Scalable and Dynamically Updatable Lookup Engine for Decision-trees 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

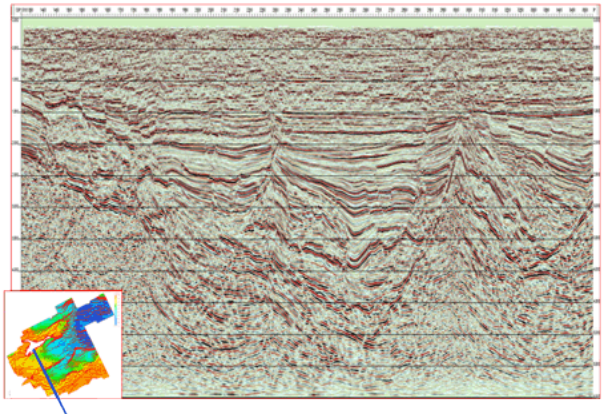
- Introduction
- Background
- Data Structures & Algorithms
- Architecture
- Evaluation
- Conclusion

Outline

- Introduction
  - High-performance computing
- Background
- Data Structures & Algorithms
- Architecture
- Evaluation
- Conclusion

High-performance Computing (HPC)

- HPC
  - Superior performance
  - Image/signal processing
  - Scientific computing, simulation
  - Networking



[1]

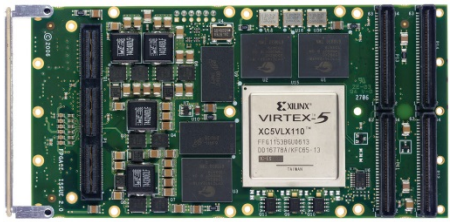
- Performance
  - High throughput
  - Fast updates

[1] http://www.bgp.com.cn/Service/Processing.html

Emerging Trends in HPC

- Software centralization
  - Adapt to the change in demand
  - Dynamically updatable

- Heterogeneous computing
  - Accelerated by hardware
  - FPGA, GPU, VLSI chips, etc.



Outline

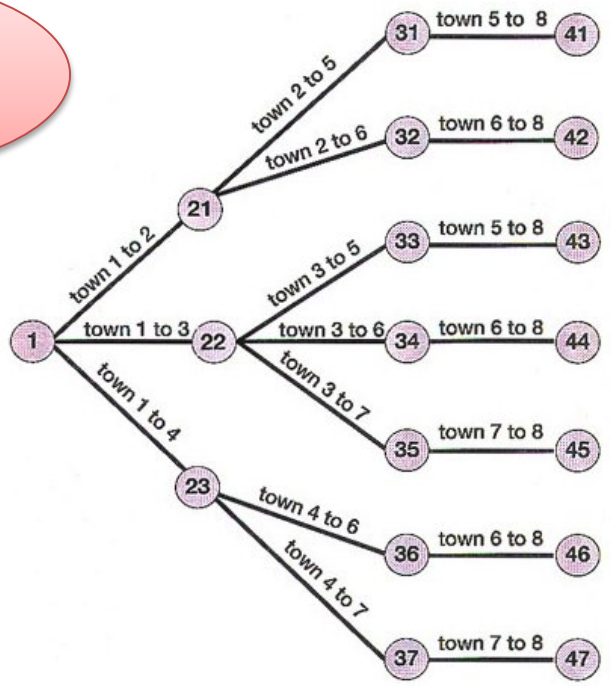
- Introduction
- Background
  - Decision-trees
  - Challenges
- Data Structures & Algorithms
- Architecture
- Evaluation
- Conclusion

Decision-trees in HPC

- Usage
  - Data mining, **classification**
- Problem definition
  - Given a decision-tree
    - N leaf nodes, M fields
  - High-throughput classification
  - Support dynamic updates

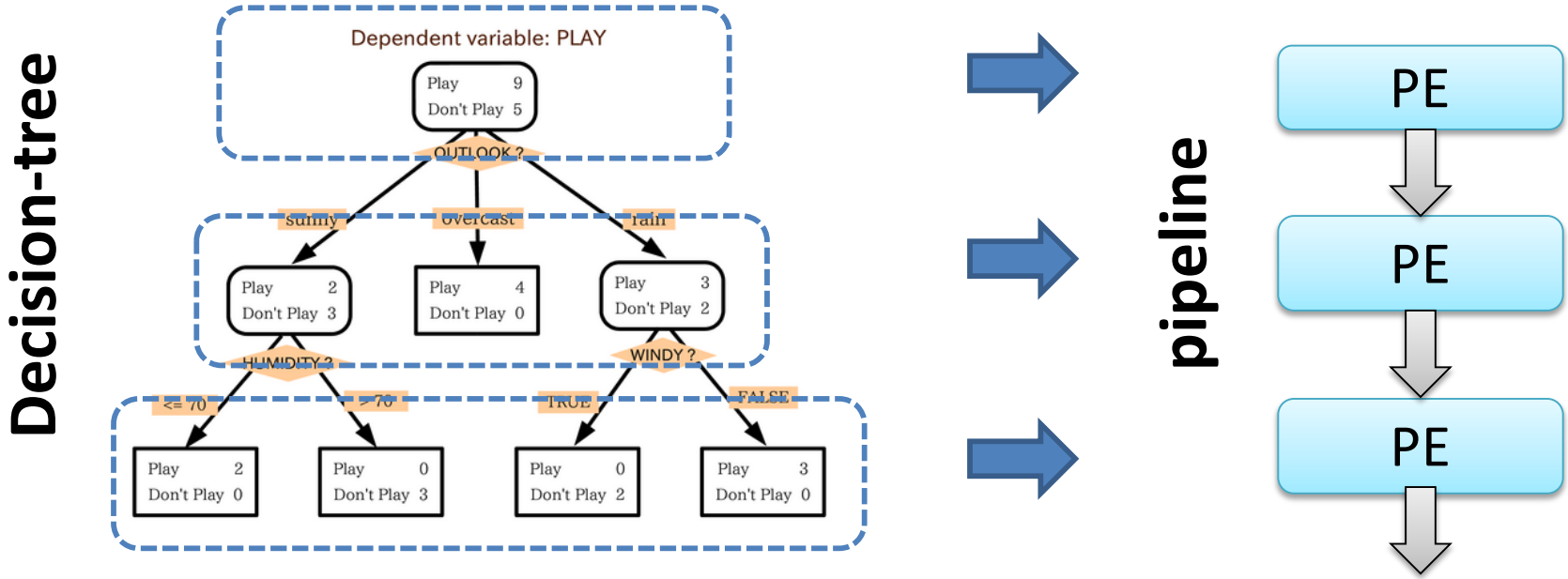
Our focus

Decision-tree



State-of-the-art Implementation

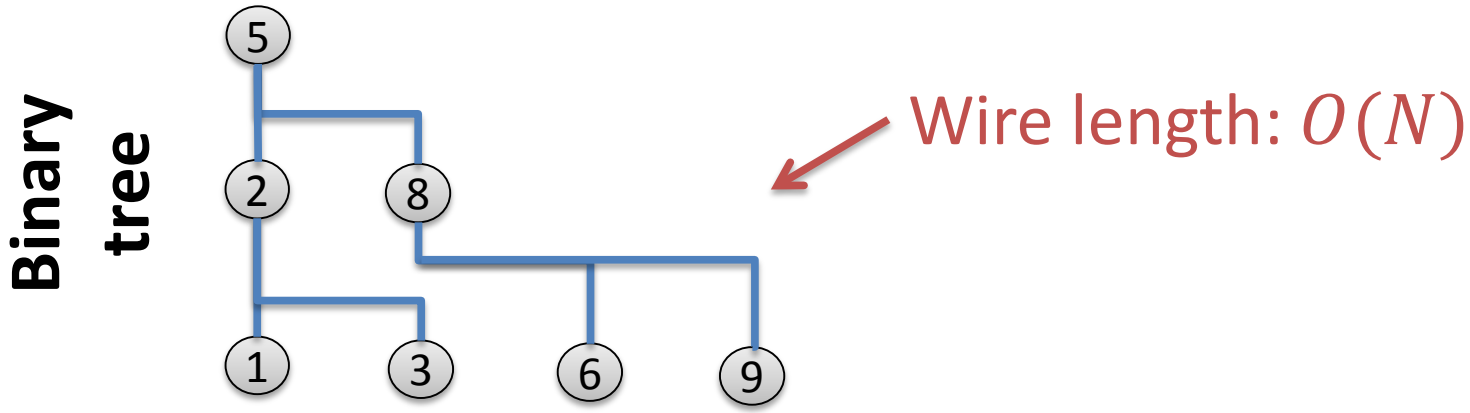
- Straightforward mapping onto hardware [2][3]
  - Each level → Processing Element (PE)
  - No. of levels ↑ → Memory consumption per stage ↑



[2] D. Sheldon and F. Vahid, "Don't Forget Memories: A Case Study Redesigning a Pattern Counting ASIC Circuit for FPGAs," CODES+ISSS, 2008.
[3] Y.-H. E. Yang and V. K. Prasanna, "High Throughput and Large Capacity Pipelined Dynamic Search Tree on FPGA," FPGA, 2010.

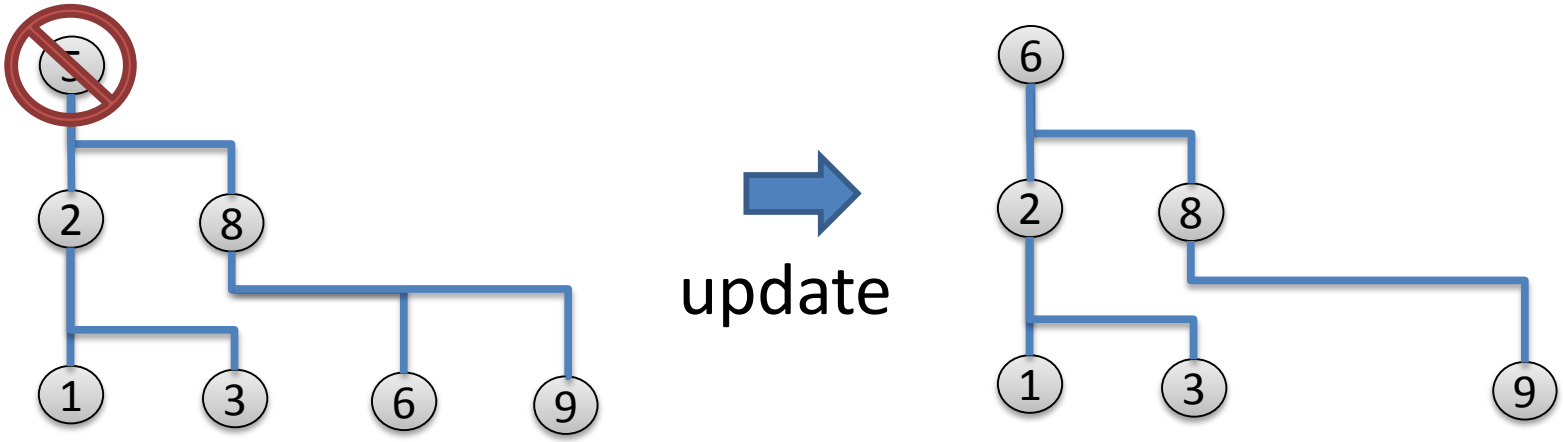
Challenges (1)

- Scalability
  - Balanced: $O(\log N)$ lookup time, $O(N)$ wire length
  - Imbalanced: $O(1)$ wire length, $O(N)$ lookup time
  - $N \uparrow \quad \rightarrow \quad$ performance (clock rate, throughput) $\downarrow$
  - Challenge 1: a **scalable** implementation



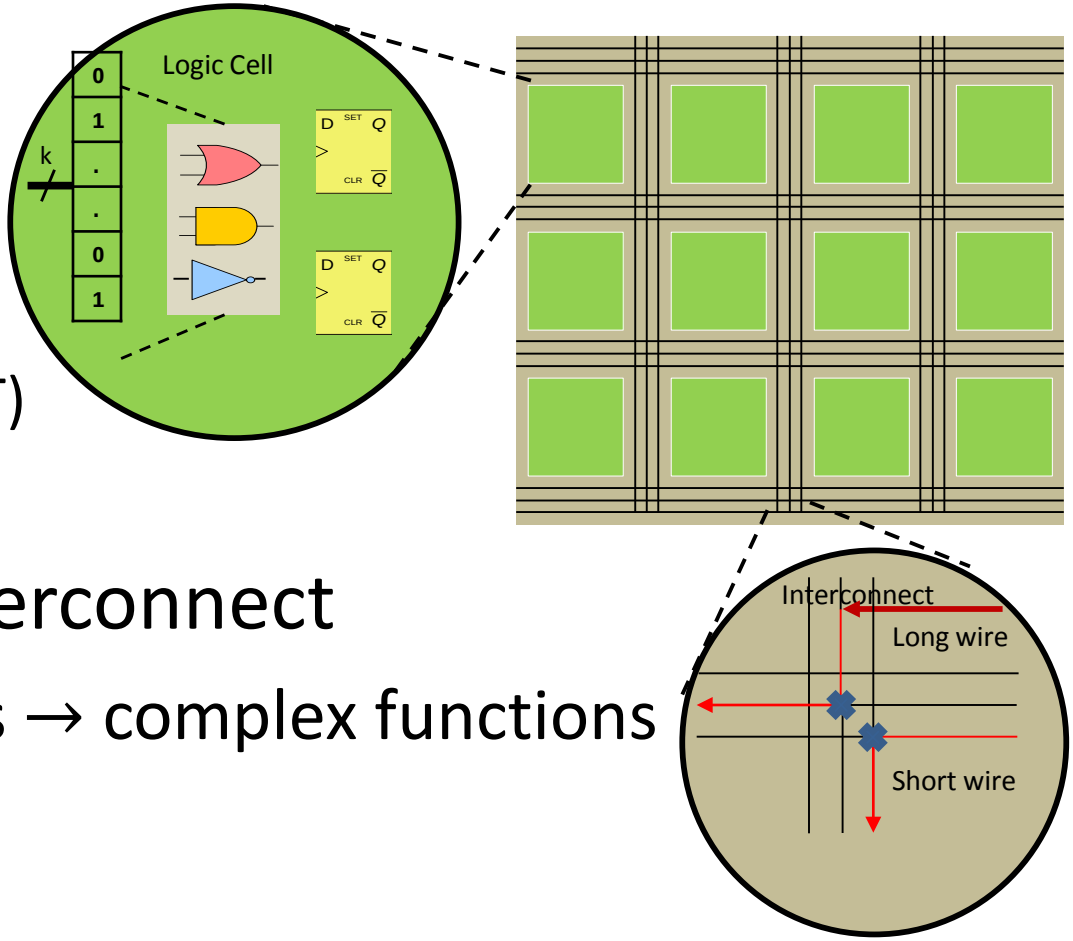
Challenges (2)

- Dynamic updates
  - Deletion/insertion: nodes being pushed around
  - Modification: as complex as deletion/insertion
  - Global changes: a single update → too many operations
  - Challenge 2: a **dynamically updatable** implementation



Field Programmable Gate Array

- Logic Cell Matrix
  - Programmable
  - Logic elements
    - Lookup Table (LUT)



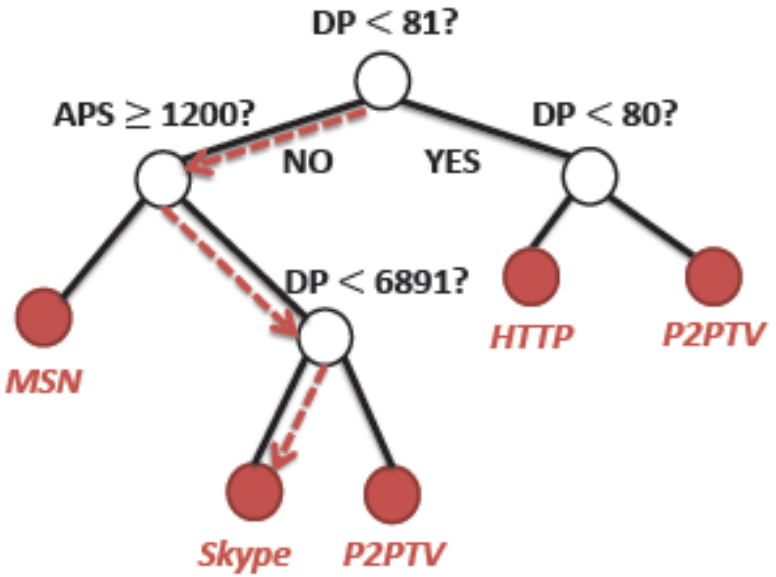
- Programmable interconnect
  - Connect logic cells → complex functions

Outline

- Introduction
- Background
- Data Structures & Algorithms
  - Rule table
  - Optimizations
  - Dynamic updates
- Architecture
- Evaluation
- Conclusion

Converting Decision-trees

- Rule table
  - Alternative representation of a decision-tree
  - Each leaf node → a root-to-leaf path → a rule
  - Intuition: parallel searching on the rule table

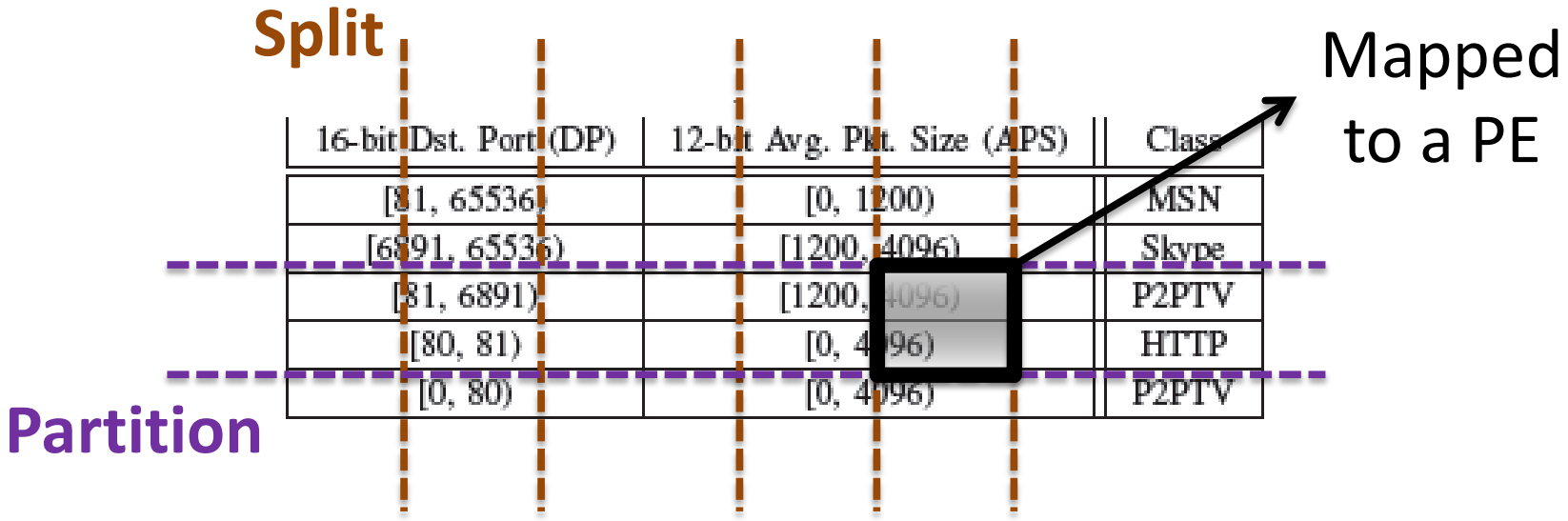


16-bit Dst. Port (DP)	12-bit Avg. Pkt. Size (APS)	Class
[81, 65536)	[0, 1200)	MSN
[6891, 65536)	[1200, 4096)	Skype
[81, 6891)	[1200, 4096)	P2PTV
[80, 81)	[0, 4096)	HTTP
[0, 80)	[0, 4096)	P2PTV

$N = 5$ rules, $M = 2$ fields,
$W = 28$ data width

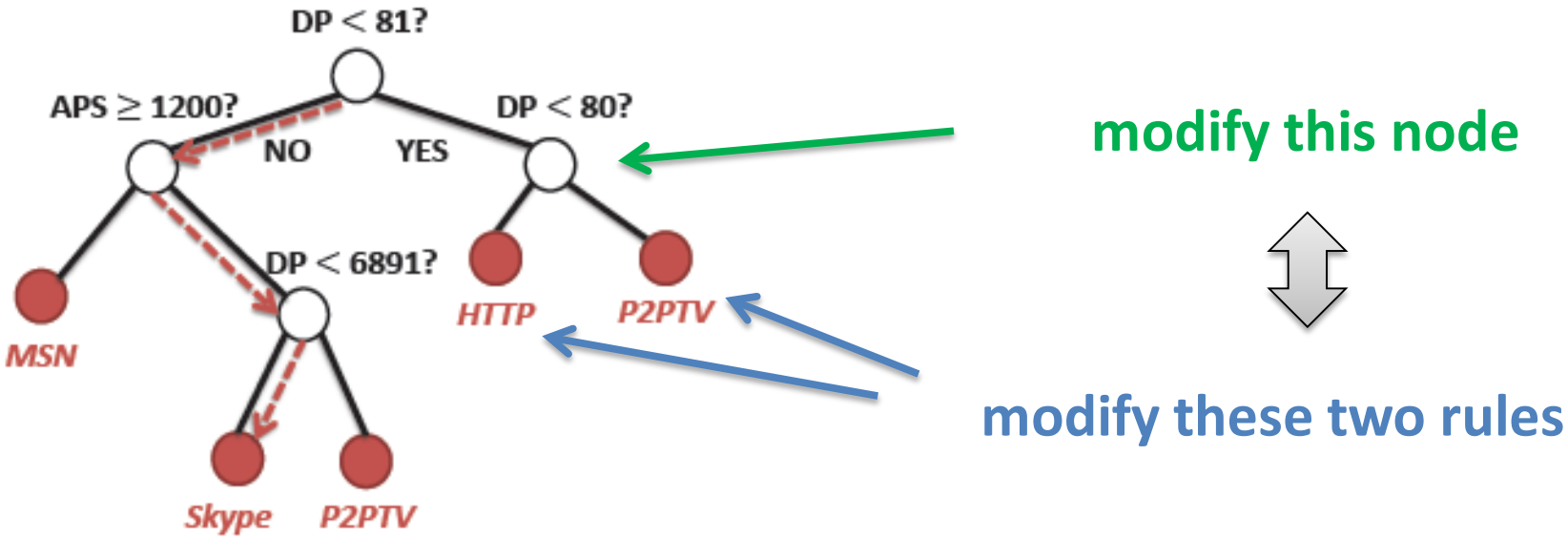
Optimization Techniques

- Splitting (vertically) and partitioning (horizontally)
  - Multiple W_m-bit sub-fields
  - Multiple sub-tables having N_n rules



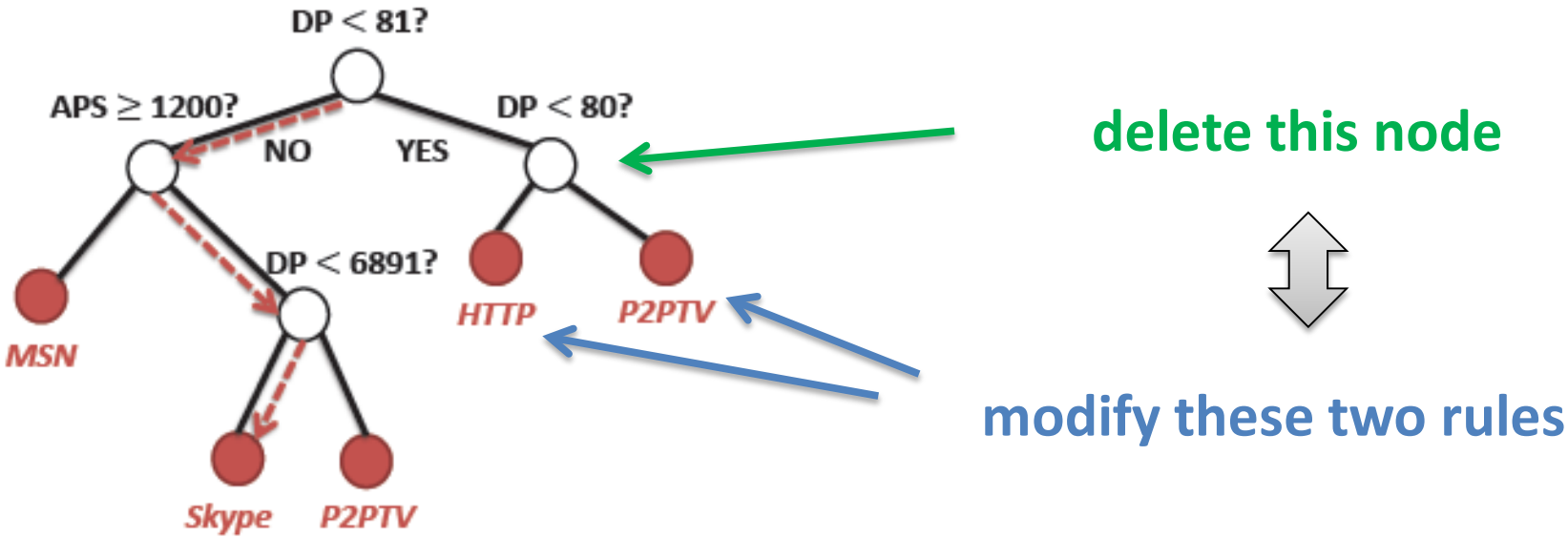
Dynamic Updates (1)

- Modification of a tree node
  - Identify the sub-rules to be modified
  - Change the values stored, or
  - Change the polarity of the comparison



Dynamic Updates (2)

- Deletion of a tree node
  - Modify sub-rule(s) to produce “invalid” sub-rule(s)



Dynamic Updates (3)

- Inserting an internal node
  - All the descendants of the new node has to be modified

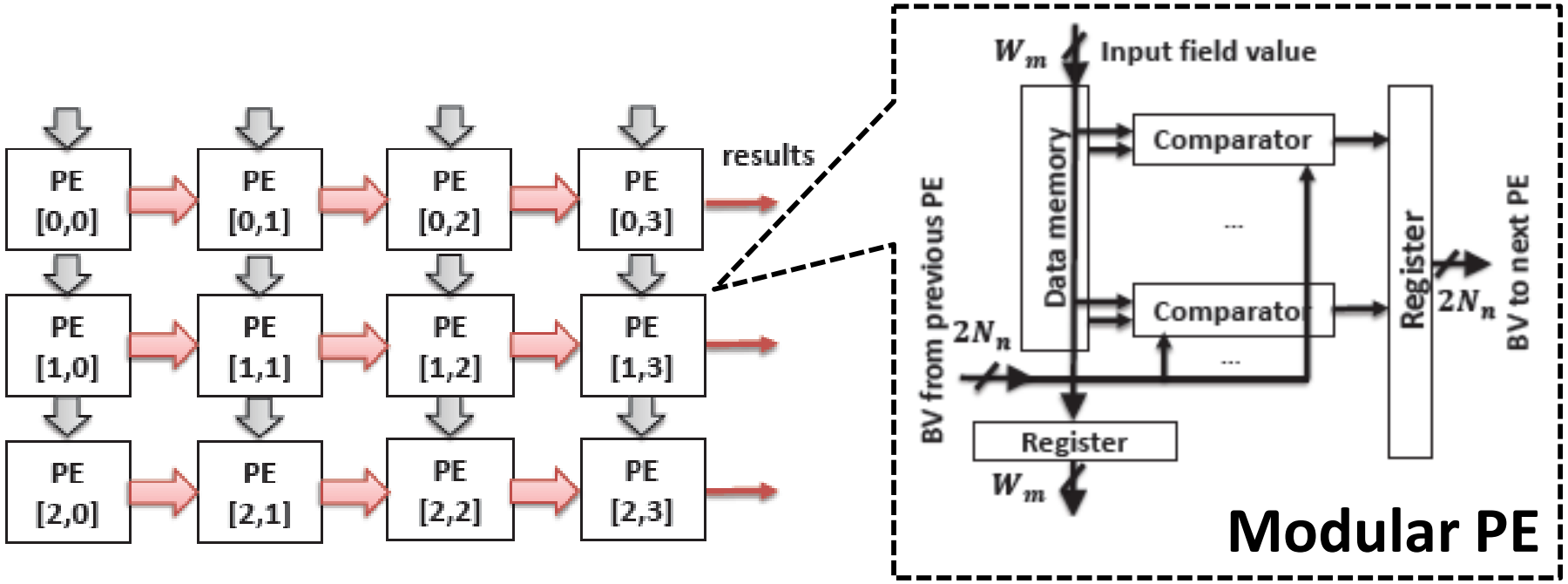
- Inserting a leaf node
  - Pre-allocate $(N_{\max} - N)$ invalid rules as placeholders
  - Modifying an invalid rule into the new rule

Outline

- Introduction
- Background
- Data Structures & Algorithms
- Architecture
  - 2-dimensional pipelined architecture
- Evaluation
- Conclusion

Architecture

- 2-dimensional pipelined architecture
  - Each sub-table → a modular PE
  - Updates: memory write accesses propagate diagonally



Outline

- Introduction
- Background
- Data Structures & Algorithms
- Architecture
- Evaluation
  - Peak throughput
  - Resource consumption
- Conclusion

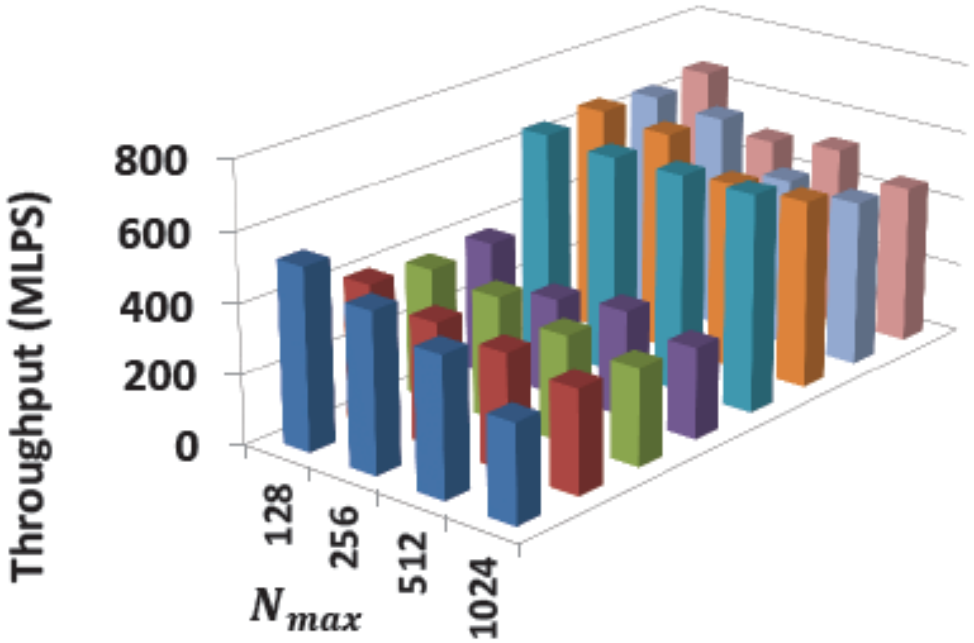
Experimental Setups

- Prototypes on state-of-the-art FPGA
  - Dual-port distributed RAM (distRAM)
  - Empirical optimizations: $W_m = 4, N_n = 2$

FPGA device	Virtex-7 XC7VX1140t FLG1930 -2L
No. of I/O pins	1100
No. of logic slices	218800
BRAM / distRAM	67 Mb / up to 17 Mb
Design Suite	Xilinx Vivado 2013.4
Programming Language	Verilog
Timing constraint	250 MHz
Reports	Post-place-and-route

Peak Throughput

- Comparison with state-of-the-art implementation
  - $1.8{\sim}2.0\times$ peak throughput (without any updates)
  - Sustain high lookup throughput as $N_{\max} \uparrow$ and/or $W \uparrow$



Resource Consumption

- No. of logic slices, no. of I/O pins
  - Largest design: $N = 512$ leaf nodes, $W = 320$ bits data

W	N_{max}	Logic slices (%)	I/O (%)	Peak Throughput (MLPS)
80	128	4.95	19.45	597.01
	256	9.04	21.63	626.37
	384	13.30	23.81	614.06
	512	18.45	26.00	569.80
160	128	9.11	35.81	700.28
	256	17.74	38.00	580.04
	384	25.95	40.18	566.09
	512	35.17	42.36	564.97
240	128	13.36	52.18	591.89
	256	26.58	54.36	565.61
	384	39.41	56.54	564.17
	512	53.61	58.72	557.56
320	128	17.64	68.54	609.57
	256	34.95	70.72	585.30
	384	55.00	72.90	538.78
	512	70.72	75.09	536.19

Resources consumed evenly

Outline

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

Conclusion

- Scalable and dynamically updatable lookup engine
  - Mapping decision-trees onto hardware more efficiently
  - Works for generic decision-trees in HPC

- Future work
  - Self-learning on this architecture
  - Investigate performance tradeoffs on various platforms

Questions?

Thanks

http://ganges.usc.edu