Lifetime-aware Design Methodology for Dynamic Partially Reconfigurable Systems

Presented by: Siva Satyendra Sahoo

Siva Satyendra Sahoo, Dr. Bharadwaj Veeravalli
Department of Electrical and Computer Engineering, National University of Singapore
satyendra@u.nus.edu, elebv@nus.edu.sg

Dr. Tuan D.A. Nguyen, Dr. Akash Kumar
Center for Advancing Electronics Design, Technische Universitat, Dresden
tuan_duy_anh.nguyen1,akash.kumar@tu-dresden.de
Outline

- Motivation
- Dynamic Partial Reconfiguration (DPR):
  - Background
  - Features
  - Aging mitigation
- System model
- System Design methodology
- Experiment and Results
- Conclusion
Outline

- Motivation
  - Dynamic Partial Reconfiguration:
    - Background
    - Features
    - Aging mitigation
  - System model
  - System Design methodology
  - Experiment and Results
  - Conclusion
Increasing fault-rates

Physical faults
- Intermittent faults
- Permanent faults

Mechanisms:
- Hot carrier injection
- Electromigration
- Gate-oxide Breakdown
- Hard Dielectric Breakdown
- Negative Bias
- Temperature Instability
- Manufacturing defects
- Insufficient voltage scaling
- High Power Density
- Increased Variability
- Higher Temperature
- Faster aging

Transistor Scaling

Increasing Fault Rate

Increasing fault-rates

Physical faults

★ Intermittent faults
★ Permanent faults

Mechanisms:

- Hot carrier injection
- Gate-oxide Breakdown
- Electromigration
- Increasing fault-rates

Reduced System Lifetime

Transistor Scaling
- Manufacturing defects
- Increased Variability

Insufficient voltage scaling

High Power Density
- Higher Temperature
- Faster aging

Insufficient voltage scaling

Increased Fault Rate

System-level effect

- Mission failures
- Reduced safety in critical systems
  - Power plants, transportation, medical etc.
- Reduced product lifetime
Run-time Resource Sharing

Meeting increasing computation demands:
  • Parallelism
  • Custom Computing
    • *Hardware Accelerators*
Run-time Resource Sharing

Meeting increasing computation demands:
- Parallelism
- Custom Computing
  - *Hardware Accelerators*

*Finite* computation resources !!

*Time-sharing* of computing resources:
- Cost-efficient *parallel* systems

http://www.responsiblelending.org/
http://www.chip-architect.com/
Run-time Resource Sharing

 Meeting increasing computation demands:
  • Parallelism
  • Custom Computing
    • Hardware Accelerators

Finite computation resources !!

 Time-sharing of computing resources:
  • Cost-efficient parallel systems
  • Multi-processor and/or Multi-core SoCs:
    • Multiple applications sharing a number of instruction-set processor pipeline
Run-time Resource Sharing

Meeting increasing computation demands:

- Parallelism
- Custom Computing
  - *Hardware Accelerators*

*Finite* computation resources !!

*Time-sharing* of computing resources:

- Cost-efficient *parallel* systems
- Multi-processor and/or Multi-core SoCs:
  - Multiple applications sharing a number of *instruction-set processor pipeline*
- FPGAs:
  - Multiple hardware accelerators sharing *reconfigurable hardware*
  - *Parallel + “Custom”*
Dynamic Partial Reconfiguration (DPR)

- Partially Reconfigurable Modules (PRMs)
- Partially Reconfigurable Regions (PRRs)
Dynamic Partial Reconfiguration (DPR)

- Partially Reconfigurable Modules (PRMs)
- Partially Reconfigurable Regions (PRRs)

![Diagram showing PR Region (PRR) and PR Module (PRM) with execution trace and modules M₁, M₂, M₃, M₄.](image)
Outline

- Motivation
- Dynamic Partial Reconfiguration:
  - Background
  - Features
  - Aging mitigation
- System model
- System Design methodology
- Experiment and Results
- Conclusion
PRM-PRR compatibility

Affects:
- PRRs’ size
- #PRRs (available parallelism)
- Bitstreams storage

<table>
<thead>
<tr>
<th>PRRs</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRMs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_1$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$M_2$</td>
<td>✓</td>
<td>✘</td>
</tr>
<tr>
<td>$M_3$</td>
<td>✘</td>
<td>✓</td>
</tr>
<tr>
<td>$M_4$</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

PRM-PRR Compatibility
Scheduling PRMs on PRRs

Deadlines

- Latency *(Timing Reliability)*
**Scheduling PRMs on PRRs**

- **Deadlines**
  - *Latency (Timing Reliability)*

- **Aging**
  - *System MTTF (Lifetime Reliability)*

![Execution Trace]

![Non-uniform Aging]

![System MTTF]
System-level Spatial Redundancy

Number of available PRRs
- Increased available parallelism
- Net aging reduced
DPR and System Lifetime

- PRM-PRR compatibility
- Scheduling: Deadlines and Aging
- System-level Spatial Redundancy

Tools to improve the system MTTF
Outline

- Motivation

- Dynamic Partial Reconfiguration:
  - Background
  - Features
  - Aging mitigation

- System model
- System Design methodology
- Experiment and Results
- Conclusion
System MTTF-aware Scheduling

- **Aim:** Reduce aging of each PRR
- **Constraints:**
  - Execution latency
System MTTF-aware system partitioning

Homogeneous v/s Heterogeneous PRRs

Effects:

- Maximum #PRRs
- Aging of each PRR
Outline

- Motivation
- Dynamic Partial Reconfiguration:
  - Background
  - Features
  - Aging mitigation
- System model
- System Design methodology
- Experiment and Results
- Conclusion
System model: Application

Task-graph

Parameters for problem formulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskID</td>
<td>Serial number of task</td>
</tr>
<tr>
<td>TaskType</td>
<td>Type of PRM used</td>
</tr>
<tr>
<td>StartT</td>
<td>Start time of task</td>
</tr>
<tr>
<td>ExecT</td>
<td>Expected execution time</td>
</tr>
<tr>
<td>EndT</td>
<td>End time of task</td>
</tr>
<tr>
<td>TaskCLBs</td>
<td>CLBs used for PRM implementation</td>
</tr>
<tr>
<td>TaskBRAMs</td>
<td>BRAMs used for PRM implementation</td>
</tr>
<tr>
<td>TaskDSPs</td>
<td>DSPs used for PRM implementation</td>
</tr>
<tr>
<td>TaskMTTF</td>
<td>Expected MTTF of the task PRM</td>
</tr>
<tr>
<td>TaskD</td>
<td>Any soft/hard deadline of the task</td>
</tr>
</tbody>
</table>

Application Task-graph

Task-level parameters
System model: Architecture [Nguyen2014]

PR-HMPSoC: PRRs and Static components

NoC-based system

**System model: Reliability** [Xiang2010]

\[
R(t) = e^{-(t/\eta)^\beta}
\]

\[
MTTF = \eta \times \Gamma (1 + 1/\beta)
\]

\[
\eta_{eff} = \frac{\sum \Delta t_i}{\sum \frac{\Delta t_i}{\eta_i}}
\]

\[
t = \sum \Delta t_i
\]

\[
\eta_i = \frac{MTTF_i}{\Gamma (1 + 1/\beta)}
\]

\[
prrMTTF_r = \frac{P_{app}}{\sum_{i=1}^{M} \frac{ExecT_i}{TaskMTTF_i}}
\]

\[
SysMTTF = \min_{all PRRs} (prrMTTF_r)
\]

Motivation

Dynamic Partial Reconfiguration:
- Background
- Features
- Aging mitigation

System model

System Design methodology

Experiment and Results

Conclusion
System Design Methodology

- Application Task-graph
- PRM Description
- System Architecture

Execution Trace Generation

PRM Characterization

MILP Formulation and Optimization

DPR Resource estimation

Floorplanning

Increase number of DPRs

DSE for DPR-based system design

Dynamic Partially Reconfigurable System
System Design Methodology

Application Task-graph

Execution Trace Generation

- **Greedy approach**
- **Execution trace**
- **Assume infinite resources**

1. DPR Resource estimation
2. PRM Characterization
3. System Architecture
4. ILP Formulation and Optimization
5. Floorplanning

DSE for DPR-based system design

Dynamic Partially Reconfigurable System

Increase number of PRRs
System Design Methodology

1. DPR Resource estimation
2. ILP Formulation and Optimization
3. Execution Trace Generation
4. PRM Characterization
5. Floorplanning

Dynamic Partially Reconfigurable System

- Determine
  - Resource requirements
  - Aging effect
- Power and max junction temp.
- Electromigration-related wear-out
System Design Methodology

- Considers the static resource requirements of the system
- Used PRFLoor for estimation

Application Task-graph

4. ILP Formulation and Optimization

5. Floorplanning

DPR Resource estimation

System Architecture

### System Design Methodology

- **Objectives:**
  - Minimize *makespan*
  - Maximize *SysMTTF*

- **Constraints:**
  - *Deadline*
  - Task-level *precedence*
  - *PRR*-level MTTF
  - *Resource* constraints
System Design Methodology

- Automatic Floorplanner
- PRFloor

Application Task-graph

PRM Description

System Architecture

3. Execution Trace Generation

4. Optimization

Floorplanning

Increase number of PRRs

DSE for DPR-based system design

Dynamic Partially Reconfigurable System

System Design Methodology

1. Application Task-graph
2. PRM Description
3. System Architecture

3. Execution Trace Generation
   - MILP Formulation and Optimization
   - DPR Resource estimation
   - Dynamic Partially Reconfigurable System

2. PRM Characterization
   - Increase number of PRRs
   - DSE for DPR-based system design

1. Application Task-graph
Outline

- Motivation
- Dynamic Partial Reconfiguration:
  - Background
  - Features
  - Aging mitigation
- System model
- System Design methodology
- Experiment and Results
- Conclusion
Experiments and Results

Experiment Setup:

- Two CPUs: Intel Xeon E5-2609 v2 @ 2.50GHz (quad-core), 32 GB of memory
- Ubuntu 14.04 LTS 64-bit
- Virtex-6 XC6VLX240T
- Gurobi Solver for finding MILP solution
- Task-graphs generated using TGFF

References:

Experiments and Results

 Experiment Setup:

- **IP Pool:**
  - 50 real-world hardware accelerators
  - Synthesized using Xilinx Vivado Suite (ver 16.2)

<table>
<thead>
<tr>
<th>PRM</th>
<th>LUT</th>
<th>BRAM</th>
<th>DSP</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFDIV</td>
<td>7309</td>
<td>1</td>
<td>24</td>
<td>CHStone</td>
</tr>
<tr>
<td>DFMUL</td>
<td>4051</td>
<td>1</td>
<td>16</td>
<td>CHStone</td>
</tr>
<tr>
<td>Log2</td>
<td>8212</td>
<td>0</td>
<td>0</td>
<td>EPFL</td>
</tr>
<tr>
<td>ADPCM</td>
<td>6222</td>
<td>6</td>
<td>126</td>
<td>OpenCores</td>
</tr>
<tr>
<td>FFT1024</td>
<td>19796</td>
<td>18</td>
<td>52</td>
<td>OpenCores</td>
</tr>
<tr>
<td>SHA</td>
<td>3069</td>
<td>20</td>
<td>0</td>
<td>OpenCores</td>
</tr>
<tr>
<td>JPEG</td>
<td>6581</td>
<td>11</td>
<td>10</td>
<td>OpenCores</td>
</tr>
<tr>
<td>Video Stream Scaler</td>
<td>524</td>
<td>2</td>
<td>11</td>
<td>Xilinx</td>
</tr>
<tr>
<td>Video Test Pattern</td>
<td>2543</td>
<td>3</td>
<td>12</td>
<td>Xilinx</td>
</tr>
<tr>
<td>Microblaze (Max Area)</td>
<td>5539</td>
<td>5</td>
<td>6</td>
<td>Xilinx</td>
</tr>
</tbody>
</table>

Some notable PRMs used in experiments.
Experiments and Results

Experiment Setup:
Optimization modes:
- homogeneous / heterogeneous
- Minimize makespan / Maximize system MTTF

Task-graph types:
- Parallelism: Fat / Slim
Experiments and Results

Results: *Fat* graphs

- System MTTF-aware scheduling: *homogeneous* PRRs

\[
sysMTTF: \text{Maximize system MTTF with deadline constraints}
\]
\[
makespan: \text{Minimize makespan with deadline constraints}
\]
Experiments and Results

Results: *Fat* graphs

- System MTTF-aware scheduling: *heterogeneous* PRRs

=sysMTTF*: Maximize system MTTF with deadline constraints

=makespan*: Minimize makespan with deadline constraints
Experiments and Results

Results: *Fat graphs*

- System MTTF-aware system partitioning: *homogeneous* v/s *heterogeneous*

Maximize system MTTF with deadline constraints using *homogeneous* PRRs

Maximize system MTTF with deadline constraints using *heterogeneous* PRRs
## Experiments and Results

### Results: Summary

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>T= 5</th>
<th>T=10</th>
<th>T=15</th>
<th>T=20</th>
<th>T=25</th>
<th>T=30</th>
<th>T=35</th>
<th>T=40</th>
<th>T=45</th>
<th>T=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fat, Large</td>
<td>0.00</td>
<td>0.21</td>
<td>0.82</td>
<td>0.75</td>
<td>1.52</td>
<td>1.37</td>
<td><strong>6.62</strong></td>
<td>7.96</td>
<td><strong>8.33</strong></td>
<td><strong>7.33</strong></td>
</tr>
<tr>
<td>Slim, Large</td>
<td>0.00</td>
<td>0.00</td>
<td>1.24</td>
<td>1.36</td>
<td>1.42</td>
<td>1.95</td>
<td><strong>9.57</strong></td>
<td>1.76</td>
<td><strong>13.16</strong></td>
<td>1.13</td>
</tr>
</tbody>
</table>

SysMTTF Improvements of Heterogeneous vs. Homogeneous Systems

\[
\frac{\text{sysMTTF}_{\text{hetero}} - \text{sysMTTF}_{\text{homo}}}{\text{sysMTTF}_{\text{homo}}}
\]
## Experiments and Results

### Results: Summary

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>T=5</th>
<th>T=10</th>
<th>T=15</th>
<th>T=20</th>
<th>T=25</th>
<th>T=30</th>
<th>T=35</th>
<th>T=40</th>
<th>T=45</th>
<th>T=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fat, Large</td>
<td>0.00</td>
<td>0.21</td>
<td>0.82</td>
<td>0.75</td>
<td>1.52</td>
<td>1.37</td>
<td>6.62</td>
<td>7.96</td>
<td>8.33</td>
<td>7.33</td>
</tr>
<tr>
<td>Slim, Large</td>
<td>0.00</td>
<td>0.00</td>
<td>1.24</td>
<td>1.36</td>
<td>1.42</td>
<td>1.95</td>
<td>9.57</td>
<td>1.76</td>
<td>13.16</td>
<td>1.13</td>
</tr>
<tr>
<td>Fat, Small</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.17</td>
<td>0.06</td>
<td>0.06</td>
<td>0.00</td>
</tr>
<tr>
<td>Slim, Small</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.05</td>
<td>0.11</td>
<td>0.00</td>
<td>0.00</td>
<td>0.08</td>
<td>0.00</td>
</tr>
</tbody>
</table>

SysMTTF Improvements of Heterogeneous vs. Homogeneous Systems

\[
\frac{\text{sysMTTF}_{\text{hetero}} - \text{sysMTTF}_{\text{homo}}}{\text{sysMTTF}_{\text{homo}}}
\]
Conclusion

- A design methodology for lifetime-aware DPR-based systems was proposed
  - Scheduling with *aging-estimation*
  - Integration of *resource constraints* into scheduler
- Investigated *homogeneous* v/s *heterogeneous* PRRs
- Investigate trade-off between *aging-related* and *externally-induced* permanent faults (*future work*)
- Use other *global* optimization methods (*future work*)
Thank You

💡 Queries ?
Experiments and Results

Results: Variation of System MTTF with #PRRs in a typical application with 25 tasks
Experiments and Results

Results: Variation of System MTTF with #PRRs in a typical application with 25 tasks

Infeasible for homogeneous PRRs
Experiments and Results

Results: Slim graphs
Experiments and Results

Results: Slim graphs

- System MTTF-aware scheduling: \textit{homogeneous} PRRs
Experiments and Results

Results: *Slim* graphs

- System MTTF-aware scheduling: *heterogeneous* PRRs
Experiments and Results

Results: *Slim* graphs

- System MTTF-aware system partitioning: *homogeneous* v/s *heterogeneous*