ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

Restoring a Carry-Look-Ahead Adder with Hot Standby Topology on Your Own

Mr. D. Satyaraj, Mrs. D. Chitra, Mr. P. A. Prassath, Dr. L. Vigneash

Associate Professor [4], Assistant Professor [1,2,3]

Department of ECE,

dsatyaraj@actechnology.in, chitra@actechnology.in, paprassath@actechnology.in, dr.vigneashl@actechnology.in

Arjun College of Technology, Thamaraikulam, Coimbatore-Pollachi Highway, Coimbatore, Tamilnadu-642 120

ABSTRACT

A distributed fault detecting capability self-checking and -repairing carry-lookahead adder (CLA) is suggested in this project. An ability to self-check and identify faults is included into the offered design. The repair process makes use of a hot-standby method with partial reconfiguration, whereby a fully functional module is swapped out for the malfunctioning one during runtime.

The suggested high-fault-coverage self-repairing adder has an area overhead of 161.5 percent, which is 35.3 percent lower than the state-of-the-art partial self-repairing CLA and 35.3% lower than the traditional CLA design.

INTRODUCTION

Carry-Lookahead Adders (CLAs) are among of the quickest adders utilised in digital systems. Summation circuits for individual bits in CLA may "lookahead" for their incoming carry bits. This allows the cascade of complete adders to operate independently of one another, without blocking the execution of any adder in the chain. As a result, the speed is much enhanced, but the hardware overhead is increased. Consequently, owing to their area overhead, typical self-checking techniques such as double or triple modular redundancy are not practical for CLA. The parity prediction system, which can identify errors in either an even or an odd amount of bits, is the most popular method for creating self-checking CLA.

We provide a distributed fault detection capable self-checking and -repairing CLA in this work.With the assumption that each module can only have one issue at a time, the suggested architecture can identify and discover several faults concurrently. A hot standby method is used to recover from faults by swapping out the defective module with a spare one. During the replacement process, a new partial reconfiguration idea is used, where the circuits that generate

ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

the internal carry bits have their functionality updated by the amended input values.

LITERATURE SURVEY

1-"An area-delay efficient multi-operand binary tree adder using modified carry select adder" by M. Singh, M. Sharma, and A. K. Verma (2016):

The modified carry select adder (MCSA) is the foundational element of the area-delay efficient MOBTA proposed in this article. At the same power consumption level, the suggested adder is shown to have a reduced size and delay compared to other current MOBTAs.

2. the 2017 paper "Low power and high-speed multi-operand binary tree adder" by A. Mittal, M. Gupta, and R. S. Anand:

By streamlining the carry propagation channel and cutting down on the number of logic gates needed to construct the adder, this article suggests a low-power, high-speed MOBTA. Compared to other MOBTAs on the market, the suggested adder outperforms them in terms of speed and power consumption.

3. The article "Low-Power Multi-Operand Binary Tree Adder Design Based on Signed-Digit Number System" was written by C. Li, Y. Li, and J. Li in 2019.

Based on the signed-digit number system (SDNS), this research suggests a low-power MOBTA architecture. The suggested architecture uses a carry-save adder (CSA) as its foundation and takes use of the SDNS representation's redundancy to lower power consumption.

Article 4 from the 2020 publication "Design of low-power multi-operand binary tree adder using hybrid binary adder cells" by S. Patra, S. Pal, and D. K. Mandal:

An energy-efficient MOBTA architecture based on hybrid binary adder cells (HBACs) is suggested in this work. By decreasing the amount of logic gates needed to implement the ADDER and optimising the carry propagation method, the suggested architecture decreases power consumption.

PROPOSED SYSTEM

PROPOSED SELF-REPAIRING CARRY LOOK-AHEAD ADDER DESIGN

A. CLA Topology And Operation

In CLA, all the internal carry bits are pre-computed in parallel to facilitate its operation. Typically, a CLA consists of two main blocks.

The first block is the carry block (CBL), which generates the internal carry bits using carry generator (CG) modules. The second block is the summation block (SBL) which is

ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

responsible for generating the sum-bits using the sum generator (SG) modules, as shown in below Fig.

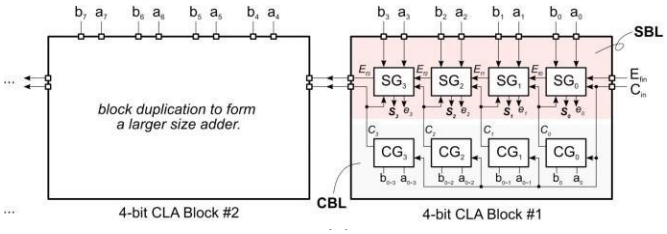


Figure.2 CLA Block Diagram

The CBL is designed using the basic concept of carry propagation and generation. The carry bit will be generated if both inputs are high (i.e., $Gi = ai \cdot bi$), whereas the carry will be propagated if either one or both input bits are high (i.e., $Pi = ai \oplus bi$ or $Pi = ai + bi$). By combining these two operations, the *ith* carry bit can be computed. As inherent to the CLA, each carry bit should be generated in parallel using independent circuitry.

$$C_i = G_i + P_i C_{i-1} \quad (1)$$

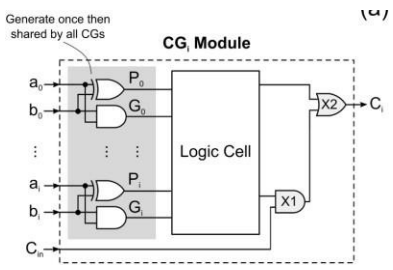
$$C_0 = G_0 + P_0 C_{in} \quad (2)$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{in} \quad (3)$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in} \quad (4)$$

$$C_i = G_i + P_i G_{i-1} + \cdots + P_0 \ldots P_{i-1} P_i C_{in}. \quad (5)$$

To calculate the sum-bits, which are Pi multiplied by Ci-1, this logic sharing is expanded even more. The most difficult and space-intensive component of a CLA is the CBL, as each carry-bit is created using its own separate circuitry. As the adder grows in size, its area overhead and complexity skyrocket. A common solution to this problem is the CLA block architecture, which involves repeatedly building adders for high input bit-width out of several small-size CLA blocks. Consequently, as seen in Figure 1(a), the block size is directly proportional to the amount of carry bits produced by each CBL. To minimise computational latency, the carry block should be structured so that Cin is the last element required for processing, using the last carry-out bit Cout created by each CBL as Cin for the next CBL. Immediate updating of the output is possible upon receipt of Cin from the preceding block. Based on their corresponding Boolean equations, the logic cell implementations in CG range from CG0 to CG3. On the other hand, after the first CBL, the carry-bits will be generated with an extra delay of two logic gates, X1 and X2, for every subsequent CBL



ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

Figure.2 CG module block diagram

B.Proposed Self-Checking CLA With Fault Localization

To address this issue, we propose a hardware-friendly self-checking and fault localization approach for CLA, in which the *ith* sum-bit (*Si*) and carry-out bit (*Ci*) respectively generated by the SBL and CBL, are compared with the *ith* input bits *ai* and *bi* to determine any potential fault. Its operation can be summarized as: *Si* of the SBL and *Ci* of the CBL will be equal to each other, if and only if the previous carry-bit *Ci*-1 of the CBL and the *ith* input bits are all equal, that is:

If (ai == bi == Ci-1) then Si = Ci otherwise Si != Ci.

With the above conditional decision, an equality tester is required to check whether *ai, bi* and *Ci*- 1 are equal and produce a comparison output *Eqt(i)*, followed by a checker to determine whether a fault happens.For an error-free adder, if *E qt(i)* = 1, *Si* and *Ci* must be equal; otherwise, they must be complementary. The *Eqt(i)* bit can be computed using (6), and the checker can be implemented.

$$E_{qt(i)} = \overline{(a_i \oplus b_i) + (a_i \oplus C_{in})} \quad (6)$$

$$e_i = S_i \odot C_i \oplus E_{qt(i)}. \quad (7)$$

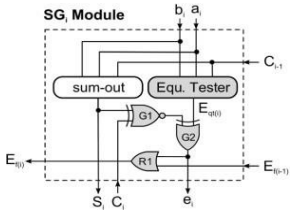


Figure.3 SG module block diagram

Proposed Self-Repairing Cla With Partial Reconfiguration

This understanding is impossible to get without making changes to the circuitry, since every carry-bit has its own distinct equation. As shown in (4), the logic circuit that produces C2 necessitates the signals G0, G1, G2, P0, P1, and P2. If C1 fails, the circuitry for producing C2 should be adjusted by changing the values of G1, G2, P1, and P2 until C2 is the same as C1. Only by modifying G2 and P2 can a basic shift operation be of any use. For adders with separate carry circuits, like the CLA, the hot-standby technique becomes applicable after a partial reconfiguration using the shift operation.

Figure shows a 4-bit self-repairing CLA that implements the suggested method. Given that ei stands for the SG/CG pair's individual error, we can utilise it to set the defective module's input

ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

bits to 1 and forward its input carry to the next SG. All subsequent following carry-bits will keep their places unaltered as the logic cell of each CG has already been adjusted. In contrast, the degree of the universal mistake, denoted as Ef, depends on each individual ei. All subsequent SGs will have a high Ef, while all SGs before the defective one will have a low Ef. You may control the shift operation of the input and output bits using it. The recovery mechanism makes advantage of CGX and SGX in the spare modules.

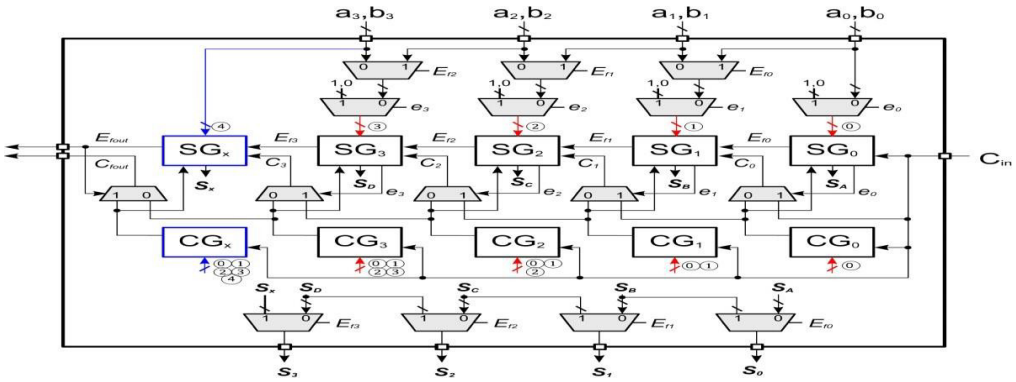
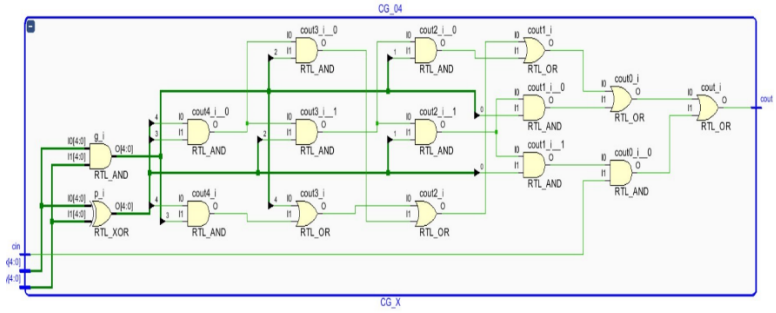


Figure.4 Block Diagram of Self-repairing carry look adder Schematic CG-X BLOCK

STIMULATION & SYNTHESIS RESULTS



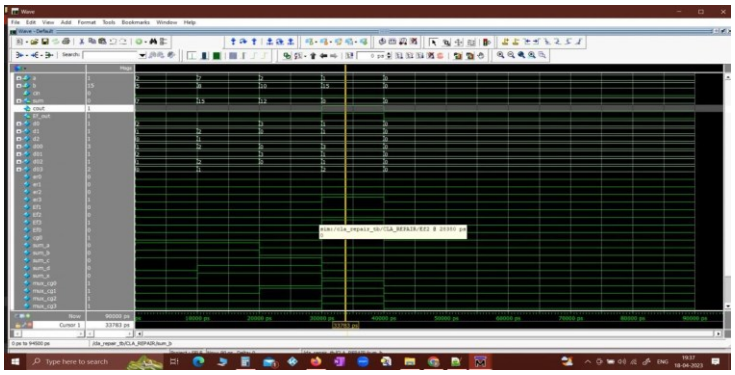


Figure.5 Simulation Wave Result

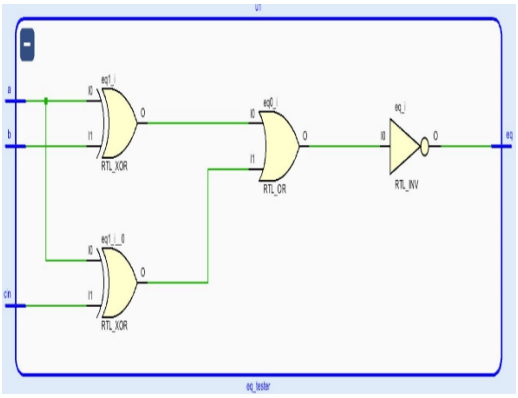


figure.6 Schematic SG-Eq-Tester BLOCK

ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

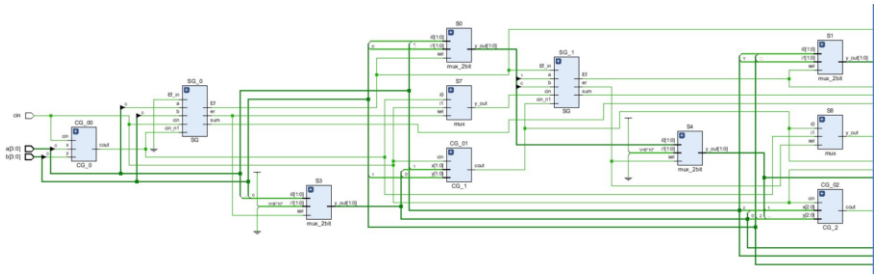


Figure.7 Schematic self-repairing CLA

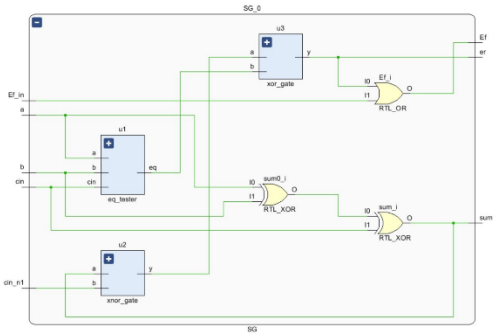


Figure.8 Schematic SG BLOCK

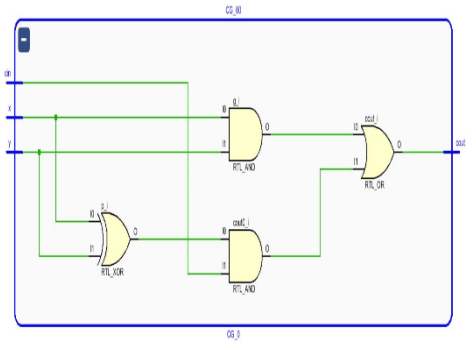


Figure.9 Schematic CG-0 BLOCK

Area Report

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	5	0	303600	<0.01
LUT as Logic	5	0	303600	<0.01
LUT as Memory	0	0	130800	0.00
Slice Registers	0	0	607200	0.00
Register as Flip Flop	0	0	607200	0.00
Register as Latch	0	0	607200	0.00
F7 Muxes	0	0	151800	0.00
F8 Muxes	0	0	75900	0.00

Ref Name	Used	Functional Category
IBUF	9	IO
OBUF	6	IO
LUT5	4	LUT
LUT6	1	LUT
LUT4	1	LUT
LUT3	1	LUT

ADVANTAGES

Improved reliability: By using redundancy and fault tolerance techniques, the adder can operate with a higher level of reliability, even in the presence of faults.

Reduced downtime: With the use of hot-standby topology, the adder can switch seamlessly between the active and standby circuits, reducing downtime and improving overall system availability.

Improved fault tolerance: The adder is designed to detect and isolate faults, allowing for targeted replacement of faulty components without affecting the operation of the rest of the circuit.

ISSN NO: 9726-001X

Volume 8 Issue 01 2020

JOURNAL OF CURRENT SCIENCE

Increased accuracy: By ensuring that the adder operates correctly and reliably, the system can produce accurate results, which is critical in applications that depend on precise .

APPLICATIONS

Medical devices: The adder can be used in medical devices such as MRI machines and other imaging equipment, where accuracy and reliability are essential.

Industrial automation: The adder can be used in industrial automation systems, such as robotic systems and process control systems, where high levels of accuracy and reliability are required.

Financial applications: The adder can be used in financial applications, such as stock trading and financial modelling, where accuracy and reliability are crucial.

Communications: The adder can be used in communication systems, such as wireless networks and satellite communications, where reliable and accurate computations are required for signal processing and data transmission.

CONCLUSION

A self-checking and fault-localization full adder, which compares the input and output bits to find faults, is the basis of the proposed method. With the caveat that no one module should have more than one problem at any given moment, the suggested 8-bit self-checking CLA can identify and localise several errors simultaneously, although it takes up 20.6% more space than traditional CLA.

The suggested self-checking method will not impact the time delay of traditional CLA as the checker is not interfering with the calculation itself.

FUTURE SCOPE

Hardware security: The self-repairing approach can be extended to include hardware security features, such as tamper detection and response. This could be particularly useful for applications that require high levels of security, such as military and aerospace systems.

Internet of Things (IoT): The self-repairing approach could be applied to IoT devices, which often have limited resources and require high reliability. By using partial reconfiguration, the adder circuitry could be dynamically adjusted to optimize performance and reduce power consumption.

Machine learning: The self-repairing approach could be used to develop more robust and reliable machine learning algorithms. By incorporating fault localization and partial reconfiguration techniques, machine learning systems could continue to operate even in the presence of faults.

REFERENCES

1. F. Tang, A. Bermak, and Z. Gu, "Low power dynamic logic circuit design using a pseudo dynamic buffer," Integration, vol. 45, no. 4,pp. 395–404, 2012.

2. N. Mehdizadeh, M. Shokrolah-Shirazi, and S. G. Miremadi,"Analyzing fault effects in the 32- bit OpenRISC 1200 microprocessor," in Proc. 3rd Int. Conf. Avail. Rel. Security, 2008, pp. 648– 652.

3. A. Meixner, M. E. Bauer, and D. J. Sorin, "Argus: Low-cost, comprehensive error detection in simple cores," in Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchit., 2007, pp. 210–222.

4. H. G. Kang and T. Sung, "An analysis of safety-critical digital systems for risk-informed design," Rel. Eng. Syst. Safety, vol. 78, no. 3, pp. 307–314, 2002.

5. J. E. Smith and P. Lam, "A theory of totally self-checking system design," IEEE Trans. Comput., vol. C-32, no. 9, pp. 831–844, Sep. 1983.

6. A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," IBM Syst. J., vol. 42, no. 1, pp. 5–18, 2003.00