ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)

Vol 3, Issue 11, November 2017

# Design of Reliable Low-Power Multiplier Using Fault-Tolerant Technique

[1] M.M.Samreen
M.tech: Reliability Engineering
JNTU Ananthapuramu

***Abstract*: -- A Reliable low-power and fast accelerated multiplier is implemented by using a redundant fault tolerant technique, "Input Output Logic Based"(IOLB) ,this technique is implemented in the fast accelerated dada multiplier is proposed by implementing two algorithms i.e. partial products are obtained in to two parts and integration of a designed hybrid final adder with dada multiplier and obtained partial products are also implemented by using array based multiplier ,tradeoffs between these two multipliers are compared with and without IOLB logic by using the fault tolerant technique power of the multiplier is reduced to 31.1 %, area overhead of the gate's required in the multiplier is 23.2% ,speed of the multiplier is increased to 40.1%.**

***Keywords* — Redundant fault-tolerant technique,"input-output logic based",column compression.**

## I. INTRODUCTION

In digital signal processing and various applications, digital multipliers play a vital role. Reliability is defined as where the probability of failure is one percent. So, it is important to design a reliable multiplier To increase the reliability of the multiplier redundant fault- tolerant technique input-output based logic is implemented in the proposed multipliers parallel structure based Dadda multiplier and array-based multiplier, in the array based multiplier minimization is done by cutting down the bit size to N/2. In the array-based multiplier [1] Baugh–Wooley algorithm was proposed to partition the partial products and in the Dadda based multiplier partial products are generated by parallel structures, In the parallel structure partial products are arranged in N-rows and the reduction of the N-row partial products to a height of one bit is done by grouping it in two sets of three bit and two bit. By using HA and FA the grouped 3-bit and 2-bit partial product computation is carried out. This paper is organized as follows section II will describe the algorithm proposed in the array based multiplier and how the multiplication is performed and the design of parallel structures, the generation of partial products by partitioning technique is implemented and section III describes the tradeoffs between an array based multiplier and a Dadda multiplier, section IV describes the comparison between the two multipliers with and without redundant fault tolerant technique.

## II. ALGORITHMS IN THE PROPOSED MULTIPLIERS

### *A. Baugh_wooley algorithm based array multiplier*

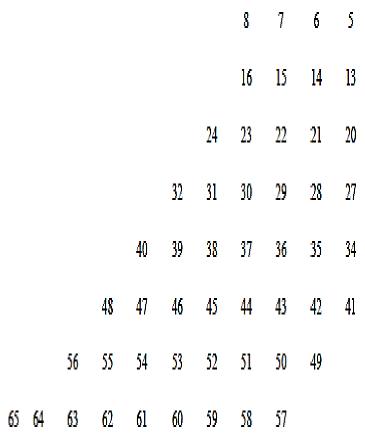
In the Baugh_wooley array multiplier multiplication is done by partitioning the partial products in two subsets as ICV(input correction vectors) and MICV(minor input correction vectors) upon which eliminating the weighted partial product according to the algorithm, based on the algorithm the full adders which are placed in the multiplier are eliminated and replaced with equivalent logic gates ,so that the no of transistors required to design a full adder gets reduced, but the array based multipliers are not high accelerated so, we go for either dadda or Wallace multiplier, comparatively dadda multiplier is the high speed multiplier, is designed by the partitioning of partial products in to two parts as part 0 and part 1 by following the algorithm consider two n-bit operands A and B for NxN baugh_wooley array multiplier which forms the partial products of a matrix of the n rows and 2n-1 columns as shown in the fig1(a)

### *B.Computation of the partial products in dadda multiplier*

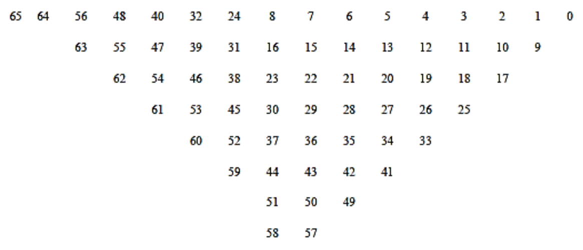
In this work, the proposed dadda multiplier partition of partial products is done in to part 0 and part 1, after partitioning in to two parts and in each part a partial product is grouped in to three bit and two-bit sized.So, that the 3-bit sized partial products are computed using fa and the 2-bit sized is computed by using ha also called as (2, 2) counter. Computation of the partial products by using HA (Half Adder) and FA (Full Adder) the different rows is classified with the different colors shown in fig.2 and fig.3 Here the partial sum and carry is denoted by s and c E.g. In fig.2 in the computation of the partial products for part 0 consider the bit positions of 6 and 13 which are computed by using ha i.e. (2, 2) counter s0 and c0 are generated. The c0 is carried to the next column. The bit positions of 7, 14 and 21 are computed using a fa i.e. (3, 2) counter which generates the sum and carry as s1 and c1.

ISSN (Online) 2395-2717

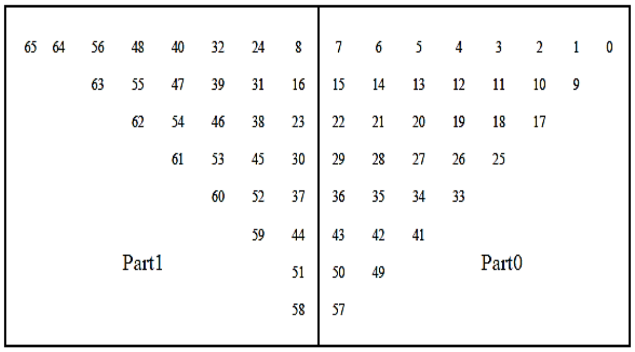
International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 3, Issue 11, November 2017



A



B



C

*Fig 1.Partition of Partial Products into Two Parts Part 0 and Part 1*

Fig .2 and Fig.3 depicts the obtained partial products are of height of one bit, it is obtained by adding the carry c1 of a fa also called as (3, 2) counter is added to the next set of (3, 2) counter set and the obtained partial products of height of one-bit is computed by using carry look-ahead adder final two rows of each part are summed using a carry look-ahead adder

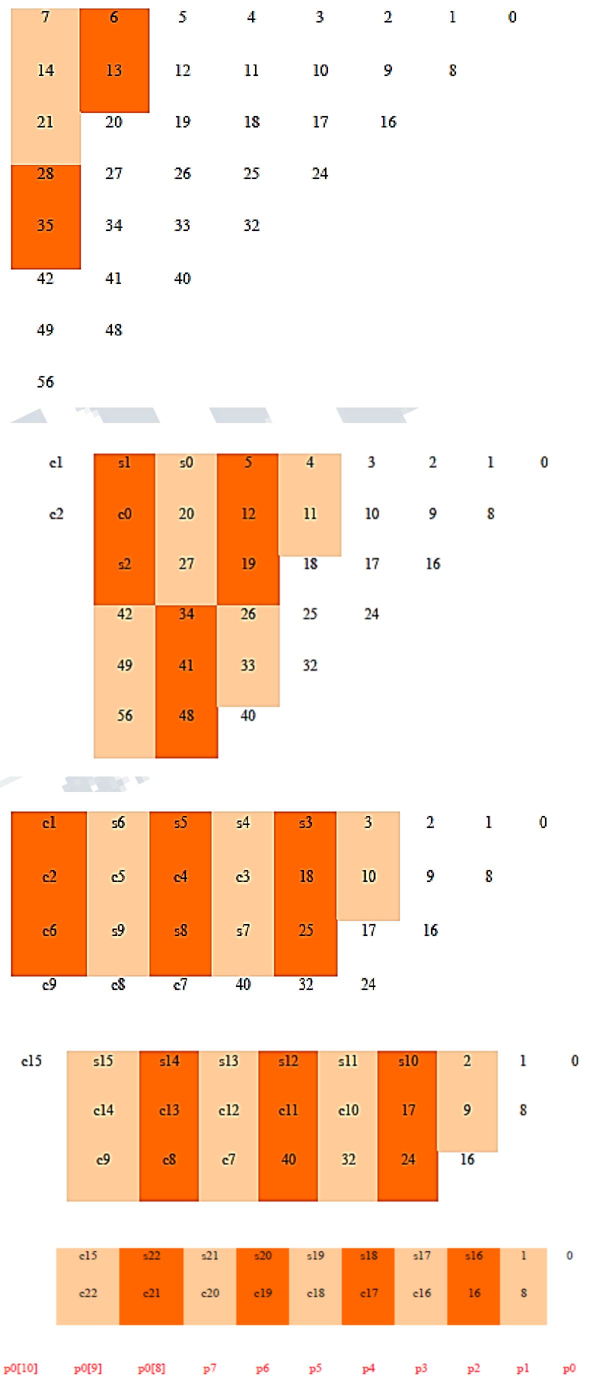


*Fig 2.Computation of Partial Products Using HA and FA Based On Dadda Approach.*

The computation procedure for part 0 and part 1 is depicted in fig.4 and the computation of partial products from part 0 and part 1 are summed by using HA and FA. Fig2 and Fig.3 gives the pictorial representation of dadda based approach p0, p1denote the partial products obtained from part 0 and part 1.Numericals are indicated on the partial products obtained from the outputs of part 0 and part 1. Outputs of the part 0 and part 1 are computed independently in parallel.

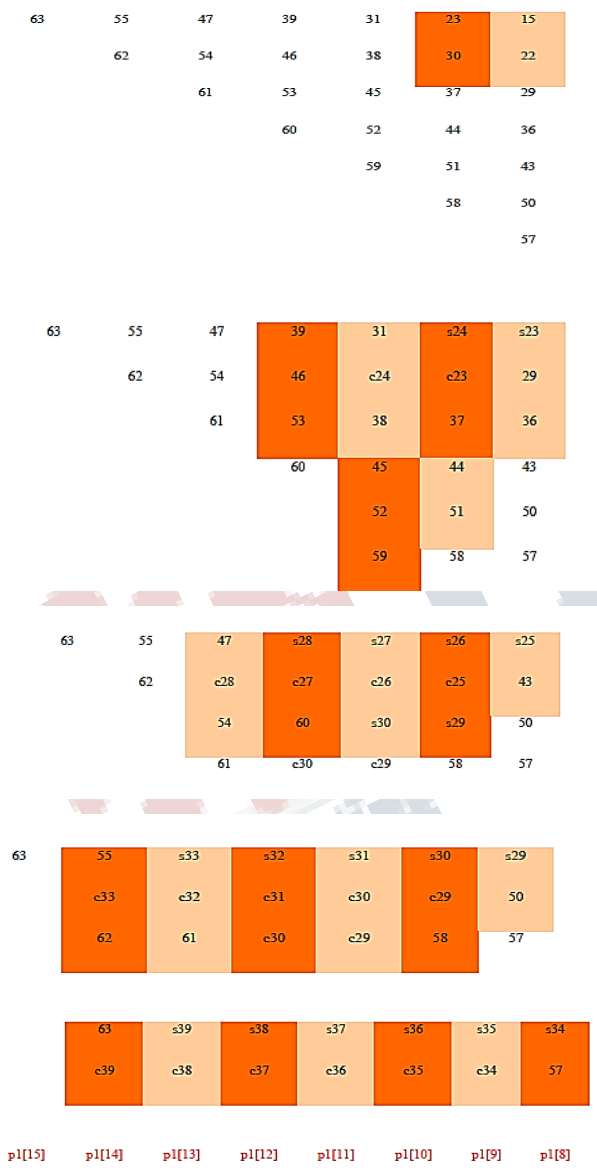


*Fig 3.Minimization of Partial Products of Part 1 Based On Dadda Reduction Algorithm*

The final computation of the partial products of part 0 and part 1 is done by using the ha and far which is known as (2, 2) counters and (3, 2) counters and obtained partial products is shown in fig 4

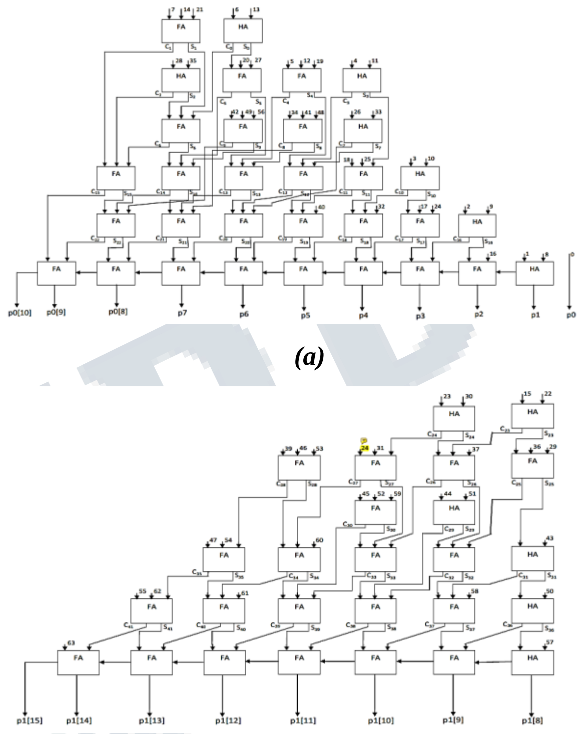


*Fig 4.Computation of Partial Products Using HA and FA (A) Computation of Part0 (B) Computation of Part1*

Integration of a hybrid adder which is designed for dadda multiplier is applicable in achieving the fast accelerated multiplication. In the previous designs of the hybrid final adder carry look ahead adder and carry select adder are integrated. But due to the design of CLSA (carry select adder), its area overhead occupancy is more. So, to achieve the optimum performance a hybrid adder is proposed in this work in this proposed hybrid adder design is done by the code converter i.e. MBEC (Multiplexer With Binary to Excess 1 Converter) and ripple carry adder for the fast summation of partial products originating from the PPST. By using the MBEC adder it provides faster performance than Carry Save Adder (CSA) and Carry Look Ahead (CLA) adder. Also, its area overhead occupancy is less and low-power consumption than Carry Select Adder (CSLA). Once each part of the partial products from part 0a and part 1 to a height of one bit obtained in two rows is as follows. By using RCA (Ripple Carry Adder) computation of these two rows is done,

*P0[10]P0[9] P0[8]P[7]P[6]P[5]P[4]P[3]P[2]P[1]*P[0]
*p1 [15] p1 [14] p1 [13] p1 [12] p1 [11] p1 [10] p1 [9] p1 [8]*

Once the partial products obtained from part 0 and part 1 the partial products obtained from the part 0 i.e. p0 [7] to p0 [0] i.e. p0 [7:0] are assigned as the final products as p [7:0], partial products from p0 [10:8] are the extra carry bits generating from part 0 and the partial products from part1 p1[15:8] is the carry part generated from part1 to find the remaining final products p[15:8] an hybrid adder with the integrated MBEC and RCA is used which is shown in fig 5.
A Single 5-bit multiplexer with binary to excess 1 converter is used in the final 8-bit hybrid adder. If the bit size increases then we require more no of MBEC such that each one requires a selection input from the preceding carry of the MBEC, so to generate the carry an additional block is developed to generate carry MBECWC, the detailed structure of the MBEC is shown Fig. 6.The BEC is used in the final adder design 16 bit and 32-bit multipliers.

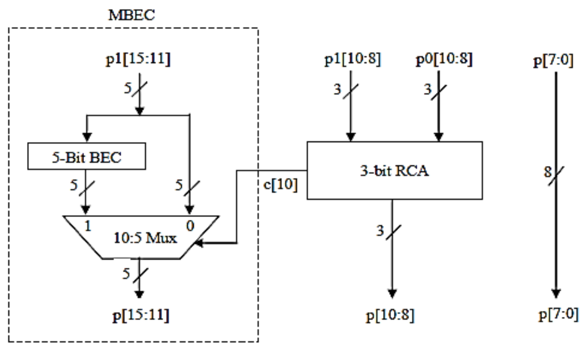


*Fig 5.Integrated Hybrid Adder Of 8-bit dadda Multiplier*

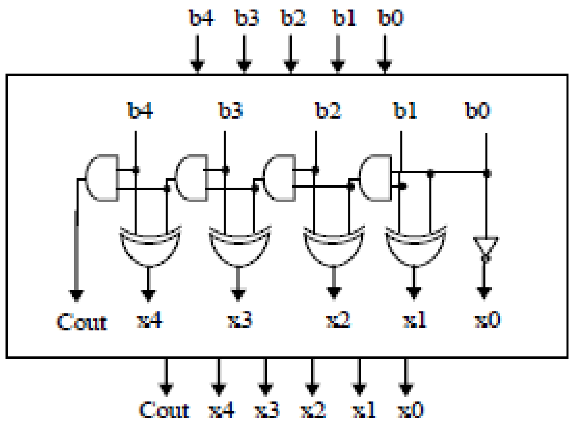


*Fig 6.The Code Converter for 5-bit binary to excess 1 code*

## III.THE REDUNDANT FAULT-TOLERANT TECHNIQUE USED IN THE PROPOSED MULTIPLIER

The motivation of the proposed redundant fault-tolerant technique i.e. Input-Output Logic Based (IOLB) is to generate an error-free signal by assessing the output i.e. change in output is caused by change in the input following system stability i.e. Bounded Input Bounded Output(BIBO). By assessing the changes in input and output signal a Xo red error signal is generated with the output signal to yield the error-free signal. The main objective of using the IOLB logic in the multipliers is, multipliers are designed with the HA and FA which again compromises of the logic gates a XOR, a NOT, and an OR gate. So, as to get the fault-free outputs and system to work reliably even though if there exists any fault in the logic gates IOLB(input-output logic based) fault-tolerant technique is implemented

### III.1 NOT GATE

We shall now consider the case of a NOT gate to illustrate the mechanism involved in designing the IOLB correction circuit. Say A is its input and B is its output. In an error-free scenario, if input A is ’1’, then B would be ’0’. However, say A is now changed to ’0’ and there is no accompanied change in B (that is, it stays at ’0’). Then, that indicates that there is an error, since, in a NOT gate, a change in input is expected to bear a change in the output too. Let us now examine a couple of other cases of input-output pair transitions. If the pair (format: ’AB’) changes from ’01’ to ’00’, this means that the output change has occurred without there being a change in the input. This is again unexpected behavior for a normal NOT gate. Similar would have been the case if the pair transitioned from ’10’ to ’11’.These above relations form the central idea in designing the error correcting circuit. Suppose Ac is the change in the input, Bc is the change in the output and E is the error signal.

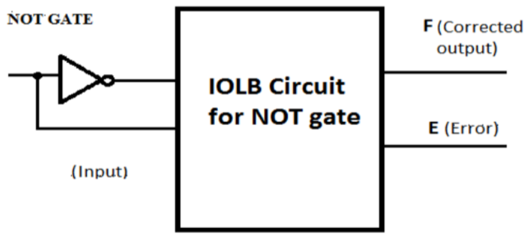


*Fig 7.IOLB circuit for a NOT gate*

For the computation of changes in variables, we take the XOR of a variable with a delayed version of itself, thereby giving us

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 3, Issue 11, November 2017

’1’ if there has been a change. The value of the delay is arbitrary. While trying to compute BC from B, we need to take care of the possibility of an error having occurred in the NOT gate. If we just perform XOR of the delayed and current values of B, we will get an erroneous Bc if the NOT gate is error-affected (since B would be erroneous). Hence, if the error signal E is ’1’ (indicating that the NOT gate is affected), we take XOR with the NOT of delayed B for the computation of Bc. A block diagram of an IOLB NOT gate is illustrated in Fig. 7 and the IOLB circuit for the NOT gate is shown in Fig 8.

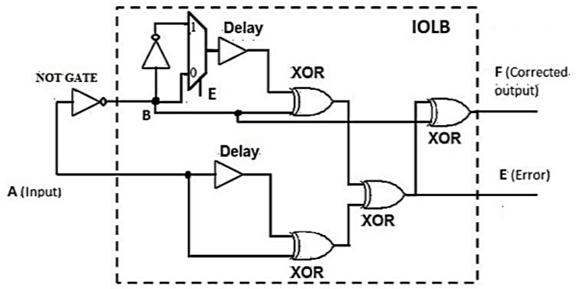


*Fig 8.An IOLB Circuit Block Diagram for a NOT Gate*

### *III.2 Exclusive OR (XOR) Gate*

We now consider the case of a two-input XOR gate. Say A and B are its inputs and S is its output. In an error-free scenario, if both A and B are same, then S would be ’0’ and if both are different then S would be ’1’. However, in the case of both A and B are same and S is ’1’ or in the case of both A and B are different and S is ’0’, that would mean there is an error. Such a case will arise when an expected output change does not succeed a change in an input/ when an unexpected output change occurs, even without a change in inputs. We shall now look at specific instances of the above cases. Say the current state (format: ’ABS’) is ’000’. Suppose a transition occurs from ’000’ to ’100’, this is unexpected because, in an XOR gate, the output is expected to change following a change in exactly in one of the input. Suppose a transition occurs from ’000’ to ’001’, this is again unexpected, since, in a XOR gate, the output cannot change without a change in its inputs. Henceforth, we shall use c-subscripted symbols to denote changes in variables. For instance, Xc denotes a change in the variable X. In other words, Xc=’1’ means there has been a change in X and Xc=’0’ means there hasn’t been a change in X For arriving at changes in variables, we take the XOR of a variable and its delayed version. However, like in the case of the NOT gate, the computation of the Sc is not straightforward. We resolve the problem in the same way as was done in the case of the NOT gate: in the computation of the Sc, we have used the NOT of delayed S if E = ’1’. For producing the fault-free output, the error signal E is XOR-ed with S. A complete picture is illustrated

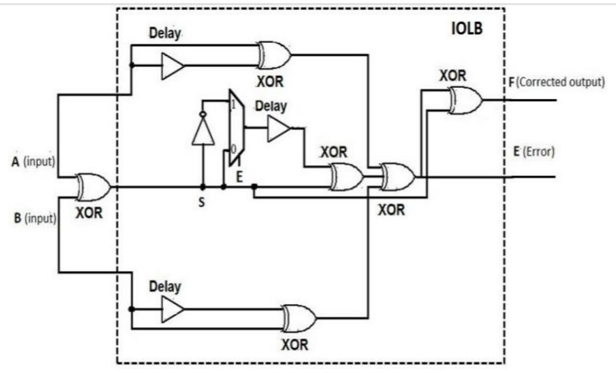


*Fig 9.Circuit Diagram for IOLB XOR Gate*

## IV.RTL(REGISTER TRANSFER LEVEL) SIMULATION RESULTS OF THE PROPOSED MULTIPLIERS

The proposed multipliers, code work is written in the Verilog hardware description language and simulation is carried out in the native compiler of the cadence suite and in Xilinx 14.1 ISE suite, and the simulation results of the array based multiplier for 12-bit, 24-bit and dadda based multiplier for 8-bit, 16-bit, 32-bit is illustrated below and their schematics are illustrated below

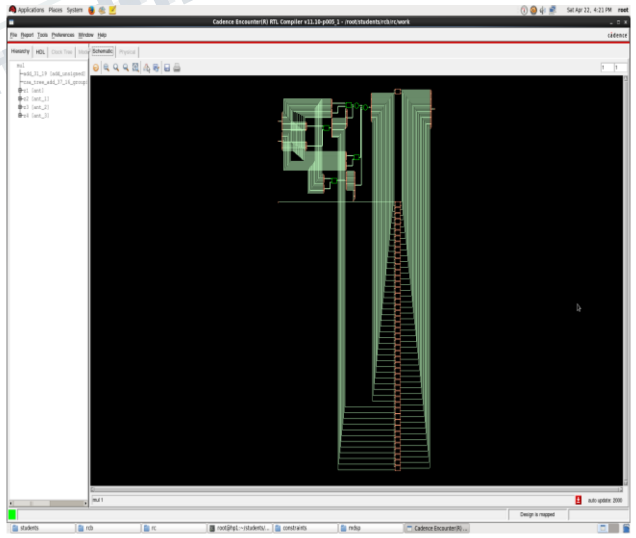


*Fig 10.Schematic Of The Array Based 12-bit Multiplier*

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 3, Issue 11, November 2017

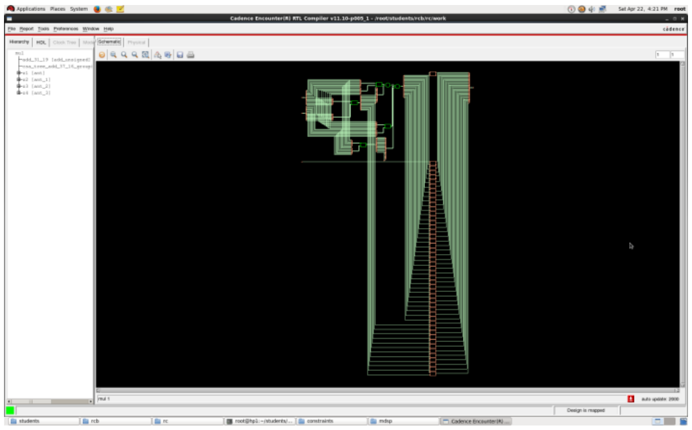


*Fig 11.Schematic Of the Array Based 24-bit Multiplier*

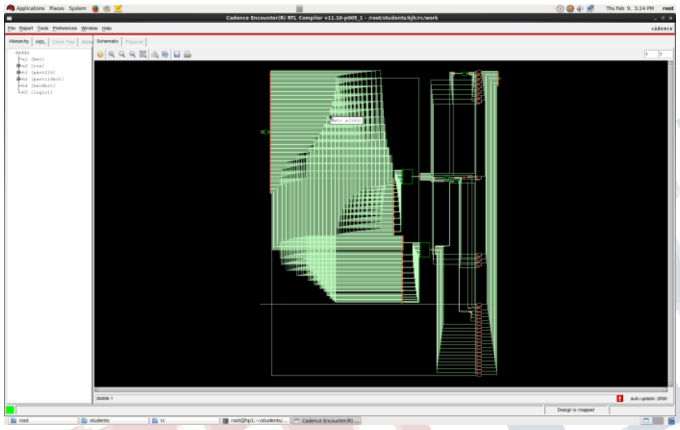


*Fig 12.Schematic Of the Dadda Based 16-bit Multiplier*

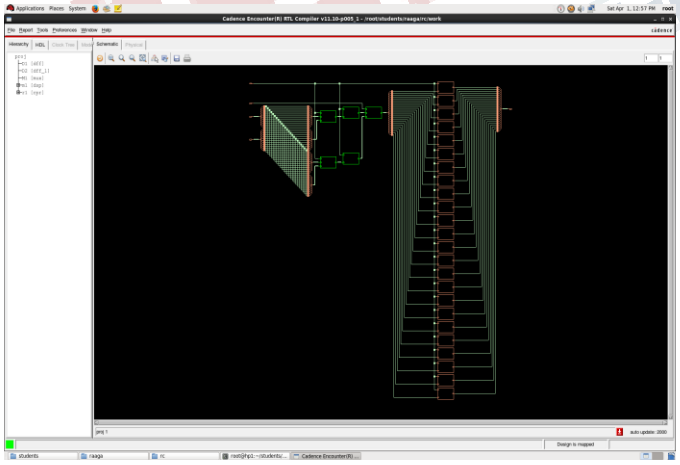


*Fig 13.Schematic of the Dadda Based 32-Bit Multiplier*

And the comparison of the proposed multipliers in the aspects of area, power and delay is done with and without IOLB (input-output logic based) which is shown in below tables and also the comparison of the proposed regular dadda multiplier with the partitioned hybrid multiplier and array-based is also done

*Table 1.An Array Based Multiplier*

| Bit length | Area in micrometer | Power in microwatts | Delay in nanoseconds |
|---|---|---|---|
| 12*12 | 1102.05 | 502.411 | 240.7 |
| 24*24 | 1112.05 | 2207.4 | 1200.9 |

*Table 2 .Dadda Based Multiplier with CLA Adder*

| Multiplier NxN | Area ( µm2) | Power (µW) | Delay (ns) |
|---|---|---|---|
| 8 bit | 8957 | 6.85 | 3.51 |
| 16 bit | 30,241 | 35.22 | 4.61 |
| 32 bit | 107,362 | 218.76 | 5.47 |

*Table 3. Dadda based Multiplier with Integrated Hybrid Adder*

| Multiplier NxN | Area (µm2) | Power (µW) | Delay (ns) |
|---|---|---|---|
| 8 bit | 9144 | 7.07 | 3.38 |
| 16 bit | 30,577 | 35.99 | 4.13 |
| 32 bit | 107491 | 221.01 | 4.07 |

## V. RESULT SUMMARY

The proposed dadda multiplier shows 5.2% lesser power consumption compared to the array based multiplier. As the bit size of the multiplier grows power requirement of the dadda multiplier is reduced when compared to the array based multiplier. A 32-bit Dadda multiplier requires only 4.7% less power than the array-based multiplier. By implementing the Redundant fault-tolerant technique for the multiplier reliability of the system is increased to 68 %.

## VI. CONCLUSION

The fast-accelerated multiplication has been achieved by partitioning of the partial products into two parts i.e. part 0 and part 1and computation of them is performed then with the parallel adder. On analyzing the simulation results and tabulation values of the area, power and speed-delay of the proposed multipliers shows that there is an improvisation in the speed-delay and power of the dadda based multiplier compared with the array-based multiplier

## REFERENCES

1. I-chyn Wey,Chien-Chang Peng, and Feng-Yu Liao “Reliable Low-power Multiplier Design Using Fixed Width Replica Redundancy Block” IEEE trans,Very Large Scale Integr.(VLSI) syst vol 23 No 1 .
2. E. E. Swartzlander, Jr. and G. Goto, "Computer arithmetic," *The Computer Engineering Handbook*, V. G. Oklobdzija, ed., Boca Raton, FL: CRC Press, 2002.
3. C. S. Wallace, “A Suggestion for a Fast Multiplier,” *IEEE Transactions on Electronic Computers*, Vol. EC-13, pp. 14-17, 1964.
4. Luigi Dadda, “Some Schemes for Parallel Multipliers,” *Alta Frequenza*, Vol. 34, pp. 349-356, August 1965
5. K.C. Bickerstaff, E.E. Swartzlander, M.J. Schulte, Analysis of column compression multipliers, Proceedings of 15th IEEE Symposium on Computer Arithmeitc,2001.
6. W. J. Townsend, Earl E. Swartzlander and J.A. Abraham, “A comparison of Dadda and Wallace multiplier delays”, Advanced Signal Processing Algorithms, Architectures and Implementations XIII. Proceedings of the SPIE, vol. 5205, 2003, pages 552-560
7. [7] B. Shim, S. Sridhara, and N. R. Shanbhag, “Reliable low-power digitalsignal processing via reduced precision redundancy,” *IEEE Trans.Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 497–510,May 2004.