the required precision, on the considered error metric and on the signedness of the multiplier. We report tradeoff curves, showing power vs. precision that can be helpful for the selection of the best suited topology. We test the approximate multipliers in some examples of image processing.

It is shown that some of the investigated approximate multipliers, while performing well for image blending, show less satisfactory behavior in the other test cases. The main contributions of the paper are:

- A comprehensive survey and comparison of the approximate 4-2 compressors presented in the literature.
- Development of a novel approximate compressor, obtained by modifying the technique of the stacking circuit.
- Highlight that the partial product reduction tree with approximate compressors should be designed by optimizing the specific connection of each partial product to each input of the approximate compressors, to minimize the error rate.
- Show that approximate compressors are well suited for unsigned multipliers, while their use in signed multipliers can result in a significant degradation of precision.
- Show power-accuracy tradeoff curves that can be helpful for the selection of the approximate compressor that is best suited for a specific application.

III EXISTING SYSTEM

Multiplication using a RB multiplier includes three steps. In the first step, a RB Booth encoder (RBBE-2) generates the PPs, in which the operands are converted from NB to RB. In the second step, allRB PPs are accumulated by a PP reduction tree (PPRT) using RB 4:2 compressors. Finally, in the third step, the RB-NB converter (i.e., a fast adder) adds the two remaining PP rows. In the second step, there are several compression stages. The overall structure of an 8-bit RB multiplier is

shown in below Fig. The basic principle of a RB multiplier is to use the RB representation during the PP reduction, such that accumulation is carry free.

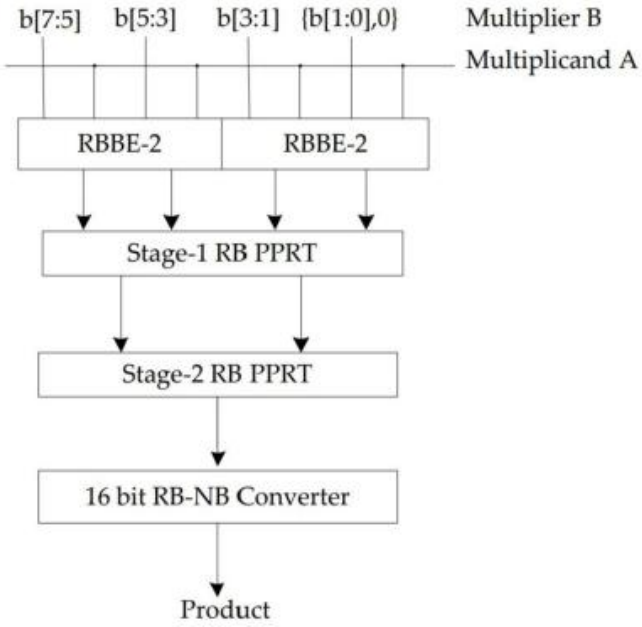

Fig.3.1. Overall structure of an 8-bit RB multiplier.

The Booth algorithm has been used to improve the sign correction issues of signed number multiplication, however, the original Booth algorithm does not reduce the number of PPs. A Modified Booth Encoding (MBE) method (also known as the radix-4 Booth algorithm) has been further proposed. It reduces the number of PP rows by half. The complexity of the parallel multiplier is reduced significantly by applying MBE. The power consumption and the delay of the entire multiplier are also reduced. Let A = aN-1aN-2…….a2a1a0 be the multiplicand and B = bN-1bN2……b2b1b0 be the multiplier. The multiplier bits are encoded so they are grouped in sets of three adjacent bits. The two side bits overlap with neighboring groups, except the first multiplier bit group. As per the encoded results from A, the Booth decoders select -2A, - A, 0, A, or 2A to generate the PP rows. 2A is obtained by a simple 1-bit left shift of the multiplicand. The negation operation is achieved by inverting each bit of A and adding 1 at its least significant bit (LSB) position. This is referred to as the correction term in this work. Therefore, the PP of each line can be easily generated by either shifting or inverting the multiplicand bits.

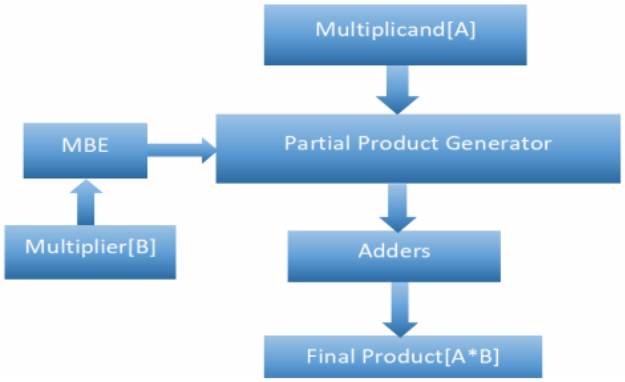

Fig.3.2. Block diagram of Existing approximate redundant binary multiplier.

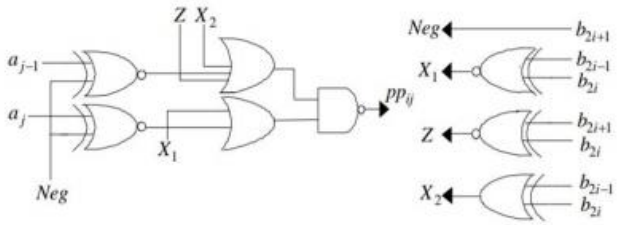

Fig.3.3. MBE scheme: encoder and decoder.

$a_j a_{j-1}$ \ $b_{2i+1}b_{2i}b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	0	1	1
01	0	0	1	0	1	0	1	0
11	0	1	1	1	0	0	0	0
10	0	1	0	1	0	0	0	1

Fig.3.4. K-Map of conventional MBE.

IV . PROPOSED SYSTEM

To accumulate the RB PPs, RB adders (RBAs) (including RB full and RB half adders) are used in the RB compression tree. As a RBA adds two RB operands (i.e., four NB operands) to produce one RB number (i.e., two NB numbers), it has four inputs and two outputs. Therefore, the RBA acts as an RB 4:2 compressor. The logic expressions of a redundant binary full adder (RBFA) are as follows

$$g_k = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+$$

$$h_k = x_k^- x_k^+ + y_k^- y_k^+$$

$$C_k^- = (x_k^- + x_k^+)(y_k^- + y_k^+)$$

$$C_k^+ = g_k C_{k-1}^- + \overline{g_k} h_k$$

$$S_k^- = g_k \oplus C_{k-1}^-$$

$$S_k^+ = C_{k-1}^+$$

Therefore, Sk - and Sk + can also be expressed as follows by combing the above equations

$$S_k^- = x_k^- \oplus x_k^+ \oplus y_k^- \oplus y_k^+ \oplus ((x_{k-1}^- + x_{k-1}^+)(y_{k-1}^- + y_k^+$$

$$\begin{aligned} S_k^+ =& (x_{k-1}^- \oplus x_{k-1}^+ \oplus y_{k-1}^- \oplus y_{k-1}^+)((x_{k-2}^- + x_{k-2}^+) \\ & (y_{k-2}^- + y_{k-2}^+)) + (x_{k-1}^- \oplus x_{k-1}^+ \oplus y_{k-1}^- \oplus y_{k-1}^+ \\ & (x_{k-1}^- x_{k-1}^+ y_{k-1}^- y_{k-1}^+) \end{aligned}$$

The RBHA can be designed with yk - = yk + = 0. The main advantage of RB multipliers that relay on RBAs, is the continuous carry-free characteristic. The RBA ensures that the addition time is fixed, so it is independent of the word length of the operands.

HIGH SPEED APPROXIMATE REDUNDANT BINARY MULTIPLIER1 (ARBM-1)

The block diagram of high speed approximate redundant binary multiplier-I is shown in below Fig. To perform the multiplication operation the multiplicand is loaded to partial product generator, the multiplier is loaded to modified booth encoder (MBE) which reduces the multiplier bits to half. The output of MBE is loaded to partial product generator. The partial products are generated by partial product generator and accumulation of partial products is done using ARBC-1.

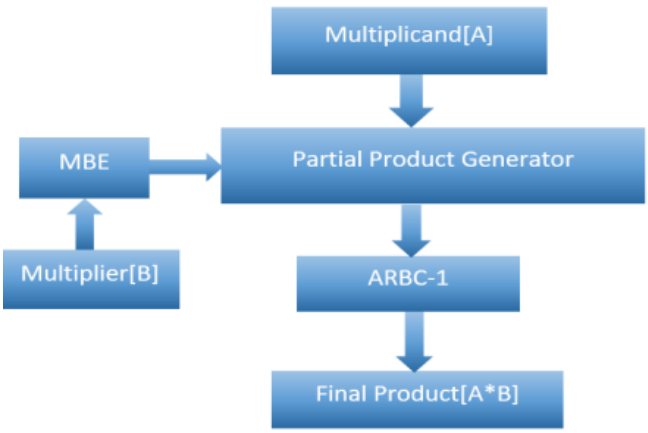

Fig.4.1. Block diagram of High speed approximate redundant binary multiplier-I.

The gate level circuit of the approximate RB compressor is given in below Fig. The approximate Sk1 + has only 3 gates, while the exact Sk1 + requires 12 gates. In total, ARBC-1 reduces the gate count from 19 to 10.

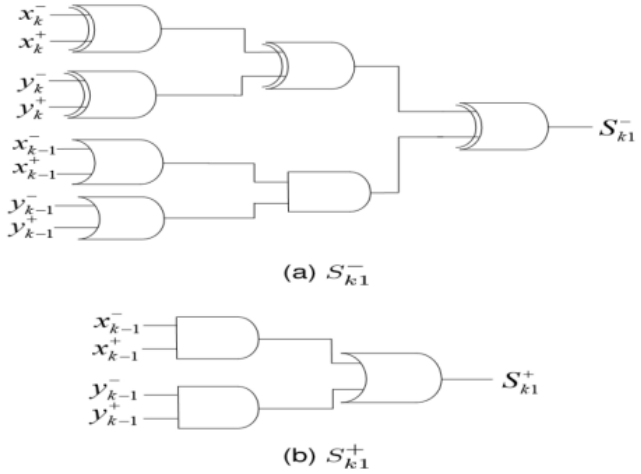

Fig.4.2. The gate level circuit of ARBC-1.

The gate level circuit of ARBC-1 is shown in Fig. 2. xk - , xk + , yk, yk + , xk-1 - , xk-1 + , yk-1, yk-1 + are the partial products generated from partial product generator. The output of ARBC-1 sk1 - (sum term) and sk1 + (carry term) and output of ARBC-1 is final product.

HIGH SPEED APPROXIMATE REDUNDANT BINARY MULTIPLIER2 (ARBM-2):

The block diagram of high speed approximate redundant binary multiplier-2 is shown in below Fig. To perform the multiplication operation the multiplicand is loaded to partial product generator, the multiplier is loaded to modified booth encoder (MBE) which reduces the multiplier bits to half. The output of MBE is loaded to partial product generator. The partial products are generated by partial product generator and accumulation of partial products is done using ARBC-2.

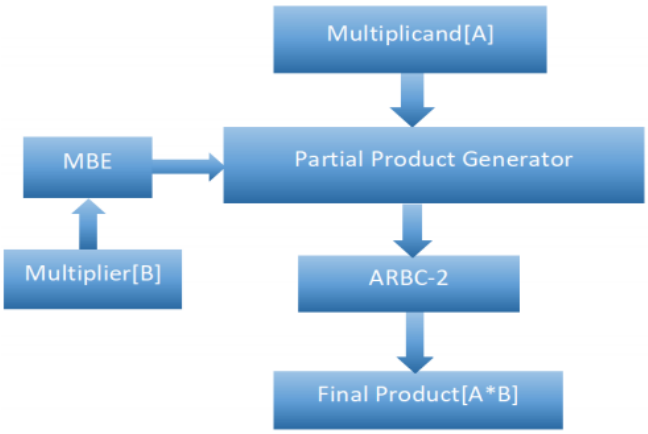

Fig.4.3. Block diagram of High speed approximate redundant binary multiplier-2.

ARBC-2 generates results that are larger than its exact counterpart. The gate level design of the approximate RB compressor is given in below Fig. ARBC-2 further reduces the gate count of Sk2 - from 7 to 5. Therefore, ARBC-2 reduces the gate count from 19 to 6, which is significantly simpler than ERBC.

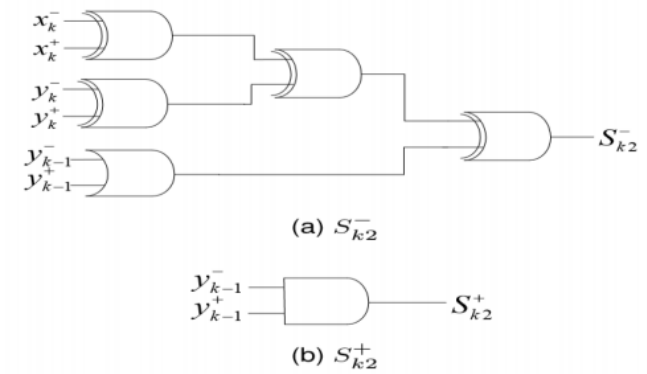

Fig.4.4. The gate level circuit of ARBC-2.

The gate level circuit of ARBC-2 is shown in above Fig xk - , xk + , yk, yk + , yk-1, yk-1 + are partial products given as input and sk2 - (sum term), sk2 + (carry term) and output of ARBC-2 is final product.

5. SIMULATION RESULTS

EXISTING APPROXIMATE REDUNDANT BINARY MULTIPLIER REPORT:

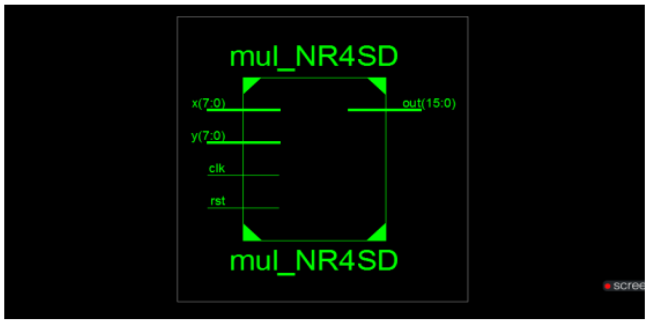

Fig.5.1. RTL diagram of ARBM.

When x=26 (8'b00011010); y=10 (8'b00001010) Out =260 (16'b0000000100000100) Here output 'out' is correct output (out=260).

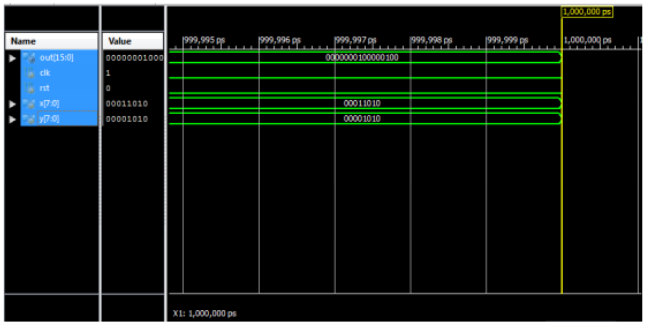

Fig.5.2. Simulation output of existing system.

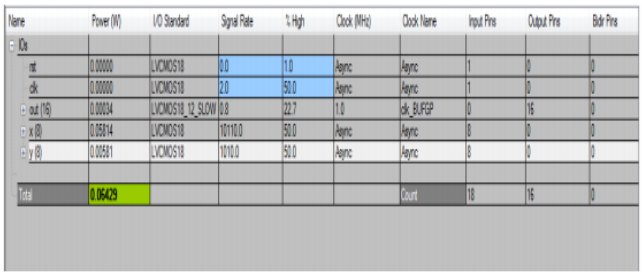

Fig.5.3. Signal rate display.

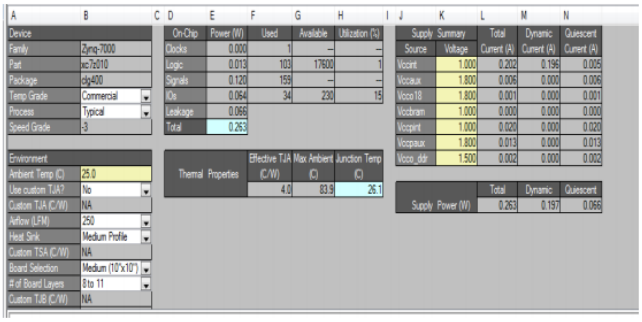

Fig.5.4. Power report.

HIGH SPEED APPROXIMATE REDUNDANT BINARY MULTIPLIER- 1 REPORT:

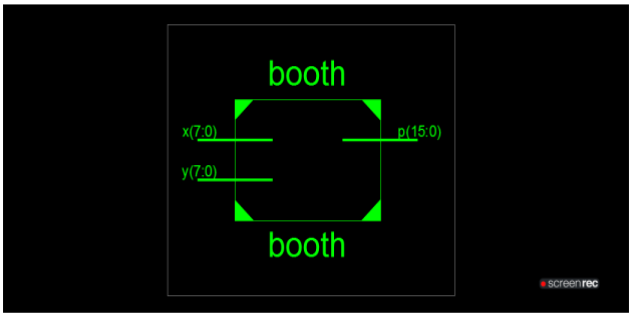

Fig.5.5. RTL diagram.

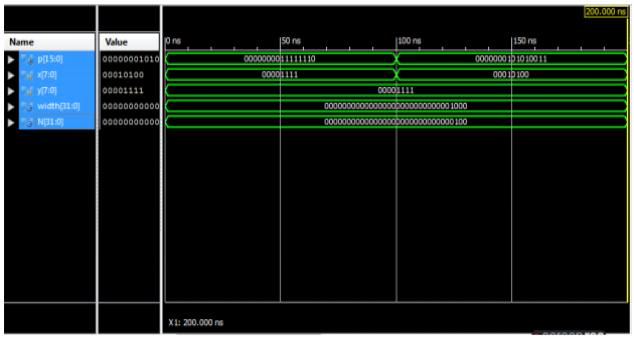

Fig.5.6. Simulation output for 8X8 multiplication using ARBM-1.

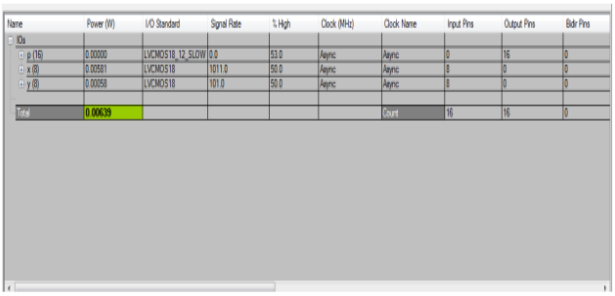

Fig.5.7. Signal rate of ARBM-1.

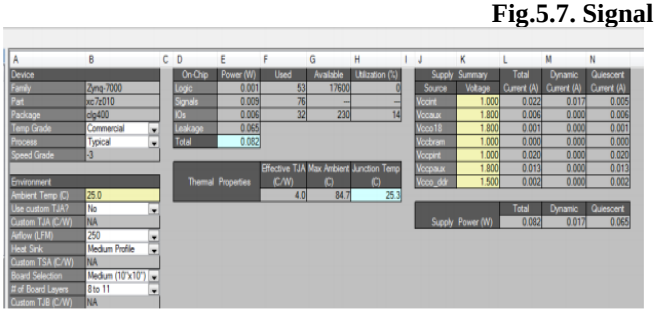

Fig.5.8. Power report of ARBM-1.

HIGH SPEED APPROXIMATE REDUNDANT BINARY MULTIPLIER- 2 REPORT:

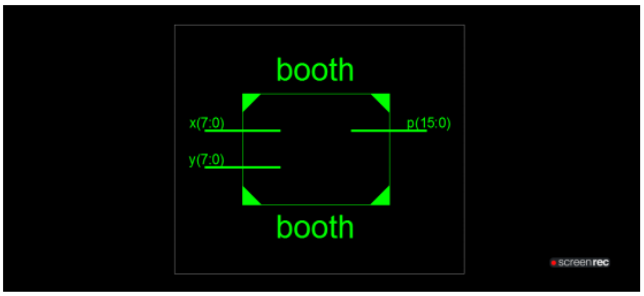

Fig.5.9. RTL SCHEMATIC of ARBM-2

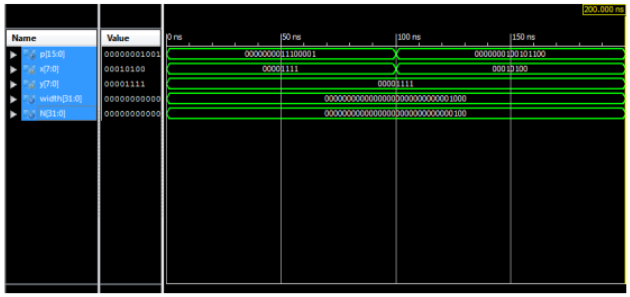

Fig.5.10. Simulation Output of ARBM circuit.

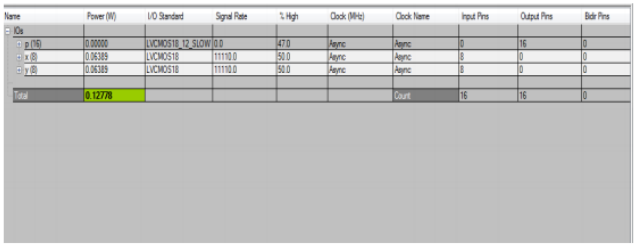

Fig.5.11. Signal rate of ARBM-2.

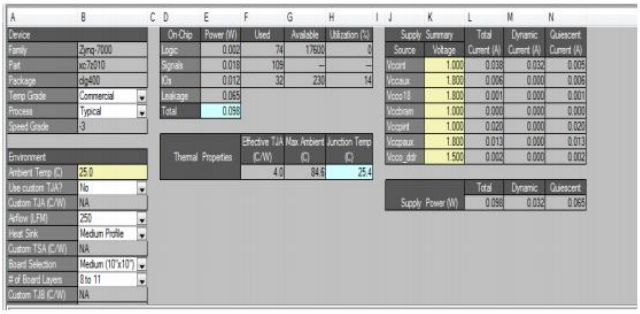

Fig.5.12. Power report of ARBM-2.

COMPARISION OF ARBM, ARBM-1 & ARBM-2:

	POWER (watts)	DELAY (ns)	SIGNAL RATE
ARBM	0.263	6.078	0.06429
ARBM-1	0.082	2.827	0.00639
ARBM-2	0.098	5.102	0.12778

5. CONCLUSION

In this project the design of existing approximate redundant binary multiplier (ARBM) using modified booth encoder is studied and implemented. In ARBM the accumulation of partial products is done using adders. Due to the propagation of carry in adders the speed of the multiplier is reduced. To overcome this issue compressors are used instead of adders. A high speed approximate redundant binary multiplier-I (ARBM-1) is designed using approximate redundant binary 4:2 compressor-1 (ARBC-1) to perform the approximate partial product accumulation. A high speed approximate redundant binary multiplier- -II (ARBM-2) is designed using approximate redundant binary 4:2 compressor-2 (ARBC-2). ARBC-2 is less complex compared to ARBC-1. ARBC-I , ARBC-II are faster than ARBC.

REFERENCES

[1]. Weiqiang Liu, Tian Cao, Peipei Yin, Yuying Zhu,(2018) “Design and Analysis of Approximate Redundant Binary Multipliers”. IEEE Transactions on computers, vol. xx, no. xx

[2]. R.S.Keote, P.Y.Karule (2016), ” VLSI design of 64bit × 64bit high performance multiplier with redundant binary encoding”, IEEE Conference ICACCA.

[3]. M. Sandhya Rani, K. Naveen Kumar Raju (2018) “Design and Implementation Radix based Booth Multiplier Using High Speed Applications” IJSRCSEIT | Volume 3 | Issue 5 | ISSN : 2456-3307.

[4]. Weiqiang Liu, Liangyu Qian, Chenghua Wang , Honglan Jiang (2017) “Design of Approximate Radix-4 Booth Multipliers for Error Tolerant Computing”, IEEE Transactions on computers, Volume: 66, Issue: 8.

[5] S. T. Chakradhar and A. Raghunathan, “Best-effort computing:Re-thinking parallel software and hardware,” in Proc. 47th ACM/IEEEDesignAutom. Conf., Jun. 2010, pp. 865–870.

[6] A. K. Verma, P. Brisk, and P. Ienne, “Variable latency speculativeaddition: A new paradigm for arithmetic circuit design,” in Proc. Design,Autom. Test Eur., Mar. 2008, pp. 1250–1255.

[7] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, “Evaluating MapReduce for multi-core and multiprocessor systems,” in Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit. (HPCA), Feb. 2007.

[8] B. Jose and D. Radhakrishnan, “Delay optimized redundant binary adders,” in Proc. 13th IEEE Int. Conf. Electron., Circuits Syst. (ICECS), Dec. 2006, pp. 514–517.

[9] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Annu. Conf.VLSI Design, Jan. 2011, pp. 346–351.M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting", in Proc. SIGGRAPH, pp. 417–424, 2000.
[10] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst., Vol. 23, No. 6, pp. 1180–1184, Jun. 2015.