**International Journal of Advanced Research in Engineering and Technology (IJARET)**
Volume 10, Issue 1, January – February 2019, pp. 88–94, Article ID: IJARET_10_01_009
Available online at http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=10&IType=1
ISSN Print: 0976-6480 and ISSN Online: 0976-6499


# DESIGN OF LOW POWER HIGH SPEED ERROR TOLERANT ADDERS USING FPGA

**Libya Thomas**
Department of ECE, MBCET, Trivandrum, India

**ABSTRACT**

*The addition of two binary numbers is the most fundamental and widely used arithmetic operation. This operation is used in microprocessors, digital signal processors, data processing application specific integrated circuits and many more. ETA (Error Tolerant Adder) is an efficient adder which speeds up binary addition. There is a huge improvement in the power and speed when we use an ETA. For increasing the speed and decreasing the power dissipation, we use the logic that in an adder circuit the delay appears mainly because of the carry propagation and also there is a lot of power dissipation. Design of ETA is done using backend tool under real time simulation conditions and then compares the performance of the ETA in terms of accuracy, delay and power consumption with that of conventional adders. The proposed architecture is then implemented using FPGA.*

**Key words:** Adder, ETA, FPGA, Xilinx, Carry free addition.

**Cite this Article:** Libya Thomas, Design of Low Power High Speed Error Tolerant Adders Using FPGA. *International Journal of Advanced Research in Engineering and Technology,* 10(1), 2019, pp 88–94.
http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=10&IType=1

## 1. INTRODUCTION

In conventional digital VLSI design, most digital data are processed and transmitted in a noisy channel. During this process, errors may occur anywhere in the processing chain. Error-tolerant concept cannot be adopted for all systems – In digital control systems, the correctness of the output signal is extremely important, and this denies the use of the error tolerant circuit. 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**.** A robust and efficient error-tolerant adder (ETA) is proposed in this project and it is compared with its conventional counterparts, with respect to power consumption and speed. 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 – But all of them suffer a trade-off between speed and power. ETA can attain great improvement in both the power consumption and speed performance.

## 2. BASIC OPERAION OF ETA

In the basic structure of ETA, the operands are divided into two parts: accurate and inaccurate one. The length of each part can be equal or unequal depending on the requirement of accuracy and speed. We are considering 12 bits inaccurate part and remaining 20 bits in inaccurate part. The addition process starts from the joining point of two parts and goes towards the two opposite direction simultaneously. Basic block diagram elaborating the concept of ETA addition arithmetic is shown in fig 1. The accurate part can be designed by using any one of the available traditional adders like ripple carry adder, carry look ahead adder and etc. The inaccurate part is designed such as to eliminate the propagation of carry from one bit to next one. This part consists of two blocks: control block and carry free addition block. The control block checks all the bits fed to the inaccurate part and monitors when both incoming bits go high. The control block output at that position and that to the right of that position are then made high. A number of control logic generating cells (CL) are connected to form this control block.

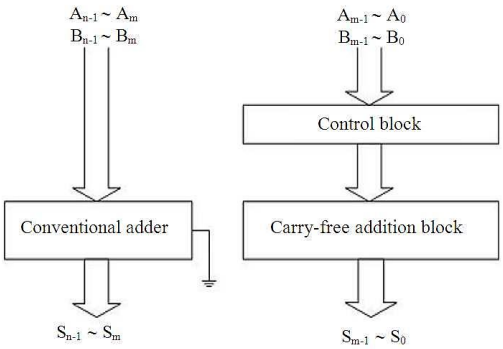


**Figure 1** Basic Block Diagram of ETA

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 is presented in fig.2.

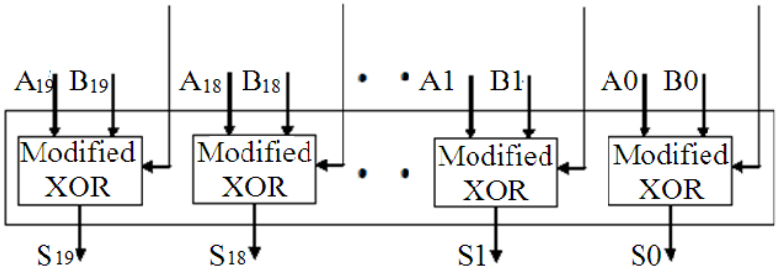


**Figure 2** (a) Overall architecture.

## 3. ARITHMETIC ADDITION

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. 3, the two 16-bit input operands, A= “1101011011010110” (54998) and B= “0111011000010101” (30229), are divided equally into 8 bits each for the accurate and inaccurate parts. 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:

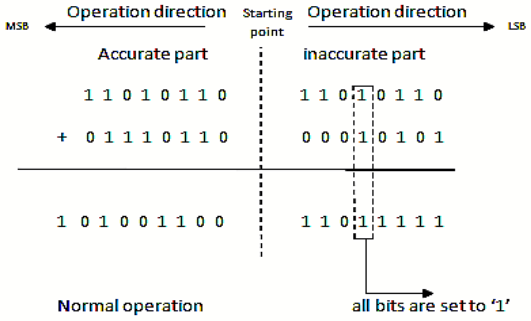


**Figure 3** Proposed addition arithmetic

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 "10100110011011111" (85215). The example given in Fig. 1 should actually yield "10100110011101011" (85227) if normal arithmetic has been applied. The overall error generated can be computed as OE=85227-85215=12.The accuracy of the adder with respect to these two input operands is obtained by eliminating the carry propagation path in the inaccurate part and performing the addition in two separate parts simultaneously, the overall delay time is greatly reduced, so is the power consumption.

# 4. CONVENTIONAL ADDERS

## 4.1. Ripple Carry Adder

The n-bit adder built from n one-bit full adders is known as a ripple carry adder, because of the way the carry is computed. Each full adder inputs a Cin, which is the Cout of the previous adder. This kind of adder is a ripple carry adder, since each carry bit "ripples" to the next full adder. Block diagram of Ripple Carry Adder is as in Fig.4. The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder.

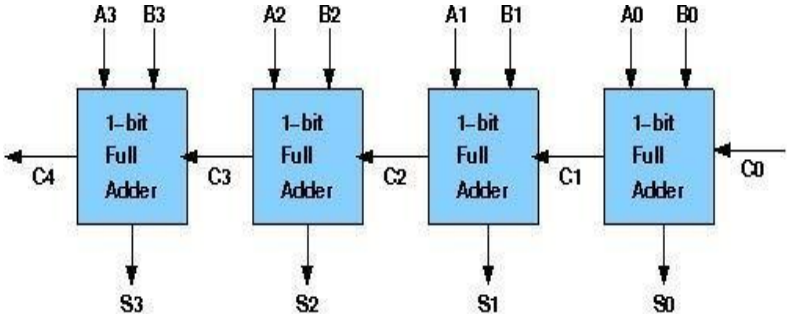


**Figure 4** Basic Block Diagram of Ripple carry adder

## 4.2. Carry Look Ahead Adder

Carry look ahead logic uses the concepts of generating and propagating carries. The addition of two 1-digit inputs A and B is said to generate if the addition will always carry, regardless of whether there is an input carry. In the case of binary addition, A+B generates if and only if both A and B are 1. The addition of two 1-digit inputs A and B is said to propagate if the addition will carry whenever there is an input carry. The propagate and generate are defined with respect to a single digit of addition and do not depend on any other digits in the sum. In the case of binary addition, A+B propagates if and only if at least one of A or B is 1. Sometimes a slightly different definition of propagate is used. By this definition, A+B is said to propagate if the addition will carry whenever there is an input carry, but will not carry if there is no input carry arithmetic, or is faster than xor and takes fewer transistors to implement. However, for a multiple-level carry look ahead adder, it is simpler to use. Block Diagram of 4bit carry-look- ahead adder is as in Fig. 5.

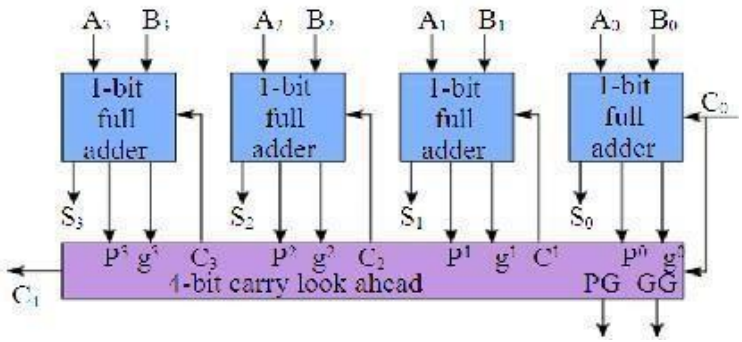


**Figure 5** Basic Block Diagram of carry look ahead adder

## 4.3. Carry Select Adder

The carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry-in being zero and the other assuming it will be one. After the two results are calculated, the correct sum, as well as the correct carry-out, is then selected with the multiplexer once the correct carry- in is known. Above is the basic building block of a carry-select adder, where the block size is 4. Two 4-bit ripple carry adders are multiplexed together, where the resulting carry and sum bits selected by the carry-in.

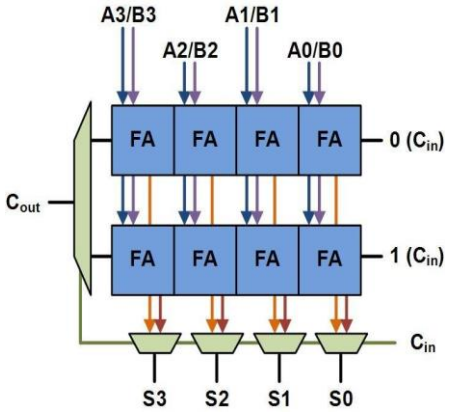


**Figure 6** Basic Block Diagram of Carry select adder

Since one ripple carry adder assumes a carry-in of 0, and the other assumes a carry-in of 1, selecting which adder had the correct assumption via the actual carry-in yields the desired result.

### 4.4. Carry Skip Adder

A carry-skip adder (also known as a carry-bypass adder) is an adder implementation that improves on the delay of a ripple-carry adder with little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder. The performance can be improved, i.e. all carries propagated more quickly by varying the block sizes. Accordingly the initial blocks of the adder are made smaller so as to quickly detect carry generates that must be propagated the furthers, the middle blocks are made larger because they are not the problem case, and then the most significant blocks are again made smaller so that the late arriving carry inputs can be processed quickly.

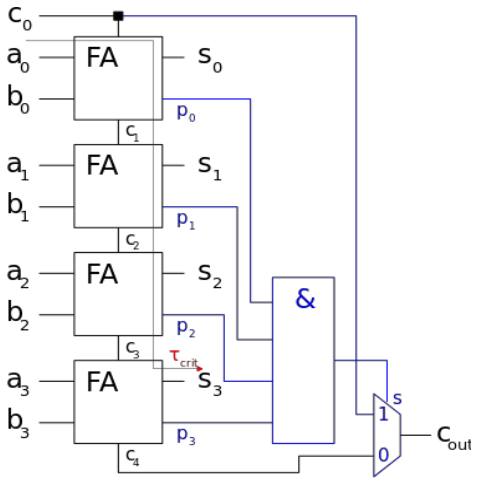


**Figure 7** Basic Block Diagram of Carry skip adder

***Block Diagram***

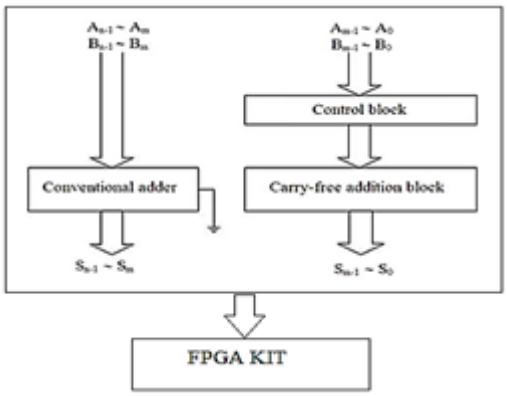


**Figure 8** General Block Diagram

## 5. SIMULATION RESULT

To demonstrate the advantages of the proposed ETA, we simulated the ETA along with four types of conventional adders, i.e., the RCA, CSLA by using XILINX 9.2 using VHDL code and simulated using modelsim. To evaluate the efficiency of the proposed architecture,

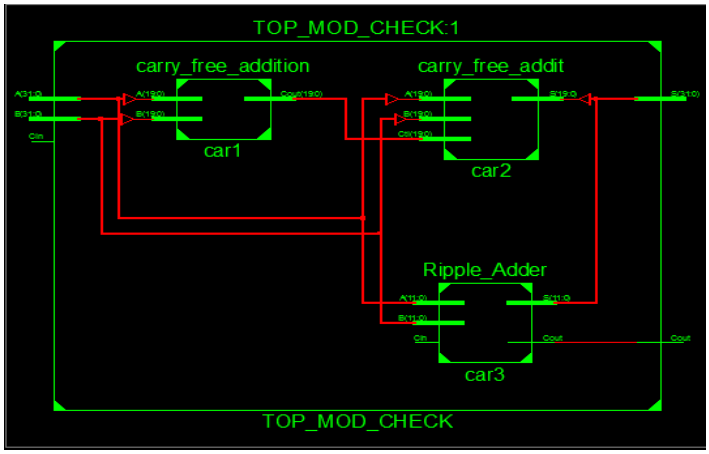


**Figure 9** RTL view of ETA

**Figure 10** Synthesis design summary of ETA

| Utilization Summary | | | | [-] |
|---|---|---|---|---|
| **Slice Device Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Registers | 0 | 18,224 | 0% | |
| Number of Slice LUTs | 33 | 9,112 | 1% | |
| Number used as logic | 33 | 9,112 | 1% | |
| Number used as ROM | 0 | | | |
| Number used as Memory | 0 | 2,176 | 0% | |
| Number of occupied Slices | 13 | 2,278 | 1% | |
| Number of LUT Flip Flop pairs used | 33 | | | |
| Number with an unused Flip Flop | 33 | 33 | 100% | |
| Number of bonded IOBs | 97 | 232 | 41% | |
| Average Fan-out of Non-Clock Nets | 1.65 | | | |

### Delay Comparison of Different Adders

| Type of adder | Delay |
|---|---|
| RCA | 20.800ns |
| CLA | 20.800ns |
| CSL | 21.694ns |
| CSK | 22.438ns |
| ETA | 17.477ns |

## 6. APPLICATION OF ERROR TOLERANT ADDER

In image processing and many other DSP applications, fast Fourier transformation (FFT or DFT) is a very important function. The computational process of FFT or DFT involves a large number of additions and multiplications. It is therefore a good platform for embedding our proposed ETA. To prove the feasibility of the ETA, we replaced all the common additions involved in a normal FFT or DFT algorithm with our proposed addition arithmetic. As we all know, a digital image is represented by a matrix in a DSP system, and each element of the matrix represents the color of one pixel of the image. To compare the quality of images Processed by both the conventional FFT or DFT and the inaccurate FFT that had incorporated our proposed ETA, we devised the following experiment. An image was first translated to a matrix form and sent through a standard system that made used of normal FFT and normal

reverse FFT. The matrix output of this system was then transformed back to an image and presented in Fig. 8. The matrix of the same image was also processed in a system that used the inaccurate FFT and inaccurate reverse FFT, where both FFTs had incorporated the 32-bit ETA described.

## 7. CONCLUSIONS

In this paper, the concept of error tolerance is introduced in VLSI design. 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 or where super low power consumption and high-speed performance are more important than accuracy. One example of such applications is in the DSP application for portable devices such as cell phones and laptops.

## REFERENCES

[1] N. Suganthi, "Design and implementation of field programmable gate array based error tolerant adder for image processing application," 2015 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, 2015, pp. 1426-1430.

[2] A. Y. Gorodilov and E. Y. Danilova, Built-In Diagnosis of The FPGA Logic Element, International Journal of Mechanical Engineering and Technology, 9(7), 2018, pp. 1347–1357.

[3] M. Pareek and M. Singhal, "Low power high speed area efficient Error Tolerant Adder using gate diffusion input method," 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), Noida, 2016, pp. 205-209.

[4] Ruchira Pradeep Pawar and Dr. S D Shirbahadurkar. Congestion Control for Scalability in Bufferless on-Chip Networks with FPGA Implementation. International Journal of Electronics and Communication Engineering & Technology, 6(8), 2015, pp. 19-27

[5] Naga Raju BOYA, Sreelekha KANDE, Vijay Kumar JINDE, Swapna CHINTAKUNTA, Mahesh UNGARALA and Ramanjappa THOGATA, Design and Development of Fpga Based Temperature Measurement and Control System, International Journal of Electronics and Communication Engineering & Technology (IJECET), Volume 4, Issue 4, July-August, 2013, pp. 86-95

[6] Santosh Pandi P and S L Gangadharaiah. FPGA Based Fixed Point LMS Adaptive Filters, International Journal of Electronics and Communication Engineering & Technology, 6(10), 2015, pp. 30-42

[7] Addanki Purna Ramesh, Dr. A.V. N. Tilak and Dr.A.M.Prasad, Fpga Based Implementation of Double Precision Floating Point Arithmetic Operations Using Verilog, International Journal of Computer Engineering & Technology (IJCET) Volume 3, Issue 2, July- September (2012), pp. 92-107

[8] Reema Karmokar, Shubham Mungekar and Trupti Vaity, FPGA Based Space Vector Pulse Width Modulation Technique Implementation for Three Phase Inverter, International Journal of Electronics and Communication Engineering and Technology, 8(2), 2017, pp. 36–45

[9] Pallavi Saxena, Urvashi Purohit and Priyanka Joshi, Analysis of Low Power, Area- Efficient and High Speed Fast Adder, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 9, September 2013. Pp: 3705-3710.

[10] Harikumar Rajaguru and Sunilkumar Prabhakar, FPGA Implementation of A Wavelet Neural Network with Particle Swarm Optimization Learning For Epileptic Seizure Detection, International Journal of Mechanical Engineering and Technology, 9(6), 2018, pp. 1141–1154.