Journal of Engineering Sciences
Vol 11, Issue 7,July/2020
ISSN NO:0377-9254

AN AREA EFFICIENT VLSI IMPLEMENTATION OF CARRY SELECT ADDER AND ALU TESTING

P. Surya Kumari[1], P.Anjaneya[2]

[1]*ECE Department, JNTUA (AITK), India, suryakumari223@gmail.com*
[2] *ECE Department , JNTUA (AITK), India, anjiaitsk@gmail.com*

Abstract---- Adders plays major role in multiplications and other advanced processors designs. Adders can be constructed for many numerical representations such as arithmetic and logical operation. The most adders operates on binary numbers. Among the different types of adders, carry select adder is a one of the fastest adder. Carry select adder was designed Full adders. This paper presented a Full adder with fault localization for the input bits. By using this scheme, instead of replacing the whole system we can now replace the particular faulty modules. This paper presented another easy testable method for ALU (Arithmetic and Logical Unit). This is an efficient & low cost fault tolerant reversible Arithmetic and logical unit .A new self-checking ALU was performed with different functional inputs. Through this ALU testing, we can easily identify the fault models.

Keywords—Concurrent error detection, carry select adder, design for testability.

1. INTRODUCTION

As the process shrink of integrated circuits advances and the integration density increases, reliability of integrated circuits in field becomes an issue. For critical systems such as server class computers and embedded systems, concurrent detection of errors is important.

Adders are the key elements of ALUs and MAC used in image and signal processing architectures as they lies in the critical path. A variety of applications require certain arithmetic operations such as incrementing the sum of two numbers by unity, finding the absolute difference between two numbers, or augmenting the sum of two numbers by a constant. One approach to perform these operations is to utilize dual adders or use multi-operand adders such as the CSAs, CSKA , CPA and CSLA. Also, multi operand addition forms a significant part of multiplication and certain DSP algorithms. Adder performance in a multi bit addition can be improved by reducing the delay due to carry propagation between different adder cells. This can be addressed by improving the structure of the basic adder block. CSLA is preferred in digital signal processors and application specific ICs designed to execute dedicated algorithms such as convolution, correlation and filtering to alleviate the problem of carry propagation delay in addition (Vijay kumar et al 2012). However, the hardware complexity of CSLA is high due to use of pair of RCAs to generate partial sum and carry corresponding to carry input 1 and 0. Then the final sum and carry are selected from the partial results by using multiplexers (Moris Mano & Michael Ciletti 2009).

The functionality of electronic equipments and gadgets has achieved a phenomenal growth over the last two decades while their physical sizes and weights have come down drastically. The major reason is due to the rapid advances in integration technologies, which enables fabrication of many millions of transistors in a single integrated circuit (IC) or chip. Every IC in the industry follows Moore's law. According to Moore's law, number of transistors (transistor density) in an IC doubles in every 1.5 years. With the recent advances in the technology, device shrinks to nanometer scale, but density and complexity of the ICs keep on increasing. This may result in many manufacturing faults and device failure. To accommodate more number of transistors, the device feature size is reduced. Reduction in the feature sizes results in increasing the manufacturing faults and fault detection becomes very difficult. A typical flow of a VLSI (Very Large Scale Integration) realization process is shown in below Figure 1 .VLSI testing is becoming more and more important and challenging to verify whether a device functions properly or not.

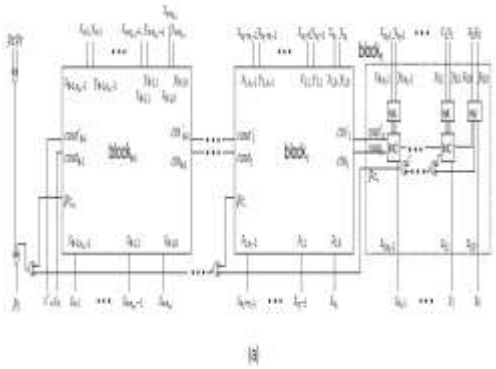
Figure 1 : VLSI realization process

Testing becomes a mandatory part of the VLSI industry. Every product must be tested before shipping to the customer. A test is defined as the crucial step to ensure the physical device which is manufactured from synthesized design has no manufacturing defects. The information collected during testing is not only used to remove the defective parts from the product, but it also helps to improve the design and manufacturing process. VLSI testing also improves the manufacturing yield level. Various stages of the VLSI testing are important to designers, product engineers, test engineers, managers, manufacturers, and end-users. A typical test procedure consists of a set of test vectors that are given as input stimuli to the CUT (circuit under test). The circuits that produce correct response for the set of test vectors is considered as good chips and those circuits that fail to produce correct response are called faulty chips. Testing is performed at various stages of a product/device including the development process, manufacturing process and sometimes even in system level operation.

2. Existing system

A concurrent error detectable adder with easy testability. In addition to operands X and Y , it receives their parities px and py, i.e., the XORed value of X and the XORed value of Y . In addition to sum output S, it produces the predicted parity pS of the sum output and two carry outputs cn and c′n. We restrict the block length nk to be even. Any erroneous output of the adder caused by a fault modeled as a single stuck-at fault can be detected by comparing the predicted parity pS with the parity of sum output S and comparing cn and c′n. One inconsistency in the two input pairs, i.e., a pair of X and pX or a pair of Y and pY, can also be detected. Each addition block except block0 has two carry inputs. Each addition block has two carry outputs, and produces parity pCk of carry signals which has the same value as ck,nk−1 _ · · · _ ck,1 _ cink in case of correct operation, where ck,nk−1, · · · , ck,1 are carry bits generated during the addition of Xk, Yk, and cink. Addition block block k (k _ 1) of the proposed adder is shown in Fig. 2(b).

The addition block receives two operands Xk and Yk, and produces sum result Sk. In addition to those inputs and output, it receives two carry inputs cink and cin′k, and produces parity of carries pCk and two carry outputs cout k and cout′k. coutk and cout′k are connected to cink+1 and cin′k+1, respectively Fig. 2(c) shows gate-level designs of HA', INC', and MUX'. Both of HA' and INC' have two carry outputs to obtain concurrent error detectability. One carry bit is used for addition, and the other is used for parity prediction HA' and INC' are designed so that effects of a single stuck-at fault in an INC' or its ascendant HA's never appear in both its sum output and one of its carry signals simultaneously because such a faulty operation generate san erroneous sum result and a predicted parity consistent with the erroneous sum result. In MUX' and INC', we use XOR gates to improve testability. Because XOR gates prevent masking of effects of a fault, effects of a fault can be observed easily. In each addition block, an XOR gate is placed for every bit position of INC0 k and INC1 k except the most significant position and the least significant position.



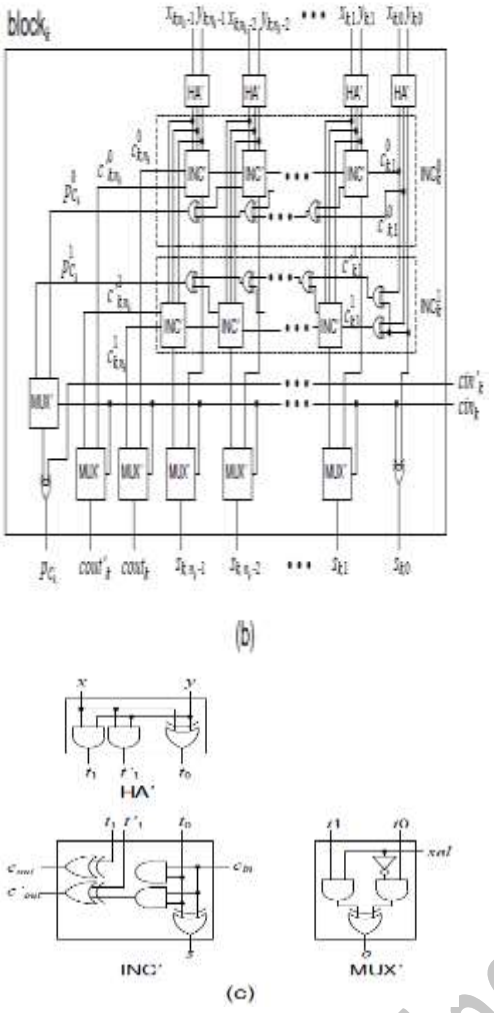


Fig. 2. Concurrent error detectable adder with easy testability (a), the design of block k (k _ 1) for the adder (b), and the gate-level designs of HA',INC' and MUX' (c).

For parity output pCk , two intermediate candidates are calculated in the two rows of INCs. One is p0 Ck which is calculated as $c'^{0}_{k,n_k-1} \oplus \cdots \oplus c'^{0}_{k,1}$,1 in INC0 k, and the other is p1 Ck which is calculated as $c'^{1}_{k,n_k-1} \oplus \cdots \oplus c'^{1}_{k,1}$,1 in INC1 k. The leftmost MUX' selects one of them according to the value of carry input cink, and the XOR gate at the output of the MUX' produces the parity of the carry bits including cin′k. With the obtained pCk , the adder calculates predicted parity pS of the sum as $(p_X \oplus p_Y) \oplus (p_{C_{b-1}} \oplus \cdots \oplus p_{C_0})$ with XOR gates because si is equal to $x_i \oplus y_i \oplus c_i$ in the correct operation.

2.1 Concurrent Error Detectability

Effects of a single stuck-at fault in an INC' or its ascendant HA' never appear in both its sum output and one of its two carry signals simultaneously as described in Section 3.1. Therefore, effect of a fault in an INC' or its ascendant HA' appears on either one of the two carry signals of the INC', the sum signal, or all the three signals (two carries and the sum). In any case, any erroneous result caused by a fault can be detected by comparing the predicted parity pS with the parity of S and comparing cn and c′n. In case one of the two carry signals of INC' is used for addition, and the other is used for parity prediction. When the carry signal for parity prediction is erroneous, only one bit of carry bits for the parity prediction is affected and the sum result is correct. Thus, erroneous results caused by the fault can be detected. When the carry signal for addition is erroneous, we let ck,j be the carry that is affected by the fault occurred at an INC'. Then, the error affecting ck,j induces also an error at the sum signal sk,j , and if it is not propagated in any subsequent positions, the error is detected because the carry signal affected by the fault is not used for parity prediction. On the other hand, if the erroneous value of ck,j is propagated at q subsequent carry signals ck,j+1, · · · , ck,j+q, they will also induce errors in the q sum signals sk,j+1, · · · , sk,j+q. Thus, the q+1 sum signals sk,j , sk,j+1, · · · , sk,j+q will be erroneous. Here, as the logic generating the carry signals c′k,j+1, · · · , c′k,j+q is identical to the logic generating the carry signals ck,j+1, · · · , ck,j+q, and they have both the same carry inputs, i.e. the carry signals ck,j , ck,j+1, · · · , ck,j+q−1, the q carry signals c′k,j+1, · · · , c′k,j+q will be also erroneous. Therefore, q + 1sum signals sk,j , sk,j+1, · · · , sk,j+q will be erroneous, and q carry signals c′k,j+1, · · · , c′k,j+q will be erroneous. Thus, the predicted parity $(p_X \oplus p_Y) \oplus (p_{C_{b-1}} \oplus \cdots \oplus p_{C_0})$ will be computed by means of q erroneous signals, while the parity of the sum signals will be computed by means of q+1 erroneous signals. Therefore, as the number of the erroneous signals used in these two parity computations differ by 1,

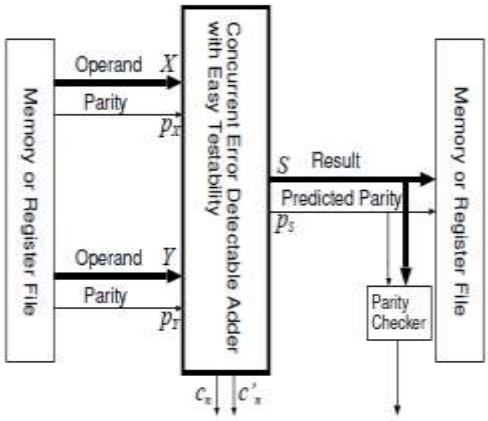


Fig 3. Example of a data path circuit with the proposed adder in a system Using parity-based error detection.

These two parities will be different and thus the error will be detected. A bit of the sum result is inverted and the parity of the sum result is different from the parity of the correct one. On the other hand, the predicted parity is calculated correctly because all the carry bits used for the prediction are correct. The effect of a fault is detected by comparing the predicted parity with the parity of the sum result. All of the carry bits and the sum bit from the INC' are incorrect. Because both of the two carry bits are incorrect, there is no inconsistency among the obtained sum bits and carry bits used for parity prediction in the upper positions than the position of the faulty INC'. In the lower positions of the INC', though carry bits for parity prediction are correct, the sum bit from the INC' is inverted. Therefore, parity of the sum result is different from the predicted parity. In the adder, each adder block has two carry inputs cink and cin′k. If there is an error on cink then sk,0 will be erroneous. Furthermore the error on cink can also induce errors on several other sum outputs (let us say at q sum outputs). Also, similarly to the arguments given in case (1), it will also create errors at q carry signals c′k,i. Thus, there will be q + 1 erroneous sum outputs and q erroneous carry signals c′k,i, and based to the same arguments as those given these errors will also be detected. The output of an XOR or a MUX' affects only one sum or carry bit or the predicted parity. Therefore, the effect of a fault in them is detected by comparing the predicted parity with the parity and comparing two carry outputs of the adder. Note that inconsistency of one of the input operands and its parity input causes an incorrect result of the predicted parity because the parity input is used for the parity prediction. Therefore, it is also possible to detect inconsistency of X and pX or inconsistency of Y and pY when there are no faults in the adder. The proposed adder is suitable for systems using parity based error detection as shown in Fig. 3. The parity-based error detection of arithmetic circuits was used in real designs. Parities fed from a memory or a register file are used as pX and pY for operands X and Y , and the predicted parity obtained by the proposed adder is used for the parity bit of the result. Any erroneous output of the adder is detected by observing the parity checker of the system and comparing two carry outputs cn and c'n.

3. Proposed method

3.1. Method 1

In this method, the testing operation was done for individual circuit. Basically, carry select adder was designed with Full adder. Carry select adder was shown in below figure 4 and the full adder circuit was tested as follows:

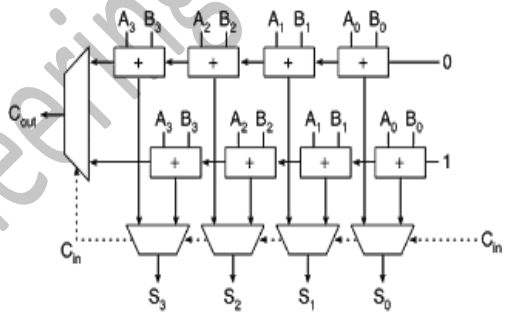


Fig 4: Carry select adder

From the truth table of full adder

- The Sum and Carry bit will be equal to each other when all the three inputs are equal.
- The Sum and Carry bit will be complemented when any of the three inputs is different.

The testing circuit for the full adder design was shown in below figure 5:

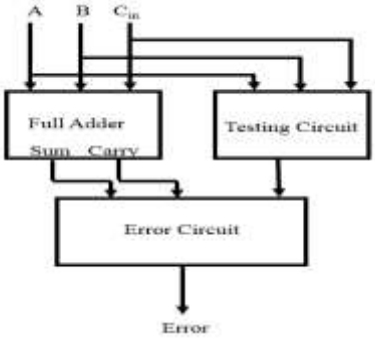


Fig 5: Testing circuit for the full adder design

The expressions for the testing circuit and error module circuits are given as:
Sum=A xor B xor C
Carry=A. B+C in. (A+B)

t he testing circuit can be expressed as

$$t=\overline{(\bar{A}\bar{B}\bar{C}+ABC)}$$

Error = Sum xnor carry xnor t

By using this testing circuitry, Carry select adder was designed. The final fault is computed by using two XNOR gates. The purpose of first XNOR gate (G1) is to check whether the Sum and Cout bit are equal or complemented. We need a second XNOR gate (G2) because of the previously mentioned observation that Sum and Cout will always be complement to each other except when all inputs are equal. Thus, the output of G1 will indicate the equality or difference of Sum and Cout and G2 will verify the output of G1 by comparing it with an equivalence tester and thus generate the final error indication. When the t is zero the output of G1 and G2 should be logic 1 and 0, respectively. While, if the t indicates logic 1 then both the XNOR gates should generate logic-0 (i. e. It =0 and Ef =0) and in any other case the fault will be indicated.

3.2. Method-2

The ALU can perform 8 different functions: NOP, ADD, SUB, SHIFT, AND, OR, PASS and MULTIPLICATION(arbitrary function). It has inputs A, B and Control. Control is the selection input to the 8:1 MUX, the MUX passes the selected function to perform on A and B. Two inputs and ALU out are 16 bits, while Control is 3-bit wise and Carry Out has only 1 bit. Ideas for Arbitrary Function: Multiplication of the lower 8 bits of inputs A and B. The ALU, or the arithmetic and logic unit is the section of the processor that is involved with executing operations of an arithmetic or logical nature.

The logic circuitry in this units is entirely combinational (i.e. consists of gates with no feedback and no flip-flops).The ALU is an extremely versatile and useful device since, it makes available, in single package, facility for performing many different logical and arithmetic operations. Arithmetic Logic Unit (ALU) is a critical component of a microprocessor and is the core component of central processing unit. ALU can perform various logic operations or different Arithmetic instructions include addition, subtraction, While logic instructions include Boolean comparisons, such as AND, OR, NAND, NOR, XOR and NOT operation.

Arithmetic and logic unit consists of two blocks for different operations are Arithmetic operations and Logical operations.

Addition and subtraction

These two tasks are performed by constructs of logic gates, such as half adders and full adders. While they may be termed 'adders', with the aid of they can also perform subtraction via use of inverters and 'two's complement' arithmetic. A binary adder-subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers. Connecting n full adders in cascade produces a binary adder for two n-bit numbers.

Logical operations

Further logic gate s are used within the ALU to perform a number of different logical tests,
including seeing if an operation produces a result of zero. Most of these logical tests are used to then change the values stored in the flag register, so that they may be checked later by separate operations or instructions.

An arithmetic logic unit(ALU) is a major component of the central processing unit of the a computer system. It does all processes related to arithmetic and logic operations that need to be done on instruction words. In some microprocessor architectures, the ALU is divided into the arithmetic unit (AU) and the logic unit (LU).

An ALU can be designed to calculate many different operations. When the operations become more and more complex, then the ALU will also become more and more expensive and also takes up more space in the CPU and dissipates more heat. That is why engineers make the ALU powerful enough to ensure that the CPU is also powerful and fast, but not so complex as to become prohibitive in terms of cost and other disadvantages.

ALU is also known as an Integer Unit (IU). The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need. Most of these operations are logical in nature. Depending on how the ALU is designed, it can make the CPU more powerful, but it also consumes more energy and creates more heat. Therefore, there must be a balance between how powerful and complex the ALU is and how expensive the whole unit becomes.

This is why faster CPUs are more expensive, consume more power and dissipate more heat.

Different operation as carried out by ALU can be categorized as follows –

- **logical operations**: These include operations like AND, OR, NOT, XOR, NOR, NAND, etc.
- **Bit-Shifting Operations**: This pertains to shifting the positions of the bits by a certain number of places either towards the right or left, which is considered a multiplication or division operations.
- **Arithmetic operations**: This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operations are more expensive to make. Multiplication and subtraction can also be done by repetitive additions and subtractions respectively.

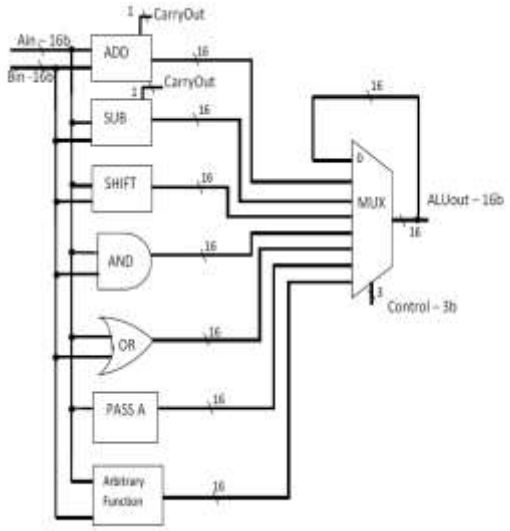


Fig 6: Proposed ALU Design with testing features

4. SIMULATION RESULTS

The total designs are done and implemented in Xilinx ISE 14.7 with Verilog HDL coding.

Table I: Comparison of Existing and Proposed method.

Parameters	Existing method	Proposed method 1	Proposed method 2 (ALU)
Area (LUT's)	164	131	326
Delay (ns)	25.686	25.455	39.488
power	0.203	0.203	0.203

5. CONCLUSION

In this paper, two different testing methods were designed. In this proposed method-I, a new design for self-checking carry select adder with fault localization for the input bits. Based on this design, We can easily identify the faulty check and replace the particular faulty modules. The proposed method-II was designed for testing the ALU design. This method is a new self-checking ALU was performed with different functional inputs. Through this ALU testing, we can easily identify the fault models.

REFERENCES

[1]. J. H. Stathis, "Reliability limits for the gate insulator in CMOS technology," *IBM Journal of Research and Development*, vol. 46, no. 2/3, pp. 265–286, 2002.

[2]. J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," *Proc. International Conferenceon Dependable Systems and Networks (DSN '04)*, pp. 177–186, June 2004.

[3]. D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of Applied Physics*, vol. 94, no. 1, pp. 1–18, 2003.

[4]. M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *IEEE Transactions on Very Large Scale Integration (VLSI)Systems*, vol. 11, no. 1, pp. 121–128, Feb. 2003.

[5]. B. Kumar and P. Lala, "On-line detection of faults in carry-select adders," *Proc. International Test Conference (ITC '03)*, vol. 1, pp. 912– 918, Sep. 2003.

[6]. D. Vasudevan and P. Lala, "A technique for modular design of self-checking carry-select adder," *Proc. 20th IEEE*

InternationalSymposium on Defect and Fault Tolerance in VLSI Systems (DFT '05), pp. 325–333, Oct. 2005.

[7]. N. Kito and N. Takagi, "Low-overhead fault-secure parallel prefix adder by carry-bit duplication," *IEICE Transactions on Informationand Systems*, vol. E96-D, no. 9, pp. 1962–1970, Sep. 2013.

[8]. J. Rivers, M. Gupta, J. Shin, P. Kudva, and P. Bose, "Error tolerance in server class processors," *IEEE Transactions on Computer-AidedDesign of Integrated Circuits and Systems*, vol. 30, no. 7, pp. 945–959, July 2011.

[9]. J. R. Black, "Electromigration - a brief survey and some recent results," *IEEE Transactions on Electron Devices*, vol. 16, no. 4, pp. 338–347, April 1969.

[10]. C. K. Hu, R. Rosenberg, H. S. Rathore, D. B. Nguyen, and B. Agarwala, "Scaling effect on electromigration in on-chip Cu wiring," *Proc. IEEE International Interconnect Technology Conference*, pp. 267– 269, May 1999.

[11]. E. T. Ogawa, J. W. McPherson, J. A. Rosal, K. J. Dickerson, T. C. Chiu, L. Y. Tsung, M. K. Jain, T. D. Bonifield, J. C. Ondrusek, and W. R. McKee, "Stress-induced voiding under vias connected to wide Cu metal leads," pp. 312–321, April 2002.

[12]. H. Matsuyama, T. Suzuki, T. Nakamura, M. Shiozu, and H. Ehara, "Re-think stress migration phenomenon with stress measurement in 12 years," *Proc. IEEE International Interconnect Technology Conference and IEEE Materials for Advanced Metallization Conference(IITC/MAM)*, pp. 307–310, May 2015.

[13]. B. Becker, R. Drechsler, and P. Molitor, "On the generation of areatime optimal testable adders," *IEEE Transactions on Computer-AidedDesign of Integrated Circuits and Systems*, vol. 14, pp. 1049–1066, Sep. 1995.

[14]. R. D. Blanton and J. P. Hayes, "Testability of convergent tree circuits," *IEEE Transactions on Computers*, vol. 45, pp. 950–963, Aug. 1996.

[15]. S. Kajihara and T. Sasao, "On the adders with minimum tests," *Proc. Sixth Asian Test Symposium (ATS '97)*, pp. 10–15, Nov. 1997.