Routing Path Reuse Maximization for Efficient NV-FPGA Reconfiguration

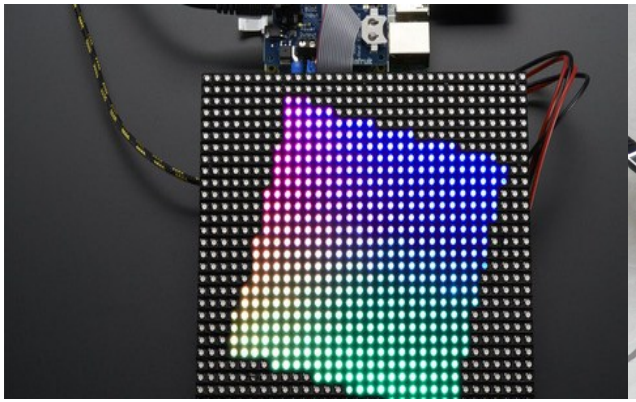
*Yuan Xue, Patrick Cronin, **Chengmo Yang** and Jingtong Hu*

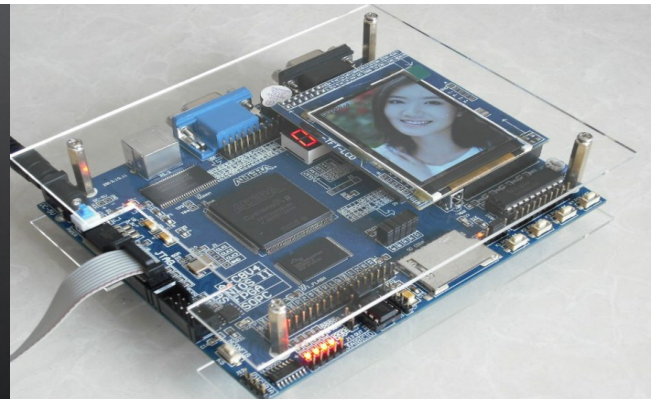
01/27/2016

Outline

- **Introduction**
  - **NV-FPGA benefits and challenges**
  - **Routing optimization strategy**
- Proposed Routing Reuse Optimizations
- Experimental Evaluation
- Conclusion

Self adaptive systems



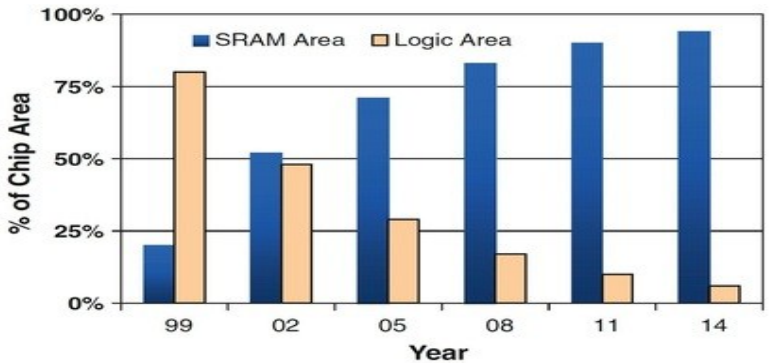


FPGA offers reconfigurability, flexibility, and low design cost to various embedded systems such as control, signal processing and many other applications areas.

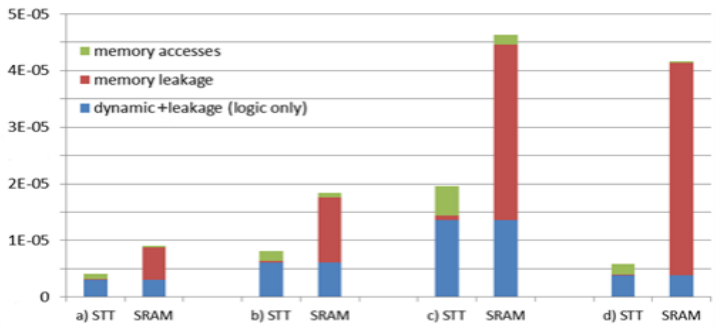
Drawbacks of traditional FPGAs

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

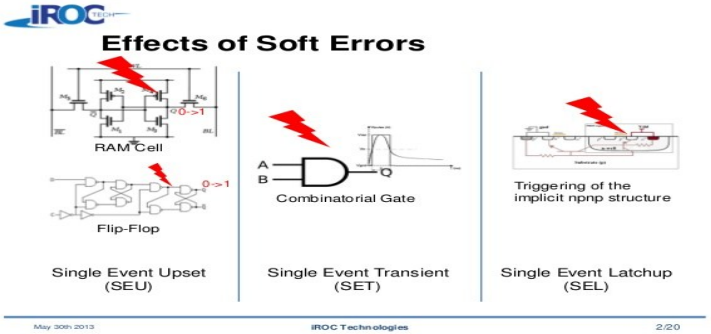
Low scalability



High leakage power



Prone to soft error



Volatile

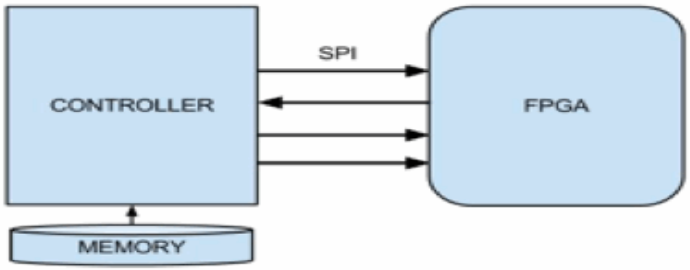
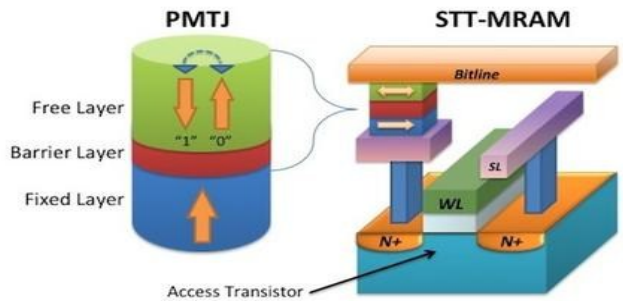


Figure 3 - FPGA Configuration in Slave Mode - SPI

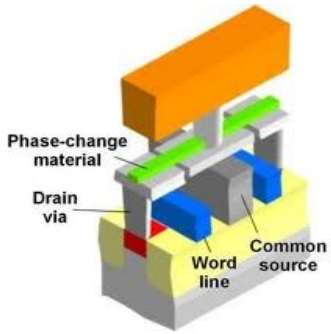
NVM-based FPGA

Non volatile Memories (NVMs) use physical characteristics to represent logic states, such as:

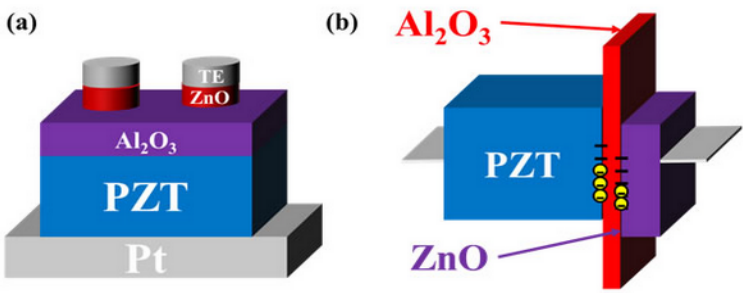
- Phase Change Memory (**PCM**)
- Ferroelectric RAM (**FRAM**)
- Spin Transfer Torque Magnetic RAM (**STT-MRAM**)



STT-MRAM



PCM



FRAM

Overcome SRAM limitations?

NVM		SRAM
• High density	→	~~Low scalability~~
• Near-zero leakage power	→	~~High leakage power~~
• Strong error resistance	→	~~Prone to soft error~~
• Near-zero power-on delay	→	~~Volatile~~

However, no rose is without a thorn!

Any issues or challenges?

COMPARISON OF SRAM AND VARIOUS NVM CELLS

Type	Area (F^2)	Read time(ns)	Write time(ns)	Write cycles
SRAM	140	0.2	**0.2**	10^{16}
PCM	4	12	**100**	10^{9}
STT-MRAM	42	35	**35**	10^{12}
NOR Flash	10	15	**1000**	10^{5}

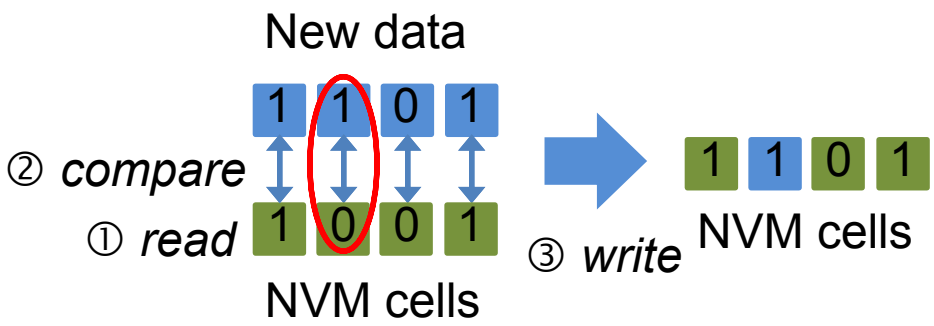
However, no rose is without a thorn!

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



Type	Read time(ns)	Write time(ns)
SRAM	0.2 ⟷	0.2
PCM	12 ⟷	100

Why this works for NVM?

For PCM

Without RBW → 4 writes → 400ns

With RBW → 4 reads+ 1 write → 148ns

Great Improvement!

For SRAM

Without RBW → 4 writes → 0.8ns

With RBW → 4 reads+ 1 write → 1ns

No benefit!

NV-FPGA

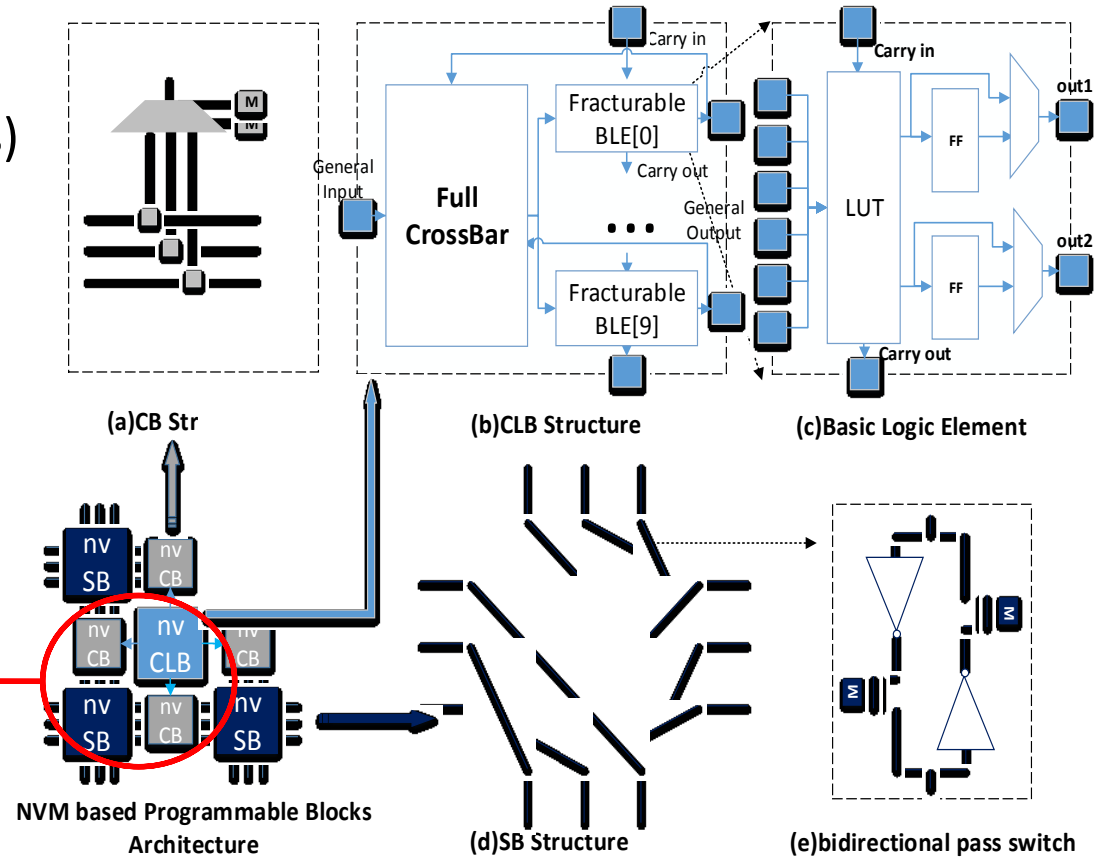
NV-FPGA: Use NVMs as on-chip memories and configurable block units on FPGA.

Main blocks include:

- Configurable logic blocks (**CLB**s)
- Connection blocks (**CB**s)
- Switch box (**SB**s)

BLOCK	AREA	POWER	DELAY
SB	**90%**	**85%**	**80%**
CLB+CB	10%	15%	20%

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?

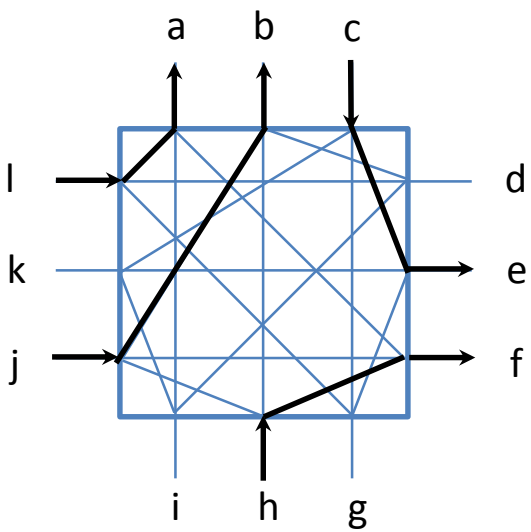
How to perform **reuse-aware routing** while preserving circuit timing?

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



Existing configuration

switch	On/ off
l→a	1
j→b	1
c→e	1
h→f	1
k→i	0

connected switch

N_{old}=4

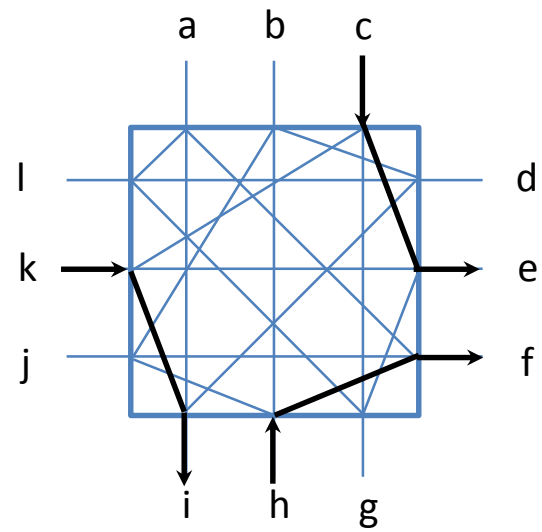
3 changed switches

Reused switch: **R=2**

switch	On/ off
l→a	0
j→b	0
c→e	1
h→f	1
k→i	1

connected switch

N_{new}=3



New configuration

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

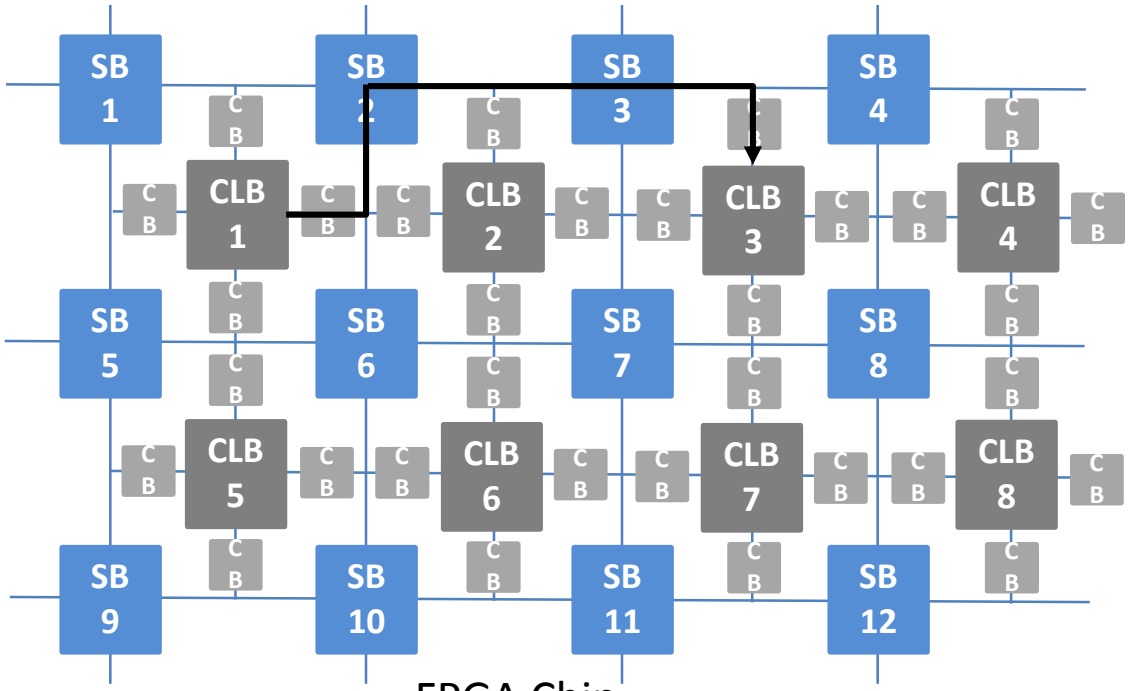
Outline

- 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.



FPGA Chip

Path Definition and Characterization

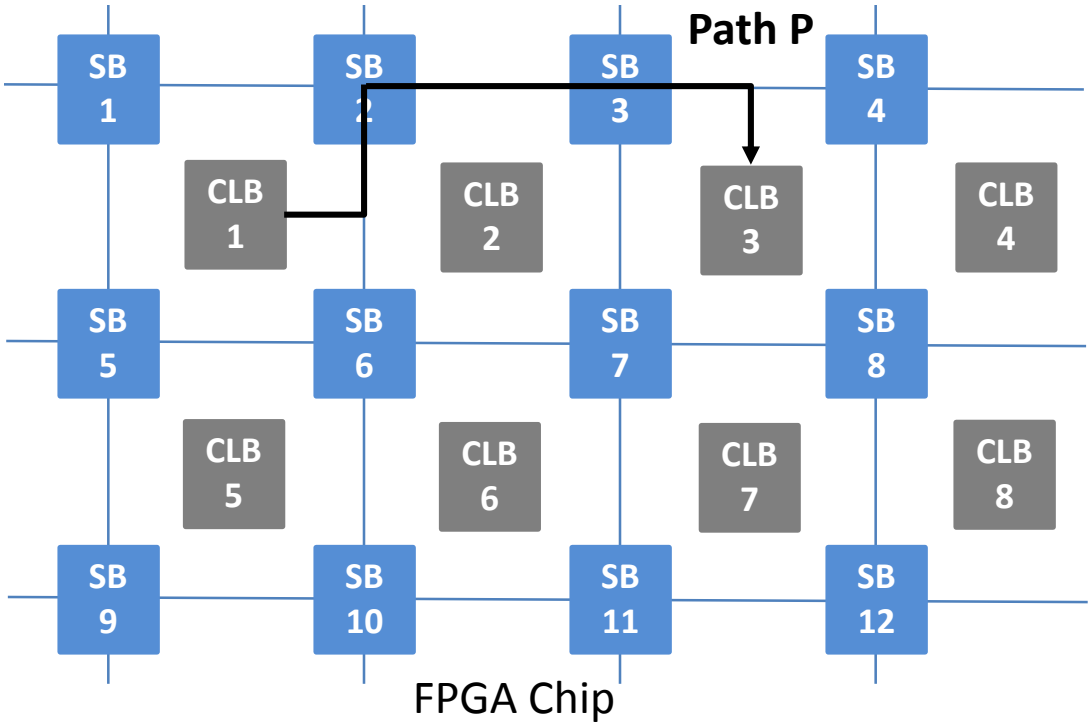
Definition: **Path** is a single source-to-sink connection.

Characterization: Path P={(i,j),(SB_{first}…SB_{last})}.

P consists of two sets:

- **CLB set: (i,j)→**starting point **CLBi**, ending point **CLBj**
- **SB set: (SB_{first}…SB_{last}) →** all SBs that P passes through

Path reuse** could efficiently translate to **SB reuse!

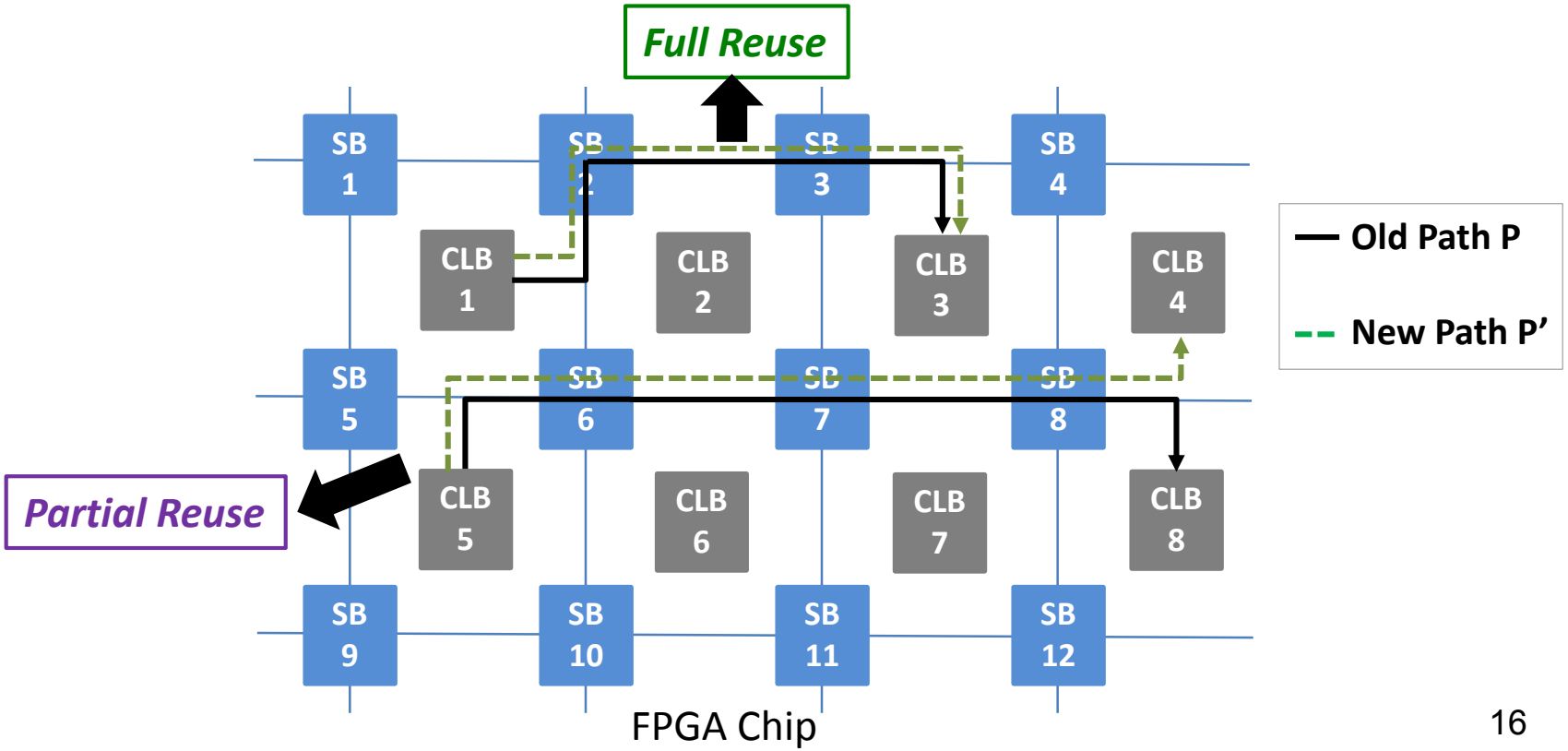


FPGA Chip

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

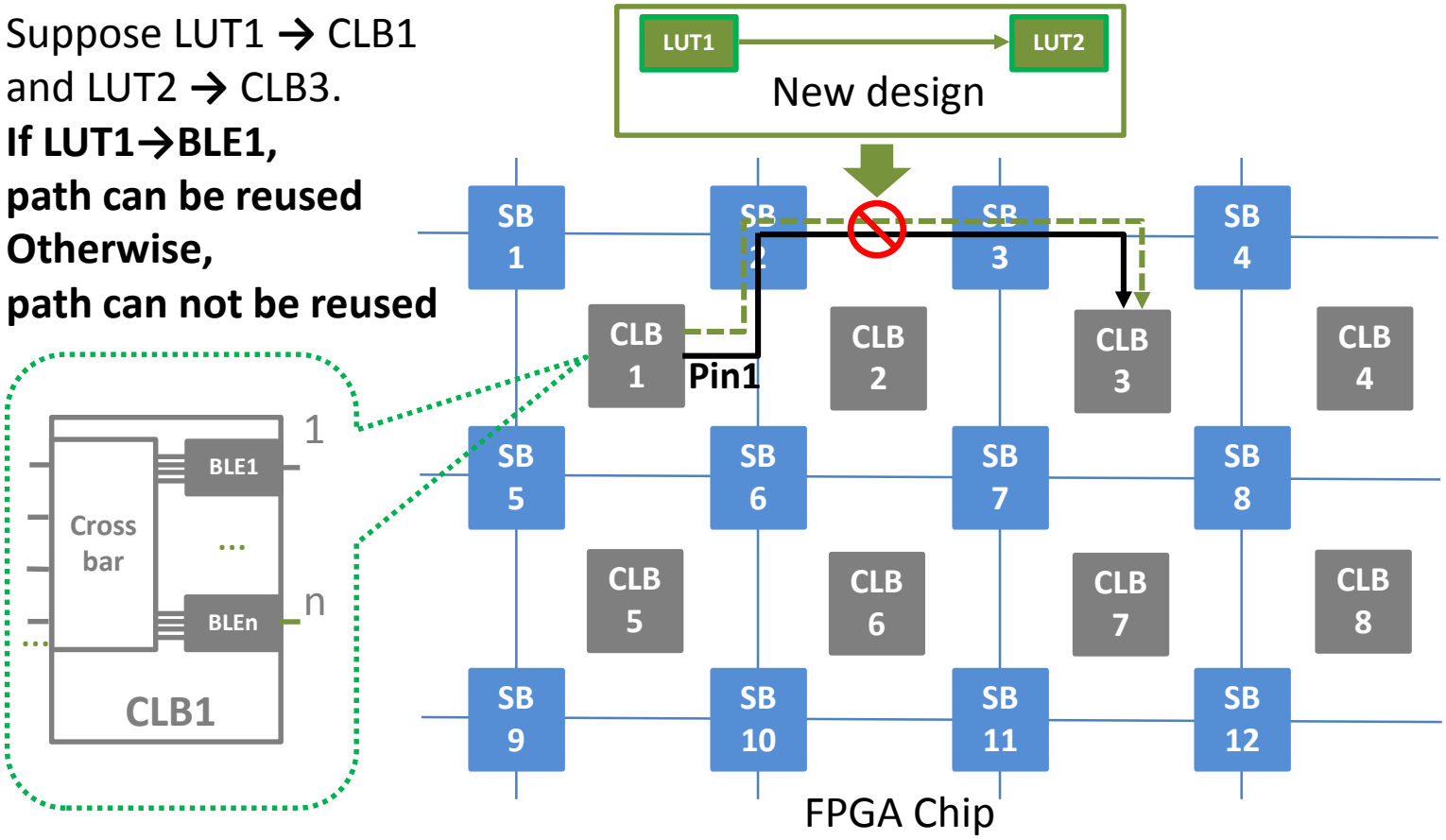
However, the same starting CLB cannot guarantee path reuse. **Starting pin matters!**

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

Suppose LUT1 → CLB1 and LUT2 → CLB3.

If LUT1→BLE1, path can be reused

Otherwise, path can not be reused

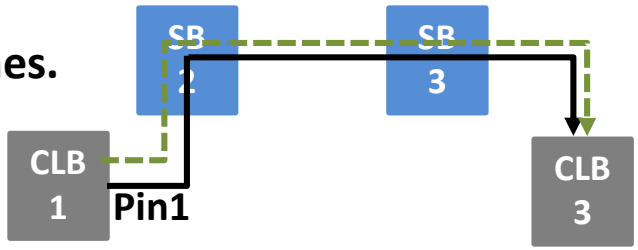


Reusable Path Maximization

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→j represents the **number of reusable switches.**

For example, LUT1→BLE1
$Edge_{11}$ =2



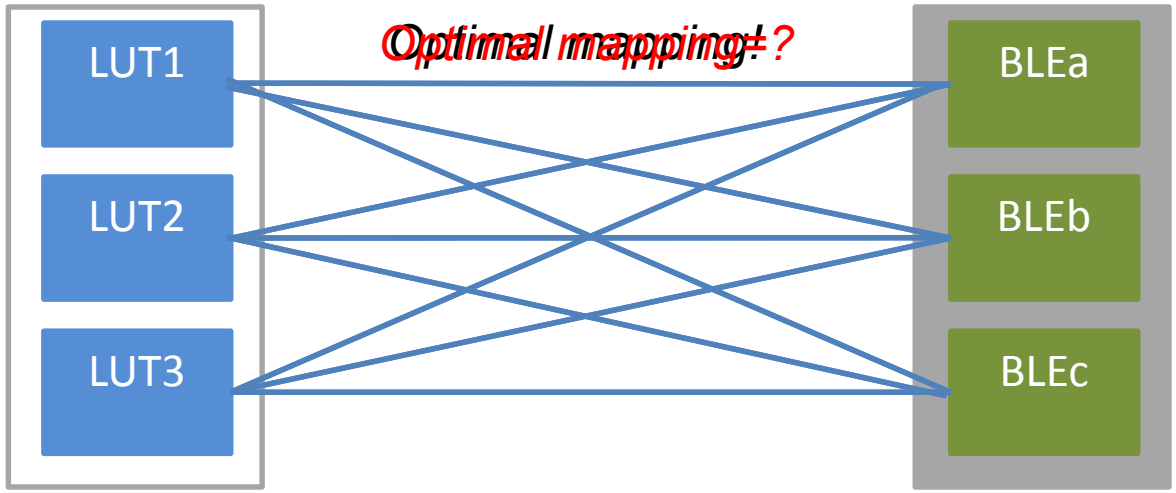
BLE 1-n

Reuse=

2	5	2
3	3	4
4	2	2

LUT 1-n

Maximal reuse!



Optimal mapping!

Optimal mapping=?

One CLB in new design

One CLB block on FPGA

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

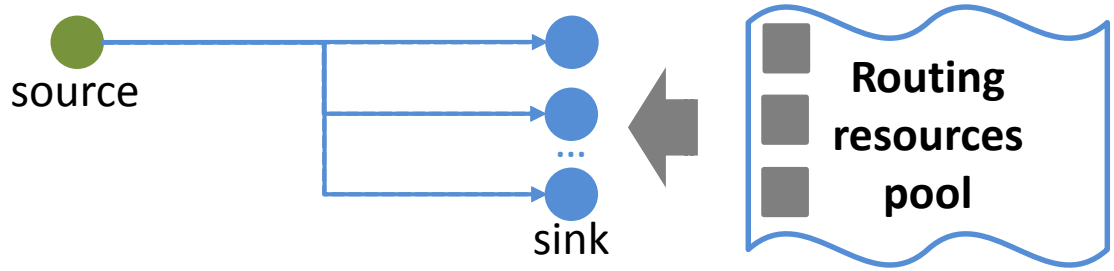
H. Kuhn, "The Hungarian method for the assignment problem," in NRLQ, Mar. 1955, pp. 83-97.

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

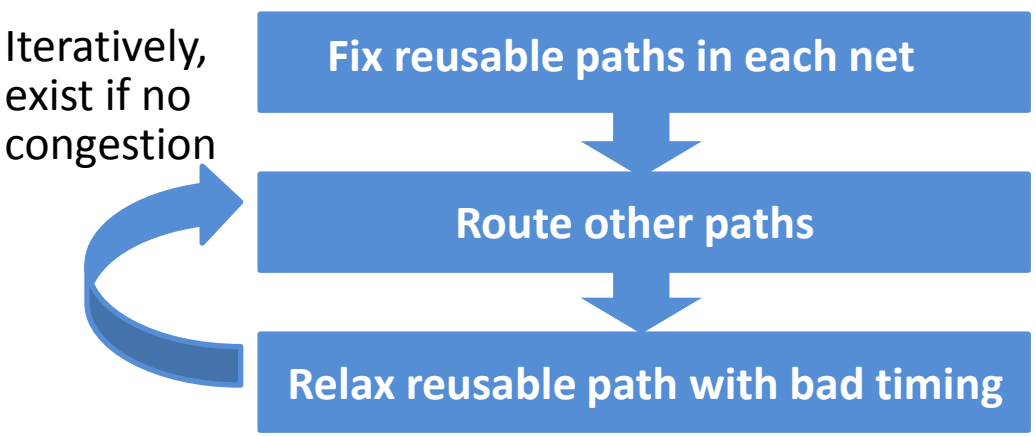
Reuse-aware Routing Algorithm

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



- Divide resource into two types: **general** (used for all nets) and **dedicated** (used by specified net)
- Rank nets by their sink count, route nets with maximum sinks first
- When relax, release related **dedicated** resources

Stage 1: fix reusable path

Stage 2: route other paths

Stage 3: relax related paths

Repeat 2,3 if congesting

Input: Netlist, Placement, Reusable paths, Relax threshold ε
Output: Routing results
Mark all routing resources rr_node:type = general;

Sort (nets, unrouted sinks);
for (loop = 0;loop<Iteration_limit;loop++)

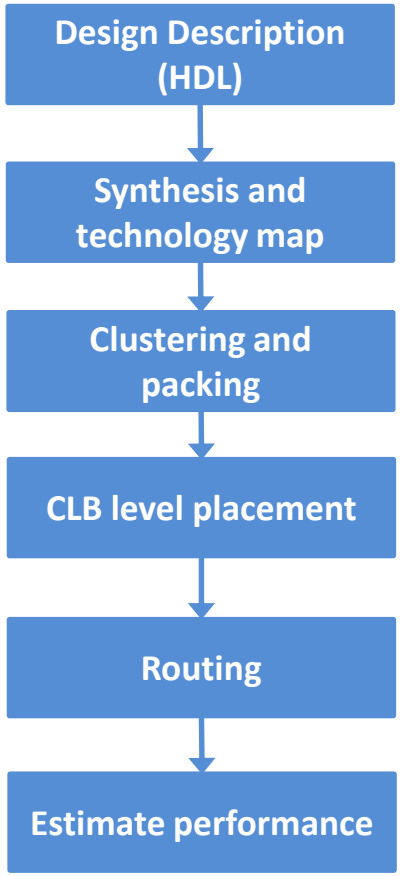
Perform timing analysis and update path criticality values;

if (no congestion exist)
Exit ;
}

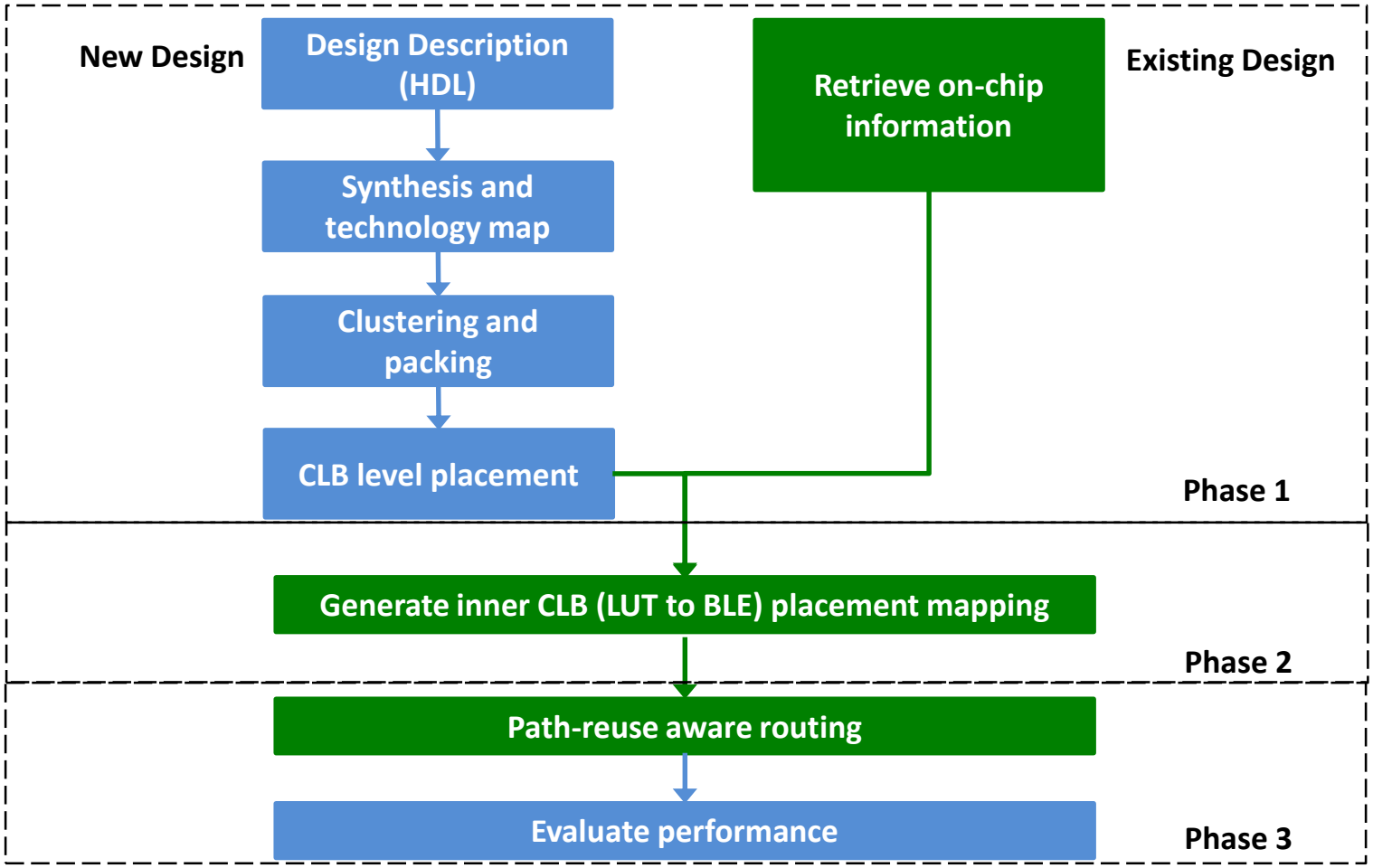
Outline

- 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



Proposed CAD flow



Outline

- Introduction
- Proposed Routing Reuse Optimizations
- **Experimental Evaluation**
- Conclusion

Methodology

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

Experimental configuration

Schemes	Read-before-write Strategy	Reusable Path Identification	Reusable Path Maximization	Reuse-aware Routing
Baseline	+	-	-	-
DIR (ε=1%)	+	+	-	+
Proposed (ε=1%)	+	+	+	+

DIR: Proposed scheme without reuse maximization

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

Methodology

10 MCNC benchmarks, 9 test pairs

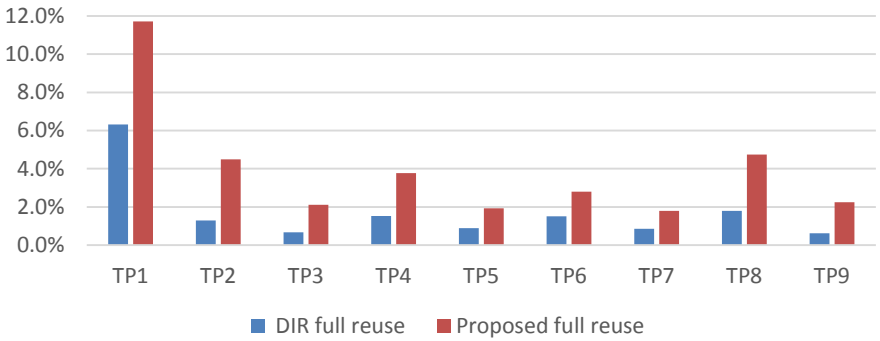
TP1

TP9

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

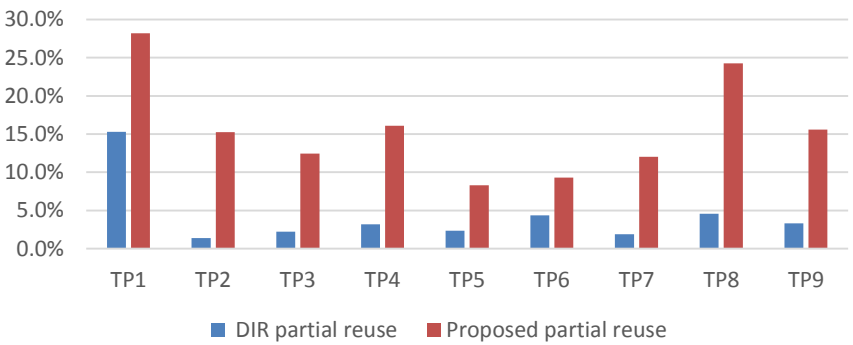
Results 1/3 – Path reuse

Full path reuse rate



- Average "DIR" = 1.7%, "Proposed" = 3.9%.

Partial path reuse rate

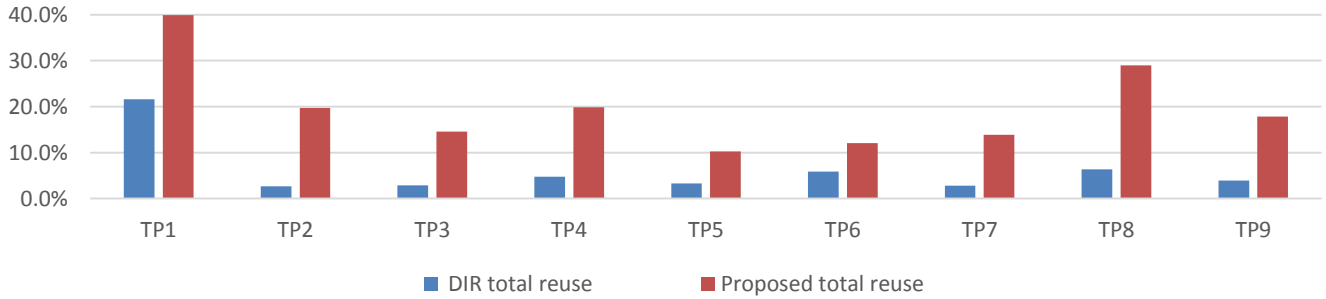


- Average "DIR" = 4.3%, "Proposed" = 15.8%.

Full + Partial

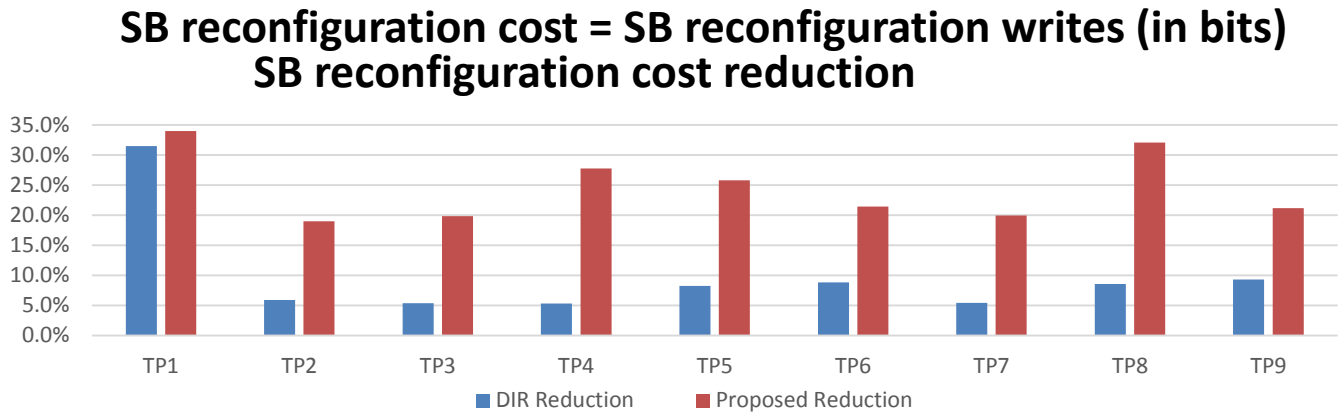
II

Total path reuse rate

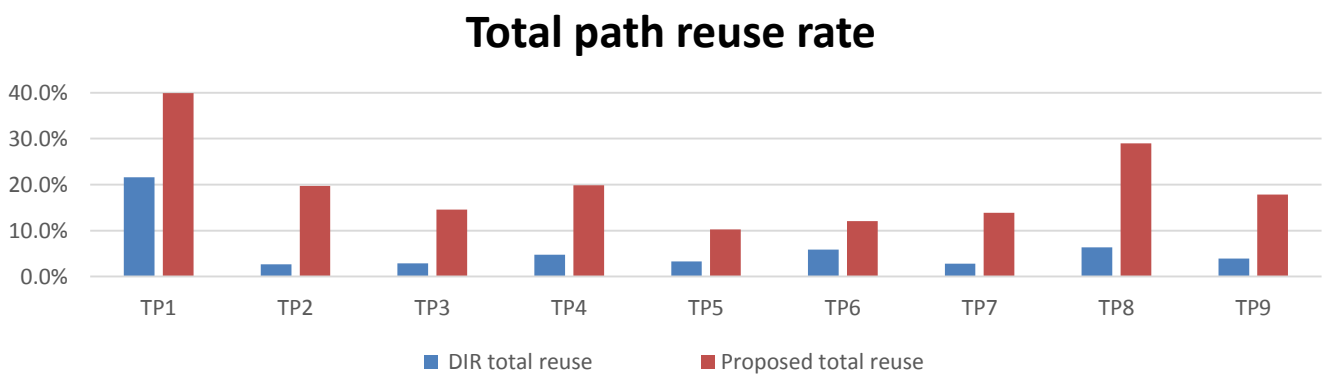


- Average "DIR" = 6%, "Proposed" = 19.7%.

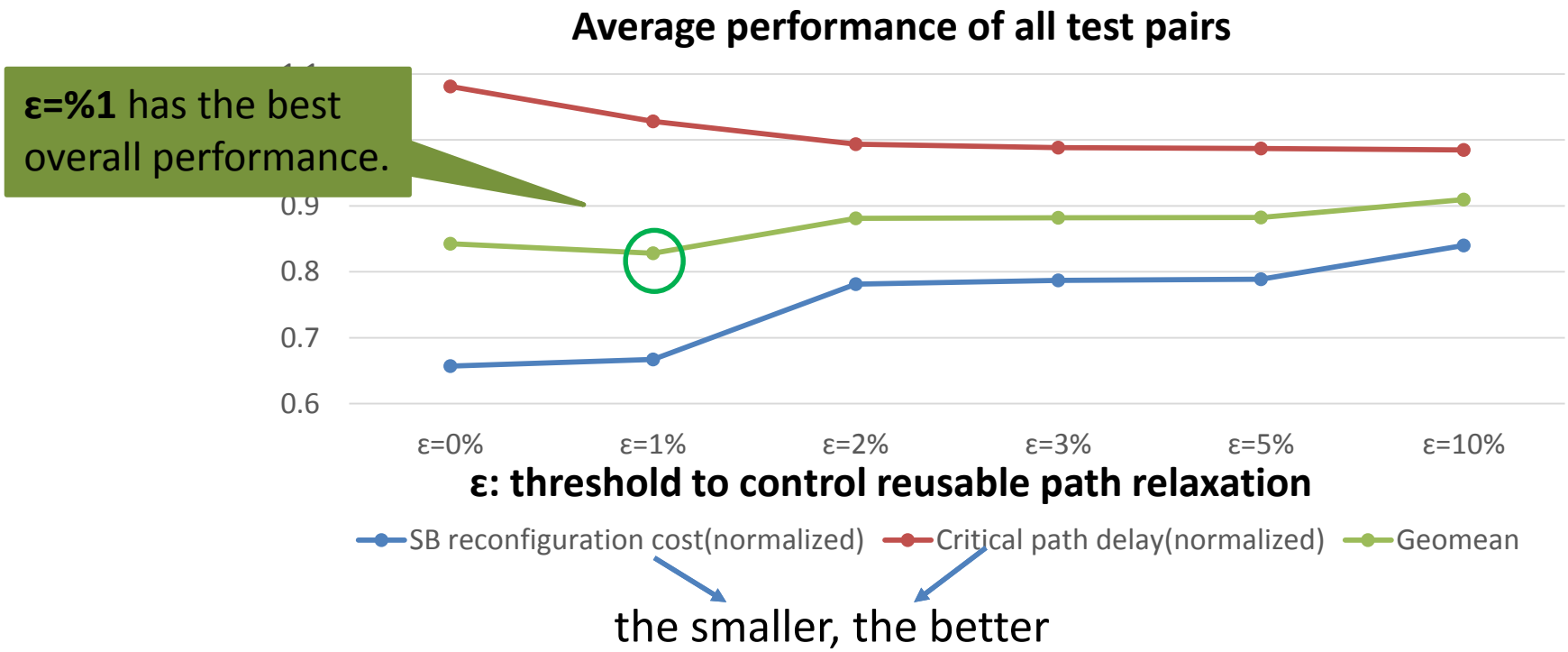
Results 2/3 – SB reconfiguration cost reduction



- 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.



Results 3/3 – Performances with different ε



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.