Self-checking combination and sequential networks design

Tatjana Nikolić
Faculty of Electronic Engineering
Nis, Serbia
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Motivation

- Modern embedded system design is oriented towards single system on chip (SoC)
- Complexity of system requires development of new:
  - design methods
  - system and circuit architectures
  - testing techniques

Prerequisite for achieving the required level of reliability - Design of ICs with built in fault-tolerance (FT) feature
  - Our focus is on self-checking (SC) circuit design
SC circuits design

- Low level methods:
  - modification of a circuit at low level (gate level)

- High level methods:
  - modification of VHDL description of a circuit based on different SC techniques,
  - VHDL code, that introduces hardware or information redundancy, is inserted at specific points in the original description.
The goal

**Disadvantages** of existing high-level methods (presented in many papers) are:
- Limited selection of SC techniques
- Performances of generated solutions, in term of area overhead and operation frequency decreasing, are insufficiently examined
- Practical application of the methods and feasibility for implementation in FPGA and CPLD technologies by using commercial CAD tools

These disadvantages present the motivation for developing of a **new methods for reliable circuit design**.

Development of a flexible design environment for automatic insertion of SC hardware, especially oriented toward:
- FPGA/CPLD technologies
- Integration with commercial CAD tools, and
- Performance/overhead estimation
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Development of electrical products

- Impressive progress in semiconductor technology
  - smaller transistor geometry
  - lower voltage operation
  - greater frequency (speeds and functionality)

- These factors lead to an increase in probability of fault occurrence.

- For realization of reliable circuits particular design techniques are needed.

- The goal of this techniques – fault-tolerant (FT) system design
  - FT systems can perform their functions even in the presence of faults.
Faults in integrated circuit

- **Fault** is physical malfunction, deficiency, or accident within a hardware or software component.

- **Error** is manifestation of a fault and presents deviation of accuracy or regularity.

- **Failure** is result of a error and presents fail of a action in expected and correct manner.
Fault models

Most models start from the fact that at any given moment may occur at most one fault.

Two basic fault models are:
- Logic-level models (stuck-fault models) – stuck-at-0 and stuck-at-1
  - Transistor-level models – stuck-on and stuck-off

Error model is defined in term of erroneous behavior that permanent and transient faults induce.
Transient faults

- Transient faults are the main error sources in VLSI IC - significant influence of environment conditions.

- Cosmic rays and alpha particles, smaller noise margins, crosstalk, power supply noise, etc. result in unacceptable soft error rate in ICs.
From fault to failure

- Fault occurrence
- Fault latency
- Error latency
- Maximal latency for FT mechanism
- Failure
- Error
- Fault influence
- Error detection
- Error activation
- Failure activation
- Time
In the FT approach, effects of faults are countered by incorporating various forms of redundancy. This is enabled by:

- Increased package density
- Reduced IC energy consumption
- Reduced IC fabrication cost

**Redundancy** - additional resources

- **Hardware** (duplication, TMR),
- **Information** (the use of coding technique – parity, Berger, Bose-Lin codes),
- **Software** (N-version programming, a number of independently written programs for a given function are run simultaneously),
- **Time** (the use of the same hardware repeatedly in time for the same inputs and comparing the results).
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Self-checking circuits

- Digital circuit (chip, board, system) has a self-checking property if it is able to detect internal faults immediately after its occurrence, without using external test signals.

- Information redundancy, which require an additional hardware, is introduced
  - The outputs of a functional block must satisfy a specific feature.
Concurrent error detecting

SC is an approach for concurrent error detection (CED) that allows:

- detection of both temporary and permanent faults;
- early detection of errors so that corrective action can be initiated before data corruption;
- easy identification of the field replicable unit (e.g. a chip or a board).

The use of a CED scheme in order to achieve the high reliability requirement of modern systems is becoming an important design technique.
Error detecting codes

- Design of SC logic circuits is based on error-detecting codes.
- Error-detecting codes - specific presentation of symbols which enables error detection inside codeword.

We use separable codes where codewords are constructed by appending check bits to the normal output bits.

The most frequently used error-detecting codes are:

- **Parity code** – the simplest separated code by which single bit errors can be detected.
- **Berger code** - optimal separated code by which single bit errors and unidirectional errors can be detected inside codeword.
- **Bose-Lin code** – code for detection of $t$ unidirectional errors ($t$-unidirectional error detecting codes, $t$-UED).
Characteristics of SC circuits

- The **Self-Testing** – ST property ensure that for every possible fault, there is at least one input code word for which the resulting output is a non-code word.

- The **Fault-Secure** – FS property ensure that for any possible fault, the circuit output is either a correct code word or a non-code word.

- A circuit is **totally self-checking** if it is both ST and FS.
A checker verifies that no fault has occurred by checking whether the output data belongs to the adopted code.

Structure of the checker circuit depends on error detection code.

A totally self-checking checker must have two outputs.

The two-rail checker has two groups of inputs and two outputs.
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
  - Self-checking network implementation
  - Partially self-checking circuits design
- Conclusion
Self-checking circuit design

- Design method is based on VHDL
  - It enables implementation of combination and sequential self-checking circuit at RTL
  - It defines template in the form VHDL code.

- The methodology provides:
  - library of pre-designed, parameterized concurrent error-detection (CED) modules
  - VHDL code template that allows easily creation of synthesizable high-level description of the SC circuit.
General structure of method for SC circuit design
The error signal reports the presence of fault if:

- the primary and complementary primary outputs configurations differ from each other,
- a fault affects one of the self-checking TRC checkers.
Berger and Bose-Lin checking scheme

The Berger check symbol of the information is defined as either:
- the binary representation of the number of zeros in the information ($B_0$)
- the ones complement of the number of ones in the information ($B_1$)

For $I$ information bits there is a need for at least $\log_2(I+1)$ check bits.

The Berger coding design technique utilizes a set of full-adder modules providing summation of the information bits.

Bose-Lin codes are similar to Berger codes and require a fixed number of check bits.
Berger and Bose-Lin checking scheme
Parity-check codes

Block diagram of self-checking circuit that combines four pairs of TSC outputs into a single pair error signal.
Design methodology

The proposed VHDL-based methodology for synthesis of SC FSM consists of three steps:

1. Creation of an initial FSM description in a form of behavior-level synthesizable VHDL code. (Designer have to use separate VHDL processes for next-state and output logic.)

2. Creation of SC FSM VHDL description by copying the next-state and output-logic code segments from initial description into a pre-designed VHDL code template and setting up several high-level design parameters.

3. Synthesis - using commercial VHDL CAD tools.
Self-checking FSM

Basic architecture  Self-checking architecture

FSM with CED
The designer is allowed to use the same or different coding scheme for the output and next-state logic.
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Design flow

- Design entry
- Simulation
- Synthesis
- Implementation
- Programming
- Conditions

- Library

- Design entry
- Simulation
- Method selection
- Synthesis
- Cost define
- Proposed cost
- Cost comparison

- Implementation
- Programming
The goal of the proposed procedure is to select the VHDL code that will require the least area to implement a SC principle at maximal operating frequency.

The area of the circuit is equal to the sum of the area of the original function logic, CED logic, and checker.

The area required by the original function logic depends on how much logic sharing is possible.

The area required by the CED logic depends on the size of the checking function that must be implemented for each code.

The area required by the checker depends on how many checking groups are there.
Implementation of SC technique on Spartan2 FPGA family

<table>
<thead>
<tr>
<th></th>
<th>original circuit</th>
<th>duplication</th>
<th>Berger</th>
<th>Bose-Lin</th>
<th>parity</th>
<th>2-b gr. parity</th>
<th>4-b gr. parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-segm. display</td>
<td>#slices</td>
<td>66</td>
<td>179</td>
<td>137</td>
<td>136</td>
<td>165</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>max.dly (ns)</td>
<td>22.2</td>
<td>43.7</td>
<td>44.4</td>
<td>41.9</td>
<td>27.8</td>
<td>34.6</td>
</tr>
<tr>
<td>priority encod.</td>
<td>#slices</td>
<td>13</td>
<td>28</td>
<td>30</td>
<td>22</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>16-u-4</td>
<td>max.dly (ns)</td>
<td>14.6</td>
<td>23.5</td>
<td>20.8</td>
<td>18.9</td>
<td>16.5</td>
<td>17.7</td>
</tr>
<tr>
<td>TAP control.</td>
<td>#slices</td>
<td>9</td>
<td>30</td>
<td>64</td>
<td>55</td>
<td>16</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>max.dly (ns)</td>
<td>15.1</td>
<td>24.8</td>
<td>32.1</td>
<td>29.0</td>
<td>22.4</td>
<td>28.5</td>
</tr>
</tbody>
</table>

The parity-check scheme is a superior one for FPGA technology, concerning both area overhead and speed decreasing.
### Implementation of SC technique on Spartan2 FPGA family

<table>
<thead>
<tr>
<th>FSM name</th>
<th>Input</th>
<th>Output</th>
<th>State</th>
<th>Area (#slices)</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b01</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>b02</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>BlackJack</td>
<td>7</td>
<td>8</td>
<td>16</td>
<td>12</td>
<td>25</td>
</tr>
<tr>
<td>CPUCont.</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>TAP</td>
<td>2</td>
<td>9</td>
<td>16</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

In general, all CED schemes involve significant area overhead as well as speed decreasing. There is no single scheme that is superior to others in all cases.
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Partially self-checking circuits design

- Previous mentioned technique for SC circuit design are focused on critical systems where very high levels of dependability are required.

- As CED becomes increasingly important in commercial electronics, there is a need for more cost-effective techniques.

- In order to make a good compromise between the hardware overhead (<100%) and error-detecting potential (<100%), we introduce the concept of partial function checking.
Partial self-checking method

The duplication with comparison (DWC) scheme.

The FC implements characteristic function, $F$, of the original function $f$ defined by $F(X,Y)=0$ if $Y=f(X)$, and $F(X,Y)=1$ when $Y\neq f(X)$. 

- standard representation -

- alternative representation - 

the DFM and the MBC, are implemented as a single module FC.
Partial function checking

- Given a logic function $Y = f(X)$ with characteristic function $F(X, Y)$, a partial function checker (PFC) in respect to $f$ is a circuit that implements function $F^*(X, Y)$, where $F^*$ represents an under-approximation of $F$.

- From one hand, the function $F^*$ under-approximates $F$ if the output of $F^*(X, Y)$ agrees with $F(X, Y)$ whenever $F(X, Y) = 0$.

- From the other hand, the bit values for $F^*(X, Y)$ when $F(X, Y) = 1$ can be arbitrary selected with a goal to reduce the complexity of $F^*$;

- The goal is to obtain as much as greater (<100%) error coverage with minimal hardware overhead (<100%).

- This under-approximation can be denoted as $F^* \subseteq F$. 
Outline

- Introduction
- Reliable systems
- Concurrent error detection
- Self-checking logic network design
- Self-checking network implementation
- Partially self-checking circuits design
- Conclusion
Conclusion

Four schemes for synthesis of totally self-checking combinational circuits based on duplication, parity-check codes, Berger codes and Bose-Lin codes derived from a VHDL description are presented.

The proposed methodology defines flexible and parameterized code template that combines pre-designed and pre-verified CED modules with original circuit processes into the self-checking structure according to the selected CED coding scheme.

Results concerning area overhead and speed decreasing when the SC circuit are implemented in FPGA technology are presented.
Conclusion

- Insertion of the concurrent error detection circuitry at the front-end of the synthesis process (at the RTL level rather than at the gate level) has the following advantages:
  - The synthesis tool can take the error detection circuitry into account when satisfying timing constraints (as well as other constraints, related to power consumption, testability, etc.)
  - Inserting the error detection circuitry at the RTL level can be easily and seamlessly incorporated into the standard design flow.

- There is no single technique that provides the best solution in all cases. Therefore, the optimal choice may be made only after the implementation and evaluation of all available techniques.