مجلة جامعة بني وليد للعلوم الإنسانية والتطبيقية

تصدر عن جامعة بني وليد – ليبيا

Website: https://jhas-bwu.com/index.php/bwjhas/index

العدد التاسع والعشرون، 2023

Design and Implementation of a Self-Checking Rotator Using Berger Code

A. M. Ejamail[1*], A. A. Khalleefah[2] , A. M. Salim[3], A. H. Maamar[4]

[1, 3]Computer engineering & IT, College of Technology Sciences, Bani-Walid -Libya

[2, 4]Department of Computer engineering, College of ELECTRONIC TECHNOLOGY, Bani-Walid –Libya

*Crosspnding author: [a]Adel.m.Ejamail@gmail.com

تاريخ الاستلام: 2023-06-17 تاريخ القبول: 2023-06-27 تاريخ النشر : 2023-09-07

Abstract: The advances in semiconductor generation have greatly improved the scale of integration. Today virtual systems are extra complicated than ever earlier than. These complex circuits are greater susceptible temporary and intermittent faulty. The complexity of the digital circuits results in extra crosstalk, noise, and different assets of temporary errors during normal operations. Conventional off-line testing techniques cannot guarantee detection of these faults; they can be detected by using on-line or concurrent error detection (CED). Concurrent error detection methods allow digital systems to affirm the correctness in their effects in the course of regular operation. Berger Code is one of the famous codes for CED applications because it could stumble on all unidirectional errors in digital structures. This paper proposes a layout to achieve the self-checking checker used of Berger code as a means of incorporating CED right into a self-checking for the rotate register.

Keywords: Berger Code, Rotator Register, Two-Rail Checker (TRC), Self-checking.

1. Introduction

Since the 1960s, concurrent error detection techniques have been used extensively in commercial digital systems [1]. Concurrent error detection's primary goal is to perform real-time checks on system outputs to ensure data integrity by identifying temporary or permanent faults while the system is functioning normally. Concurrent error detection makes it possible to find errors as the system is running. is essential for a broad range of applications, from avionics and space to telephone switching networks, online transaction processing, etc. An online error detector of your choice can be added to a computer system. These detectors operate under the fundamental principle of redundancy in hardware replication, information (redundant codes), software, or execution time. Redundancy is the use of additional resources above and beyond what is necessary for an unchecked system [2]. By some other words this redundancy can take the form of additional hardware (hardware redundancy), additional software (software redundancy), additional information (information redundancy), or multiple code executions on a single piece of hardware (time redundancy) [3]. Only information redundancy using Berger code checkers in Totally Self-Checking (TSC) is the focus of this paper.

2. Rotate register

A rotate register is a virtual circuit includes a cascade of flip flops, wherein the output of one flip flop is connected to the input of the subsequent turn flop. The data stored within the rotate register can be shifted from one location to the following vicinity. An N-bits rotate register can be shaped by means of connecting N-flip- flops where each flip- flop keep a single bit of data. Registers are used for data storage or for the motion of data; they're normally used interior Calculators or computer systems to save data inclusive of two Binary numbers before they're added together, or to convert data from both a parallel to serial or serial to parallel layout [4]. Figure 1 shows the impact of facts motion from left to right through a rotate register.

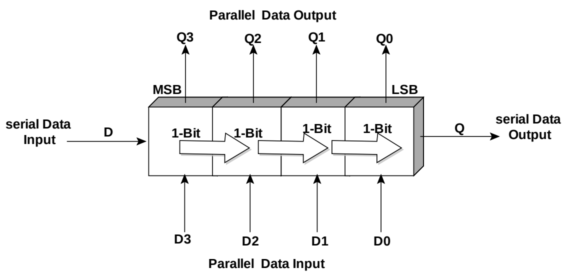

Fig. 1: rotate register.

Figure 2 shows rotate right operation, the serial output of the register is connected to the serial input of the register.

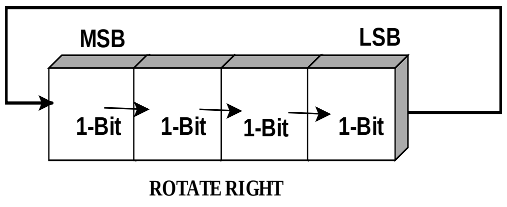

Fig. 2: rotate Right register.

By applying clock pulses data is shifted right. The data shifted out of the serial out pin at the right hand side is recirculated back into the shift register input at the left hand side. Thus the data is rotated right within the register. Figure 3 shows rotate left operation, the serial output of the register is connected to the serial input of the register.

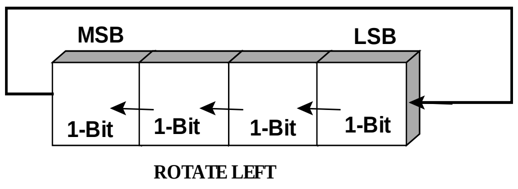

Fig. 3: rotate Left register.

By applying clock pulses data is shifted left. The data shifted out of the serial out pin at the left hand side is recirculated back into the shift register input at the right hand side. Thus the data is rotated left within the register.

3. SELF-CHECKING CIRCUITS

Self-checking is the capability of a logic component (chips, boards, or assembled system) to automatically check for errors [5]. Self-checking circuits enable on-line error detection, which enables faults to be found as the circuit is being used normally. It has the ability to identify both temporary and irreversible faults [6]. To the original circuit (functional unit), it is intended to add some circuitry. If there is an error in the functional unit, this additional circuit will check the output of the functional unit to ascertain it [7]. Self-checking circuits can be designed using a variety of codes. Only separable codes that separate the information bits from the check bits are the focus of this paper. The Berger code converts a word's number of 1s into a binary representation. The count is added to the data after the binary word is completed. The best AUED (All Unidirectional Error Detecting) codes are called Berger codes. It can identify all unidirectional bit errors, such as when one or more ones become zeros, but it cannot recognize when zeros become ones. The error won't be noticed if the same number of bits flip from one to zero as they do from zero to one [8]. Two-Rail Checker (TRC) A one bit output is insufficient for a Totally Self-Checking (TSC) circuit because a stuck-at-fault on the output resulting in the "good" output could not be detected [2, 9, 10, 11]. A two rail (1-out-of-2) code is typically used to encode the output of a TSC circuit. Two complementary codewords are compared using a two-rail checker unit (TRC). The checker decides whether the functional circuit's output is an acceptable or unacceptable codeword [2]. Figure 4's block diagram depicted the suggested design for the self-checking rotator register checker. The proposed 4-bit Rotator Left/Right register resembles a shift register with taps added to each stage of input and output in a serial-in/serial-out design. Din (Serial Input) is where serial data shifts in. The first Din bit data appears at (Q4) after a certain number of clocks that is equal to the number of bits. Start by entering the rightmost bit of the four bits (1001) into the register. As soon as data are entered into the register, the Check Symbol Generator (CSG) generates a code word for the data, which is then stored in the 3-bit register and used to compare the inputs from the two rails.

Self-checking can be defined as the ability to automatically check logic (chip, board, or assembled system) for errors [5]. A self-checking circuit allows online fault detection. This means that faults can be detected during normal operation of the circuit. It can detect the presence of both transient and permanent errors [6]. The idea is to add some circuits to the original circuit (functional unit). This additional circuitry examines the outputs of the functional units and determines if the functional units are faulty [7]. Many codes can be used to design self-checking circuits. This article will only focus on separable code where the information bit is separated from the check bit. A Berger code counts the number of ones in a word and represents it in binary. Complement binary words and add counts to the data. The Berger code is the best systematic AUED (All Unidirectional Error Detecting) code detects all unidirectional errors. If one or more 1's are converted to 0's, they can be identified, but if 0's are converted to 1's at the same time, they are not. If the same number of bits changes from 1 to 0 as they change from 0 to 1, no error is detected [8]. A 1-bit output is not sufficient for a full self-test (TSC) circuit because it cannot detect stuck-at faults in the output that lead to a 2-Rail Checker (TRC) "good" output [2, 9], 10, 11]. The output of the TSC circuit is typically encoded with a two-rail (1 of 2) code. A Two-Rail Checker Unit (TRC) is used to compare two complementary codewords. A validator determines whether the output of a functional circuit is a valid or invalid codeword [2]. The block diagram in Figure 4 shows a proposed design for implementing a self-checking rotator register checker. The proposed 4-bit rotator left/right register looks like a serial-in/serial-out shift register with additional taps at each input and output stage. Serial data is inserted into Din (serial input). After a number of clocks equal to the number of bits, the first Din bit data appears on (Q4). Enter 4 bits (1001) into the register, starting with the rightmost bit. As data enters a register, a check symbol generator (CSG) generates codewords for the data. A 3-bit register contains complementary codewords used at the two rail inputs for comparison.

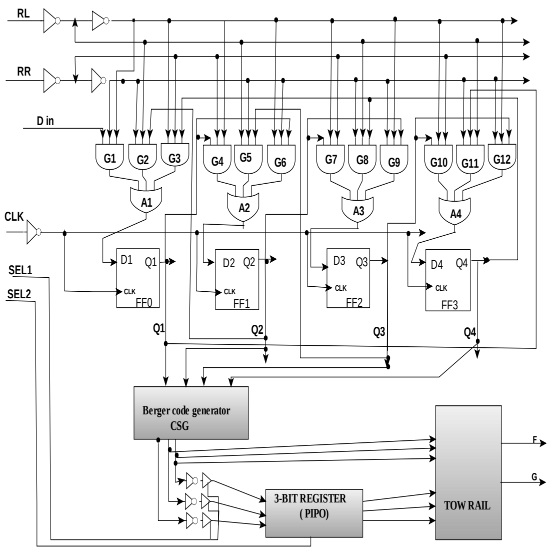

Fig. 4: 4 Bit Rotator Left/Right register.

Since no information bit is discarded except for their position the Berger check of the result of rotate operation is simply the Berger check of the operand, since no information bit and the numbers of 1's is the same, if no error is detected. After the operation of register rotate (left/right) done, the CSG generates new check bits code for that word. In order to detect faults on the word after the operation we drive the stored check bits code and the inverted created one to the Two -rail checker. The steps in the checking procedure can be implemented as shows the flowchart in figure 5. The designs are implemented in Verilog HDL for simulation and synthesis. The Waveforms below shows the output of the Simulation Processor. The waveform in Figure 6 shows the Reset register, loading data to the rotator register and execute the rotate left / rotate right to the loaded Data, so the signals as follow:

CLK: clock signal. Din: Data in shows the input data into the register.Q1: first Bit of the output data shows the value of first bit in the register.Q2: second Bit of the output data shows the value of second bit in the register. Q3: third Bit of the output data shows the value of third bit in the register.Q4: fourth Bit of the output data shows the value of fourth bit in the register. The table 1 shows the control state for the used register. The waveform in

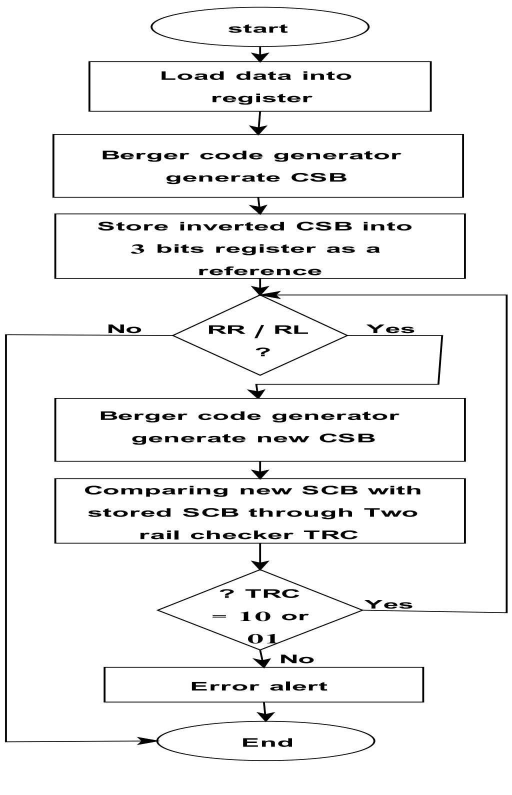
start
Load data into register
Berger code generator generate CSB
Store inverted CSB into 3 bits register as a reference
RR / RL ?
No
Yes
Berger code generator generate new CSB
Comparing new SCB with stored SCB through Two rail checker TRC
? TRC = 10 or 01
Yes
No
Error alert
End

Fig. 5: Flowchart shows steps in the checking procedure.

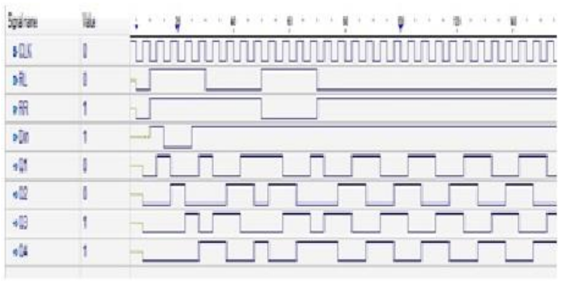

Fig. 6: Waveform shows loading data to the rotator register.

Table 1: control state for the register.

RL	RR	Operation
0	0	Reset register
0	1	rotate right
1	0	rotate left
1	1	Load data

Figure 7 shows the Berger Code check symbol Generator for 4-bit, where the input for CSG is the output for the 4-bit rotator register, the signals as follow:

CLK: clock signal. Q1: first Bit of the output data shows the value of first bit in the register.Q2: second Bit of the output data shows the value of second bit in the register.

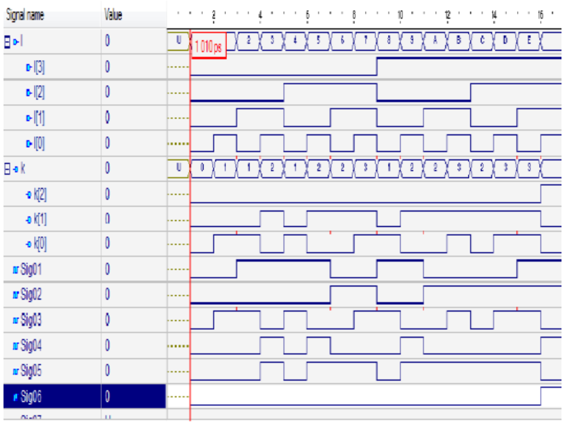

Fig. 7: Waveform shows Berger code check symbol generator.

Q3: third Bit of the output data shows the value of third bit in the register. Q4: fourth Bit of the output data shows the value of fourth bit in the register.Sig21: first Bit of the CSG output shows the value of first bit in the CSG output. Sig22: second Bit of the CSG output shows the value of second bit in the CSG output. Sig23: third Bit of the CSG output shows the value of third bit in the CSG output. The waveform in the Figure 8 shows the implementation loaded data to the register, generate the check symbol bits word, pushed in complement of the check symbol bits word to 3-bit register, rotate left/right the register, drive data to two rail checker, and obtain the result.

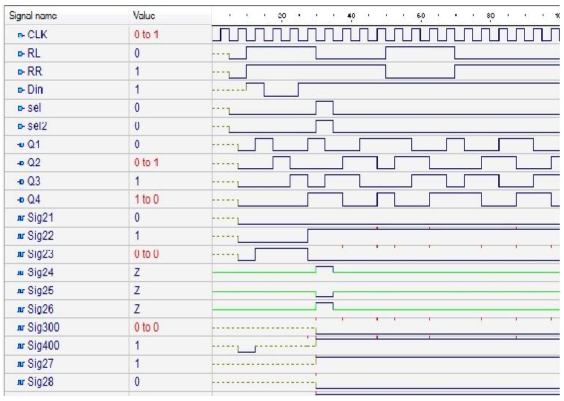

Fig. 8: Waveform shows the Simulation of rotator implementation processor.

4. Conclusion

This paper presents a technique for implementing fault secure rotators based on Berger code. In this design Berger code generator generates check bits code for each input word of the shift register, since no information bit and the numbers of 1's is the same, if no error is detected. After the operation of register rotate left/right done, the Berger code generator generates check bits code for that word. In order to detect faults on the word after the operation we drive the stored check bits code and the inverted created one to the Two -rail checker. This design achieves high levels of reliability by means of moderate hardware cost. The system can detect all Unidirectional Error in the rotator.

5. References

[1]. Subhasish Mitra, "Diversity Techniques for Concurrent Error Detection", Technical Report, Center for reliable computing, May 2000.

[2] P. K. Lala. Self-Checking and Fault-Tolerant Digital System Design. Morgan Kaufman Publishers, San Francisco, 2001.

[3]. Manoj Franklin, "A Study of Time Redundant Fault Tolerance Techniques for Superscalar Processors" Department of Electrical & Computer Engineering, Clemson University, Clemson, USA, 1995 IEEE.

[4]. Tony R. Kuphaldt Lessons In Electric Circuits,
Volume IV Digital Fourth Edition, last update
November 01, 2007, openbookproject.net/
electricCircuits.

[5] Huda Abugharsa, and Ali Maamar," Self Checking Systolic LIFO Stack",7th WSEAS Int. Conf. on Instrumentation Measurement, Circuits and Systems (IMCAS '08), Hangzhou, China, April 6-8, 2008.

[6] KHADIJA F. O. ALGHEITTA. AMAL J. MAHFOUD. ALI H. MAAMAR. Design of a Self-Checking Up Counter. International Conference on Advanced in Computing, Engineering and Learning Technologies, Abu Dhabi, UAE, 2013.

[7] Mustafa Abd-El-Barr," Design and Analysis of Reliable and Fault-Tolerant Computer Systems ", Imperial College Press, 2007, ISBN 1-86094-668-2.

[8] VARADAN SAVULIMEDU VEERAVALLI. Diagnosis And Error Correction For A Fault-Tolerant Arithmetic And Logic Unit For Medical Microprocessors, Graduate School- New Brunswick Rutgers, The State University of New Jersey October, 2008.

[9] B. W. Johnson. Design and Analysis of Fault Tolerant Digital Systems. Addison-Wesley, Reading, MA, 1989.

[10] D. K. Pradhan. Fault-Tolerant Computing: Theory and Techniques, volume I. Prentice Hall, Englewood Cliffs, New Jersey, 2003.

[11] T. R. Stankovic, M. K. Stojcev, and G. Ordjevic. Design of Self-Checking Combinational Circuits. In Proc. of the International Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services, volume 17, pages 763-768, October 2003.