Routing Path Reuse Maximization for Efficient NV-FPGA Reconfiguration

Yuan Xue, Patrick Cronin, Chengmo Yang and Jingtong Hu

01/27/2016
Outline

• Introduction
  – NV-FPGA benefits and challenges
  – Routing optimization strategy

• Proposed Routing Reuse Optimizations

• Experimental Evaluation

• Conclusion
FPGA offers reconfigurability, flexibility, and low design cost to various embedded systems such as control, signal processing and many other applications areas.
Drawbacks of traditional FPGAs

Unfortunately, traditional **SRAM-based FPGAs** cannot meet increasing design requirements:

- Low scalability
- High leakage power
- Prone to soft error
- Volatile
Non volatile Memories (NVMs) use physical characteristics to represent logic states, such as:

- Phase Change Memory (PCM)
- Ferroelectric RAM (FRAM)
- Spin Transfer Torque Magnetic RAM (STT-MRAM)
Overcome SRAM limitations?

<table>
<thead>
<tr>
<th>NVM</th>
<th>SRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>• High density</td>
<td>Low scalability</td>
</tr>
<tr>
<td>• Near-zero leakage power</td>
<td>High leakage power</td>
</tr>
<tr>
<td>• Strong error resistance</td>
<td>Prone to soft error</td>
</tr>
<tr>
<td>• Near-zero power-on delay</td>
<td>Volatile</td>
</tr>
</tbody>
</table>

However, no rose is without a thorn!
Any issues or challenges?

**COMPARISON OF SRAM AND VARIOUS NVM CELLS**

<table>
<thead>
<tr>
<th>Type</th>
<th>Area (F²)</th>
<th>Read time(ns)</th>
<th>Write time(ns)</th>
<th>Write cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>140</td>
<td>0.2</td>
<td>0.2</td>
<td>$10^{16}$</td>
</tr>
<tr>
<td>PCM</td>
<td>4</td>
<td>12</td>
<td>100</td>
<td>$10^{9}$</td>
</tr>
<tr>
<td>STT-MRAM</td>
<td>42</td>
<td>35</td>
<td>35</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>NOR Flash</td>
<td>10</td>
<td>15</td>
<td>1000</td>
<td>$10^{5}$</td>
</tr>
</tbody>
</table>

*However, no rose is without a thorn!*

Two major issues of NVM FPGA:

- **Slow Writes** make the reconfiguration time non-trivial!
- **Short Endurance** limits device lifetime!
How to solve?

Basic scheme: reduce writes and increase reuse with a bit-level read-before-write (RBW) strategy

Why this works for NVM?

For PCM

Without RBW → 4 writes → 400ns
With RBW → 4 reads+ → 148ns 
1 write

Great Improvement!

For SRAM

Without RBW → 4 writes → 0.8ns
With RBW → 4 reads+ → 1ns 
1 write

No benefit!
**NV-FPGA**

**NV-FPGA:** Use NVMs as on-chip memories and configurable block units on FPGA.

**Main blocks include:**
- Configurable logic blocks (CLBs)
- Connection blocks (CBs)
- Switch box (SBs)

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>AREA</th>
<th>POWER</th>
<th>DELAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>90%</td>
<td>85%</td>
<td>80%</td>
</tr>
<tr>
<td>CLB+CB</td>
<td>10%</td>
<td>15%</td>
<td>20%</td>
</tr>
</tbody>
</table>

All can be NVMs!

We target **SBs** – the dominating blocks on the FPGA
Related work on routing optimization

Three categories:

Hierarchical routing match & preserve
• **Idea**: Construct cluster routing graph, match and preserve route hierarchically
• **Related work**: M.M Ozdal *ICCAD’09*, Ching-Yu Chin *ICCAD’14*

Coarse-grained partial reconfiguration
• **Idea**: Partition bitstream into dynamic and static parts, reuse static parts under partial reconfiguration framework.
• **Related work**: E. Vansteenkiste *FPL’12*, B. Al Farisi *FPL’13*

Incremental design routing reuse
• **Idea**: At engineering change order (ECO) stage, compare netlists to find possible reusable metal wire sections, and preserve these metal layer wires.
• **Related work**: Yun-Ru Wu *VLSI-DAT’10*, Hsi-An Chien *ASP-DAC’14*

Proposed work differs from them in:
- Single **Path** level routing optimization
- Fine-grained **bit level reuse**, require no partial reconfiguration support
- Can be applied to both highly-similar and **dissimilar** designs.
Overview

Our goal

Minimize bit-flips when reconfiguring on NVM-based FPGA

Strategy

Maximize bit-level reuse of switch boxes during routing

Questions to be addressed

How to model SB reconfiguration cost?
What types of flexibilities can be exploited to maximize reuse?
How to perform reuse-aware routing while preserving circuit timing?
Outline

• Introduction

• Proposed Routing Reuse Optimizations
  – Modeling routing reconfiguration cost
  – Identifying & Maximizing reusable path
  – Proposed reuse-aware routing algorithm
  – Proposed CAD flow

• Experimental Evaluation

• Conclusion
SB Reconfiguration cost

Wilton Switch type

<table>
<thead>
<tr>
<th>Switch</th>
<th>On/Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>l→a</td>
<td>1</td>
</tr>
<tr>
<td>j→b</td>
<td>1</td>
</tr>
<tr>
<td>c→e</td>
<td>1</td>
</tr>
<tr>
<td>h→f</td>
<td>1</td>
</tr>
<tr>
<td>k→i</td>
<td>0</td>
</tr>
</tbody>
</table>

3 changed switches

<table>
<thead>
<tr>
<th>Switch</th>
<th>On/Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>l→a</td>
<td>0</td>
</tr>
<tr>
<td>j→b</td>
<td>0</td>
</tr>
<tr>
<td>c→e</td>
<td>1</td>
</tr>
<tr>
<td>h→f</td>
<td>1</td>
</tr>
<tr>
<td>k→i</td>
<td>1</td>
</tr>
</tbody>
</table>

Connected switch: Reused switch: $R=2$

Existing configuration: $N_{old}=4$

New configuration: $N_{new}=3$

Single SB reconfig cost:

$$RR_{SB} = N_{old} + N_{new} - 2R$$

Total SB reconfig cost:

$$Cost_{reconfig} = \sum_{i=1}^{NUM_{SB}} RR_{SB_i}$$

Maximize reuse $R$

Min reconfig cost
Outline

• Introduction

• Proposed Routing Reuse Optimizations
  – Modeling routing reconfiguration cost
  – Identifying & Maximizing reusable path
  – Proposed reuse-aware routing algorithm
  – Proposed CAD flow

• Experimental Evaluation

• Conclusion
Path Definition and Characterization

Definition: Path is a single source-to-sink connection.

CBs locally connect CLBs to SBs, can be omitted in our structure model.
Path Definition and Characterization

**Definition:** Path is a single source-to-sink connection.

**Characterization:** Path $P=\{(i,j),(SB_{\text{first}}...,SB_{\text{last}})\}$.

$P$ consists of two sets:
- **CLB set:** $(i,j) \rightarrow$ starting point $\text{CLB}_i$, ending point $\text{CLB}_j$
- **SB set:** $(SB_{\text{first}}...,SB_{\text{last}}) \rightarrow$ all SBs that $P$ passes through

---

Path reuse could efficiently translate to SB reuse!
Two types of Reusable Paths

**Full reuse:** $P'$ shares both **starting and ending** CLBs with $P$.

**Partial reuse:** $P'$ shares **starting** CLB and **last SB** with $P$.
Reusable Path Maximization

However, the same starting CLB cannot guarantee path reuse. **Starting pin matters!**

LUT-to-BLE mapping will result in different path reuse and reconfiguration costs.

Suppose LUT1 $\rightarrow$ CLB1 and LUT2 $\rightarrow$ CLB3. If LUT1 $\rightarrow$ BLE1, path can be reused. Otherwise, path can not be reused.

Suppose LUT1 $\rightarrow$ CLB1 and LUT2 $\rightarrow$ CLB3. If LUT1 $\rightarrow$ BLE1, path can be reused. Otherwise, path can not be reused.
Reusable Path Maximization

To maximize path reuse, we exploit LUT-to-BLE mapping flexibilities. Assume $n$ BLEs in each CLB, the problem can be translated to **bipartite graph matching**.

Weight of edge $i \rightarrow j$ represents the number of reusable switches. For example, LUT1 $\rightarrow$ BLE1

$\text{Edge}_{11} = 2$

Optimal mapping can be identified with **Kuhn-Munkras Algorithm**.

Outline

• Introduction
• Proposed Routing Reuse Optimizations
  – Modeling routing reconfiguration cost
  – Identifying & Maximizing reusable path
  – Proposed reuse-aware routing algorithm
  – Proposed CAD flow
• Experimental Evaluation
• Conclusion
Basic routing:
Select routing resources to finish connection in each net.

- **Net:** *single* source to *multiple* sinks connection.
- Each net contains *several paths* with the *same starting point*.

### Proposed three-stage routing algorithm

- **Fix reusable paths in each net**
  - **Divide resource into two types:** *general* (used for all nets) and *dedicated* (used by specified net)
- **Route other paths**
  - Rank nets by their sink count, route nets with maximum sinks first
- **Relax reusable path with bad timing**
  - When relax, release related *dedicated* resources

Iteratively, exist if no congestion
Input: Netlist, Placement, Reusable paths, Relax threshold $\varepsilon$
Output: Routing results
Mark all routing resources $rr\_node\_type = \text{general}$;

Stage 1: fix reusable path

Sort (nets, unrouted sinks);
for (loop = 0; loop < Iteration\_limit; loop++)

Stage 2: route other paths

Perform timing analysis and update path criticality values;

Stage 3: relax related paths

if (no congestion exist)
Exit;

Repeat 2,3 if congesting
Outline

• Introduction

• Proposed Routing Reuse Optimizations
  – Modeling routing reconfiguration cost
  – Identifying & Maximizing reusable path
  – Proposed reuse-aware routing algorithm
    – Proposed CAD flow

• Experimental Evaluation

• Conclusion
Original CAD flow

1. Design Description (HDL)
2. Synthesis and technology map
3. Clustering and packing
4. CLB level placement
5. Routing
6. Estimate performance
Proposed CAD flow

New Design

- Design Description (HDL)
- Synthesis and technology map
- Clustering and packing
- CLB level placement

Existing Design

- Retrieve on-chip information

Phase 1

- Generate inner CLB (LUT to BLE) placement mapping

Phase 2

- Path-reuse aware routing

Phase 3

- Evaluate performance
Outline

• Introduction
• Proposed Routing Reuse Optimizations
• Experimental Evaluation
• Conclusion
Methodology

- FPGA architecture: Altera Stratix IV
- CAD toolkit: VTR 7.0

Experimental configuration

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Read-before-write Strategy</th>
<th>Reusable Path Identification</th>
<th>Reusable Path Maximization</th>
<th>Reuse-aware Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DIR (ε=1%)</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Proposed (ε=1%)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**DIR:** Proposed scheme without reuse maximization

**ε:** Threshold for reusable path relaxation, path timing within \( \varepsilon=1\% \) of critical path will be relaxed.
Methodology

10 MCNC benchmarks, 9 test pairs

<table>
<thead>
<tr>
<th>No</th>
<th>Benchmark</th>
<th>CLB#</th>
<th>LUT#</th>
<th>Net#</th>
<th>Track Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bigkey</td>
<td>170</td>
<td>1699</td>
<td>829</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>s298</td>
<td>194</td>
<td>1930</td>
<td>683</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>frisc</td>
<td>356</td>
<td>3539</td>
<td>1859</td>
<td>56</td>
</tr>
<tr>
<td>4</td>
<td>elliptic</td>
<td>361</td>
<td>3602</td>
<td>1950</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>spla</td>
<td>369</td>
<td>3690</td>
<td>1866</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>pdc</td>
<td>458</td>
<td>4575</td>
<td>2292</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>ex1010</td>
<td>460</td>
<td>4598</td>
<td>2668</td>
<td>62</td>
</tr>
<tr>
<td>8</td>
<td>s38584</td>
<td>635</td>
<td>6177</td>
<td>3697</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>s38417</td>
<td>636</td>
<td>6042</td>
<td>3613</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>clma</td>
<td>837</td>
<td>8365</td>
<td>4981</td>
<td>66</td>
</tr>
</tbody>
</table>
Results 1/3 – Path reuse

Full path reuse rate

- Average “DIR” = 1.7%, “Proposed” = 3.9%.

Partial path reuse rate

- Average “DIR” = 4.3%, “Proposed” = 15.8%.

Total path reuse rate

- Average “DIR” = 6%, “Proposed” = 19.7%. 
Results 2/3 – SB reconfiguration cost reduction

SB reconfiguration cost = SB reconfiguration writes (in bits)

SB reconfiguration cost reduction

Average “DIR” = 9.8%, “Proposed” = 24.5%.

Reconfiguration cost reduction is strongly correlated with reuse rate but not exactly the same, since paths contain different numbers of SBs.

Total path reuse rate
Results 3/3 – Performances with different $\varepsilon$

$\varepsilon=1\%$ has the best overall performance.

$\varepsilon$: threshold to control reusable path relaxation

- the smaller, the better
Outline

• Introduction
• Proposed Routing Reuse Optimizations
• Experimental Evaluation
• Conclusion
Conclusion

Challenges:
- NVM-based FPGAs are promising to self-adaptive applications, but slow writes and short endurance of NVMs need to be addressed

Our goal:
- **Minimize** reconfiguration costs of switch boxes through reuse-aware routing

Our approaches:
- Model SB reconfiguration cost and identify two types of reusable paths
- Maximize path reuse through exploiting LUT-to-BLE mapping flexibilities
- Enhance VTR CAD flow with a reuse-aware routing algorithm

Results Summary:
- Proposed schemes **deliver as much as 40% path reuse and 34% reduction in SB reconfiguration cost**, within 3.5% overhead in critical path delay.
THE END

Questions are welcome.