Dynamic Partial Reconfiguration Approach to the Design of Sustainable Edge Detectors

Ronald F. DeMara, Jooheung Lee
Rawad Al-Haddad, Rashad Oreifej, Rizwan Ashraf
University of Central Florida
Orlando, FL

Brian Stensrud and Michael Quist
Soar Technology, Inc.
Orlando, FL

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract #W31P4Q-08-C-0168
Introduction

- **Overall Design Objectives:**
  - Mission-critical applications in harsh operating environments
  - Sustainable design to increase reliability and application life-span
  - Autonomous operation and adapted to situation at hand
  - Self-recovery using flexible and dynamic fault tolerance strategy
  - Software control and monitoring of the hardware system

- **Use Case:** Sobel-Edge Detection (Image-Processing Application)

- **Reconfigurable Adaptive Redundancy System (RARS):**
  - Dynamically adapts to operational environment by changing topology
  - Maximizes system performance based on run-time criteria
  - Self-aware of component status to reconfigure fabric when needed
  - Interfaced with PC-based monitoring system using JTAG to perform higher-level control and management
  - implemented on Xilinx Virtex-4 (XCV4SX35) device
Related Work

- **Evolutionary Algorithms and Evolvable Hardware:** Evolution of *fault-tolerant electronic circuits* on devices such as FPGAs [12], [16] extensive, also FPTAs.
- **Image Processing and Evolution:** Tyrrell [15] and Ross [20] used genetic programming with *software model-based* fitness evaluation to evolve edge detectors.
- **Multilayer Runtime Reconfiguration Architecture (MRRA):** A software framework [16] capable of communicating with the FPGA through high level API calls to perform *direct bitstream manipulation*.

- **Fault Detection:**
  - ✓ Modular redundancy
  - CBS readback & compare
  - Concurrent Error Detection
  - BIST

- **Fault Repair:**
  - Blind Scrubbing
  - TMR with recovery
  - Reconfiguration (A-priori Synthesized Allocations)
  - Roving STARs (Online BIST)
  - ✓ Evolutionary Techniques
System Design

Software Monitoring and Control
- Graphical User Interface Monitor
- Communication Controller
- Refurbishment Manager

Software Hardware

System Level Schematic
Autonomic Element (AE): Autonomous and application-independent

Functional Element (FE): Any application can be implemented that provides specified status signals for monitoring and control for reconfiguration
RARS Supported Modes

**Simplex**: One FE is functional

**Duplex**: Two FEs are functional with discrepancy detection

**Hybrid mode**: Temporal combination of the other modes

**TMR**: All three FEs are functional with voter enabled
• **Functional overview:**
  - *Performance Manager (PM)* compares the high-level throughput against system requirement
  - PM monitors status of each FE, configuration of AE and the overall performance level of system
  - Implements special-purpose communication protocol
  - Carries messages through the JTAG interface to the hardware via the General-purpose Native JTAG Tester (GNAT)
  - Serves two purposes:
    1. **Provide the monitoring module with graphical interface (Java applet)**
    2. **Enable higher-level fault-detection and recovery techniques (GA)**

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE_STATUS_REQUEST</td>
<td>Request status of a certain FE</td>
</tr>
<tr>
<td>AE_STATUS_REQUEST</td>
<td>Request status of a certain AE</td>
</tr>
<tr>
<td>PERFORMANCE_REQUEST</td>
<td>Request performance value</td>
</tr>
<tr>
<td>FE_STATUS_REPORT</td>
<td>Report certain FE status (online healthy, online faulty, offline, etc…)</td>
</tr>
<tr>
<td>AE_STATUS_REPORT</td>
<td>Report certain AE status (Duplex, Voter)</td>
</tr>
<tr>
<td>PERFORMANCE_REPORT</td>
<td>Report performance value</td>
</tr>
</tbody>
</table>

Table: Software to hardware messages

Table: Hardware to software messages
Dynamic Partial Reconfiguration

- Early Access Partial Reconfiguration (EAPR) design flow was used to achieve dynamic partial reconfiguration
- Reduce repair time due to small bitstream size (PR configures the FE in in 1.8% of the time required to reconfigure whole system)
- Facilitates repair while system is kept online

PRR’s FE1, FE2 and FE3, and full design respectively

<table>
<thead>
<tr>
<th>Approach</th>
<th>Virtex-4 Full</th>
<th>Virtex-4 Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Virtex-4</td>
<td>Virtex-4</td>
</tr>
<tr>
<td>Bitstream Size</td>
<td>1.633 MB</td>
<td>30.61 KB</td>
</tr>
<tr>
<td>JTAG Cable</td>
<td>parallel cable IV 5Mbps</td>
<td>parallel cable IV 5Mbps</td>
</tr>
<tr>
<td>Config time (msec)</td>
<td>2613</td>
<td>48</td>
</tr>
</tbody>
</table>

Partial Vs Full bitstream Comparison
Evolutionary Repair Strategies

- Genetic Algorithms: based on principles of natural selection.
- Genetic operators such as *mutation* and *crossover* used to search within a large irregular complex solution space, especially multi-objective optimization.

### Extrinsic Evolution

- Genetic Algorithm
- Simulation in the loop
- Software model
- Done? Build it

### Intrinsic Evolution

- Genetic Algorithm
- Hardware in the loop

- Fitness function is measured out of the physical device output
- Constraints imposed by the device’s internal structure
- Demonstrates that the resultant design will actually fit on to the implementation platform

- Functional models that abstract the physical aspects of the real device
- Representation has to undergo placement and routing before implementation.
Monitoring & Control Components

- **GA Engine**: C++ application implements a customizable standard GA and is also able to directly read the fitness value from hardware.

- **Chromosome Manipulator**: Hardware abstraction from the GA engine perspective.

- **Multi Runtime Reconfiguration Architecture (MRRA)**: partial bitstream manipulation and decoding, modular architecture provides the logic, translation, and reconfiguration layers which facilitates communication with the FPGA.

- **JTAG (IEEE 1149.1)**: serial port on hardware side, also used for dynamic partial reconfiguration purposes.

- **General-purpose Native JTAG tester (GNAT)**: configured on the device to support Input/Output operations with the user implemented circuit.
System Repair Cycle

Virtex-4

Duplex
“no fault”
100% Throughput

fault

AE detected
faulty FEx
Trigger
watchdog timer

Triplex
“one faulty FE”
100% Throughput

Discrepancy?
yes

no

no fault

Transient fault
cleared

no

Watchdog expired?
yes

no

Initiate repair
request to host

yes

no

Triplex
“FEx under repair”
100% Throughput

Host PC

GA Process

Download
Individual to
FPGA via JtaG
Partial
Reconfig

Read fitness
JtaG-GNAT
communication

yes

no

no

Repaired?
Edge Detection Use Case

PC

Monitor

PC

Original Video Stream

Edge-Detected Video Stream

VGA

VGA

VGA-IN

VGA-OUT

ARS

FE-1

FE-2

FE-3

AE

GNAT

Control/Status

Parallel Port

JTAG

Software Layer

Virtex-4 FPGA

FPGA Board

Control
Physical Design of the Use-Case on the Xilinx VSK platform

ML402 MOTHERBOARD

PowerPC
BlockRAM
Config
ICAP
Compact Flash
Terminal Program
UART

VGA-In
AD9887 IC
Input Image Buffer (BRAM) on XCV2P7 FPGA

Config:
- start
- length
- addrA, dataA, weA
- addrB, dataB
- ce_ICAP, we_ICAP, busy_ICAP, Out_ICAP

ICAP:
- clock
- reset

PLB BUS

VGA-Out
AE

Output Image Buffer (BRAM) on XCV4SX35 FPGA

64-bit VIOBUS

VIDEO IO DAUGHTER CARD

XCV4SX35 FPGA

ML402 MOTHERBOARD

VGA-In
Video Input

AD9887 IC
Input Image Buffer (BRAM) on XCV2P7 FPGA

64-bit VIOBUS

XCV4SX35 FPGA

Video Output
Results

- Normal fault-free output
- Degraded output
- Refurbished 100% performance output
GA-Based Repair Results

- GA operated on 8 LUTs: Critical LUT’s that are highly influential on the performance of the Edge Detector
- 5 runs. The maximum fitness is 2048, which means that out of 2048 discrepancy reading samples, the correct solution does not show any discrepancy when its output is compared to a completely healthy Sobel edge detector.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Generic images (fairly simple)</td>
<td>Unfragmented localized thin edges in medical images.</td>
<td>Microscopic images from mineral samples</td>
<td>Generic (satellite images, uniform patterns, etc...)</td>
</tr>
<tr>
<td><strong>Methodology</strong></td>
<td>Exploiting inherent parallelism in images</td>
<td>Split image into linked sub-images. Maintain links between adjacent pixels</td>
<td>Training stage (requires sampling 23.6% of image), followed by Genetic programming.</td>
<td>Evolving a subset of the Edge Detector (critical LUTs) in order to recover from faults.</td>
</tr>
<tr>
<td><strong>Fitness Evaluation</strong></td>
<td>Software model</td>
<td>Software model</td>
<td>Software model</td>
<td>Intrinsic Evolution (HW in the loop)</td>
</tr>
<tr>
<td><strong>Evolutionary Algorithm</strong></td>
<td>Genetic Programming</td>
<td>2D Genetic Algorithm problem-specific operators.</td>
<td>Genetic Programming Training stage (~25%) Evolution (~75%)</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td><strong>Genetic String Coding</strong></td>
<td>Four node functions (and, or, not, xor), 8 terminal values for pixels around the evolved one</td>
<td>Edge Map: image pixels are masked with corresponding values in pixel map (0: not edge, 1: edge)</td>
<td>High-level functions (avg, min, max, stdev) Terminal Pixels and high-level ephemeral (gradient, intensity)</td>
<td>Direct Bitstream Evolution. The solution coding is the actual bitfile.</td>
</tr>
<tr>
<td><strong>Fitness Function</strong></td>
<td>Pratt Figure of Merit (PFM) relative to Sobel edge-detector F = 1/(1+Pef + Pnf)</td>
<td>Highly complex cost function based on 5 cost factors</td>
<td>Biased random sampling fitness evaluation for training. Program fitness is similar to PFM.</td>
<td>Model-Free triplex discrepancy based. No application specific a-priori knowledge needed.</td>
</tr>
<tr>
<td><strong>Evolution Speed</strong></td>
<td>Partial solution in 2333 generations (24 hours of evolution time)</td>
<td>2300 generations for rings image. 300 generations for thin, well-localized edges</td>
<td>75 generations. 25% of the images for training, Very large population size of 2000</td>
<td>361 generations, low population size of 10 on Lena benchmark. 8 critical LUTs evolved.</td>
</tr>
<tr>
<td><strong>Best Fitness</strong></td>
<td>Not reported</td>
<td>0.85 PFM with scaling factor of 0.01.</td>
<td>0.590 for Image 1 0.633 for Image 2</td>
<td>100% compared to a pristine Sobel edge detector output.</td>
</tr>
</tbody>
</table>


