Self Checking FIFO Queue

AMAL J. MAHFOUD
Academy of Higher Studies
Tripoli, LIBYA
amal_11081@yahoo.com

ALI H. MAAMAR
Higher Institute of Electronics
Beni Waled, LIBYA
Ali_h_maamar@yahoo.com

***Abstract*— The advances in VLSI technology have made many changes not only in the architecture of computer and digital systems, but also in the amount of hardware that can be implemented in a single chip which makes it possible to include the entire system in single chip. But as the scale of integration has increased so also has the occurrence of intermittent faults. The detection of intermittent faults requires the use of concurrent error detection CED (coding) techniques. This paper investigates the use of Berger code as a mean of incorporating CED into a self checking FIFO queue.**

Key-Words: FIFO Queue, Self checking, Berger code, Concurrent Error Detection, Unidirectional errors.

I. Introduction

As integrated circuit technology achieves higher and higher densities of active components and thus capable of implementing more complex structures. There is an increasing trend for implementing complex algorithms in hardware. Generally the best implementations are defined from the structures with high degree of regularity. As an example of algorithms that can be implemented in hardware is the queue, hardware queue is capable of higher speed than software queue using a conventional microprocessor memory. High speed queue can be useful in many applications. A queue is a particular kind of collection in which the entities in the collection are kept in order, and the principal operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure. Unfortunately as the scale of integration has increased so also has the occurrence of intermittent faults. The characteristics of these types of faults render them undetectable by standard test strategies. The detection of intermittent faults requires the use of *Concurrent Error Detection* (CED) techniques, which continually monitor the operation of the circuit and compared it with some known reference; this is achieved by incorporating some form of redundancy into the system [1]. One method of implementing CED in VLSI 1 circuit is through the use of information redundancy. This paper investigates the use Berger Code as means of incorporating CED into a self checking queue. Berger code is an optimal code [2], it can detect all unidirectional errors and it is a separable code which means it is eases to decode and encode. Self-checking circuit can be defined as the ability to verify automatically whether there is any fault in the circuit (chips, boards, or assembled system), thus, self-checking circuits allow on-line error detection, which means faults can be detected during the normal operation of the circuit [3]. The self-checking could be achieved by the redundancy techniques; one way to achieve self-checking design is through the use of error detecting codes (the information redundancy technique) [4].

II. FIFO Queue

In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that whenever an element is added, all elements that were added before have to be removed before the new element can be invoked. A FIFO consists of an array of registers as shown in figure 1, and a controller that manages the traffic of data to and from the FIFO. FIFO's architecture provides access to only one register cell at a time, not to the entire array of registers. A *FIFO* has two address pointers, one for writing to the next available cell, and another one for reading the next unread cell [5]. The pointers for reading and writing are relocated dynamically as command to read or write are received. The *FIFO* buffer can receive data until it is full and can be read until it is empty. A pointer is moved after each operation. Then will describe the designing of two cases of queue:

- *Queue is Full.*
- *Queue is Empty.*

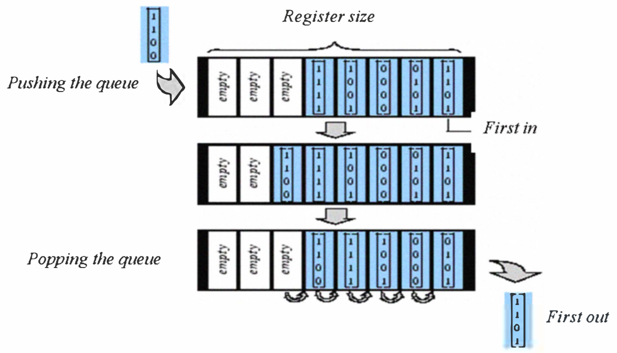

Figure 1. FIFO behavior

It's very important to know the status of the queue before any pop or push operation is performed. Figure 2 shows the typical waveform in the FIFO. CLOCK signal is free running. The 2 writing of new data into the FIFO is initialized by a rising edge on the CLK line and write signal is high. The data are written into the FIFO with the rising edge of CLOCK signal . The reading of the data from the

FIFO is also by a rising positive edge of the clock signal and read signal is high.

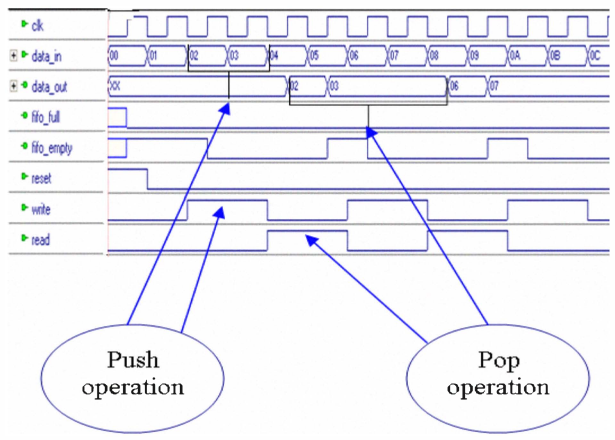

Figure 2. Timing diagram of the FIFO

A queue data structure is the fact that allows access to only the front and back of the structure. Furthermore, elements can only be removed from the front and can only be added to the back as shown in figure 3. The storage array is the main circuit of the FIFO, it consists of an array of registers (c_0, c_1, c_2, ………, c_n). All the registers have the same length, and has (input and output) control signals to control the data movement. Each row of the array is a full word which is controlled by states controller.

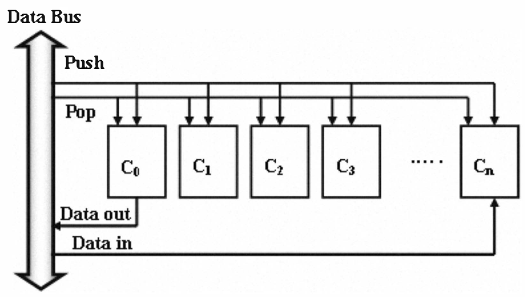

Figure 3. FIFO queue architecture

III. Berger Code

Berger code is a separable and unordered code, it is separable because the information bits and the check bits (check symbol) in the codeword are separate, it is an unordered code as it is not possible to change one codeword into another codeword by simply changing either 1's to 0's or 0's to 1's, this means that the code can detect all unidirectional errors. The codeword of the Berger code is formed by appending the check bits to the information bits, the check bits of the code is the binary representation of the number of *1's* (or the complement of the number of *0's*) in the information bits, the number of check bits [k=log (I+1)], where *I* is the number of bits in the information bits (data word), the number of bits in the codeword $n = I + k$ bits. If the number of information bits in a codeword is $I = 2^k - 1$, $k \geq 1$ then the code is known as a Maximal Length Berger code; otherwise it is known as a Non Maximum Length Berger code. For example, if $I = 7$ and $k = 3$, it is Maximal Length Berger code because $I = (2^k - 1)$, whereas $I = 6$ and $k = 3$ is Non-Maximal Length Berger code because $I \neq (2^k - 1)$.

IV. Self Checking Checker

Self-checking circuits allow on-line error detection, that means faults can be detected during the normal operation of the circuit. It can detect the presence of both transient and permanent faults. A self-checking circuit, see figure 4, consists of a functional circuit (F), which produces encoded output vectors, and a checker (C), which checks the vectors to determine if an error has occurred. The checker has the ability to give an error indication even when a fault occurs in the checker itself. Self-checking logic is typically designed using coding techniques; one way to achieve self checking design is through the use of error detecting codes (the information redundancy technique)

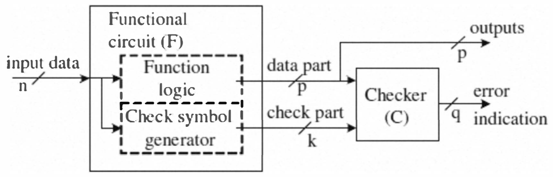

Fig .4 General structure of self-checking circuit

V. Two Rail Checker

Two-rail checker unit (TRC) is used to compare two complementary code words. The checker determines whether the output of the functional circuit is a valid or invalid codeword. Two-rail checker unit has two groups of inputs: (x_1,x_2, …..x_n) and (y_1,y_2,…..y_n). It also has two outputs: *f* and *g*. The signals observed on the outputs should always be complementary. Consider a two rail checker with *n*=2, as shown in Figure 5, the two input groups are (x_1,x_2) and (y_1,y_2). In a non-error situation where (y_1=x_1') and (y_2=x_2'), the result of this is(*f*=*g*'). In situation where due to a fault where (y_1=x_1) or (y_2=x_2), this will then produce (*f*=*g*), that means a non codeword output thus giving an error indication.

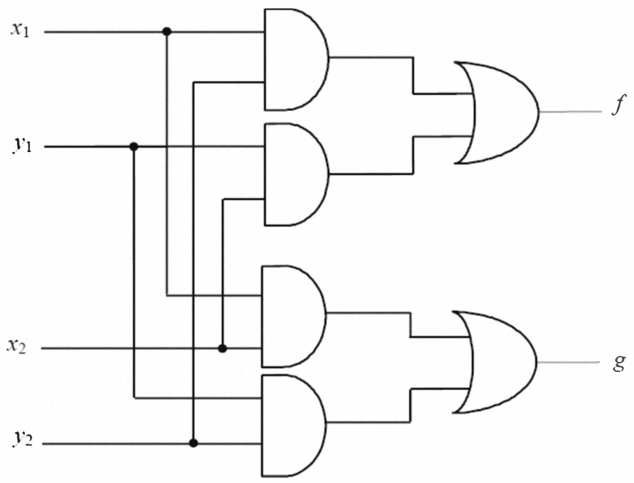

Figure 5. two Rail checker with 2 inputs

VI. SELF CHECKING HARDWARE

Figure 6, shows the self checking FIFO queue . In self checking FIFO queue two check symbol generators (CSG) circuits are needed, one checker, and an extra storage cells (K) attached to each word, these cells are used to store the check symbol generated for each word pushed into the FIFO, the number of the bits of the check symbol depends on the code used and also on the size of the data word.

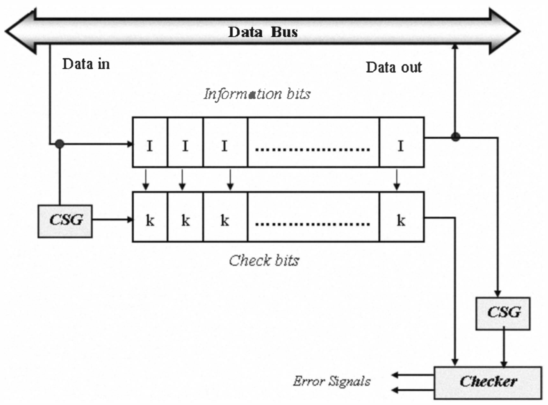

Figure 6. Self checking FIFO Queue

For example, if the size of the data word that can be pushed into the FIFO is 8 bits(I=8), and since we used Berger code, then the number of bits of the check symbol is 4 bits (K=4). When data word is pushed into the FIFO, its check symbol should be generated and pushed into the FIFO, as the data word moves to or from the FIFO its check symbol should also follows the data word. Figure 6, Self checking FIFO queue. When data is to be popped out from the FIFO via the bus, the data should immediately be checked for any detectable errors. The check symbol generator (CSG) is a one counter, it counts the number of ones in the information bits of any information word, and gives the number of ones which represents the check symbol. When the check symbol becomes available it is then compared with the stored check symbol (which generated when the data word pushed in the FIFO). If the stored check symbol and the generated check symbol of the popped word are match then the data word is error free and can be moved out from the FIFO, but if they not match then the data word is not error free word.

VII. CONCLUSION

The work in this paper was concerned with the investigation of Berger code as a means of integrating a Concurrent Error Detection (CED) scheme into a FIFO Queue. Berger code has the advantage that it can detect all unidirectional errors. The design of a Self-Checking hardware FIFO Queue using Berger code have been presented, the Queue is self checking against errors affecting the information bits and the check bits.

REFERENCES

[1] Russell, G.; Maamar, A.H., "Check bit prediction scheme using Dong's code for concurrent error detection in VLSI processors," Computers and Digital Techniques, IEE Proceedings - , vol.147, no.6, pp.467-471, Nov 2000.

[2] Hao Dong, "Modified Berger Codes for Detection of Unidirectional Errors", Computers, IEEE Transactions on, vol.C-33, no.6, pp.572-575, June 1984.

[3] Miron Abramovici, Melvin A.Breuer, and Arthur D.Friedman, "DIGITAL SYSTEMS TESTING AND TESTABLE DESIGN" , 1990,ISBN 0-7803-1062-4, Chapter 13:SELFCHECKING DESIGN, pp.569-587.

[4] 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.

[5] Michael D.Ciletti, "Advanced Digital Design with the Verilog HDL", Upper Saddle River, New Jersey, Chapter 9, pp.628-641.