MODIFIED DESIGN OF LOW-POWER HIGH SPEED TRUNCATION-ERROR-TOLERANT ADDER AND ITS APPLICATION IN DIGITAL SIGNAL PROCESSING

ANNA JOHNSON

M-tech student, Mangalam College Of Engineering Ettumanoor
E-mail: annajohnsonnadackal@gmail.com

Abstract: In modern VLSI technology, the occurrence of all kinds of errors has become inevitable. By adopting an emerging concept in VLSIdesign and test, error tolerance (ET), a novel error-tolerant adder (ETA)is proposed. The ETA is able to ease the strict restriction on accuracy, and at the same time achieve tremendous improvements in both the power consumption and speed performance. One important potential application of the proposed ETA is in digital signal processing systems that can tolerate certain amount of errors**.**Thus by modifications, we can reduce power,delay and area.

I. INTRODUCTION

In conventional digital VLSI design, one usually assumes that a usable circuit/system should always provide definite and accurate results. But in fact, such perfect operations are seldom needed in our non digital worldly experiences. The world accepts "analog computation," which generates "good enough" results rather than totally accurate results. The data processed by many digital systems may already contain errors. Error tolerance (ET) and the PCMOS technology are two of them. According to the definition, a circuit is error tolerant if: 1) it contains defects that cause internal and may cause external errors and 2) the system that incorporates this circuit produces acceptable results. However, for many digital signal processing (DSP) systems that processsig nalsrelating to human senses such a shearing, sight, smell, and touch, e.g., the image processing and speech processing systems, the error-tolerant circuits may be applicable.

II. ERROR TOLERANT ADDER

Increasingly huge data sets and the need for instant response require the adder to be large and fast. The traditional ripple-carry adder (RCA) is therefore no longer suitable for large adders because of its low-speed performance. Many different types of fast adders, such as the carry-skip adder (CSK) , carry-select adder (CSL) ,and carry-look-ahead adder (CLA) , have been developed. Also, there are many low-power adder design techniques that have been proposed . However, there are always trade-offs between speed and power. The error-tolerant design can be a potential solution to this problem. By sacrificing some accuracy, the ETA can attain great improvement in both the power consumption and speed performance.

In a conventional adder circuit, the delay is mainly attributed to the carry propagation chain along the critical path, from the least significant bit (LSB) to the most significant bit (MSB). Meanwhile, a significant proportion of the power consumption of an adder is due to the glitches that are caused by the carry propagation. Therefore, if the carry propagation can be eliminated or curtailed, a great improvement in speed performance and power consumption can be achieved. In this paper, we propose for the first time, an innovative and novel addition arithmetic that can attain great saving in speed and power consumption.
We first split the input operands into two parts: an accurate part that includes several higher order bits and the inaccurate part that is made up of the remaining lower order bits. The length of each part need not necessary be equal.

The addition process starts from the middle (joining point of the two parts) toward the two opposite directions simultaneously. In the example of Fig. 1, the two 16-bit input operands, "1011001110011010" (45978) and "0110100100010011" (26899), are divided equally into 8 bits each for the accurate and inaccurate parts.

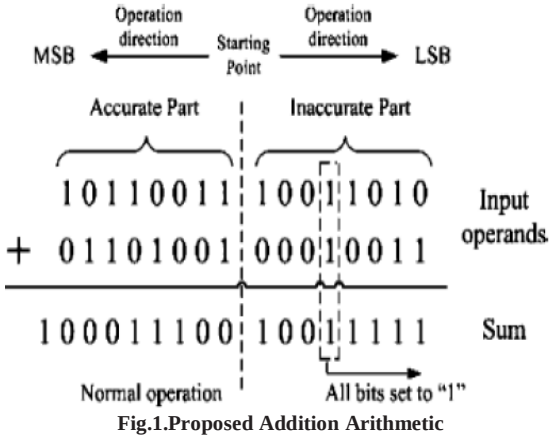


Fig.1.Proposed Addition Arithmetic

The addition of the higher order bits (accurate part) of the input operands is performed from right to left (LSB to MSB) and normal addition method is applied. This is to preserve its correctness since the higher order bits play a more important role than the

lower order bits. The lower order bits of the input operands (inaccurate part) require a special addition mechanism. No carry signal will be generated or taken in at any bit position to eliminate the carry propagation path. To minimize the overall error due to the elimination of the carry chain, a special strategy is adapted, and can be described as follow: 1) check every bit position from left to right (MSB to LSB); 2) if both input bits are "0" or different, normal one-bit addition is performed and the operation proceeds to next bit position; 3) if both input bits are "1," the checking process stopped and from this bit onward, all sum bits to the right are set to "1." The addition mechanism described can be easily understood from the example given in Fig. 1 with a final result of "10001110010011111" (72863). The example given in Fig. 1 should actually yield "10001110010101101" (72877) if normal arithmetic has been applied. The overall error generated can be computed as OE=72877-72863=14. The accuracy of the adder with respect to these two input operands is ACC=(1-(14/72877))*100%=99.98%.

The block diagram of the hardware implementation of such an ETA that adopts our proposed addition arithmetic is provided in Fig.2. This most straightforward structure consists of two parts: an accurate part and an inaccurate part. The accurate part is constructed using a conventional adder such as the RCA, CSK, CSL, or CLA. The carry-in of this adder is connected to ground. The inaccurate part constitutes two blocks: a carry-free addition block and a control block. The control block is used to generate the control signals, to determine the working mode of the carry-free addition block. A 32-bit adder is used as an example for our illustration of the design methodology and circuit implementation of an ETA.

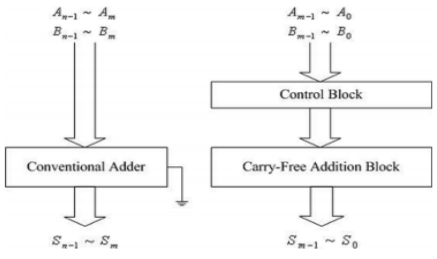


Fig.2.Block Diagram

III. DESIGN OF A 32-BIT ERROR TOLERANT ADDER

The first step of designing a proposed ETA is to divide the adder into two parts in a specific manner. The dividing strategy is based on a guess-and-verify stratagem, depending on the requirements, such as accuracy, speed, and power. First, we define the delay of the proposed adder as Td=max(Th,Tl), where Th is the delay in the accurate part and Tl is the delay in the inaccurate part. With the proper dividing strategy, we can make Th approximately equal to Tl and hence achieve an optimal time delay. With this partition method defined, we then check whether the accuracy performance of the adder meets the requirements preset by designer/customer. and the acceptance probability to be 98%. The proposed partition method must therefore have at least 98% of all possible inputs reaching an accuracy of better than the above, we divided the 32-bit adder by putting 12 bits in the accurate part and 20 bits in the inaccurate part. In our proposed 32-bit ETA, the inaccurate part has 20 bits as opposed to the 12 bits used in the accurate part. The overall delay is determined by the inaccurate part, and so the accurate part need not be a fast adder. The ripple-carry adder, which is the most power-saving conventional adder, has been chosen for the accurate part of the circuit. The inaccurate part is the most critical section in the proposed ETA as it determines the accuracy, speed performance, and power consumption of the adder. The inaccurate part consists of two blocks: the carry free addition block and the control block. The carry-free addition block is made up of 20 modified XOR gates, and each of which is used to generate a sum bit.

The block diagram of the carry-free addition block and the schematic implementation of the modified XOR gate are presented in Fig. 3. In the modified XOR gate, three extra transistors, M1, M2, and M3, are added to a conventional XOR gate. CTL is the control signal coming from the control block of Fig. 4 and is used to set the operational mode of the circuit. When CTL=0, M1 and M2 aret urned on, while M3 is turned off, leaving the circuit to operate in the normal XOR mode. When CTL=1 , M1 and M2 are both turned off,while M3 is turned on, connecting the output node to VDD, and hence setting the sum output to "1."

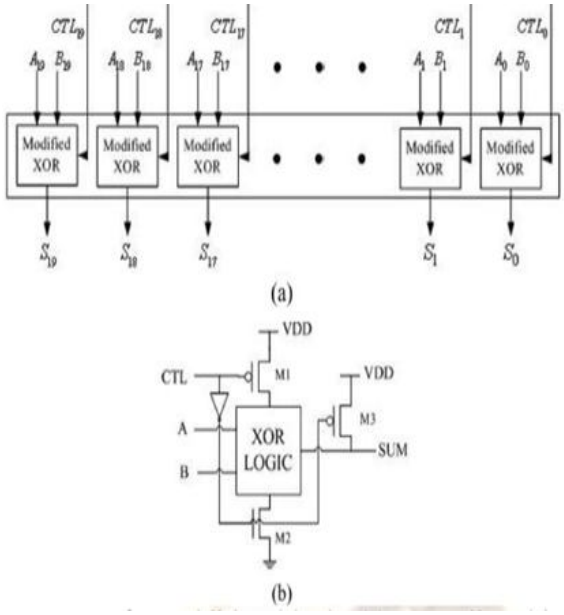


Fig.3.Carry-free addition block. (a) Overall architecture and (b) schematic diagram of a modified XOR gate.[1]

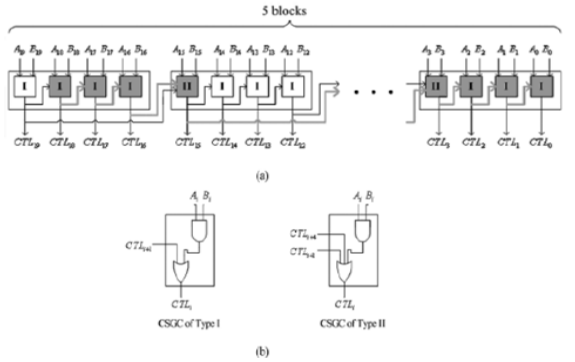


Fig.4. Control block. (a) Overall architecture and (b) schematic implementations of CSGC

The function of the control block is to detect the first bit position when both input bits are "1," and to set the control signal on this position as well as those on its right to high. It is made up of 20 control signal generating cells (CSGCs) and each cell generates a control signal for the modified XOR gate at the corresponding bit position in the carry-free addition block. Instead of a long chain of 20 cascaded GSGCs, the control block is arranged into five equal-sized groups, with additional connections between every two neighboring groups. Two types of CSGC, labeled as type I and II in Fig. 4(a), are designed, and the schematic implementations of these two types of CSGC are provided in Fig. 6(b). The control signal generated by the leftmost cell of each group is connected to the input of the leftmost cell in next group. The extra connections allow the propagated high control signal to "jump" from one group to another instead of passing through all the 20 cells. Hence, the worst case propagation path [shaded in gray in Fig. 4(a)] consists of only ten cells.

IV. SIMULATION

4.1 INACCURATE PART OF ETA

4.1.1 SCHEMATIC DIAGRAM

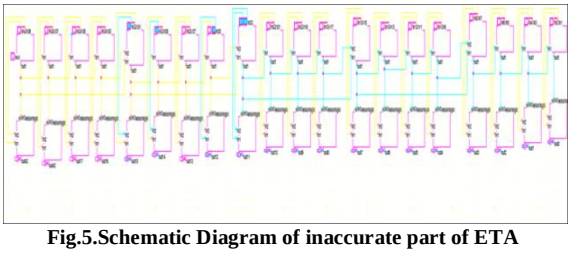


Fig.5.Schematic Diagram of inaccurate part of ETA

4.1.2 OUTPUT SHOWING POWER

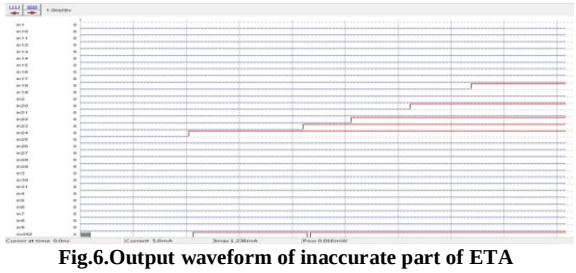


Fig.6.Output waveform of inaccurate part of ETA

4.1.3 OUTPUT SHOWING DELAY

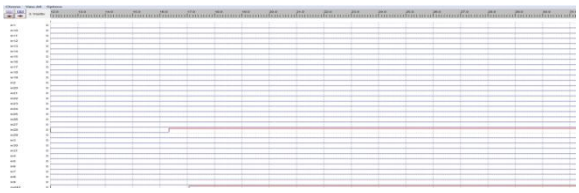


Fig.7.Output waveform showing delay

4.2 ACCURATE PART OF ETA: RCA-12 BIT

4.2.1 OUTPUT WAVEFORM

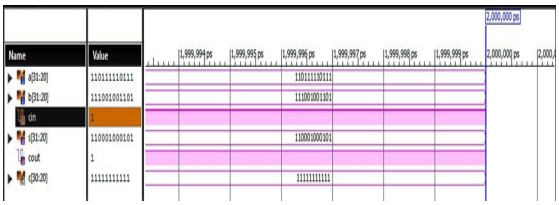


Fig.8.Output waveform of RCA-12 bit

4.2.2 OUTPUT SHOWING POWER

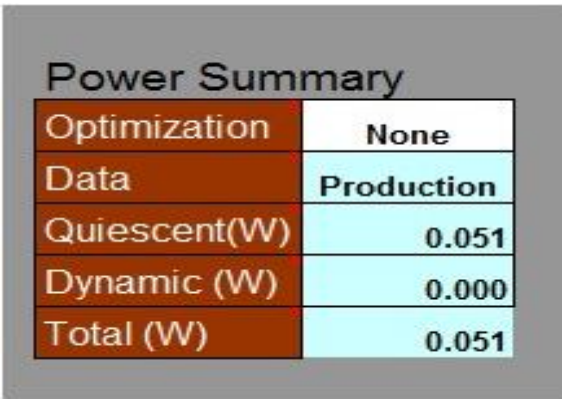


Fig.9.Output showing power of RCA-12 bit

4.2.3 OUTPUT SHOWING DELAY

```
                                    Gate     Net
Cell:in->out        fanout   Delay   Delay  Logical Name (Net Name)
----------------------------------------  ------------
 IBUF:I->O               2   1.218   0.622  b_20_IBUF (b_20_IBUF)
 LUT3:I0->O              2   0.704   0.526  A0/cout1 (c<20>)
 LUT3:I1->O              2   0.704   0.526  A1/cout1 (c<21>)
 LUT3:I1->O              2   0.704   0.526  A2/cout1 (c<22>)
 LUT3:I1->O              2   0.704   0.526  A3/cout1 (c<23>)
 LUT3:I1->O              2   0.704   0.526  A4/cout1 (c<24>)
 LUT3:I1->O              2   0.704   0.526  A5/cout1 (c<25>)
 LUT3:I1->O              2   0.704   0.526  A6/cout1 (c<26>)
 LUT3:I1->O              2   0.704   0.526  A7/cout1 (c<27>)
 LUT3:I1->O              2   0.704   0.526  A8/cout1 (c<28>)
 LUT3:I1->O              2   0.704   0.526  A9/cout1 (c<29>)
 LUT3:I1->O              2   0.704   0.526  A10/cout1 (c<30>)
 LUT3:I1->O              1   0.704   0.420  A11/cout1 (cout_OBUF)
 OBUF:I->O                   3.272          cout_OBUF (cout)
----------------------------------------
Total                       19.766ns (12.938ns logic, 6.828ns route)
                                     (65.5% logic, 34.5% route)
```

Fig.10.Output showing delay of of RCA-12 bit

4.3 RCA 32

4.3.1 OUTPUT WAVEFORM

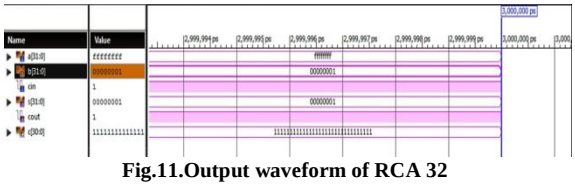


Fig.11.Output waveform of RCA 32

4.3.2 OUTPUT SHOWING POWER

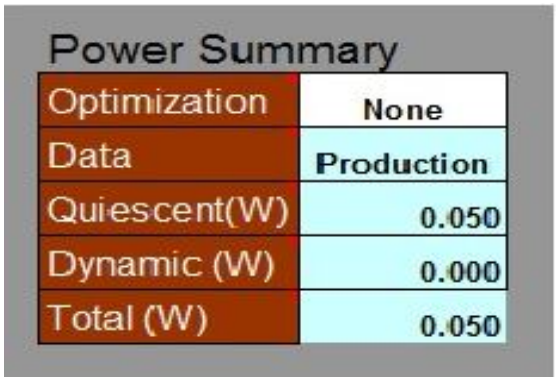


Fig.12.Output showing power of RCA 32

4.3.3 OUTPUT SHOWING DELAY

```
LUT3:I0->O              2    0.612    0.449  A0/cout1 (c<0>)
LUT3:I1->O              2    0.612    0.449  A1/cout1 (c<1>)
LUT3:I1->O              2    0.612    0.449  A2/cout1 (c<2>)
LUT3:I1->O              2    0.612    0.449  A3/cout1 (c<3>)
LUT3:I1->O              2    0.612    0.449  A4/cout1 (c<4>)
LUT3:I1->O              2    0.612    0.449  A5/cout1 (c<5>)
LUT3:I1->O              2    0.612    0.449  A6/cout1 (c<6>)
LUT3:I1->O              2    0.612    0.449  A7/cout1 (c<7>)
LUT3:I1->O              2    0.612    0.449  A8/cout1 (c<8>)
LUT3:I1->O              2    0.612    0.449  A9/cout1 (c<9>)
LUT3:I1->O              2    0.612    0.449  A10/cout1 (c<10>)
LUT3:I1->O              2    0.612    0.449  A11/cout1 (c<11>)
LUT3:I1->O              2    0.612    0.449  A12/cout1 (c<12>)
LUT3:I1->O              2    0.612    0.449  A13/cout1 (c<13>)
LUT3:I1->O              2    0.612    0.449  A14/cout1 (c<14>)
LUT3:I1->O              2    0.612    0.449  A15/cout1 (c<15>)
LUT3:I1->O              2    0.612    0.449  A16/cout1 (c<16>)
LUT3:I1->O              2    0.612    0.449  A17/cout1 (c<17>)
LUT3:I1->O              2    0.612    0.449  A18/cout1 (c<18>)
LUT3:I1->O              2    0.612    0.449  A19/cout1 (c<19>)
LUT3:I1->O              2    0.612    0.449  A20/cout1 (c<20>)
LUT3:I1->O              2    0.612    0.449  A21/cout1 (c<21>)
LUT3:I1->O              2    0.612    0.449  A22/cout1 (c<22>)
LUT3:I1->O              2    0.612    0.449  A23/cout1 (c<23>)
LUT3:I1->O              2    0.612    0.449  A24/cout1 (c<24>)
LUT3:I1->O              2    0.612    0.449  A25/cout1 (c<25>)
LUT3:I1->O              2    0.612    0.449  A26/cout1 (c<26>)
LUT3:I1->O              2    0.612    0.449  A27/cout1 (c<27>)
LUT3:I1->O              2    0.612    0.449  A28/cout1 (c<28>)
LUT3:I1->O              2    0.612    0.449  A29/cout1 (c<29>)
LUT3:I1->O              2    0.612    0.449  A30/cout1 (c<30>)
LUT3:I1->O              1    0.612    0.357  A31/cout1 (cout_OBUF)
OBUF:I->O                    3.169          cout_OBUF (cout)
----------------------------------------
Total                       38.665ns (23.859ns logic, 14.806ns route)
                                     (61.7% logic, 38.3% route)
```

Fig.13 Output showing delay of RCA 32.

4.4 CSK 32-BIT ADDER

4.4.1 OUTPUT WAVEFORM



Fig.14.Output waveform of CSK 32-BIT adder

4.4.2 OUTPUT SHOWING POWER

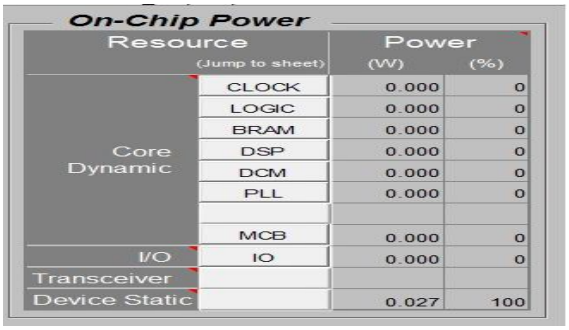


Fig.15.Output showing power of CSK 32-BIT adder

4.4.3 OUTPUT SHOWING DELAY

```
LUT3:I1->O              2    0.612    0.449  s_5_or00011 (s_5_or0001)
LUT3:I1->O              2    0.612    0.449  C_6_or00001 (C_6_or0000)
LUT3:I1->O              2    0.612    0.449  s_8_or00001 (s_8_or0000)
LUT3:I1->O              2    0.612    0.449  s_8_or00011 (s_8_or0001)
LUT3:I1->O              2    0.612    0.449  C_9_or00001 (C_9_or0000)
LUT3:I1->O              2    0.612    0.449  s_11_or00001 (s_11_or0000)
LUT3:I1->O              2    0.612    0.449  s_11_or00011 (s_11_or0001)
LUT3:I1->O              2    0.612    0.449  C_12_or00001 (C_12_or0000)
LUT3:I1->O              2    0.612    0.449  C_15_or00011 (C_15_or0001)
LUT3:I1->O              2    0.612    0.449  C_15_or00021 (C_15_or0002)
LUT3:I1->O              2    0.612    0.449  C_15_or00001 (C_15_or0000)
LUT3:I1->O              2    0.612    0.449  C_18_or00011 (C_18_or0001)
LUT3:I1->O              2    0.612    0.449  C_18_or00021 (C_18_or0002)
LUT3:I1->O              2    0.612    0.449  C_18_or00001 (C_18_or0000)
LUT3:I1->O              2    0.612    0.449  C_21_or00011 (C_21_or0001)
LUT3:I1->O              2    0.612    0.449  C_21_or00021 (C_21_or0002)
LUT3:I1->O              2    0.612    0.449  C_21_or00001 (C_21_or0000)
LUT3:I1->O              2    0.612    0.449  C_24_or00011 (C_24_or0001)
LUT3:I1->O              2    0.612    0.449  C_24_or00021 (C_24_or0002)
LUT3:I1->O              2    0.612    0.449  C_24_or00001 (C_24_or0000)
LUT3:I1->O              2    0.612    0.449  C_27_or00011 (C_27_or0001)
LUT3:I1->O              2    0.612    0.449  C_27_or00021 (C_27_or0002)
LUT3:I1->O              2    0.612    0.449  C_27_or00001 (C_27_or0000)
LUT3:I1->O              2    0.612    0.449  s_29_or00001 (s_29_or0000)
LUT3:I1->O              2    0.612    0.449  s_29_or00011 (s_29_or0001)
LUT3:I1->O              3    0.612    0.520  C_30_or00001 (C_30_or0000)
LUT3:I1->O              1    0.612    0.357  Cout1 (Cout_OBUF)
OBUF:I->O                    3.169          Cout_OBUF (Cout)
----------------------------------------
Total                       37.676ns (23.247ns logic, 14.429ns route)
                                     (61.7% logic, 38.3% route)
```

Fig.16.Output showing delay of CSK 32-BIT adder

4.5 CLA 32-BIT

4.5.1 OUTPUT WAVEFORM

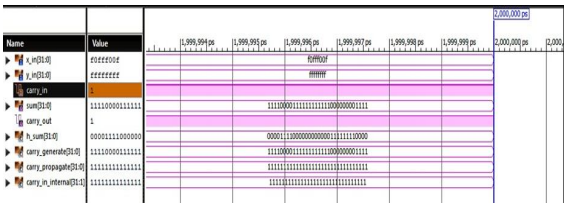


Fig.17.Output waveform of CLA 32-BIT adder

4.5.2 OUTPUT SHOWING POWER

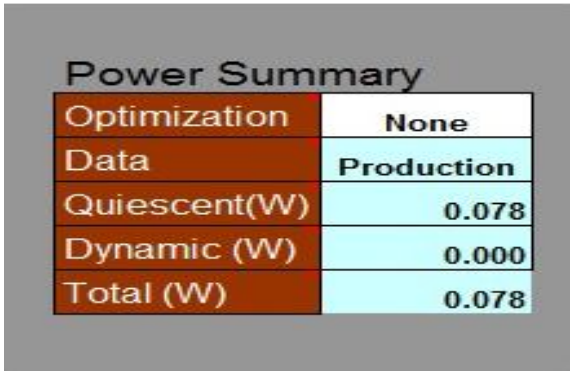


Fig.18.Output showing power of CLA 32-BIT adder

4.5.3 OUTPUT SHOWING DELAY

```
LUT3:I2->O              2    0.612    0.410  carry_in_internal_2_or00001 (carry_in_internal<2>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_3_or00001 (carry_in_internal<3>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_4_or00001 (carry_in_internal<4>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_5_or00001 (carry_in_internal<5>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_6_or00001 (carry_in_internal<6>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_7_or00001 (carry_in_internal<7>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_8_or00001 (carry_in_internal<8>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_9_or00001 (carry_in_internal<9>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_10_or00001 (carry_in_internal<10>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_11_or00001 (carry_in_internal<11>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_12_or00001 (carry_in_internal<12>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_13_or00001 (carry_in_internal<13>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_14_or00001 (carry_in_internal<14>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_15_or00001 (carry_in_internal<15>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_16_or00001 (carry_in_internal<16>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_17_or00001 (carry_in_internal<17>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_18_or00001 (carry_in_internal<18>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_19_or00001 (carry_in_internal<19>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_20_or00001 (carry_in_internal<20>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_21_or00001 (carry_in_internal<21>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_22_or00001 (carry_in_internal<22>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_23_or00001 (carry_in_internal<23>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_24_or00001 (carry_in_internal<24>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_25_or00001 (carry_in_internal<25>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_26_or00001 (carry_in_internal<26>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_27_or00001 (carry_in_internal<27>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_28_or00001 (carry_in_internal<28>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_29_or00001 (carry_in_internal<29>)
LUT3:I2->O              2    0.612    0.410  carry_in_internal_30_or00001 (carry_in_internal<30>)
LUT3:I2->O              2    0.612    0.532  carry_in_internal_31_or00001 (carry_in_internal<31>)
LUT3:I0->O              1    0.612    0.357  Mxor_sum<31>_Result1 (sum_31_OBUF)
OBUF:I->O                    3.169          sum_31_OBUF (sum<31>)
----------------------------------------
Total                       37.578ns (23.859ns logic, 13.719ns route)
                                     (63.5% logic, 36.5% route)
```

Fig.19.Output showing delayof CLA 32-BIT adder

4.6 CSLA 32-BIT

4.6.1 OUTPUT WAVEFORM

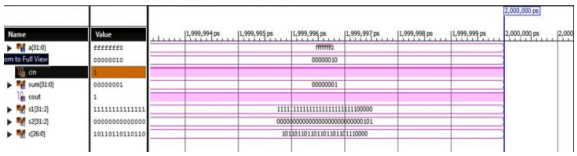


Fig.20.Output waveform of CSLA 32-BIT adder

4.6.2 OUTPUT SHOWING POWER

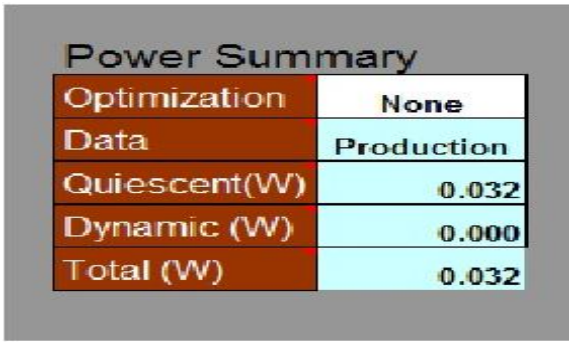


Fig.21.Output showing power of CSLA 32-BIT adder

4.6.3 OUTPUT SHOWING DELAY

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	2	1.106	0.532	b_0_IBUF (b_0_IBUF)
LUT3:I0->O	2	0.612	0.449	F1/A1/ap1/y11 (N5)
LUT3:I1->O	2	0.612	0.449	F1/A1/cout1 (c<0>)
LUT3:I1->O	3	0.612	0.481	M2/y11 (N8)
LUT4:I2->O	2	0.612	0.383	M4/y128 (M4/y128)
LUT4:I3->O	3	0.612	0.451	M7/y1 (c<6>)
MUXF5:S->O	1	0.641	0.387	M8/y183_SW1_f5 (N94)
LUT3:I2->O	2	0.612	0.410	M8/y183 (N16)
LUT3:I2->O	6	0.612	0.721	M12/y1 (c<9>)
LUT4:I0->O	1	0.612	0.000	M14/y11 (M14/y1)
MUXF5:I1->O	2	0.278	0.410	M14/y1_f5 (N2)
LUT3:I2->O	2	0.612	0.410	M13/y136 (N17)
LUT3:I2->O	2	0.612	0.449	M18/y1 (c<12>)
LUT3:I1->O	2	0.612	0.449	M19/y11 (N7)
LUT3:I1->O	2	0.612	0.449	M21/y1 (c<15>)
LUT3:I1->O	3	0.612	0.481	M22/y11 (N6)
LUT4:I2->O	2	0.612	0.383	M25/y128 (M25/y128)
LUT4:I3->O	5	0.612	0.568	M28/y1 (c<21>)
LUT4:I2->O	2	0.612	0.383	M29/y128 (M29/y128)
LUT4:I3->O	6	0.612	0.599	M33/y1 (c<24>)
LUT4:I2->O	1	0.612	0.000	M34/y11 (M34/y1)
MUXF5:I1->O	2	0.278	0.410	M34/y1_f5 (N20)
LUT3:I2->O	1	0.612	0.357	M39/y1 (cout_OBUF)
OBUF:I->O		3.169		cout_OBUF (cout)
Total		26.711ns	(17.100ns logic, 9.611ns route) (64.0% logic, 36.0% route)	

Fig.22.Output showing delay of CSLA 32-BIT adder

V. TABULATION RESULT

ADDER	POWER(W)	DELAY(ns)
RCA	.050	38.665
CSK	.027	37.676
CLA	.078	37.578
CSLA	.032	26.711
ETA	.0000016	1

VI. MODIFICATION IN XOR LOGIC

Modification done in xor logic by removing a single NMOS transistor.

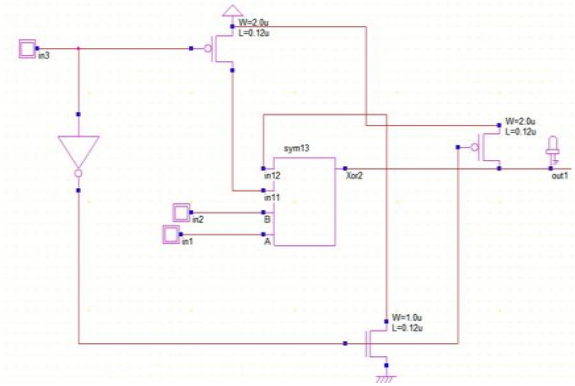


Fig.23.XOR logic before modification

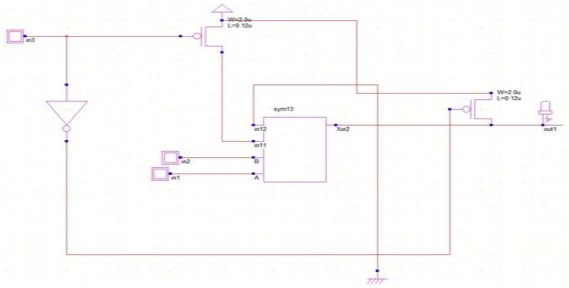


Fig.24.XOR logic after modification

PROPOSED ETA

101010011011,00111011001101100101+
101100110100,11100111010110100110

01011100111111111111111111111111 =>1560281087 (ETA value)

01011101000000100010100100001011 =>1560422667 (Actual value)

Accuracy=(1-(actual value-ETA value)/actual value)*100%=**99.99%**

MODIFIED ETA

101010011011,00111011001101100101+
101100110100,11100111010110100110

01011100111,11011100111011011011 =>1560137435 (ETA value)

01011101000000100010100100001011 =>1560422667 (Actual value)

Accuracy=(1-(actual value-ETA value)/Actual value)*100%=**99.98%**

6.1 INACCURATE PART OF ETA

6.1.1 SCHEMATIC DIAGRAM

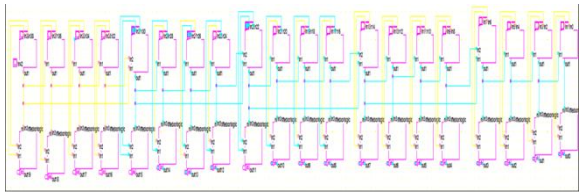


Fig.25.Schematic Diagram of inaccurate part of ETA

6.1.2 OUTPUT SHOWING POWER

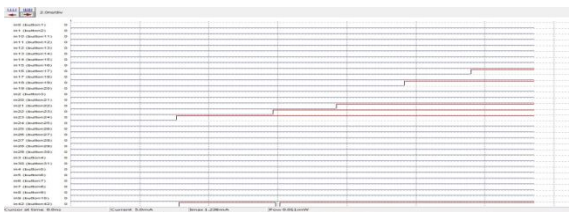


Fig.26.Output waveform of inaccurate part of ETA

6.1.3 OUTPUT SHOWING DELAY



Fig.27.Output waveform showing delay

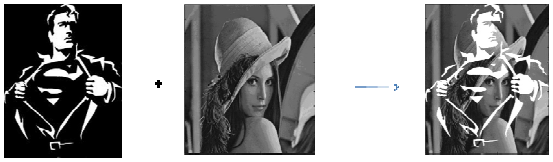
6.2.COMPARISON

ETA	POWER (mw)	DELAY (ns)	TRANSISTOR COUNT
ACTUAL	.000016	1	1006
MODIFIED	.000011	.9	986

VII. APPLICATION OF ERROR TOLERANT ADDER IN DIGITAL SIGNAL PROCESSING

Digital signal processing (DSP) system that process signals relating to human senses such as hearing, sight, smell, and touch, e.g., the image processing and speech processing systems.

❑ ADDING TWO IMAGES

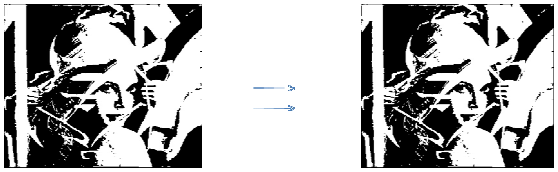


❑ NORMAL ADDITION



❑ ETA ADDITION





Thus by comparing both images we can't see any difference.

COMPARISON OF DIFFERENT LOGICS IN ETA

❑CONVENTIONAL ADDER USING PASS TRANSISTOR LOGIC

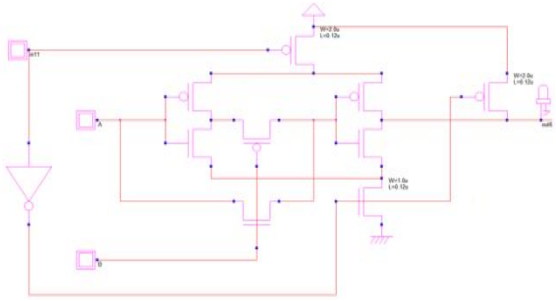


Fig.28.conventional adder using pass transistor logic

○ ETA

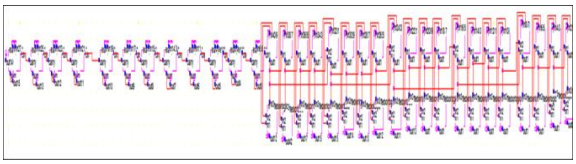


Fig.29.ETA

○ POWER

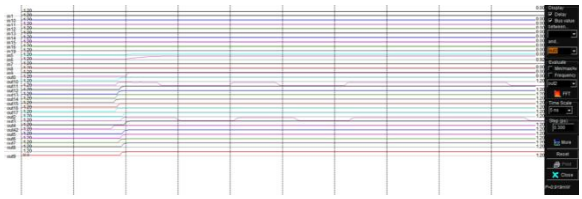


Fig.30.Power

○ AREA

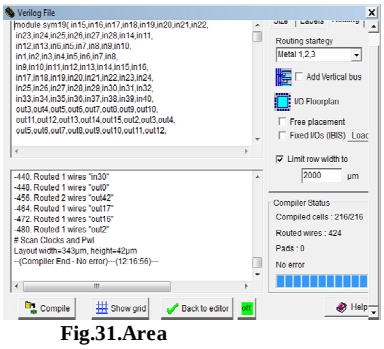


Fig.31.Area

❑ 3T XOR GATE BASED ADDER

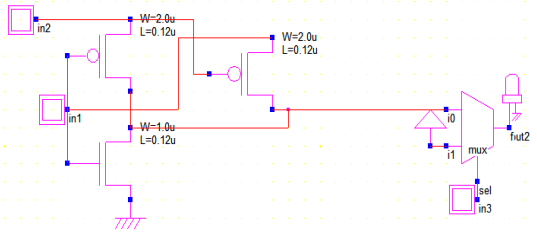


Fig.32.3t xor gate based adder

o ETA

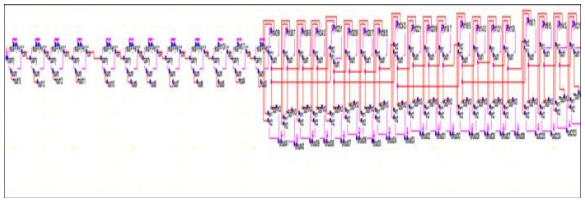


Fig.33.ETA

o POWER

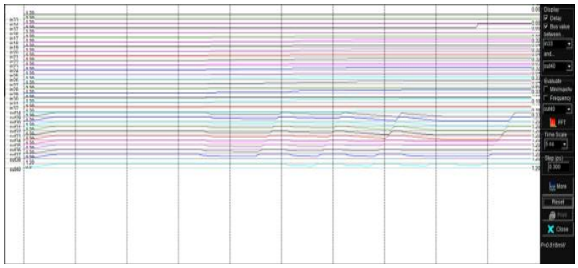


Fig.34.ETA

o AREA

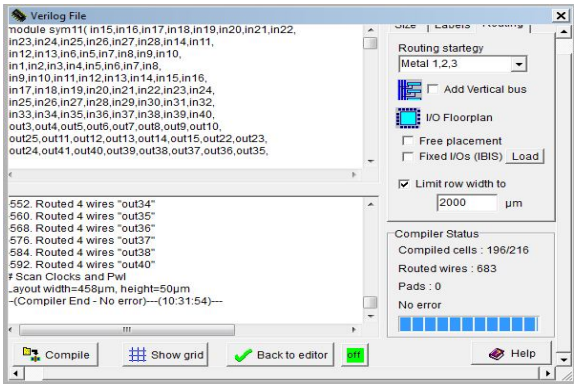


Fig.35.Power

❑ OR GATE BASED ADDER

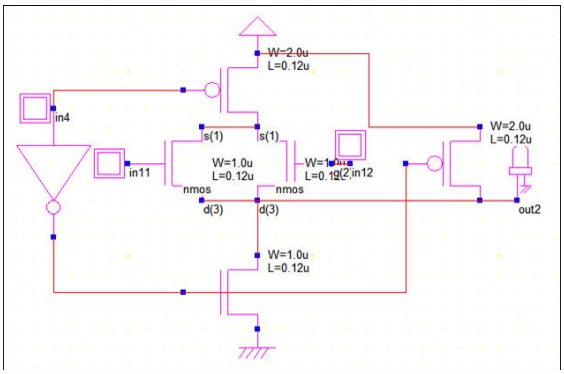


Fig.36.or gate based adder

o ETA

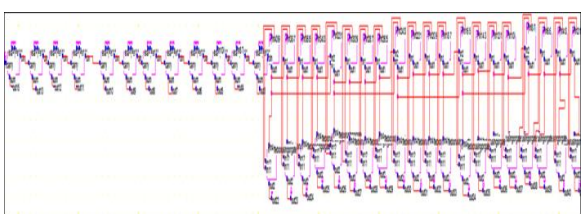


Fig.37.ETA

o POWER

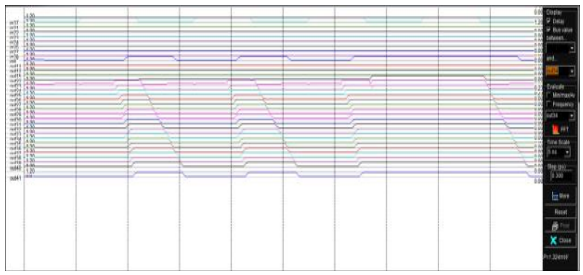


Fig.38.POWER

o AREA

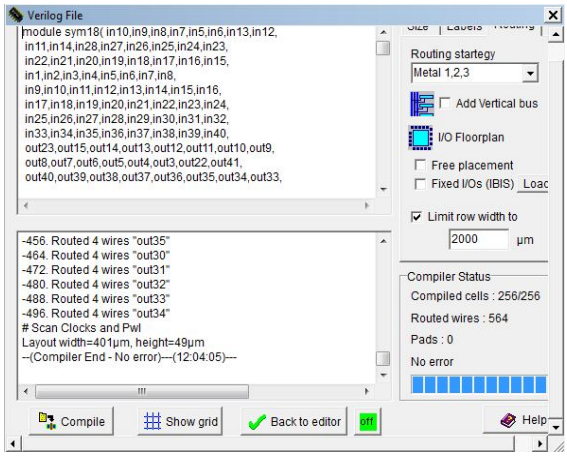


Fig.39.Area

❑ 2 TRANSISTOR BASED LOGIC

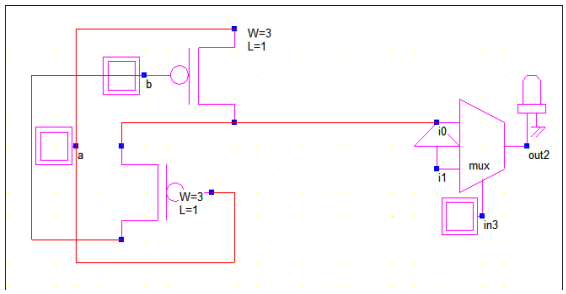


Fig.40.2T transistor based logic

o ETA

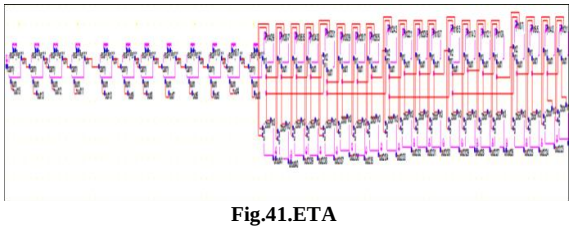


Fig.41.ETA

o POWER

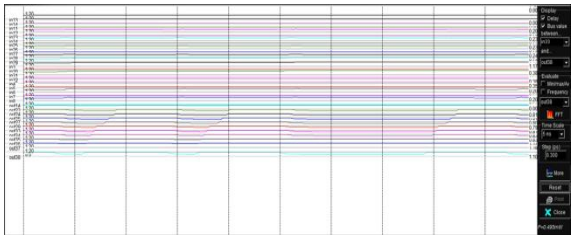


Fig.42.POWER

o AREA

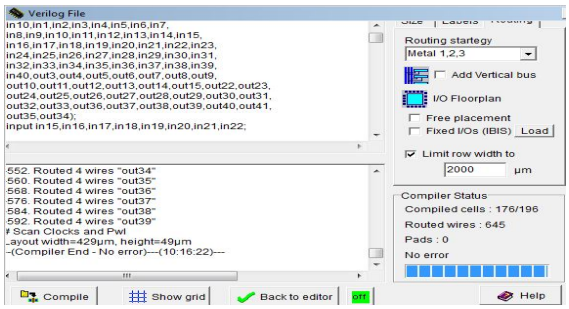


Fig.43.AREA

COMPARISON TABLE

ETA	POWER(mw)	AREA(NO: OF CELLS)
PASS TRANSISTOR LOGIC	.919	216
3T XOR BASED LOGIC	.818	216
OR BASED LOGIC	1.324	256
2T BASED LOGIC	.495	196

CONCLUSION

In this paper, the concept of error tolerance is introduced in VLSI design. A novel type of adder, the error-tolerant adder, which trades certain amount of accuracy for significant power saving and performance improvement, is proposed. Extensive comparisons with conventional digital adders showed that the proposed ETA outperformed the conventional adders in both power consumption and speed performance. The potential applications of the ETA fall mainly in areas where there is no strict requirement on accuracy .One example of such applications is in the DSP application for portable devices such as cell phones and laptops. Thus by using XILINX ISE 14.7, DSCH, MICROWIND, MATLAB etc, the simulation is done. And the xpower estimator and analyzer also used for finding the power.

REFERENCES

[1] Ning Zhu, Wang Ling Goh, Weija Zhang, Kiat Seng Yeo, and Zhi Hui Kong" Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing", IEEE Transaction On Very Large Scale Integration(VLSI)Systems, Vol.18,No.8,August 2010.

★★★