Fast and Smoothed Packet Classification

Recent advancement of research partnership
with V. Prasanna and S.-H. Teng

Jun Li & Yibo Xue
with contributions from our students Yaxuan Qi, Jeffrey Fong, Xiaoqi Ren, et al.

Outline

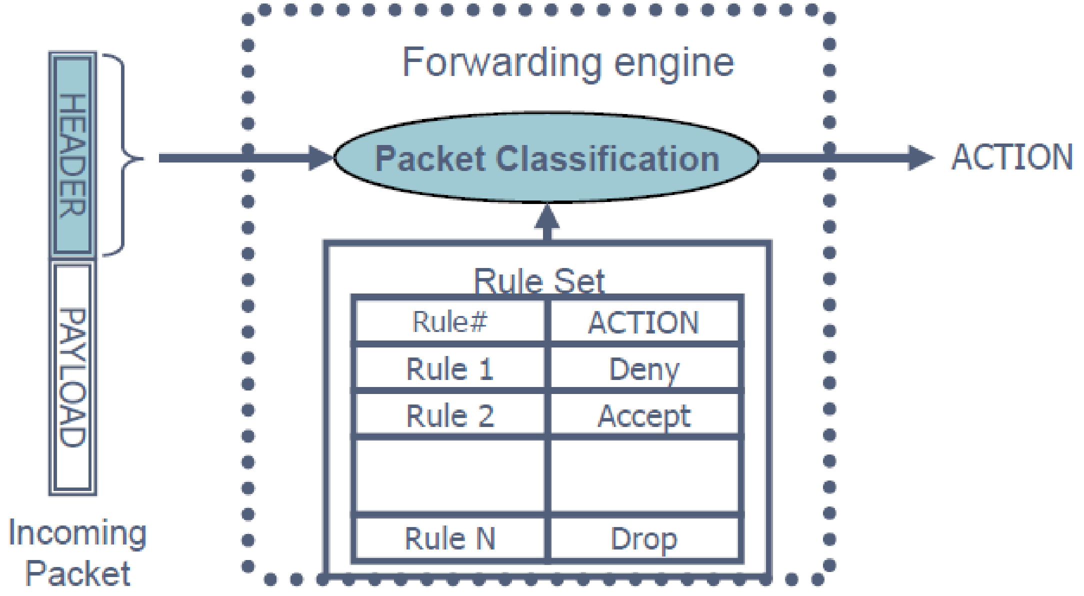
- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- Future Work

Outline

- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- Future Work

Packet Classification Problem

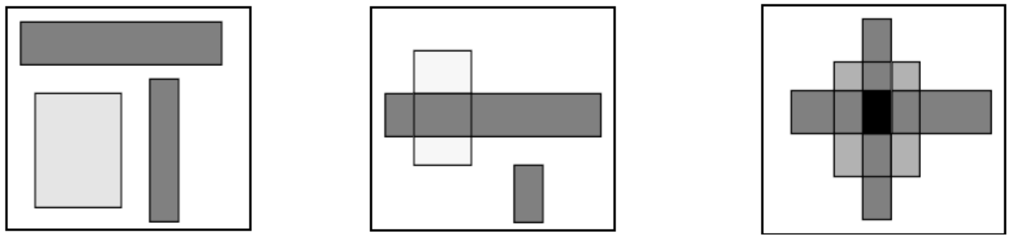
- To identify and associate each packet to a specific rule
- May match multiple rules
- Used for:
  - Routing
  - FW, IDS/IPS, &AV
  - LB & TE
  - OpenFlow & SDN



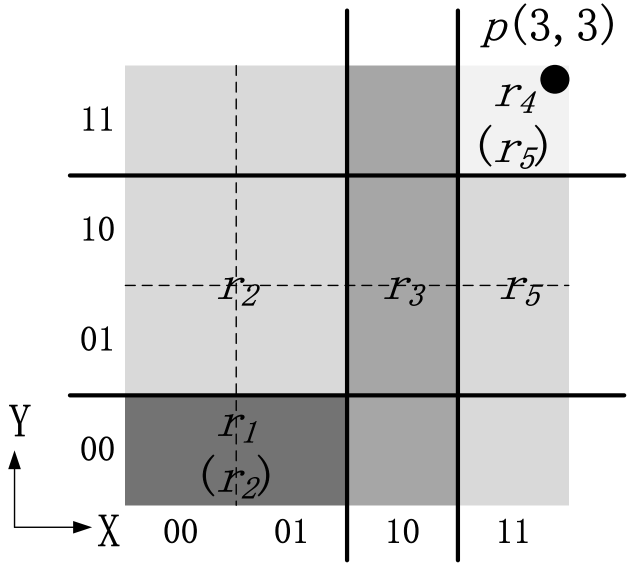
	Field 1 (sIP)	Field 2 (dPort)	...	Field F (protocol)	Action
Rule 1	166.111.72.50/21	80	...	UDP	Deny
Rule 2	166.168.3.0/24	53	...	TCP	Accept
...	...	...	...	...	...
Rule N	0.0.0.0/0	0~65535	...	ANY	Drop

Theoretical Complexity

- Point location problem



- Very high complexity



		H-Trie	H-Tree	S-Trie	S-Tree
d=1	time	$\Theta(W)$	$\Theta(\log n)$	$\Theta(W)$	$\Theta(\log n)$
	space	$\Theta(n * W)$	$\Theta(n * \log n)$	$\Theta(n * W)$	$\Theta(n * \log n)$
d>1	time	$\Theta(W^{d-1})$	$\Theta(\log^{d-1} n)$	$\Theta(d * W)$	$\Theta(d * \log n)$
	space	$\Theta(n * W^{d-1})$	$\Theta(n * \log^{d-1} n)$	$\Theta(n^d * dW)$	$\Theta(n^d * d\log n)$

Performance in Practice

THEORETICAL VS. PRACTICAL COMPLEXITY FOR REAL-LIFE RULES

Rule Sets	*# rules*	*# non-over ranges in each field (theoretical)*	*# non-over ranges in sIP (practical)*	*# non-over ranges in dIP (practical)*	*# non-over ranges in sPT (practical)*	*# non-over ranges in dPT (practical)*	*# non-over rectangles (theoretical)*	*# non-over rectangles (practical)*
FW1	269	539	100	111	23	77	8.44×10^{10}	1.97×10^{7}
FW1-100	92	185	19	45	20	48	1.17×10^{9}	8.21×10^{5}
FW1-1K	791	1583	221	314	23	75	6.28×10^{12}	1.20×10^{8}
FW1-5K	4653	9307	3429	5251	23	77	7.50×10^{15}	3.19×10^{10}
FW1-10K	9311	18623					1.20×10^{17}	1.71×10^{11}
ACL1	752	1505					5.13×10^{12}	9.67×10^{6}
ACL1-100	98	197					1.51×10^{9}	4.03×10^{5}
ACL1-1K	916	1833					1.13×10^{13}	1.32×10^{7}
ACL1-5K	4415	8831					6.08×10^{15}	2.42×10^{8}
ACL1-10K	9603	19207					1.36×10^{17}	1.90×10^{9}
IPC1	1550	3101					9.25×10^{13}	3.40×10^{8}
IPC1-100	99	199					1.57×10^{9}	9.49×10^{6}
IPC1-1K	938	1877	559	796	49	78	1.24×10^{13}	1.70×10^{9}
IPC1-5K	4460	8921	886	2125	59	93	6.33×10^{15}	1.03×10^{10}
IPC1-10K	9037	18075	2377	4604	59	94	1.07×10^{17}	6.07×10^{10}

Note: sIP, dIP, sPT and dPT are source IP, destination IP, source Port and destination Port; FW, ACL, IPC are firewall policies, access control lists, and IP chain rules

wst-case: 5.13×10^{12}
practical: 9.67×10^{6}

- Few applications reach the worst case bound
- Real-life rule sets have geometrical redundancy

Progress of Joint Research

- Efficient Algorithms
  - Exploiting real-life rule set redundancy
  - HyperSplit Algorithm (Infocom)
- Fast Speed
  - Using SRAM-based solution on FPGA
  - 100Gbps Throughput (FPT)
- Smoothed Analysis
  - Introducing Sampling-based Smoothed Analysis
  - Practical evaluation (submitted)

Outline

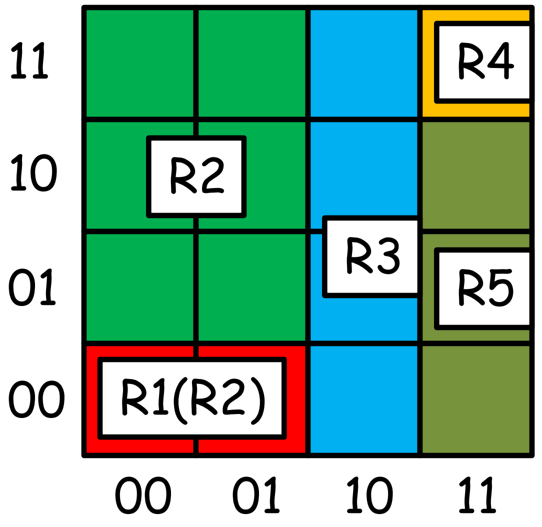
- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- Future Work

HyperSplit

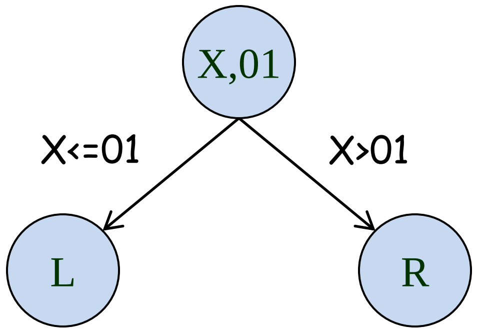
- Memory-efficient packet classification algorithm
  - Uses 10% of the memory that other comparable algorithms requires
- Optimized k-d tree data structure
- Uses heuristics to select the most efficient splitting point on a specific field

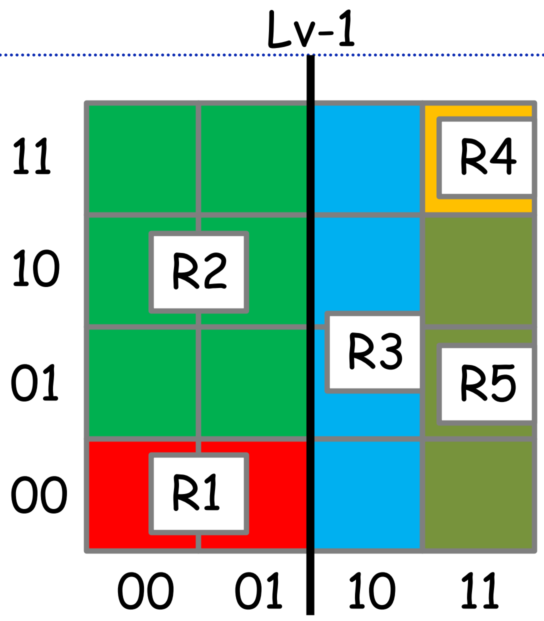
Example

Rule	*Priority*	*Field-X*	*Field-Y*
R1	1	00~01	00~00
R2	2	00~01	00~11
R3	3	10~10	00~11
R4	4	11~11	11~11
R5	5	11~11	00~11

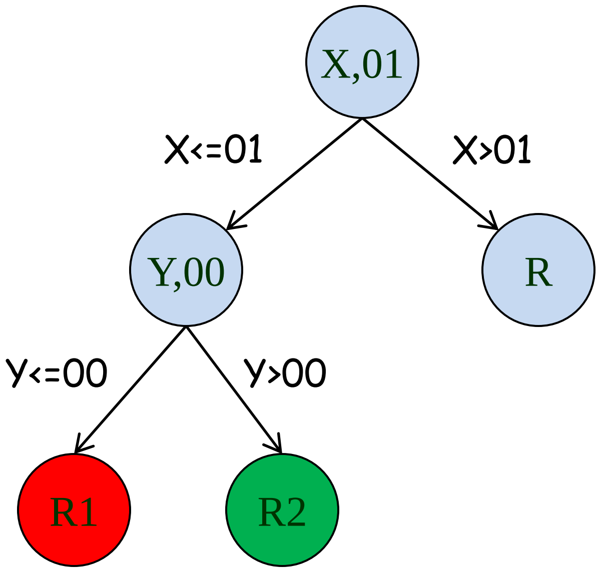


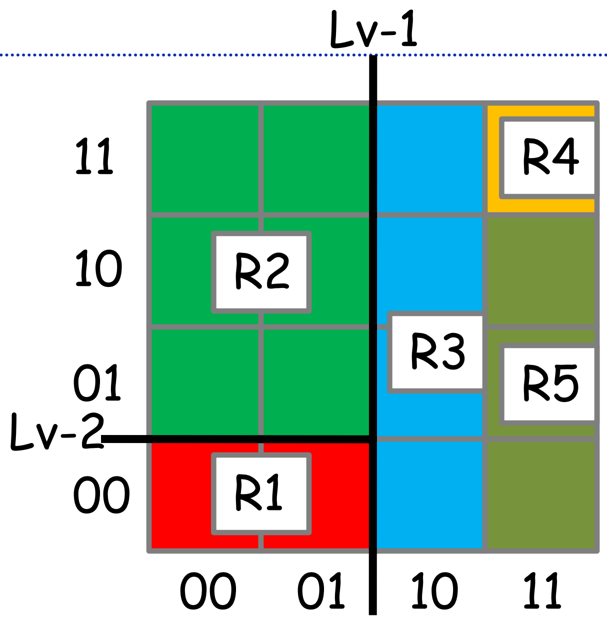
Example



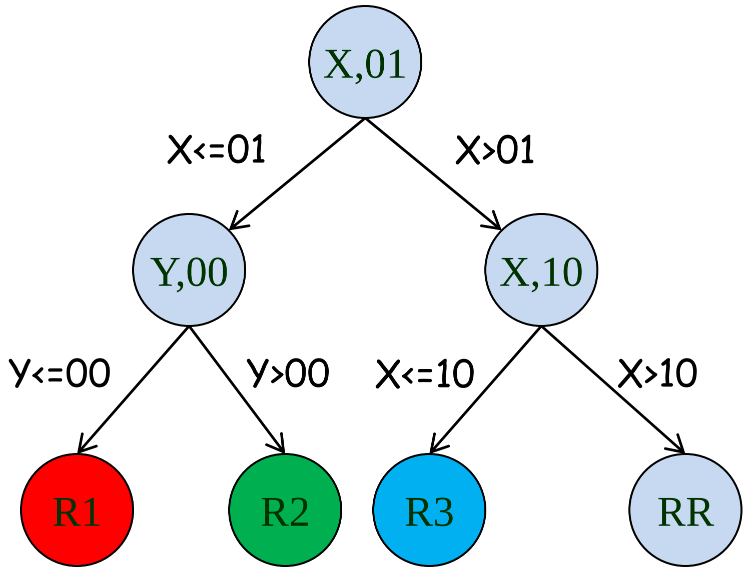


Example

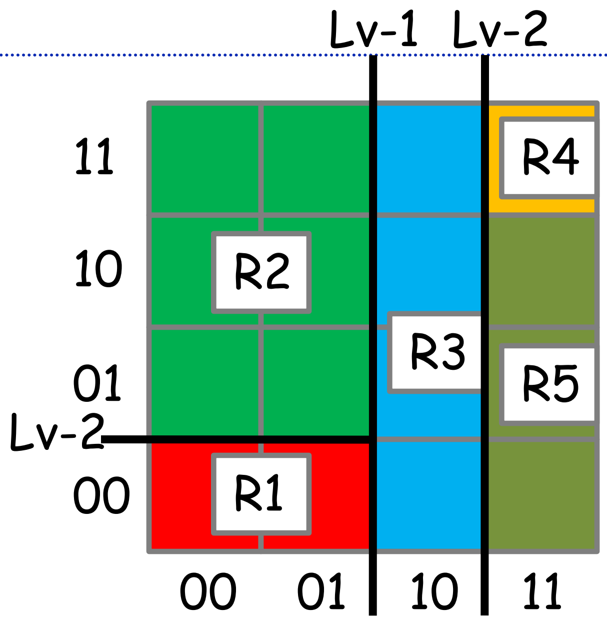

X,01
X<=01
X>01
Y,00
R
Y<=00
Y>00
R1
R2


Lv-1
Lv-2
11
10
01
00
00
01
10
11
R4
R2
R3
R5
R1

Example

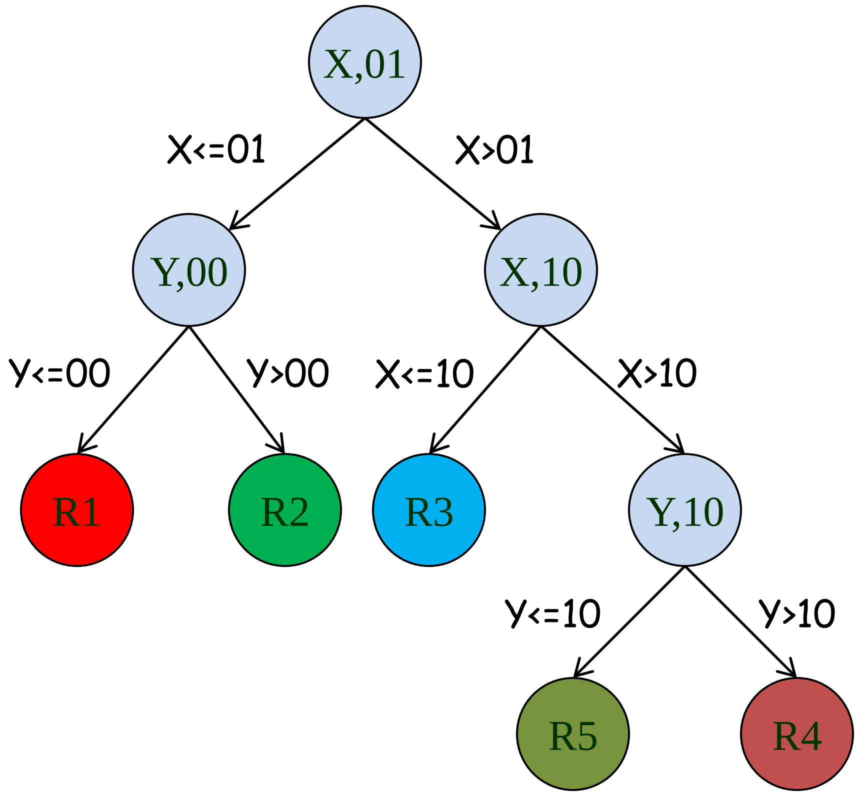


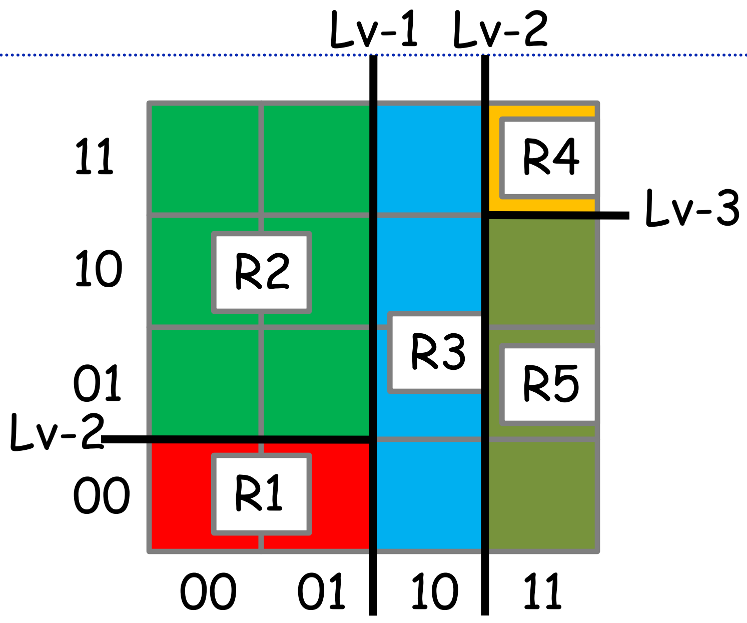
X,01
X<=01
X>01
Y,00
X,10
Y<=00
Y>00
X<=10
X>10
R1
R2
R3
RR



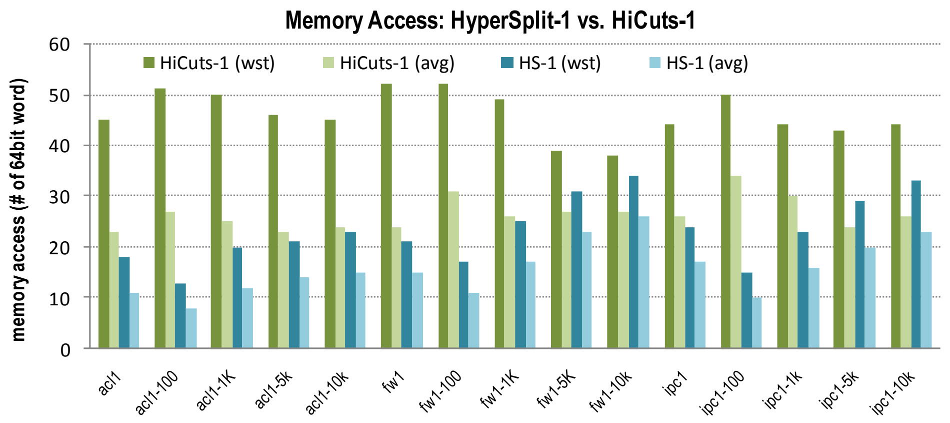
Lv-1
Lv-2
11
10
01
Lv-2
00
R1
R2
R3
R4
R5
00
01
10
11

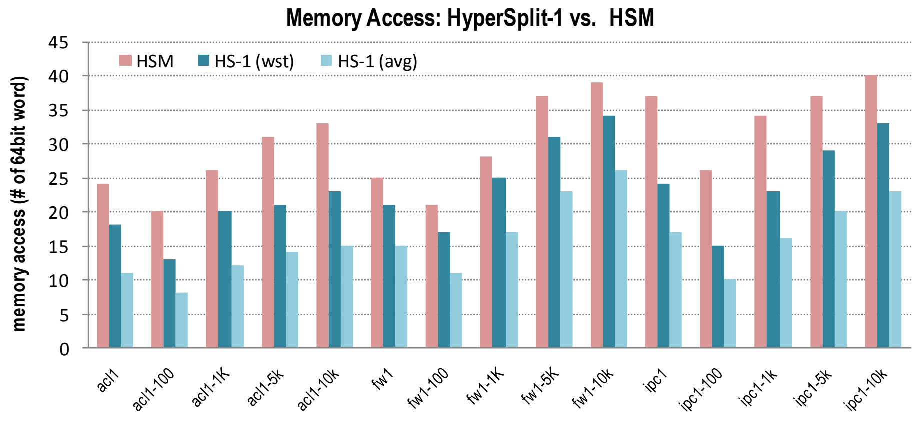
Example

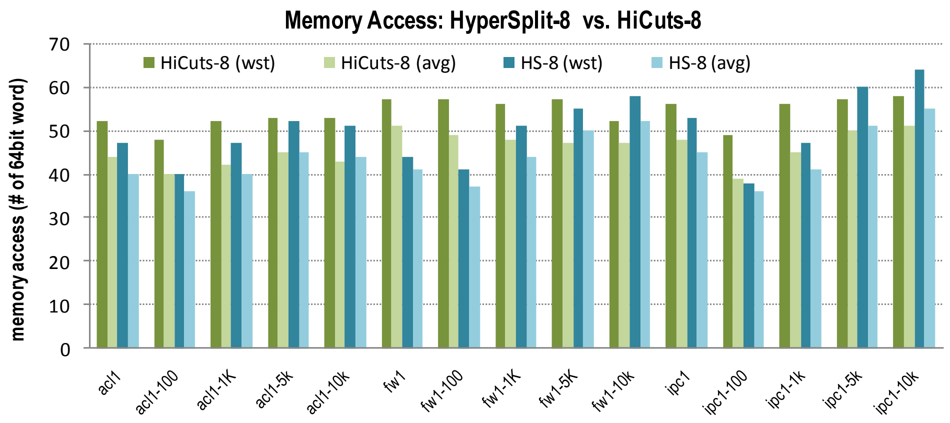

X,01
X<=01
X>01
Y,00
X,10
Y<=00
Y>00
X<=10
X>10
R1
R2
R3
Y,10
Y<=10
Y>10
R5
R4


Lv-1
Lv-2
Lv-3
Lv-2
11
10
01
00
00
01
10
11
R1
R2
R3
R4
R5

Memory Access



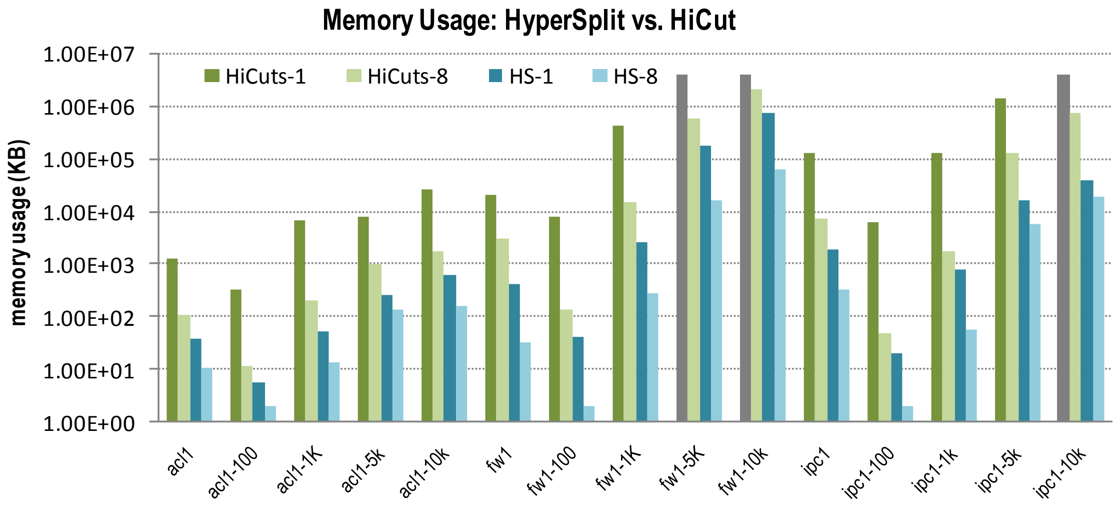


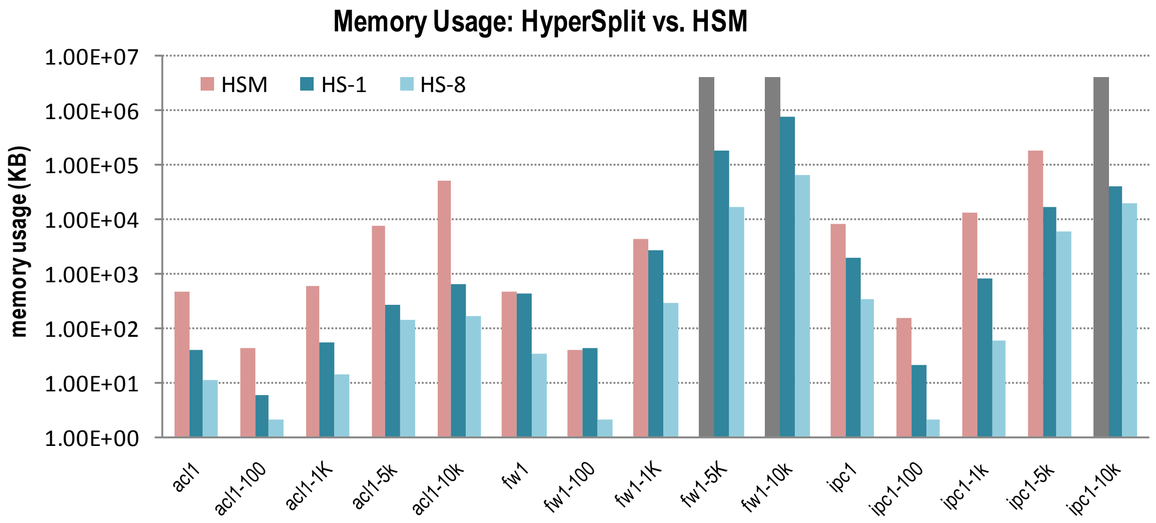


- HyperSplit-1 vs. HiCuts-1
  - 50~80% less access
- HyperSplit-8 vs. HiCuts-8
  - 10~30% less access
- HyperSplit-1 vs. HSM
  - 20~50% less access

Y. Qi, L. Xu, B. Yang, Y. Xue, and J. Li, Packet classification algorithms: from theory to practice, INFOCOM, pp. 648-656, 2009.

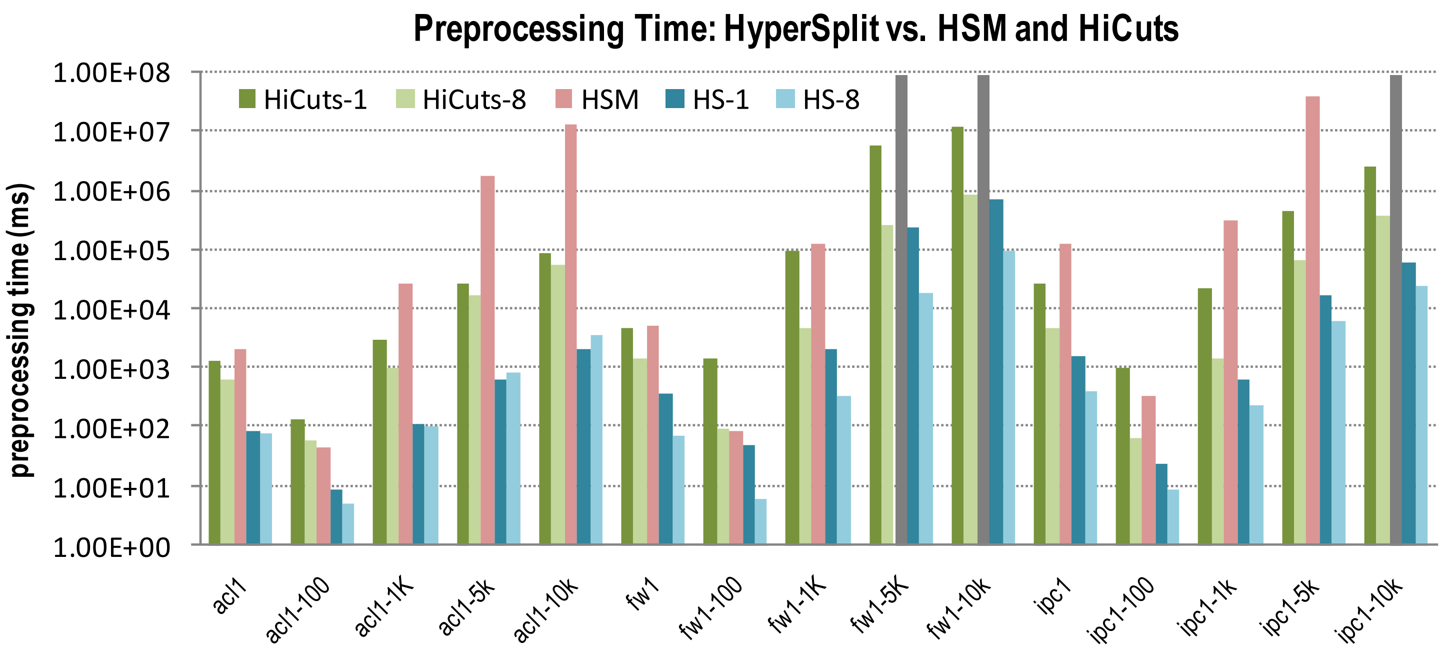
Memory Usage





- HyperSplit-1 vs. HiCuts-1
  - 1~2 orders less memory
- HyperSplit-8 vs. HiCuts-8
  - 1~2 orders less memory
- HyperSplit-1 vs. HSM
  - 1~2 orders less memory

Preprocessing Time



- HyperSplit-1 vs. HiCuts-1: 1~2 orders less time
- HyperSplit-8 vs. HiCuts-8: About 1 orders less time
- HyperSplit-1 vs. HSM: 1~4 orders less time

Outline

- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- Future Work

Existing Solutions

SRAM Based

- Advantage:
  - Price
  - (generally) # of Rules
- Disadvantage
  - Speed

TCAM Based

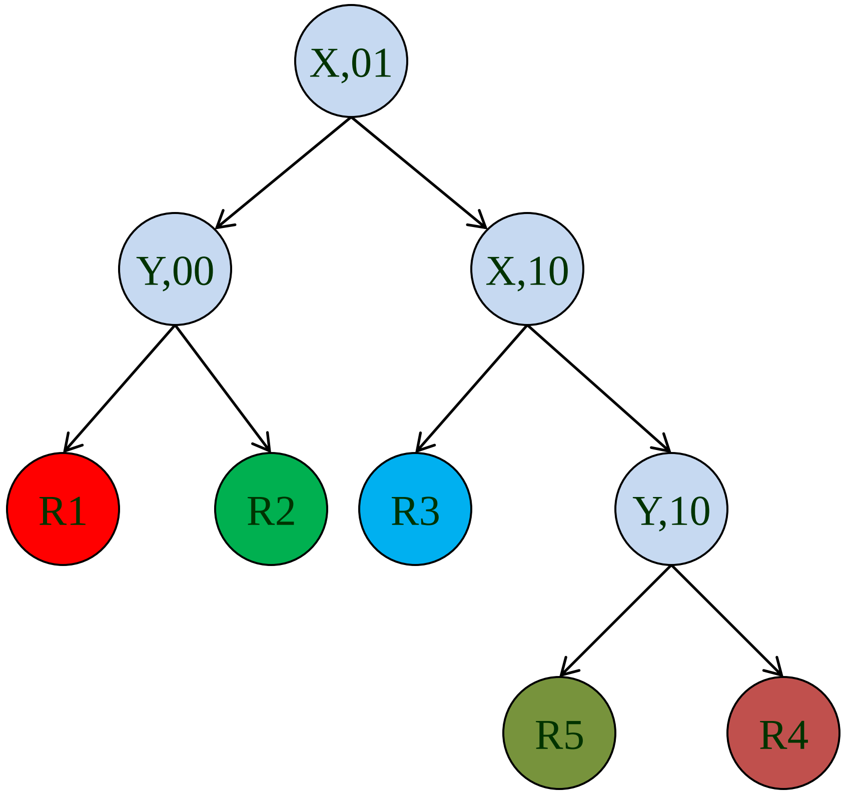
- Advantage
  - Speed
- Disadvantage
  - Price
  - Power consumption
  - Chip size
  - Range to Prefix Conversion

W. Jiang and V.K. Prasanna, Field-split parallel architecture for high performance multi-match packet classification using FPGAs, SPAA, pp. 648–656, 2009.

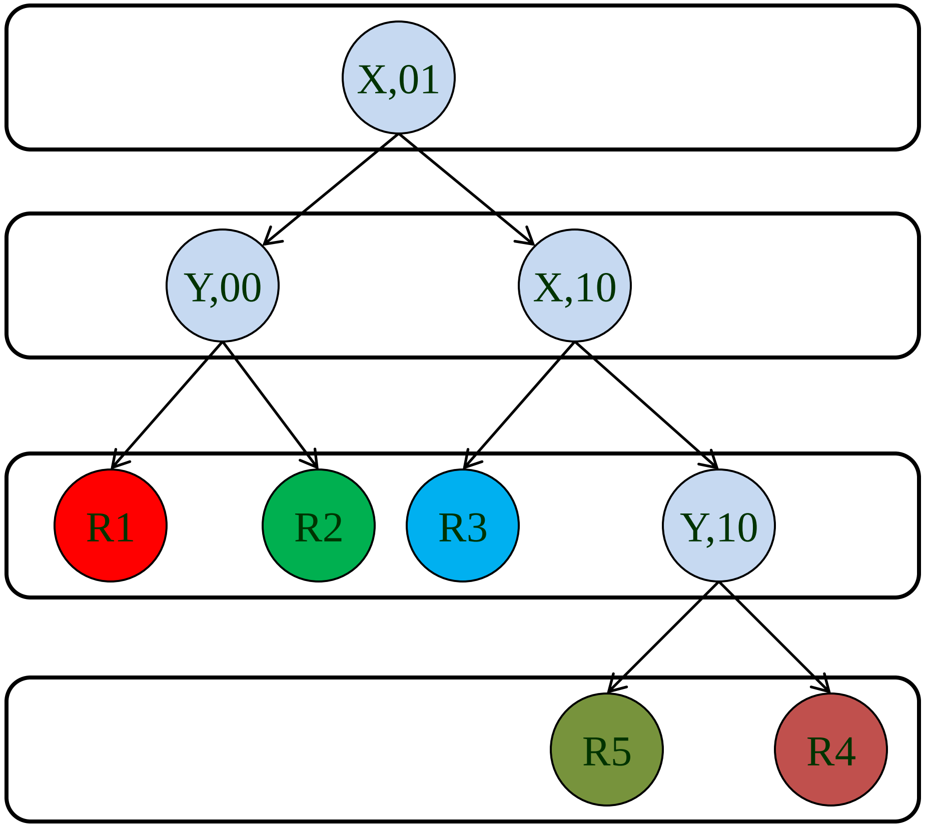
Challenges & Goals

- Memory Usage
  - Needs to be memory efficient that can support large rule sets
- High Performance
  - Requires high throughput and deterministic performance
- On-the-fly update
  - To allow rules to be changed and updated without downtime

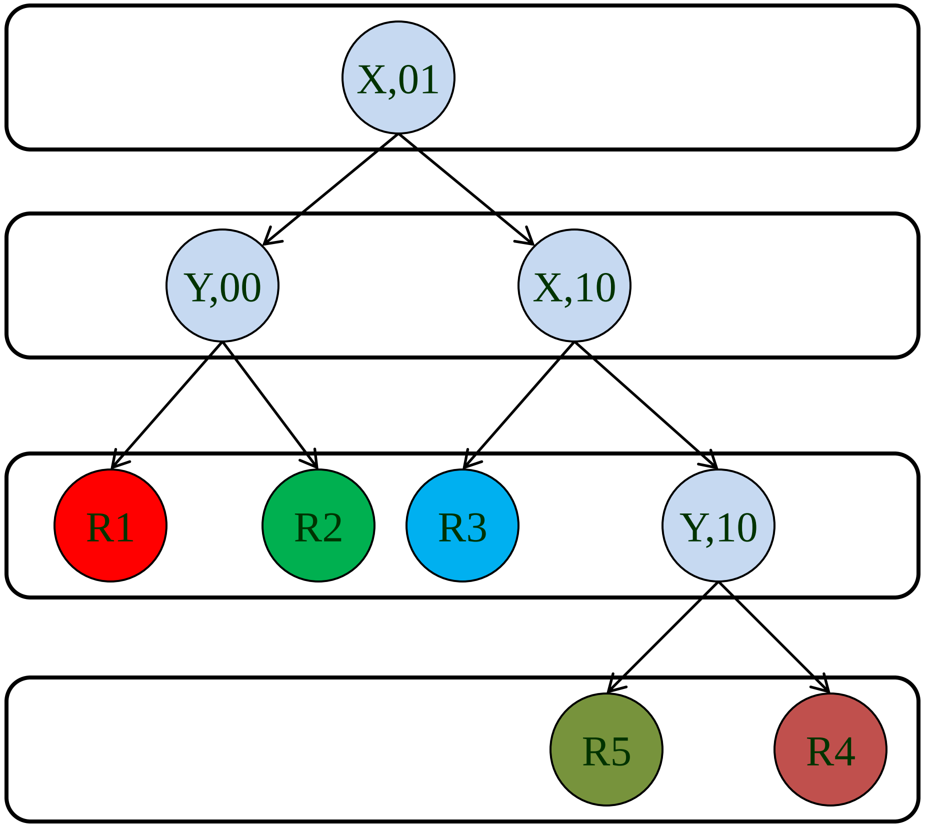
Mapping Decision Tree into Hardware

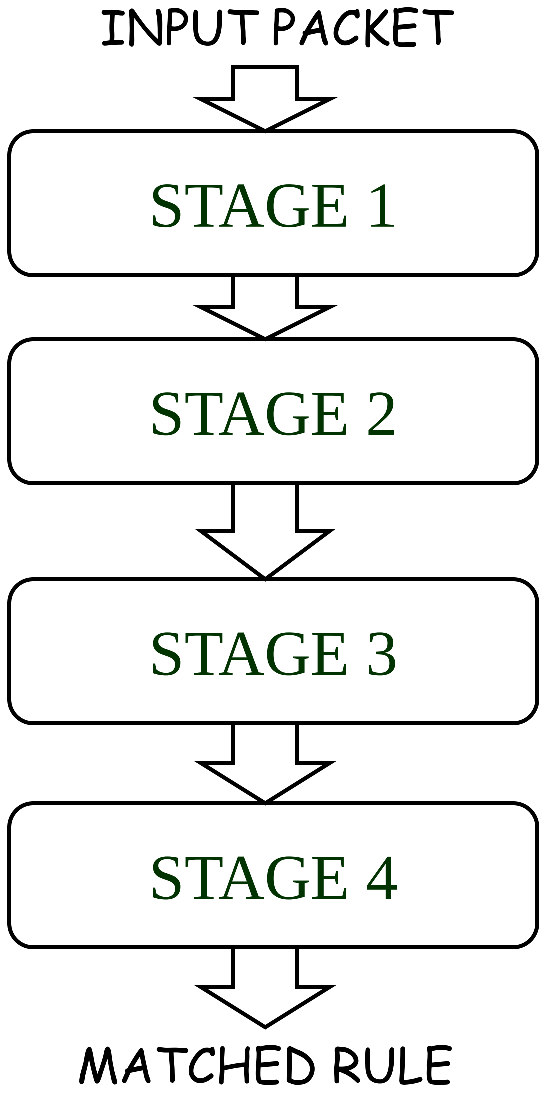


Mapping Decision Tree into Hardware



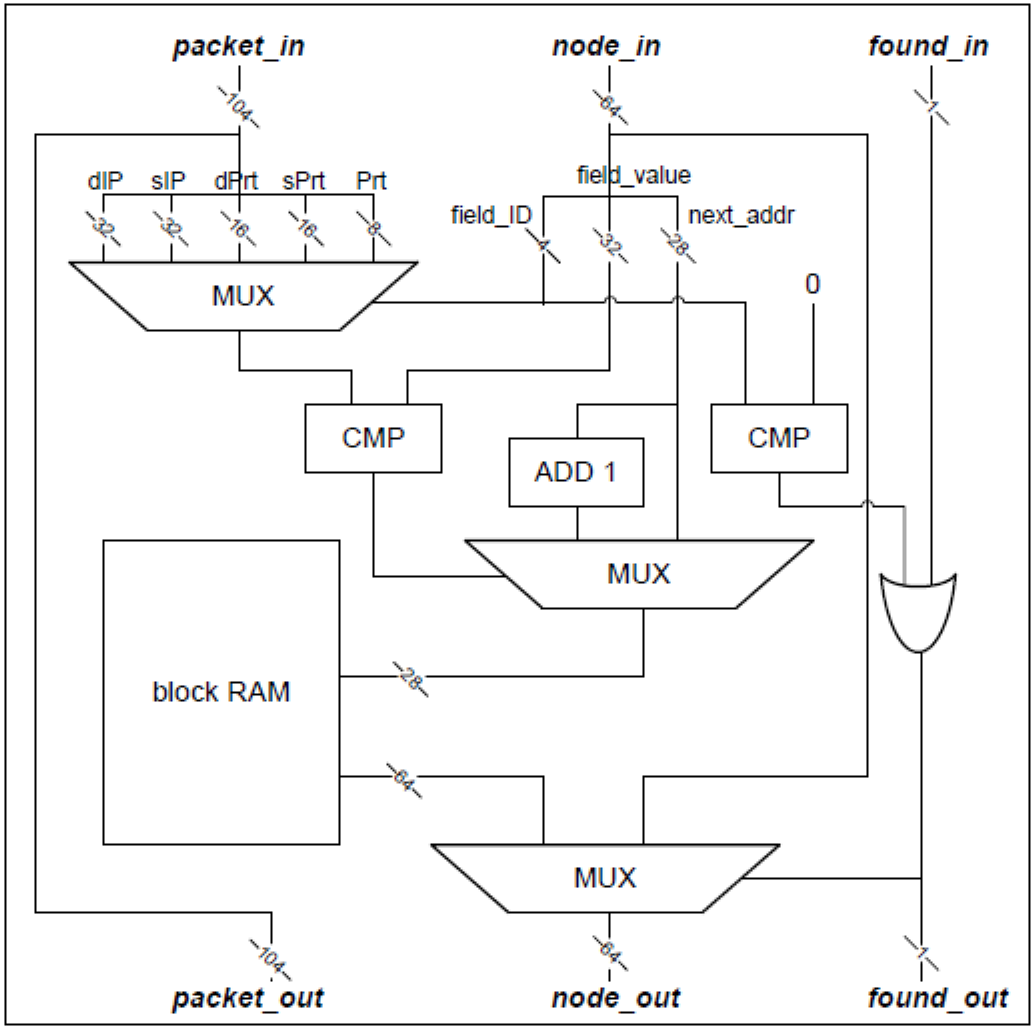
Mapping Decision Tree into Hardware





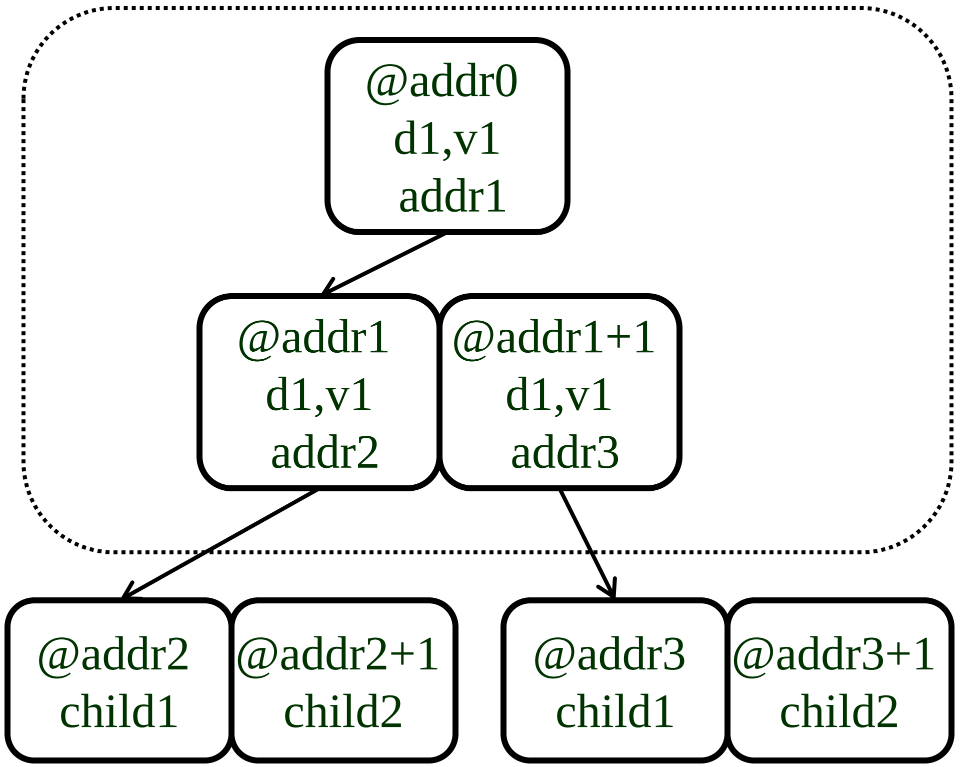
Hardware Implementation

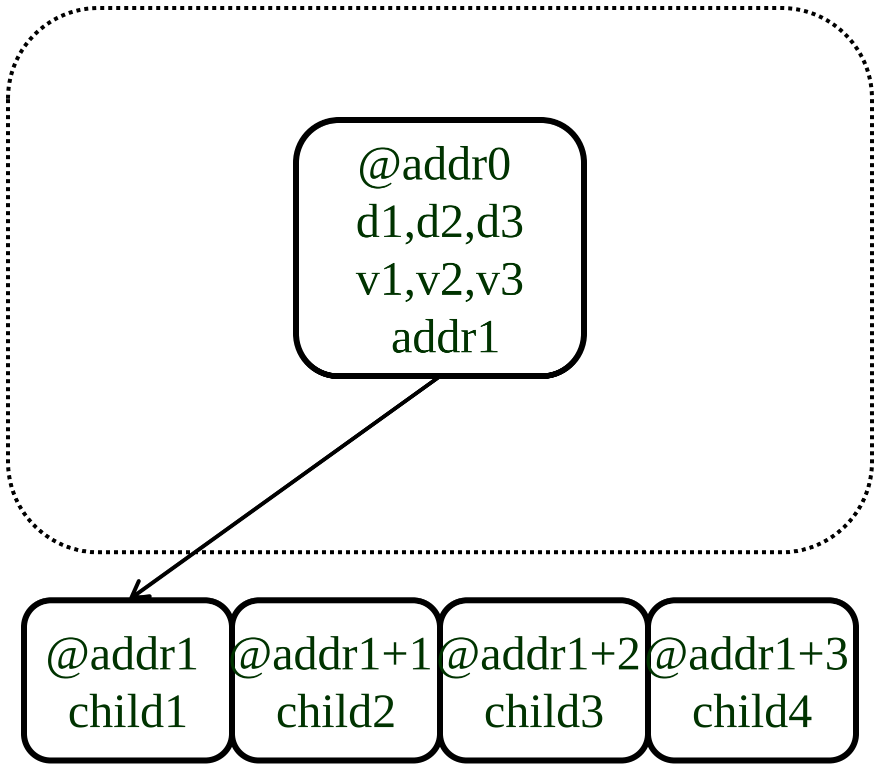
STAGE n

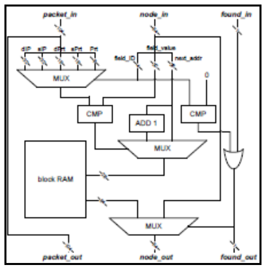


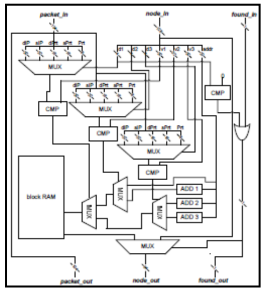
Architecture Optimization (1)

- Node-merging: Pipeline Depth Reduction



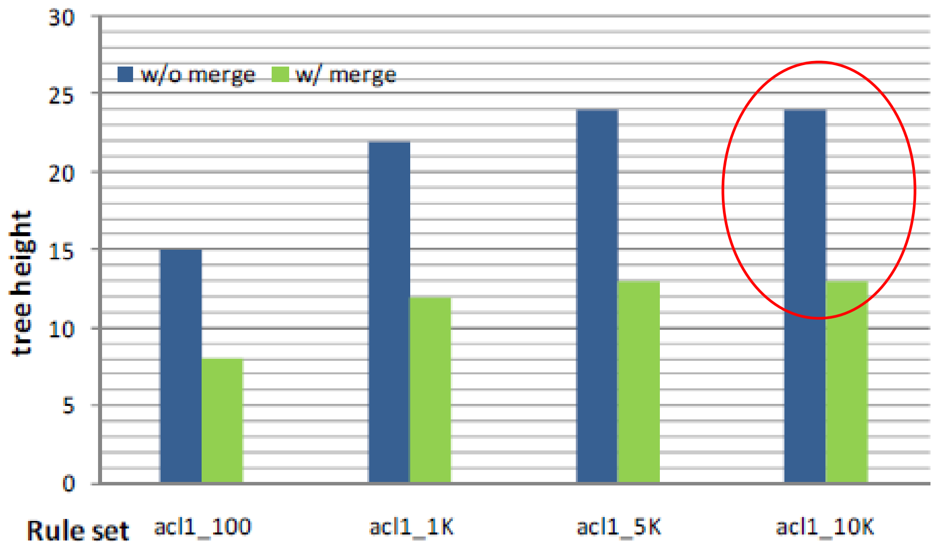




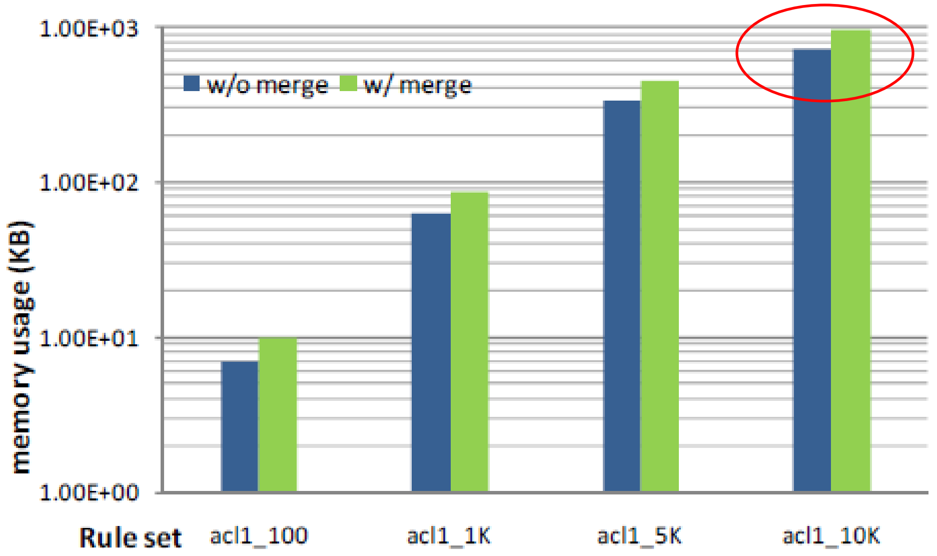


Algorithm Evaluation (1)

Node-merging Optimization



Tree heights with and without node-merging



Memory usage with and without node-merging

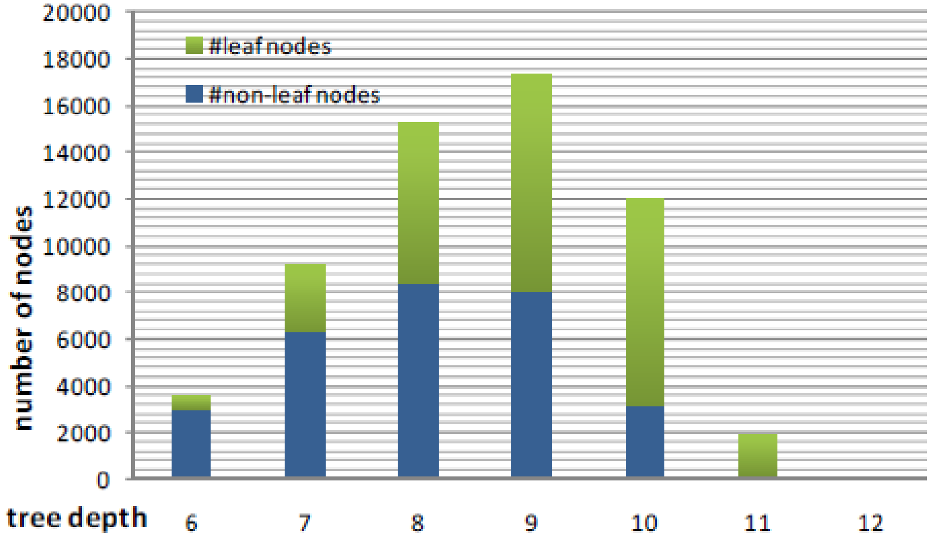
- Reduce tree height (pipeline depth) by almost 50%!
- Minimal memory overhead

Architecture Optimization (2)

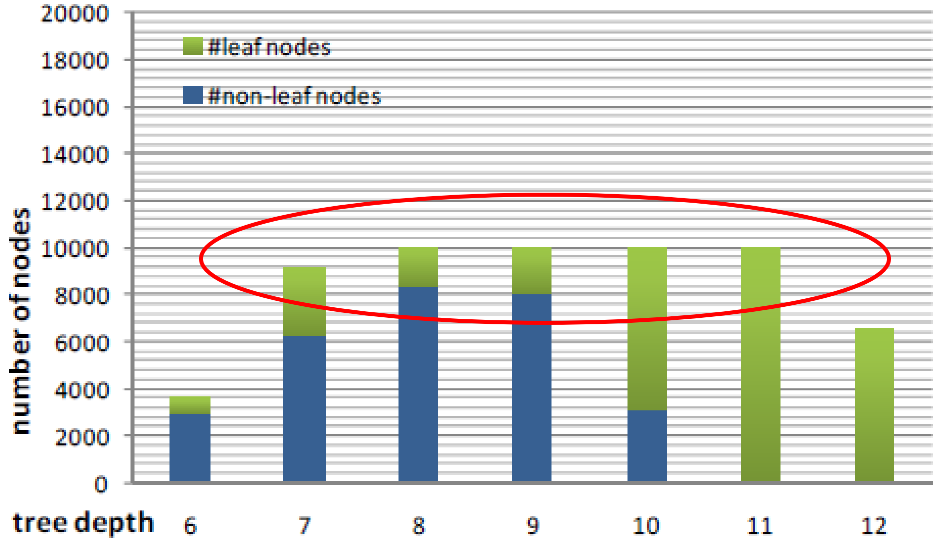
- Leaf-pushing: Controlled BRAM Allocation
  - Sizes of BRAM on each stage needs to be predetermined
  - Different rule sets will result in different memory usage per stage
  - Limits the size of a certain stage by pushing leafs to lower levels of the pipeline

Algorithm Evaluation (2)

- Leaf-pushing Optimization



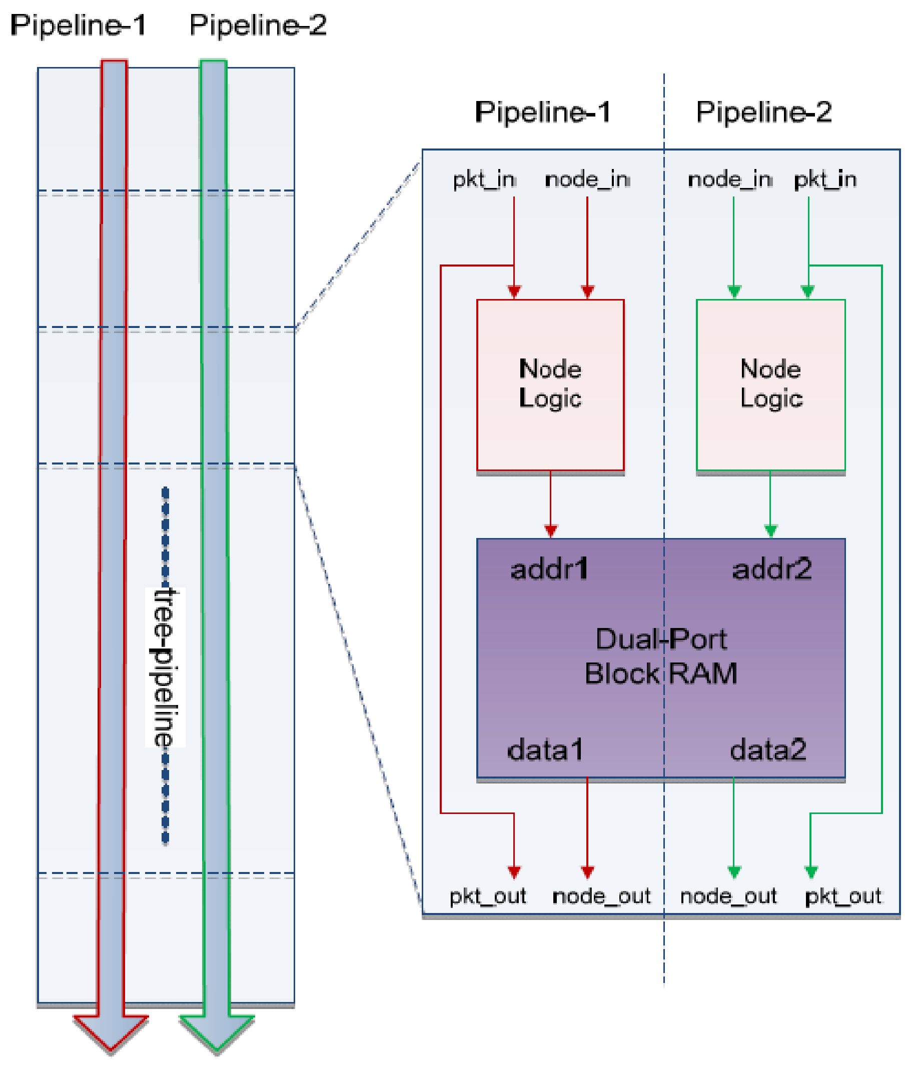
Nodes distribution without leaf pushing



Nodes distribution with leaf pushing

Architecture Optimization (3)

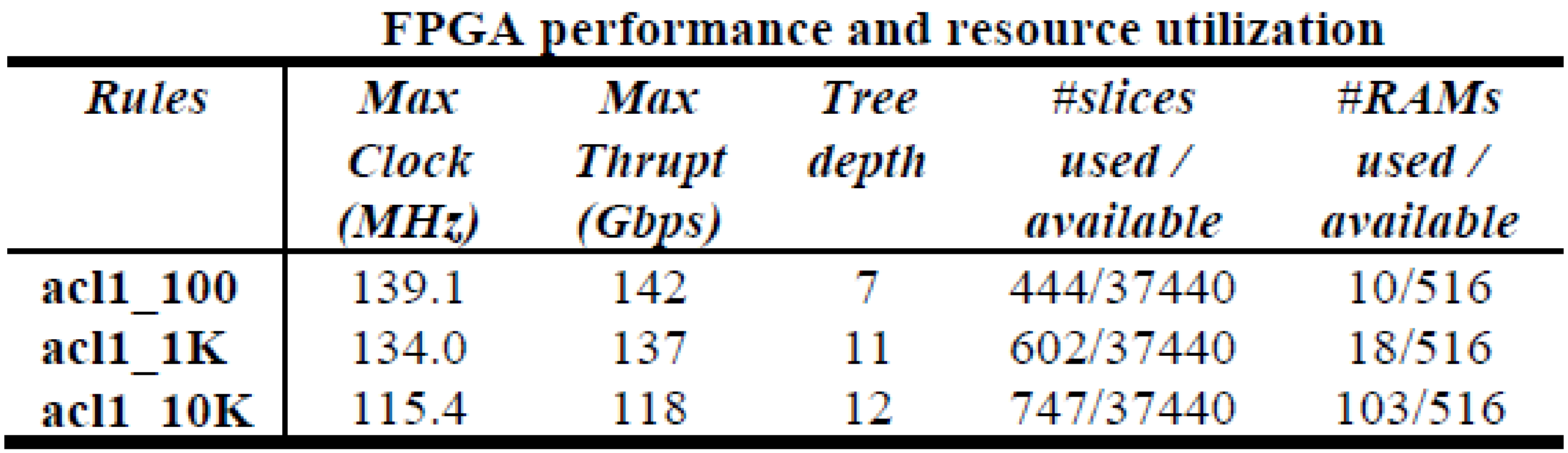
- Dual Pipeline
  - Take advantage of dual-port BRAM
  - Double the throughput without increasing memory usage



Test Setup

- Tested with a publicly available rulesets from Washington University
  - Used the ACL 100, 1K, 5K, 10K rulesets
- Design is implemented on a Xilinx Virtex-6
  - Model: VC6VSX475T
  - Containing 7,640Kb Distributed RAM and 38,304Kb Block RAM
  - Using Xilinx ISE 11.5 tool

FPGA Performance

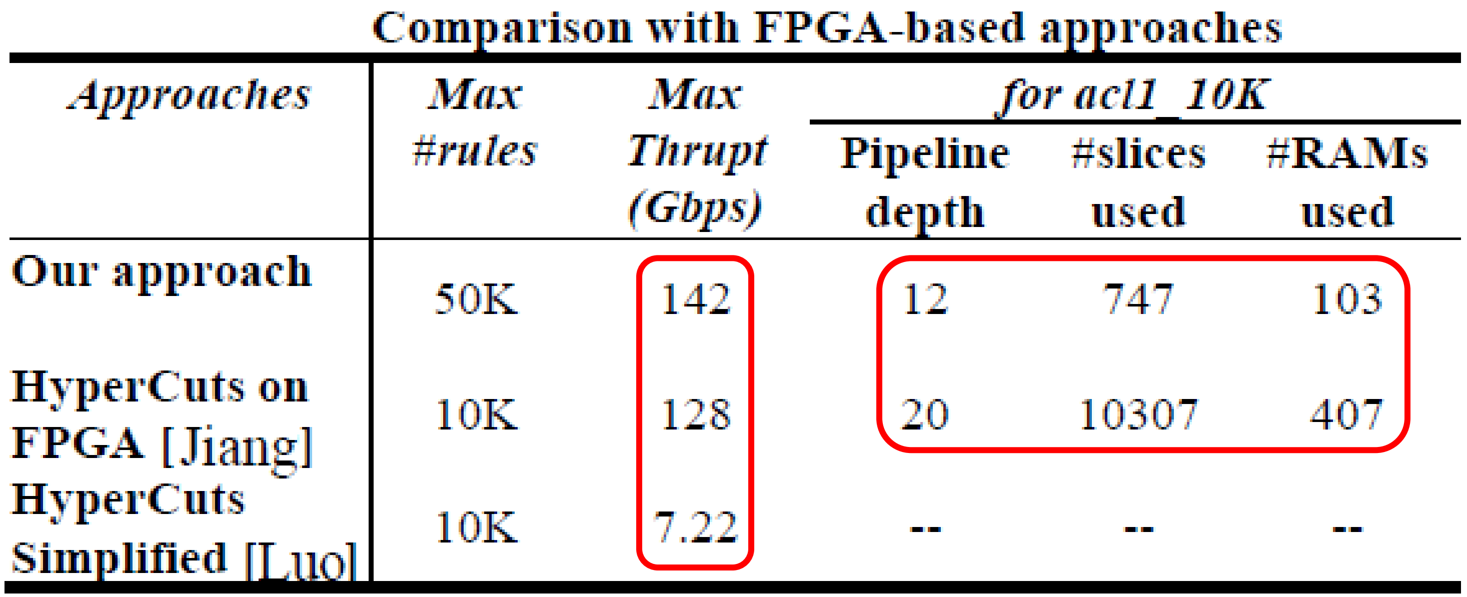


FPGA performance and resource utilization

Rules	*Max Clock (MHz)*	*Max Thrupt (Gbps)*	*Tree depth*	*#slices used / available*	*#RAMs used / available*
acl1_100	139.1	142	7	444/37440	10/516
acl1_1K	134.0	137	11	602/37440	18/516
acl1 10K	115.4	118	12	747/37440	103/516

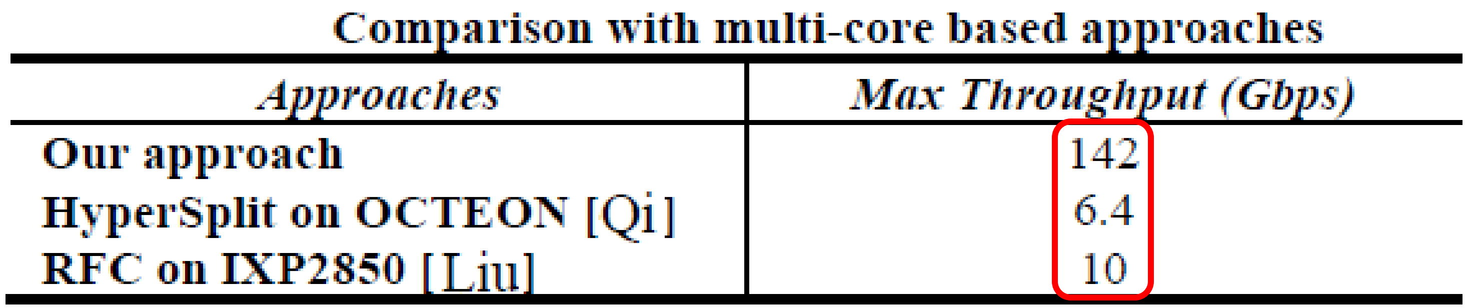
Y. Qi, J. Fong, W. Jiang, B. Xu, J. Li and V. Prasanna, Multi-dimensional packet classification on FPGA: 100 Gbps and beyond, FPT, pp. 241-248, 2010.

FPGA Comparison



Comparison with FPGA-based approaches

Approaches	*Max #rules*	*Max Thrupt (Gbps)*	*for acl1_10K*		
			Pipeline depth	#slices used	#RAMs used
Our approach	50K	142	12	747	103
HyperCuts on FPGA [Jiang]	10K	128	20	10307	407
HyperCuts Simplified [Luo]	10K	7.22	--	--	--



Comparison with multi-core based approaches

Approaches	*Max Throughput (Gbps)*
Our approach	142
HyperSplit on OCTEON [Qi]	6.4
RFC on IXP2850 [Liu]	10

Conclusion

- FPGA provides a flexible and excellent solution to the packet classification problem
- HyperSplit algorithm is suitable to hardware implementation with an efficient mapping
    - optimizations used to reduce tree length, constraint the memory usage of each stage, and improve performance
- Consume less resource than other FPGA-based solutions and much faster than multicore based solutions

Outline

- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- Future Work

Current Algorithm Evaluation

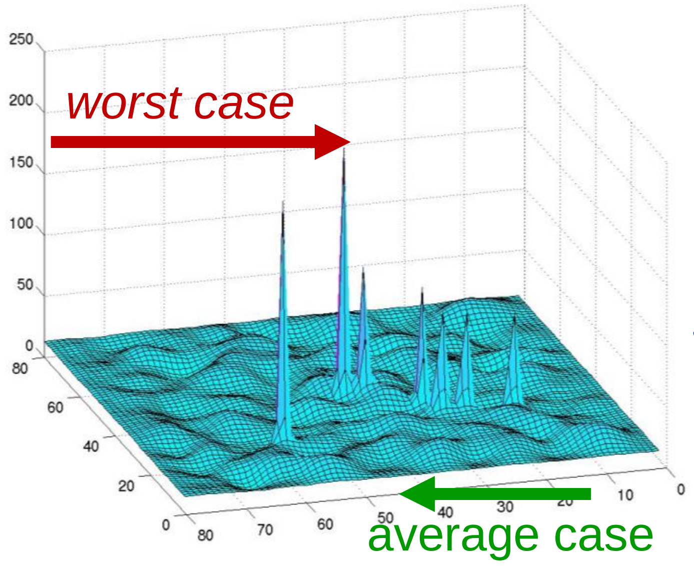
- Worst-case Evaluation:
  - Use the worst-case performance to evaluate the practical performance
- Drawbacks
  - may be defined in a contrived and extreme circumstance
  - may provide a significantly pessimistic evaluation result

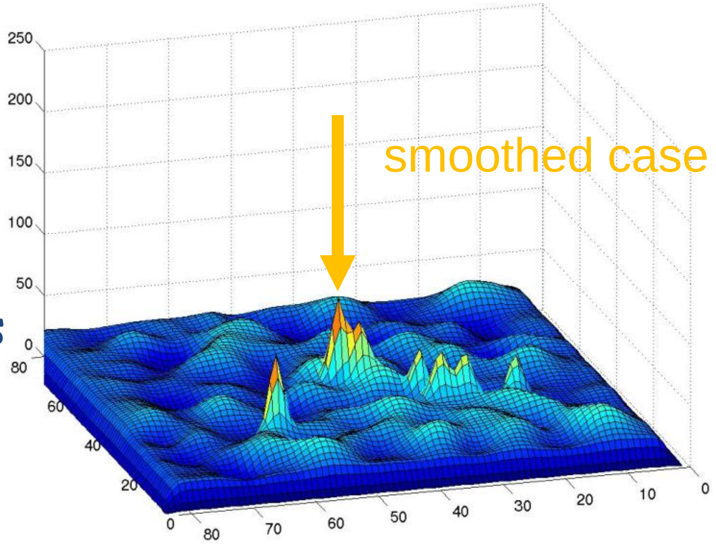
Current Algorithm Evaluation

- Average-case Evaluation:
  - Measures the expected performance of an algorithm over a pre-defined distribution of the inputs
- Drawbacks
  - may vary greatly from distribution to distribution
  - is usually difficult to model the 'practical' distribution of inputs in complex applications
  - tend to result in an overly optimistic evaluation

For complicate network algorithms, worst-case and average-case analyses cannot reveal practical performance!

New Algorithm Evaluation Method





Not Practical! ✗

More Accurate! 🙂

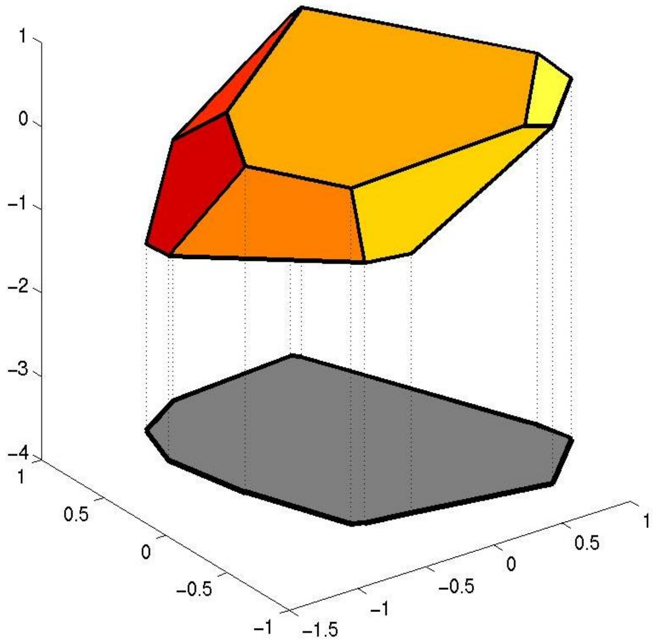
New Algorithm Evaluation Method

- Smoothed Analysis:

  $\max(E_g(M_A(x+\sigma g)))$

- First Use:
  - shadow-vertex simplex algorithm
  - worst-case complexity: exponential

    smoothed complexity: polynomial



D. Spielman and S. Teng, Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time, Journal of the ACM (JACM), vol. 51, no. 3, pp. 385–463, 2004.

SA → Sampling-based SA (SSA)

- To facilitate analysis for **COMPLICATE** algorithms in **COMPLEX** environment…
  - **Aim:**

    Simplified while maintaining **Accuracy**
  - **Method:**

    Sampling-based method (SSA)
  - **Formula:**

    $$\max_x(E_g(M_A(x+\sigma g)))$$

SSA Framework

- STEP1: Inputs Generation
  - gather *N* worse cases and constitute set *W*
- STEP2: Sampling
  - sample in the neighborhood of each instance *x* in *W*
- STEP3: Calculate Results
  - calculate expectations of result set for each *x*
  - obtain the maximum of all the expectations as SSA result

SA vs. SSA

- Divergences
  - Smoothing "each input" vs. "local maximums"
  - "not sampling" vs. "sampling"
- SA and SSA reach ALMOST THE SAME evaluation results!
  - With proper parameter selection
    - e.g., choose enough cases into the particular set, with a high enough sampling rate

Case Study

- Two algorithms for **Packet Classification** Problem
  - Computational Geometry Algorithm (CG)
  - HyperSplit Algorithm (HS)
- Evaluate and compare **worst-case**, **average-case**, and **SSA performances**

Case Study : Memory Usage

- Memory Usage (KB):

Algorithm	Worst-case	Average-case
CG	14061	1769.01
HS	7606	763.24

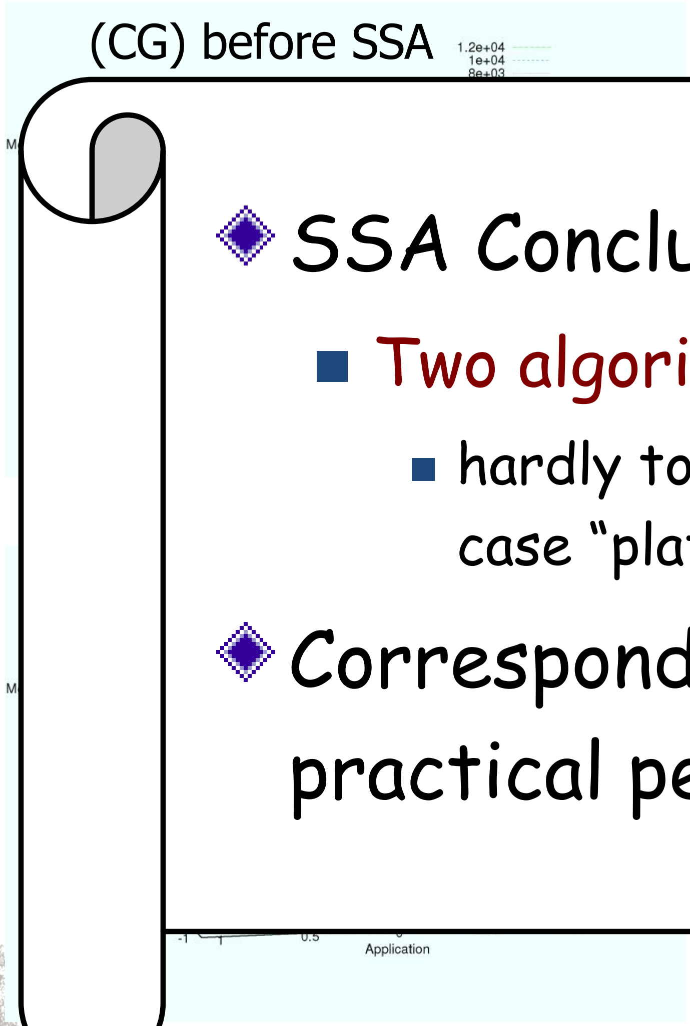
Worst-case Performance : Bad!

Average-case Performance: Good!

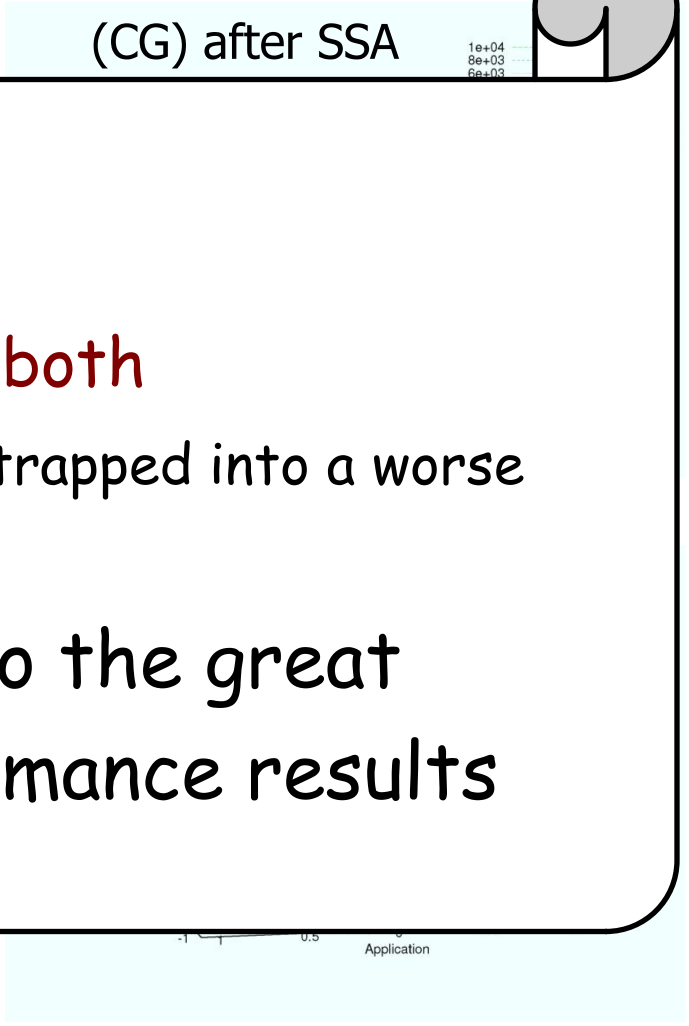
} **Conflict!**

Case Study : Memory Usage

(CG) before SSA



(CG) after SSA



- SSA Conclusion
  - Two algorithms both
    - hardly to be entrapped into a worse case “plateau”
- Corresponding to the great practical performance results

Case Study : Tree Depth

- Tree Depth:

Algorithm	Worst-case	Average-case
CG	28	24.12
HS	29	21.59

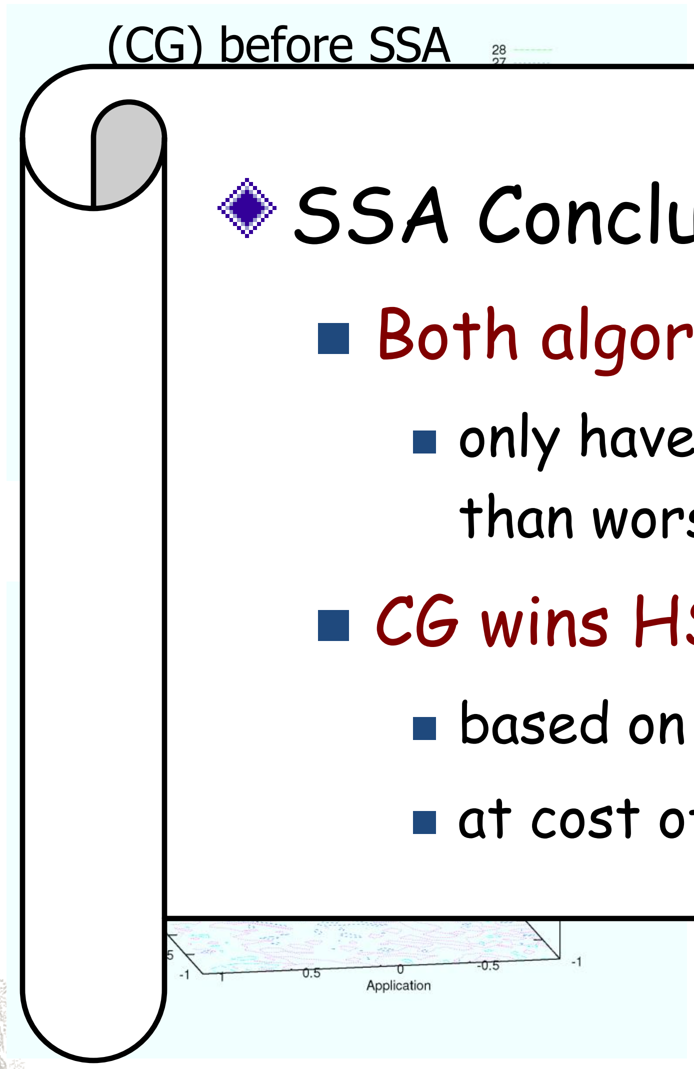
Worst-case Performance : CG > HS

Average-case Performance: CG < HS

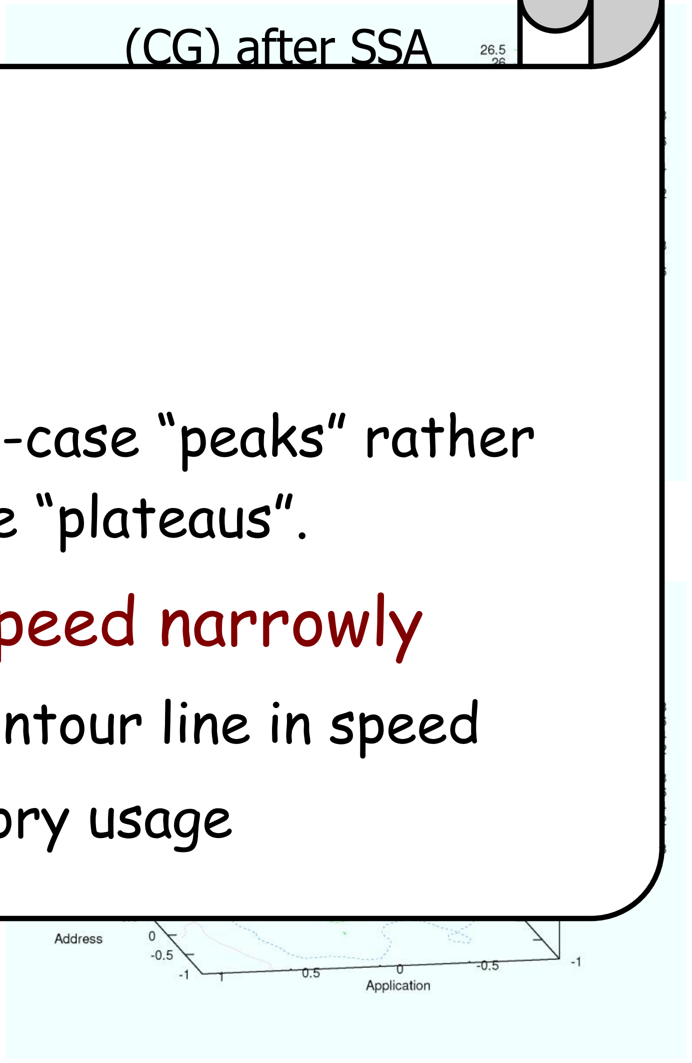
$\}$ **Conflict!**

Case Study : Tree Depth

(CG) before SSA



(CG) after SSA



- SSA Conclusion
  - Both algorithms
    - only have worse-case "peaks" rather than worse-case "plateaus".
  - CG wins HS in speed narrowly
    - based on the contour line in speed
    - at cost of memory usage

Conclusions

- SSA reveals **PRACTICAL, CLOSE-TO-REAL PERFORMANCE**
- SSA can enhance existing benchmark generator
- "Fast algorithms, smoothed analysis, and hardness results"

  X. Ren, Y. Qi, B. Yang, J. Li, and S.-H. Teng, Sampling-based smoothed analysis for network algorithm evaluation, (submitted)

Outline

- Background
- HyperSplit Algorithm
- Architecture for FPGA Implementation
- Evaluation with Smoothed Analysis
- **Future Work**

Future Work

- Regular Expression Matching algorithmic study
- Novel “explosion free” algorithm
- Many-core and FPGA: architecture and parallel processing
- Sampling-based Smoothed Analysis: further empirical validation and evaluation

Y. Qi, K. Wang, J. Fong, Y. Xue, J. Li, W, Jiang, and V. Prasanna, FEACAN: front-end acceleration for content-aware network processing, INFOCOM, pp. 2114 - 2122, 2011.

Thank you and Questions?