Implementation and Delay Estimation of Concurrent Error Detection
Arithmetic Adders Using Hardware Redundancy Based on Dual Rail Encoding

Susrutha Babu Sukhavasi, Suparshya Babu Sukhavasi and Navarun Gupta
Department of Electrical Engineering
University of Bridgeport

Arithmetic functions are the most used operations in VLSI circuits. So the design of adders with high reliability and speed operation are of major concern in such circuits. This paper presents a methodology for designing totally self-checking Arithmetic adders for VLSI circuits and FPGA implementation using Verilog HDL. It detects the presence of all single stuck-at faults on-line that may occur in digital systems. The design encodes sum/carry/propagate bits of an adder based on 2-rail codes and then encoded bits are checked by using self-checking checker. The design is proposed for carry-skip and carry look-ahead adders. The total overhead and delay varies slightly when compared to the adders without self-checking capability; thus the reliability glance can be inserted with little increased overhead & delay. The totally self-checking system (TSC) can be implemented through an Application Specific Integrated Circuit or Field Programmable Gate Array. FPGA implementation is preferable, as its design is less complex and low cost compared to ASIC design. This paper illustrates the low cost advanced self-checking scheme using Xilinx ISE 11 and Digilent Nexys2 FPGA board.

Corresponding Author: Dr. Navarun Gupta, navarung@bridgeport.edu

Introduction

With the evolution of science and technology in recent years, the complexity of digital systems has increased hysterically to seize high performance.

Increase in level of complexity makes these systems highly desirable for maintenance and repair. In addition to this, the large scale integration has significantly increased the occurrence of intermittent and transient faults. The characteristics of these faults which occur only in the normal functioning of system make them undetectable through standard test strategies. So interest in concurrent fault detection continues to grow as VLSI circuits get more and more complex. The probability of occurrence of transient faults can be found using a self-checking design; constantly monitors the functioning and indicates whether it is faultless or not. One means of implementing this on-line fault detection method in VLSI circuits is through the use of 2-Rail Encoding technique. Through the extensive course of time, miscellaneous arithmetic adder types have been developed for more performance and less delay. Some of them are ripple-carry, carry-skip, carry look-ahead, carry-select and conditional sum adder. Area, overhead and delay constraints always vary for each one of them individually, as depicted in [1]. The main difference between all these adders is how the carry is generated/propagated/selected/ passed to the consecutive bit. Over the years, advanced versions of the adder styles are introduced improving their operation, but their reliability is left behind. To ensure the genuine operation of these adders, detecting the transient faults is eminently important.

As discussed above, transient faults are detected by the on-line or concurrent fault detection i.e., built-in self-checking capability. Several techniques have been proposed for the self-checking design of adders, for example information redundancy or coding technique[2]-[4], parity prediction[5][6], time redundancy[7]. The disadvantages of all the approaches include non-detection of all single faults, complex checker designs for residue codes, unnoticed
faults in carry is propagated, increase in computation time leading to slow addition process. In recent years, various techniques for the concurrent fault detection have been presented in [8]-[13], but these techniques result in increased overhead and area because of the configuration logic block. In [14], carry-select self-checking design has been proposed based on two-rail encoding, a rather simple and effective technique compared to the others with less increased delay and overhead.

From this inspiration, we propose a self-checking scheme for all the adder styles in this paper and compare the constraints in the absence of self-checking module. The results clearly suggest that the configuration logic block in the two-rail encoding design increase the constraints to a minute level and the adders are provided with a system that has the following features:

- Integrated compact checker.
- Stuck at-0/1 faults at input, intermediate and output lines.
- Detects all the single faults i.e., totally self-checking.
- Faults are immediately detected upon occurrence. This prevents corruption of data.

The encoded sum/carry/propagate bits are fed (inputs) to the two rail checker for the fault detection. The 2-rail checker has two sets of inputs $X0, X1,...,Xn$ & $Y0, Y1,...,Yn$ and two output signals $f$ and $g$. It follows a rule that the inputs should be a complemented pair i.e., $X0, Y0$ and $X1, Y1$ should be complement to each other. The encoded bits are produced to the checker in this complemented pair manner. When it deviates from the actual routine due to a fault, the checker gives the error indication. For example, when $X0X1=11$ and $Y0Y1=00$ the checker produce a non-error indication i.e., $f=0$, $g=1$. Where in a situation due to a fault $Y0Y1=01$, it produces $f=1$ and $g=1$, a non-code giving the error indication somewhere in the circuit. Below Fig. 1 illustrates the gate level model of 2-rail checker.

This paper is structured as follows. Section 2 describes the self-checking design module based on two rail encoding. Section 3 illustrates the configuration logic required by the 2-bit carry-select adder. Section 4, 5, 6, 7 presents the configuration logic block for the 4-bit ripple-carry, carry-skip, carry look-ahead adder and conditional sum adders. FPGA implementation details and simulation results of the implemented design is presented in section 8. The resultant constraints are compared and discussed in section 9 and paper is culminated in section 10.

**Self-Checking design**

In the past, the field of fault coverage and design of self-checking systems has grabbed minimal curiosity from the researchers. The increased complexity of the systems made them to look for a solution so that the systems have a capability of self-checking. Informally, it is a facility to verify whether there is a fault in the system without any externally exercised test strategies. The formal definition that distinguishes the TSC circuits can be extracted from self-checking design theory [15] which is given below.

![Fig. 2 Totally self-checking module](image-url)
A circuit is “totally self-checking” if it is both fault-secure and self-testing.

- **Self-testing:** A circuit is self-testing if, for every modeled fault, the normal operation produces an invalid code output for at least one code input.
- **Fault secure:** A circuit is fault secure if, for every modeled fault, the circuit never produces an erroneous code output that belongs to the output code for all code inputs.

Self-testing implies the detection of occurrence of any fault or deviation from the prescribed set by at least one input set. Whereas, fault secure property says that the faulty system cannot have a code word as its output and in addition it should be different from correct code word. All the faults from a given set would cause erroneous output in a digital system during the normal operation. So these compact totally self-checking designs are crucial for the trustworthy digital designs. The theoretical self-checking model is illustrated in fig. 2. By observing the output of the checker it is possible to detect the fault but it is not possible to locate that fault from the information provided by checker output. The output combinations 01 and 10 are considered as faultless, whereas the checker output 00 and 11 indicates a non-code word or fault.

Considering the self-checking design primitives of the arithmetic adders, the main block design of a totally self-checking n-bit arithmetic adder based on two rail encoding can be presented as below [Fig. 3].

![Fig. 3 Self-Checking Arithmetic Adders design](image)

The sum/carry/propagate bits which are prior of a particular adder are two-rail encoded by using a configuration logic block [CLB]. Configuration logic for the input of the checker is designed by observing the addition property for a particular adder type. Each type of adder uses different addition property or logic, in regard, the relation between generated sum and carry bits are evolved and eventually their CLB is obtained. In other words, the normal functioning of individual adders is given to the checker as a faultless design. Occurrence of transient faults leads to the deviation of normal operation and indicated by a non-code word i.e., 00 or 11. This technique is very much preferable as the configuration logic is built by using few conventional gates, consequently varying the constraints to a minimal level.

From the details in above discussion it should be understandable that a configuration logic block that generates valid code word as input to the checker from the corresponding adder outputs is needed. Therefore, an arithmetic adder style could be concurrently tested if the two-rail checker is provided with complimentary inputs i.e., valid code words (01 or 10). The addition property and the design of configuration logics for mentioned adder types are presented in the following sections.

**Design of 4-bit self-checking carry-skip adder**

Carry-skip or carry-bypass adder is an improved implementation on the delay of a Ripple-carry adder with a minimal effort. To speed-up the operation, the carry propagation is skipped to the last position without waiting for rippling to next stage. For each input pair \((a_i, b_i)\), the propagate signal is generated by Xor, \(P_i = a_i \oplus b_i\). When all the propagate signals are logic-1, then the initial \(cin\) determines the carry-out. Each propagate signal bit generated by the ripple carry block is connected to the n-input AND gate. The ensuing bit is fed as the select bit to a MUX that switches either the nth carry-bit or the initial \(cin\) to the carry-out output. In this manner propagate signal plays a key role in the adder operation. As a result the two rail encoding is performed including the propagate signal. The possible combinations of addition are shown in table-1.
For $\text{carrying-in}=0$, so $S_00'$ and $P_0$ is one complemented pair. Subsequently, it can observed that $C_{00} \oplus S_{01} = P_1$ and $C_{01} \oplus S_{02} = P_2$ so on… Therefore, $P_1$ and $(S_{01} \oplus C_{00})$ becomes another complementary pair.

So on i.e, $P_2$ and $(S_{02} \oplus C_{01})$ is the next complementary pair. $P_3$ and $(S_{03} \oplus C_{02})$ is the subsequent pair.

For carry-in =1

It is observed that $S_{10}$ and $P_0$ is one complementary pair. The relation between successive bits is derived as follows

$C_{10} \oplus S_{11} = P_1$ and $C_{11} \oplus S_{12} = P_2$ so on…

Which implies that $P_1$ and $(S_{11} \oplus C_{10})$ becomes a complementary pair.

So on i.e., $P_2$ and $(S_{12} \oplus C_{11})$ is the next complementary pair.

$P_3$ and $(S_{13} \oplus C_{12})$ is the subsequent pair.

The design can be summarized as, the sum bit of first full adder is to modified for initial carry-in i.e., the first sum bit is inverted for the carry-in =0 so as to obtain a complementary pair which is shown in the Fig. 5.

![Four-bit Self-Checking Carry-skip adder design](Image)

**Table-1**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A</th>
<th>B</th>
<th>S</th>
<th>C</th>
<th>S</th>
<th>C</th>
<th>S</th>
<th>C</th>
<th>P</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

From the above table, it is evident that $S_{00} = P_0$ for carry-in=0, so $S_00'$ and $P_0$ is one complemented pair. Subsequently, it can observed that $C_{00} \oplus S_{01} = P_1$ and $C_{01} \oplus S_{02} = P_2$ so on… Therefore, $P_1$ and $(S_{01} \oplus C_{00})$ becomes another complementary pair.

The encoded sum, carry bits with the propagate signal is fed to the two-rail checker according to the complementary rule, discussed in the initial sections. The occurrence of a transient fault is immediately noticed by the checker and produces a non-valid code word indicating the fault in the operation or in itself. The same is coded by using Verilog in Xilinx ISE 11 tool and implemented on FPGA. The simulation constraints are compared with other schemes and the same is discussed in the later section.
Design of 4-bit self-checking carry-look ahead adder

As already discussed in the earlier sections, parity prediction scheme has been proposed for all the adder types, but its non-detectability against the single faults is a major drawback. Despite the low hardware overhead and the compatibility with memory systems, it is not fault secure. So the scheme proposed below is more beneficial in accounts of fault secure property and less complexity of design.

The design of self-checking 4-bit carry look-ahead adder slice design is illustrated in fig. 5. Adder-bit slice is elaborated in the manifestation of each adder-bit slice using the carry duplication/replication in Fig. 6.

Fig. 5 4-bit Self-Checking Carry-look ahead adder-slice design

This technique uses carry replication method in the adder bit slice for the encoding of complementary pair. We know that the carry generation logic is the major part of the fast carry look ahead operation. So the carry bits are encoded for the fault secure design. As mentioned, the encoding is mainly evaluated from the carry replication part. In order to achieve the low hardware overhead, the carry check bit is implemented using the carry slice of a conventional ripple carry adder. The speed aspect can be maintained as; the carry in of each carry check slice comes from normal carry bits generated by the block, instead of preceding slice of carry check bits.

Fig. 6 shows the block diagram of the self-checking design using the carry encoding. Fig. 6 illustrates the adder slice bits detailing the carry replication. The compact circuit generates the carry check bits with a delay similar to the adder output delay. The carry generation block provides the inputs to both of them and this merges logic generating both the normal and carry check bits. Thus a fault in the normal and carry checks is detected in the block generating them. In this case, all the errors are noticed and indicated by the two-rail checker. The generate (Gi) and propagates (Pi) signals are used in the generation of normal carry by using initial carry-in signal.

\[
Pi = a_i \oplus b_i \\
Gi = a_i \cdot b_i
\]

Fig. 6 Adder bit-slice design using the carry replication

The two-pair two-rail checker receives the pair of inputs \((C1, C1')\) and\((C2, C2')\), where \(x0 = y0'\) and \(x1 = y1'\). The output of the two rail checker is also in two-rail form and indicates the fault with the invalid code word. The FPGA implementation and comparison constraints detailed in the last sections evident the low-hardware overhead and slightly varied speed of operation when compared a normal carry look-ahead adder.

Simulation Results

The self-checking arithmetic adder schemes were coded using Xilinx ISE11 synthesis tool and simulated using ModelSim SE 5.7f tool and output waveforms are shown in following
The simulations include the detection of stuck-at faults at varies nodes i.e., sum bits, carry and propagate bits. Each simulation is noted for the fault indication bits i.e. $f$ & $g$, the erroneous sum produced by the adder operation is noticed by the non-cord word i.e. 11 or 00

**Faultless Carry skip adder**

![Fig. 7 Faultless carry skip adder](image)

**Faultless Carry look ahead adder**

![Fig. 9 Faultless carry look ahead adder](image)

**Faulty Carry skip adder when sum $[1] = s-a-1$**

![Fig. 8 Fault detection in carry skip adder](image)

**Faulty Carry look ahead adder when carry bit $c[1] = s-a-0$**

![Fig 10 Fault detection in carry look ahead adder](image)

**Comparison and discussion of results**

The self-checking schemes were implemented using the Spartan 3E digilent nexys FPGA board. The resulting constraints are compared for all the adder models with and without the self-checking capability. The average fanout of the non-clock nets used by the designs is a constraint that details about how the design is efficiently routed and likely to meet the overhead constraints. Higher fanout means more load that decides the delay variation. Lower the fanout, lesser will be delay. The occupied slices infer about the percentage increase in overhead and area of the proposed designs when compared to the normal adder styles. Each slice contains a number of LUT’s, carry logic elements which makeups logic of the design before mapping.

![Fig. 11 Comparision (4-bit) for avg. fanout of non-clock nets](image)

Fig. 11, Fig. 12 and Fig.13 illustrates the comparison graphs of average fanout of non-clock nets path delay in $ns$ and slices overhead respectively. The results portray that high degree of reliability level can be included in the adder operation with very minimum variation of the delay and overhead. Thus, two-rail encoding
technique proves to be efficient for the TSC design.

![Fig. 12 Comparison(4-bit) for Path delay(ns)](image)

**Fig. 12** Comparison(4-bit) for Path delay(ns)

![Fig. 13 Comparison(4-bit) of slices occupied](image)

**Fig. 13** Comparison(4-bit) of slices occupied

**FPGA Implementation**

\[ a=0001, b=1111, \text{sum}=0001, \text{cout}=1, \text{f}=1, \text{g}=0 \]

(no fault)

![Fig. 14 Faultless 4-bit carry skip adder operation](image)

**Fig. 14** Faultless 4-bit carry skip adder operation

\[ a=0001, b=1111, \text{sum}=0011, \text{cout}=1, \text{f}=1, \text{g}=1 \]

(Fault)

![Fig. 15 Faulty 4-bit carry skip adder operation when sum[1]=s-a-1](image)

**Fig. 15** Faulty 4-bit carry skip adder operation when sum[1]=s-a-1

\[ a=1011, b=0001, \text{sum}=1101, \text{cout}=0, \text{f}=0, \text{g}=1 \]

(no fault)

![Fig. 16 Faultless 4-bit carry lookahead adder operation](image)

**Fig. 16** Faultless 4-bit carry lookahead adder operation

\[ a=1011, b=0001, \text{sum}=1101, \text{cout}=0, \text{f}=0, \text{g}=0 \]

(Fault)

![Fig. 17 Faulty 4-bit carry lookahead adder operation when c[1]=s-a-0](image)

**Fig. 17** Faulty 4-bit carry lookahead adder operation when c[1]=s-a-0
Conclusion

The two-rail encoding technique and the CLB design for the self checking arithmetic adders is proposed and their performance constraints are compared for all the models without self checking capability. These adder styles are totally self checking for each and every single fault happening at sum bits, carry bits, input, output, and configuration logic block. From the resultant constraints, we conclude that reliability can be achieved in digital systems with a minimum increase in delay and overhead. In the presence of a fault, invalid code word is fed as input to the checker yielding an invalid output code word.

References


[14] Dilip P. Vasudevan, Parag K. Lala, and James Patrick Parkerson, "Self-Checking Carry-Select Adder Design Based on Two-Rail Encoding", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR
