Enabling High Throughput and Virtualization for Traffic Classification on FPGA

Yun R. Qu, Viktor K. Prasanna

Ming Hsieh Dept. of Electrical Engineering

University of Southern California

Los Angeles, CA 90089

Email: yunqu@usc.edu prasanna@usc.edu

USC Viterbi School of Engineering

University of Southern California

Outline

- Introduction
- Background
- Algorithms & Architecture
- Evaluation
- Conclusion

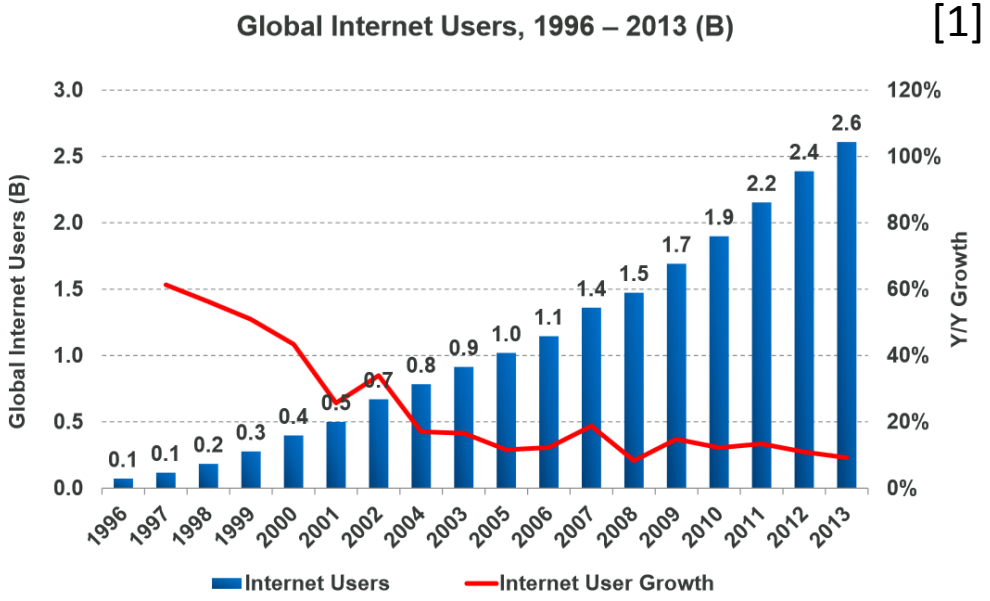
Outline

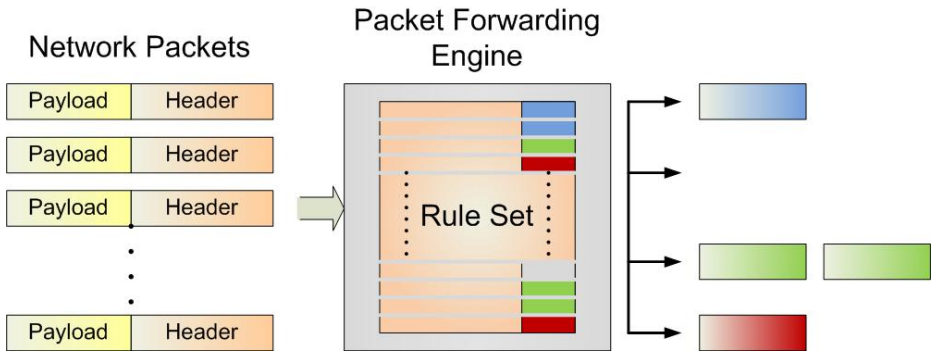
- Introduction
  - Routing devices, virtualization
- Background
- Algorithms & Architecture
- Evaluation
- Conclusion

Internet & Routing Devices

- Internet
  - Internet users ↑
  - network traffic ↑
- Gateway/router/switch
  - Packet forwarding
  - Access control
  - Packet/traffic classification

[1]

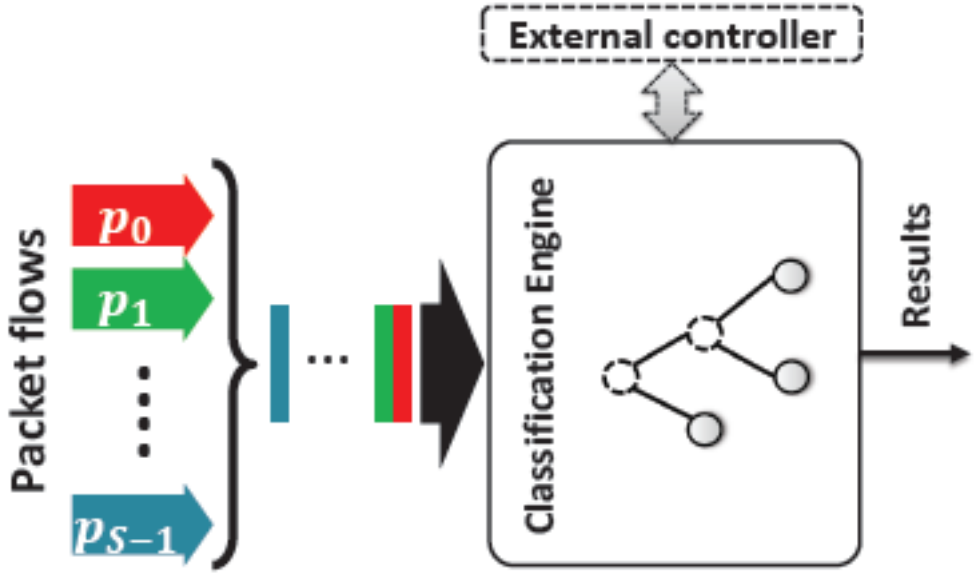




[1] http://www.kpcb.com/internet-trends

Trend: Virtual Machines

- Hardware Virtualization
  - Sharing expensive hardware devices
  - *e.g.*, a virtualized classification engine
  - Require **efficient dynamic updates** !



Outline

- Introduction
- Background
  - Traffic Classification
  - Related works
  - Motivations
- Algorithms & Architecture
- Evaluation
- Conclusion

Internet Traffic Classification

- Definition
  - Classifying traffic into application classes
  - Based on various features extracted from traffic
    - *e.g.*, port numbers, packet length, *etc.*
  - Extract features from
    - Packet headers → accuracy issue
    - Deep packet inspection → privacy issue
  - Major challenges
    - Performance (*e.g.*, throughput)
    - Hardware virtualization

Existing Approaches (1)

- Port-number-based
  - Transport-layer port number
    - *e.g.*, HTTP: 80, FTP control: 21
  - Dynamic port → no longer reliable

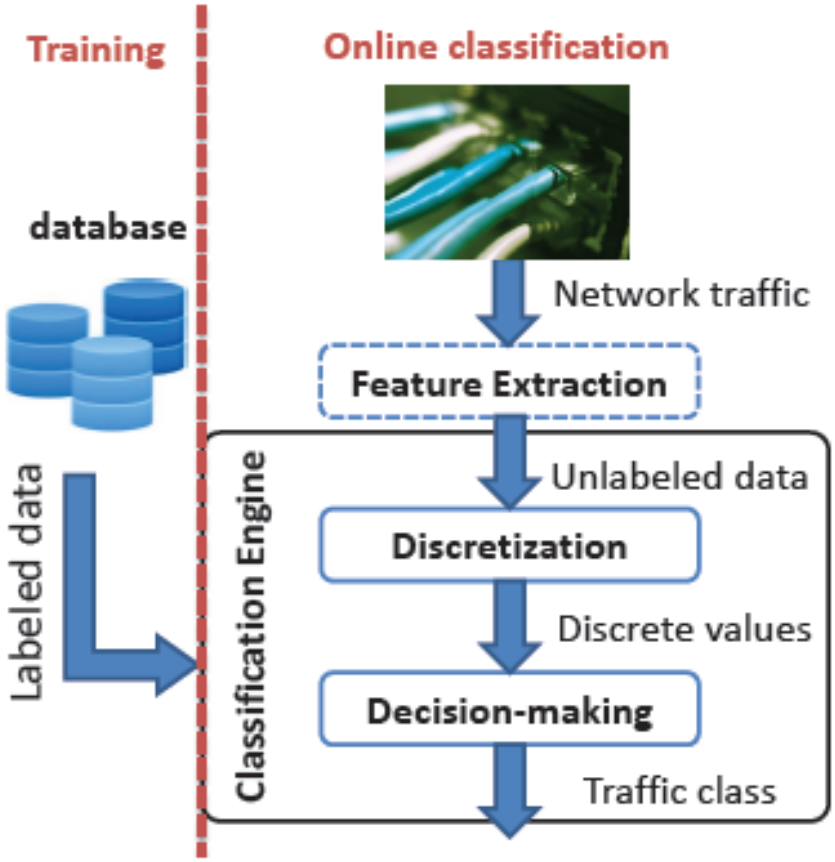
- Deep Packet Inspection (DPI)
  - Slow, privacy issue, *etc.*

- Heuristic-based
  - Large training data, low accuracy

Existing Approaches (2)

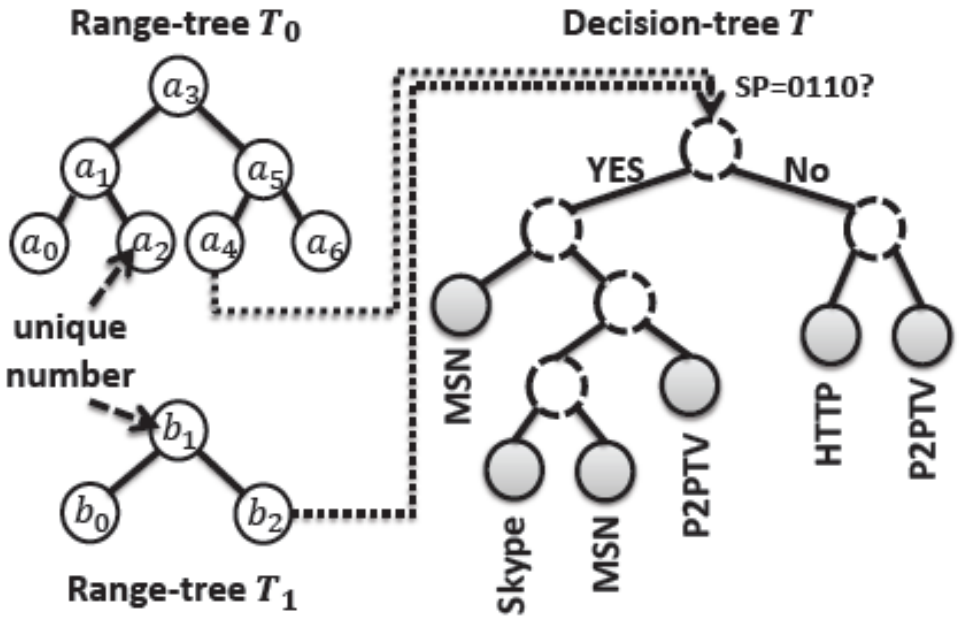
- Machine Learning (ML)
  - Supervised / unsupervised
  - Extract features
  - **Discretization**
    - Into unique numbers
  - **Decision-making**
    - Determine traffic class
    - *e.g.*, **C4.5 decision-tree**

Our focus



Existing Approaches (3)

- C4.5 decision-tree
  - Range-trees (discretization) + decision-tree (decision-making)
  - Not easy for dynamic updates



Motivation

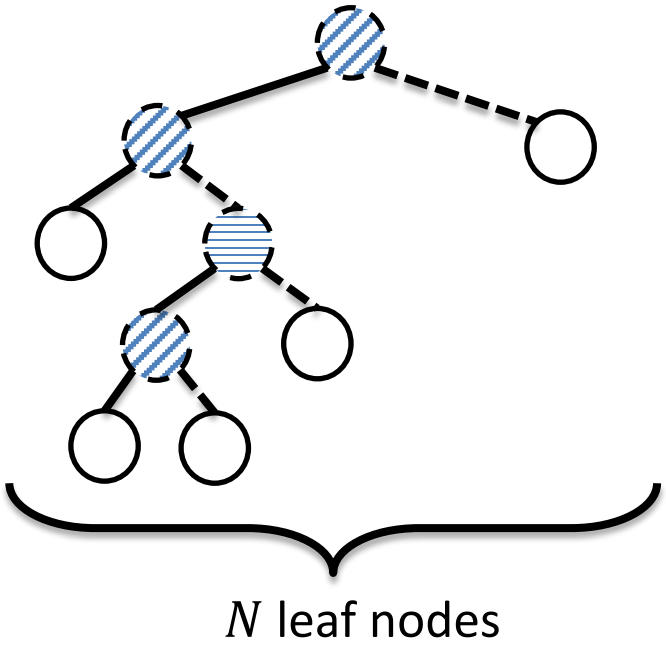
- Facilitate Dynamic Updates
  - Combine range-trees and decision-tree
  - Architectural support
- Explore Parallel Searches
  - Search multiple root-to-leaf paths in parallel
  - Search multiple features in parallel

Outline

- Introduction
- Background
- Algorithms & Architecture
  - Converting Tree to Rule Set Table (RST)
  - Modular Processing Element (PE)
  - 2-dimensional Pipelined Architecture
  - Efficient Dynamic Updates
- Evaluation
- Conclusion

Converting Decision-tree

- Conversion techniques
  - Each root-to-leaf path ↔ a set of criteria
  - All the criteria of a path → an entry of RST



Rule Set Table (RST)

Feature 0	Feature 1	Feature 2

Check feature 0

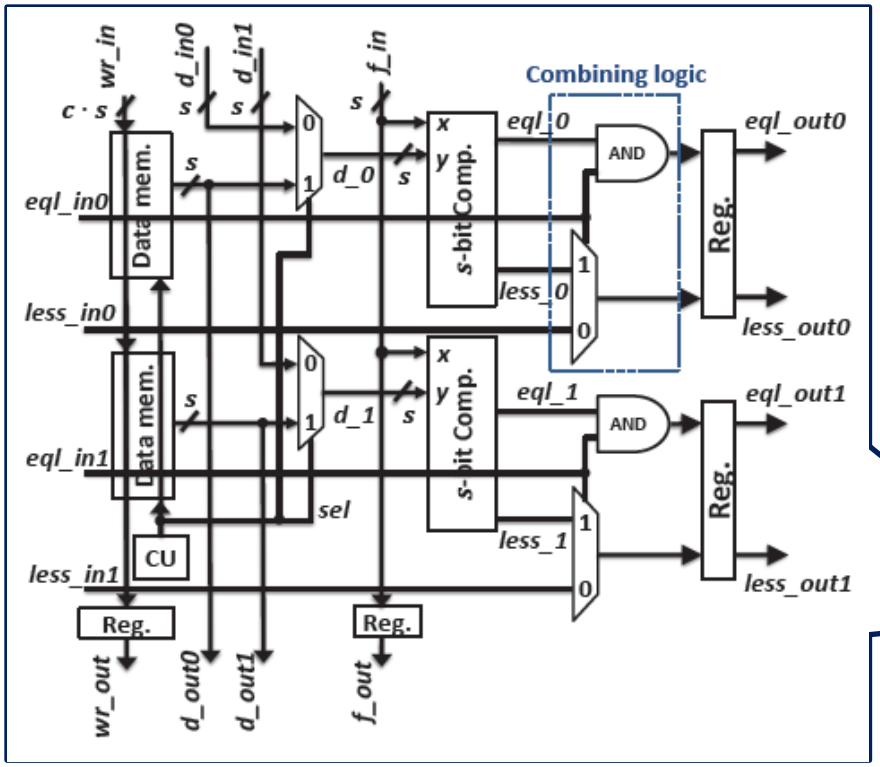
Check feature 1

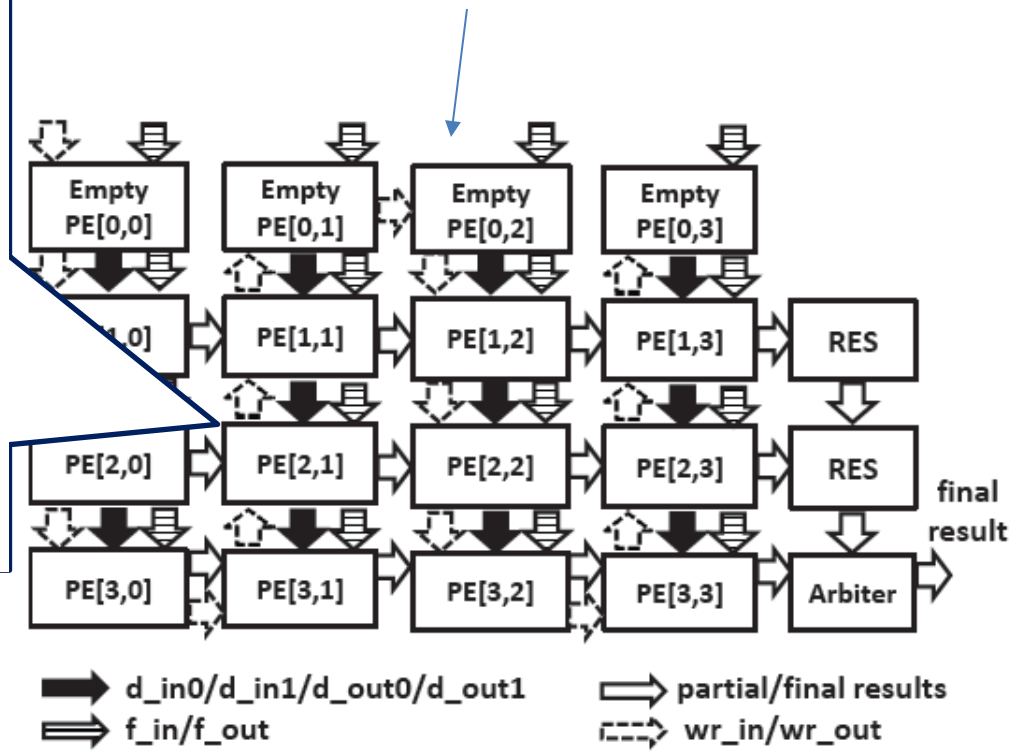
Check feature 2

Architecture (1)

- Modular Processing Element (PE)
  - Compare a stride of input against a stride of a range boundary
- 2-dimensional Pipelined Architecture
  - Propagate packet headers vertically (*f_in* / *f_out*)
  - Propagate partial / final results horizontally
- Snake-like Pipeline
  - Scan chain for writing data into data memory (*wr_in* / *wr_out*)

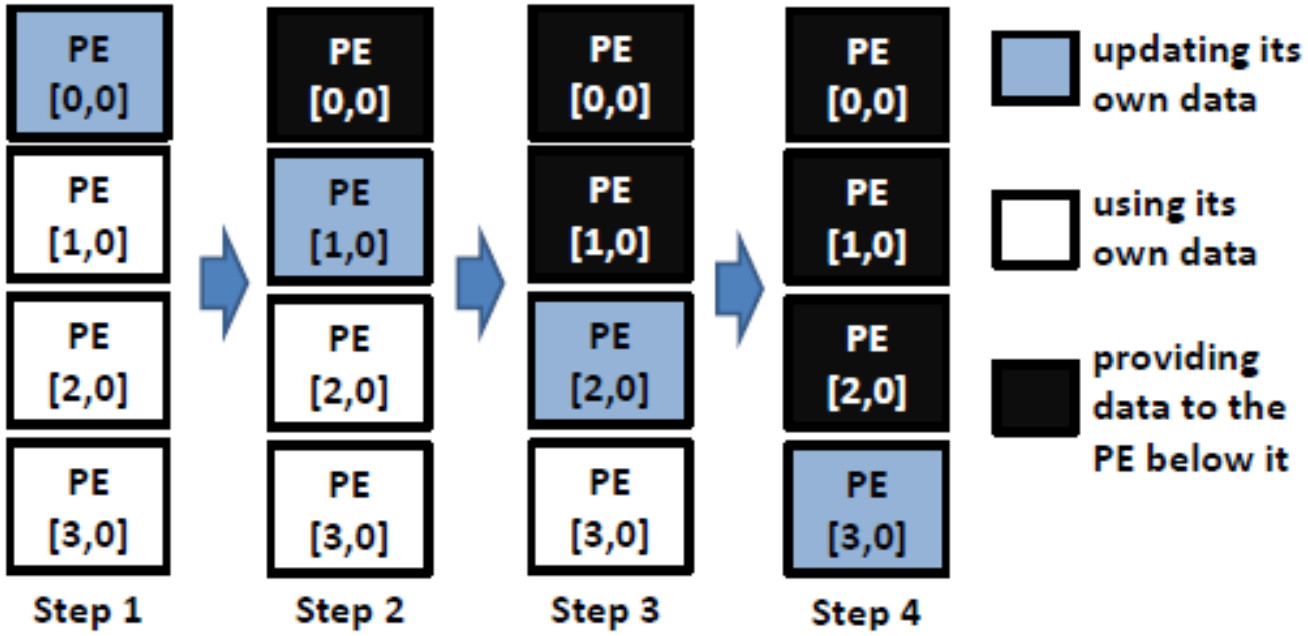
Architecture (2)





Virtualization

- Dynamic Updates
  - Tree update ↔ deletion, insertion, modification of rules on RST
  - *e.g.*, a top-down **ping-pong** update on the first column



Outline

- Introduction
- Background
- Algorithms & Architecture
- Evaluation
  - Throughput, resource consumption
  - Comparison with prior works
- Conclusion

Experimental Setup

- FPGA
  - XC7VX1140t FLG1930 -2L
  - 1100 I/O, 218800 logic slices
  - 67 Mb BRAM, up to 17 Mb distributed RAM (dual-ported)
  - Verilog on Vivado Design Suite 2013.4
- Data Sets
  - Real decision-tree: 42-levels, 96 leaves
  - Similar synthetic trees
  - 6 features, 8 application classes

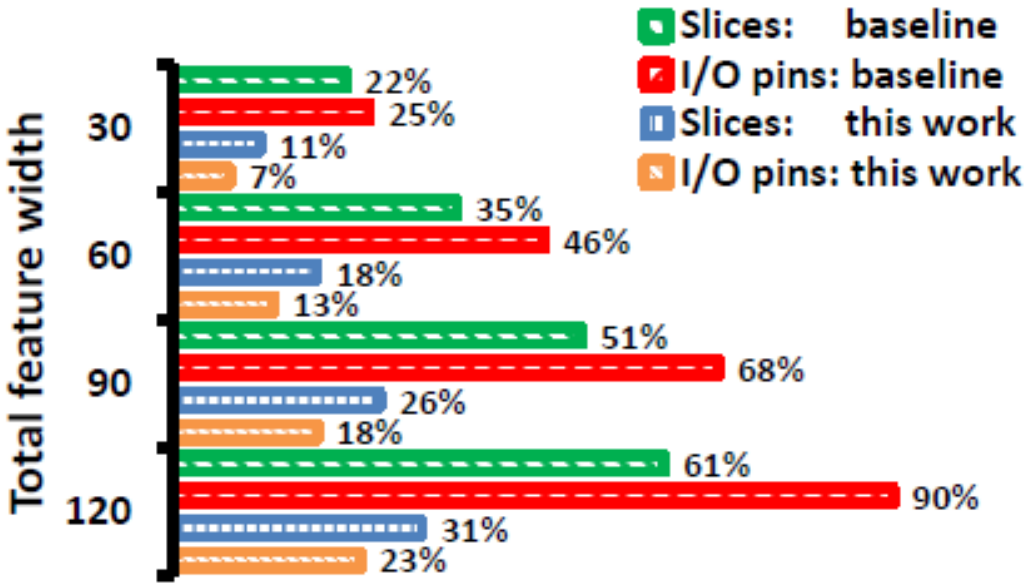
Throughput

- Parameterized Design
  - Best: each PE compares 3-bit strides against 2 boundaries
- Overall Throughput
  - Sustain 2× throughput of state-of-the-art

Parameters		State-of-the-art				This work			
Feature With		**80**	**160**	**240**	**320**	**80**	**160**	**240**	**320**
Max. no. of leaves	**32**	512	493	458	421	759	715	701	679
	64	437	425	401	374	705	682	653	621
	96	399	382	354	305	645	620	609	588

Resource Consumption

- No. of used logic slices, I/O pins, *etc.*
  - Set max. no. of leaves : 64
  - Baseline: two copies of the same architecture



Comparison

Approach		Algorithm	Platform	Throughput
w/o dynamic updates	Este's	SVM	Dual Xeon, 2.6 GHz, 24 cores, 48 GB RAM	2 MCPS
	Gringoli's	SVM	Dual Xeon, 2.6 GHz, 24 cores, 48 GB RAM	7 MCPS
	Groleat's	SVM	Synthesized on Virtex-5	290 MCPS
	Tong's	C4.5	Place-and-routed on Virtex-7	399 MCPS
w/ dynamic updates	Jiang's	k-NN	Place-and-routed on Virtex-5	125 MCPS
	This work	C4.5	Place-and-routed on Virtex-7	645 MCPS

Outline

- Introduction
- Background
- Algorithms & Architecture
- Evaluation
- Conclusion

Conclusion

- Conclusion
  - Based on a conversion from decision-tree to RST
  - Parallel searching on FPGA
  - Achieve high performance & scalability

- Future work
  - Power, latency, *etc*.
  - Self-learning mechanism

Questions?

Thanks

http://ganges.usc.edu