FPGA Implementation of an Asynchronous Processor with Both Online and Offline Testing Capabilities

Nikolaos Minas
Matthew Marshall
Gordon Russell
Alex Yakovlev
Outline

- Introduction.
- Error Detection/Correction overview.
- Information Redundancy Scheme.
- Dong’s Code.
- CED Pipeline.
- Asynchronous Reconfigurable Tester.
- Results.
- Conclusions.
Introduction

- Technological advances reduce reliability of components due to:
  - Process variation
  - Reduction in power supply voltages
  - High operating frequencies

- These factors increase the occurrence of transient and intermittent faults.
Intermittent and Transient Fault characteristics

**Intermittent**
- Poor fabrication.
- Process Variation.
- Occur repeatedly at a give location.
- Errors occur in bursts once activated.

**Transient**
- Alpha or neuron particles.
- Power supply transients.
- Interconnect noise.
- EMI
- Random and short duration.
Error Detection/Correction Overview

General Architecture of CED Scheme

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Time</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Area</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Power</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Input

Operation

Characteristic Prediction

Comparison

Output

Error
Information Redundancy Schemes

- Check bits are attached to the data bits to form a code word.
- For all input combinations only a subset represents valid information.
- In Berger code the number of check bits is a function of the data bits.
- In Dong’s code the number of check bits are a function of error coverage.
The completed Check Symbol is made of two parts

*C1* is a count of the zeroes within the data word, modulo \( (m+1) \)

\( (‘m’ \text{ is the maximum weight of unidirectional errors to be detected by the code }) \)

*C2* is a count of the number of zeroes in *C1*.

**Completed codeword is - Data word||C1||C2.**

\[ C1 = \log_2(m+1) \]

As a result, check bits are not a function of the data word.
- No single code can detect both:
  - data processing errors.
  - data transfer errors.

Consequently the technique of Check Symbol Prediction is used.
Pipeline Processor

- To demonstrate the applicability of Dong’s Code, a 32-bit asynchronous RISC based processor was implemented.

- The processor has a repertoire of 32 instructions related to:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU Operation</td>
<td>18</td>
</tr>
<tr>
<td>Program Flow</td>
<td>9</td>
</tr>
<tr>
<td>Memory Access</td>
<td>2</td>
</tr>
<tr>
<td>System set Op.</td>
<td>6</td>
</tr>
</tbody>
</table>
CED Pipeline Architecture

- **Fetch Instruction**
- **Decode**
- **Register file**
- **Check Symbol Generator (CSG)**
- **Check Symbol Prediction (CSP)**
- **Execute in ALU**
- **Check**
- **Writeback to register**

- Registers Required Values Check symbols
- Check Symbol
- ALU output
- Check Symbol
- Value
- Error No error
- Error signal
- Value & Check Symbol
Automatic Test Equipment (ATE) are not capable of fully testing Asynchronous circuits because of the absence of a global clock.

The physical cost of testing can be reduced by using FPGAs as an embedded test platform.
- FIFO stages have been designed using a GALS approach to take advantage of the FPGA hardware resources.
- Asynchronous communication was achieved using controllers to generate the Request (Req), Acknowledge (Ack) and Enable signals.
Error Mapping

Fault-Free Output
Results – Error Detection

Operand Error

Opcode Error
Results- Power consumption and area overheads

- If direct comparison is to be made between different processor design styles it is essential that they have a common:
  - Architecture.
  - Instruction Set.
  - Technology.

- To this end 4 designs of an identical processor architecture were undertaken, that is,
  - Synchronous processor with/without CED.
  - Asynchronous processor with/without CED
Results – Power Dissipation ASIC
Results – Power Dissipation FPGA

![Graph showing power dissipation for different FPGA architectures.]

- Async
- Async CED
- Sync
- Sync CED

The graph compares the power dissipation in milliwatts (mW) across different FPGA architectures. Sync CED has the highest power dissipation, followed by Sync, Async CED, and Async.
Results – Area Overhead

ASIC

Sync CED 17%
Async CED 13%
Sync 0%
Async -4%

FPGA

Sync CED 26%
Async CED 20%
Sync 0%
Async -4%
The asynchronous CED processor and the asynchronous tester were implemented in a Virtex2-1000 FPGA from Xilinx.

The system utilised 57% of the total FPGA area.

The processor comprises 5375 LUTs and the tester 517 LUTs.
Asynchronous Circuit on FPGAs

**Problems**

- Timing closure
- Place and Route
- Delay Chains

**Solutions**

- Control Signals placed as clocks.
- Manual P&R.
- Use of carry chain gates to create predictable delays.
Conclusions

- 32-bit asynchronous RISC based processor with CED was designed in both ASIC and FPGA.
- Implementation of an asynchronous reconfigurable tester.
- Results showed that the asynchronous CED processor offers significant advantages over the synchronous equivalent, in area overheads and power consumption.

<table>
<thead>
<tr>
<th></th>
<th>ASIC</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>4%</td>
<td>6%</td>
</tr>
<tr>
<td>Power</td>
<td>25%</td>
<td>29%</td>
</tr>
</tbody>
</table>
Thank you!!
Any Questions?
Check Symbol Prediction Circuit

- XcYc generator
- AND/OR MUX
- Carry Generator
- MUX
- Zeros Counter
- Shift MUX
- Add/Sub
- 2’s complement
- Check Symbol

- X
- Y
- Logic/Arith/Mul
- Select
- Cin
- Xck
- Yck
- Mul
- Cout
- Mul Carries (Cc) from ALU
- Control
### Example of Dong’s Code

<table>
<thead>
<tr>
<th>Information Bits (I)</th>
<th>Number of Zeros in ‘I’</th>
<th>Zeros mod 8</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000 00000000 00000000 00000000</td>
<td>32</td>
<td>0</td>
<td>000</td>
<td>11</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000001</td>
<td>31</td>
<td>7</td>
<td>111</td>
<td>00</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000011</td>
<td>30</td>
<td>6</td>
<td>110</td>
<td>01</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000111</td>
<td>29</td>
<td>5</td>
<td>101</td>
<td>01</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00001111</td>
<td>28</td>
<td>4</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00011111</td>
<td>27</td>
<td>3</td>
<td>011</td>
<td>01</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00111111</td>
<td>26</td>
<td>2</td>
<td>010</td>
<td>10</td>
</tr>
<tr>
<td>00000000 00000000 00000000 01111111</td>
<td>25</td>
<td>1</td>
<td>001</td>
<td>10</td>
</tr>
<tr>
<td>00000000 00000000 00000000 11111111</td>
<td>24</td>
<td>0</td>
<td>000</td>
<td>11</td>
</tr>
</tbody>
</table>
## Error Coverage for Dong’s Code

<table>
<thead>
<tr>
<th>Information Bits</th>
<th>Value of ‘m’</th>
<th>Bits in C1</th>
<th>Error Coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>3</td>
<td>2</td>
<td>93.74</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>2</td>
<td>93.75</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>2</td>
<td>93.75</td>
</tr>
<tr>
<td>64</td>
<td>3</td>
<td>2</td>
<td>93.75</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>3</td>
<td>99.04</td>
</tr>
<tr>
<td>32</td>
<td>7</td>
<td>3</td>
<td>98.54</td>
</tr>
<tr>
<td>48</td>
<td>7</td>
<td>3</td>
<td>98.33</td>
</tr>
<tr>
<td>64</td>
<td>7</td>
<td>3</td>
<td>98.47</td>
</tr>
</tbody>
</table>

‘m’ is the maximum weight of unidirectional errors to be detected by the code
### Dong’s Code Error Detection Ability

<table>
<thead>
<tr>
<th>Type of error affecting the information bits</th>
<th>Type of error affecting the check bits</th>
<th>Number of errors detected by the code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unidirectional</strong> 1→0 OR 0→1</td>
<td><strong>Error free</strong></td>
<td><strong>Errors of weight ≠ (m+1) or multiples</strong></td>
</tr>
<tr>
<td><strong>Unidirectional</strong> 1→0 OR 0→1</td>
<td><strong>Unidirectional</strong> 1→0 OR 0→1</td>
<td><strong>All errors</strong></td>
</tr>
<tr>
<td><strong>Bi-directional</strong> 1→0 AND 0→1</td>
<td><strong>Unidirectional</strong> 1→0 OR 0→1</td>
<td><strong>All errors</strong></td>
</tr>
</tbody>
</table>
Area Overheads

Asynchronous vs. Synchronous CED Area

CED Code

- NO CED
- Berger Code
- Dong's Code

Area (x10^3 μm^2)

- Asynchronous
- Synchronous