ParaSplit: A Scalable Architecture on FPGA for Terabit Packet Classification

Jeffrey Fong[1], Wang Xiang[1]
Yaxuan Qi[2], Jun Li[2], Weirong Jiang[3]

HOTI 2012
2012.08.21

[1]Reserach Institute of Information Technology, Tsinghua University, Beijing, China
[2]Tsinghua National Lab for Information Science and Technology, Beijing, China
[3]Ericsson Inc., San Jose, CA, USA

Outline

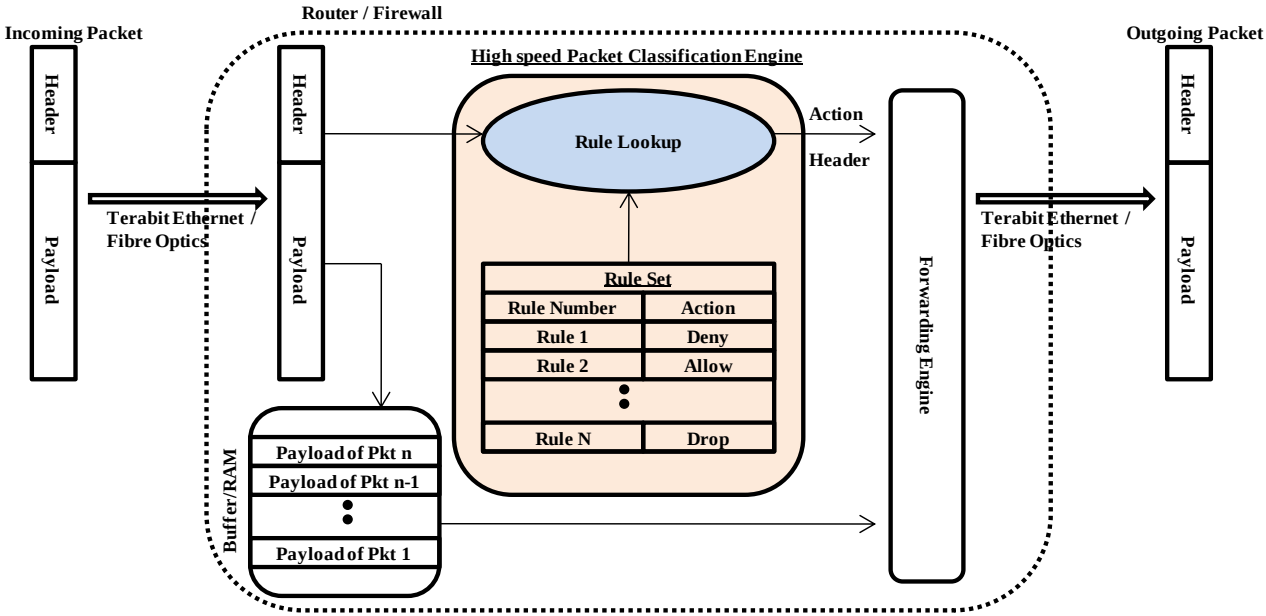
- Background and Motivation
  - The Packet Classification Problem
- ParaSplit
  - Range Point Conversion Set Partitioning
  - Simulated Annealing
  - Hardware Design Implementation
- Performance Evaluation
- Conclusion

Outline

- **Background and Motivation**
  - The Packet Classification Problem
- ParaSplit
  - Range Point Conversion Set Partitioning
  - Simulated Annealing
  - Hardware Design Implementation
- Performance Evaluation
- Conclusion

The Packet Classification Problem

- To identify and associate each packet to a specific rule
- May match multiple rules
- Used for:
  - Routing
  - Firewall/ Intrusion Detection System
  - Quality of Service



Rule	SA Range	DA Range	SP Range	DP Range	Action
(0)	0.0.0.0~64.0.0.0	32.0.0.0~64.0.0.0	0~65535	128~256	deny
(1)	32.0.0.0~255.255.255.25	0.0.0.0~64.0.0.0	64~256	0~65535	permit
(2)	32.0.0.0~128.0.0.0	0.0.0.0~255.255.255.255	128~65535	128~65535	deny

Related Works

SRAM Based

Software running on general hardware, i.e. multicore server

- Different algorithms gives different search speed and/or number of rules

Advantage:

- Price
- (generally) # of Rules

Disadvantage

- Speed

TCAM Based

Dedicated packet matching hardware

- Different hardware architecture gives different speed

Advantage

- Speed

Disadvantage

- Price
- Energy consumption

Challenges and Goals

Increasing Bandwidth

- Increasing number of application that needs high bandwidth: ie. VoIP, Video Streaming, Data Center/SDN (OpenFlow)
  - Needs to achieve high throughput
  - Needs deterministic performance

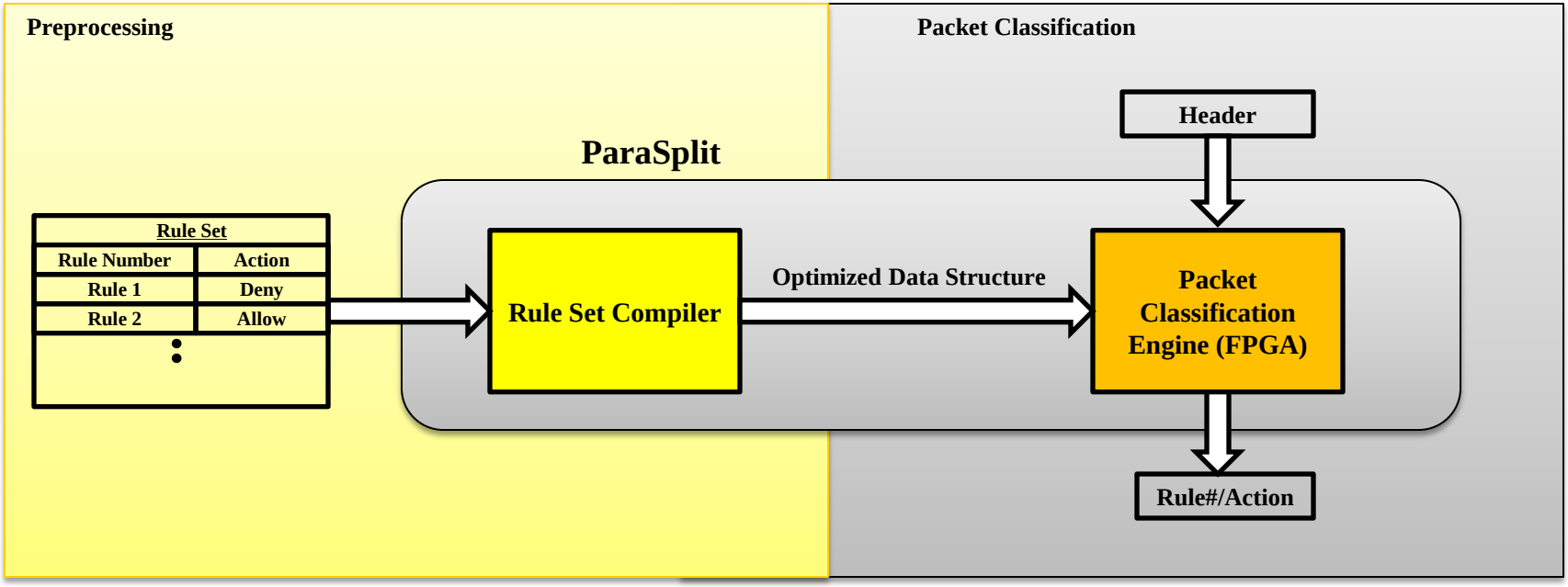
Rule sets are becoming large and complex

- Complex with many header fields (multi-dimensional)
- Even state-of-the-art algorithm requires a few to hundreds of GB
  - Needs low memory consumption

Outline

- Background and Motivation
  - The Packet Classification Problem
- **ParaSplit**
  - Range Point Conversion Set Partitioning
  - Simulated Annealing
  - Hardware Design Implementation
- Performance Evaluation
- Conclusion

ParaSplit



ParaSplit is an optimized software-hardware solution

Rule Set Compiler: Generate data structure used by hardware

Packet Classification Engine: Find best matching rule

Algorithmic Motivation

Worst case decision tree spatial complexity:

$$C=\Theta(n^d)$$

Depends on the intrinsic property of rule set

Rule Replication and Memory Consumption in HyperSplit

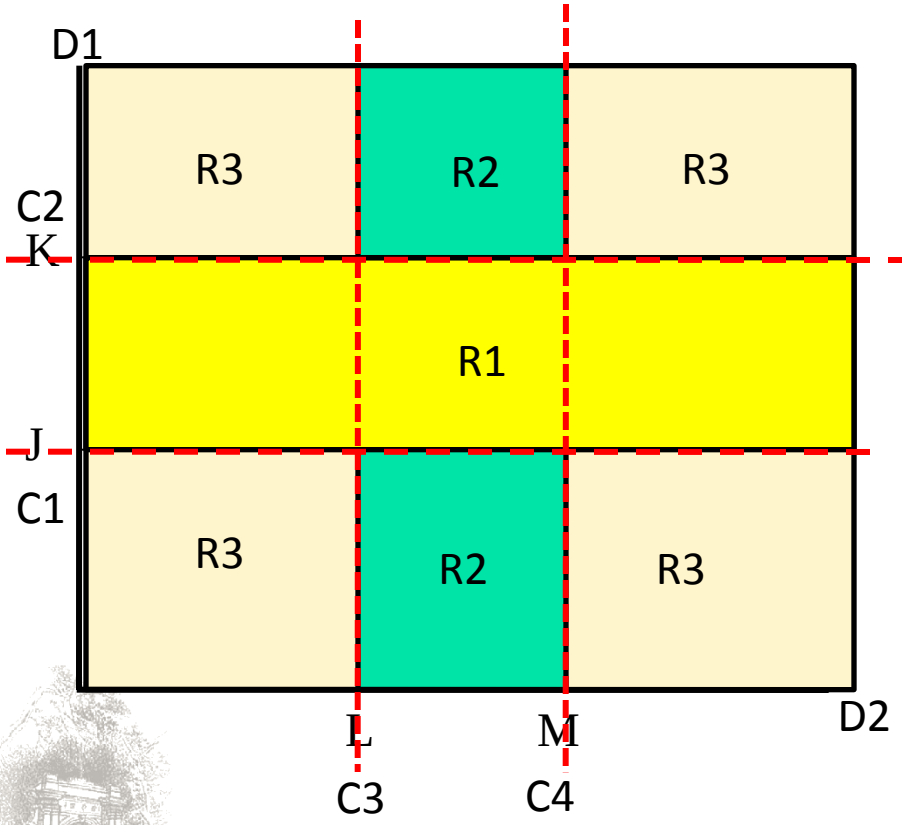
Rule Set	*Number of Rules*	*Number of Leafs*	*Replications*	*Memory Consumption*
ACL1_10K	9603	60657	6.32	947.8KB
IPC1_10K	9037	4278300	473.42	65.3MB
FW1_10K	9311	64499809	6927.26	984.2MB

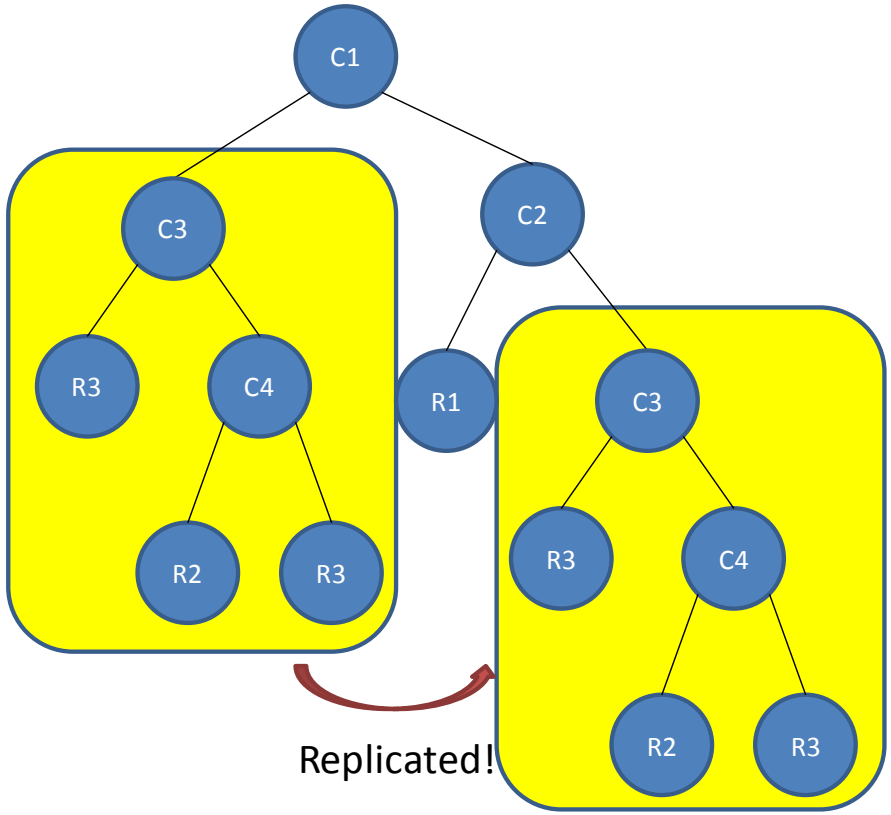
Difference in memory consumption caused by overlapping/conflicting rules leading to rule replication

Algorithmic Motivation

Example of overlapping rules causing replication

Rule	D1	D2
R1	J<x<K	*
R2	*	L<y<M
R3	*	*





Rule Set Partitioning

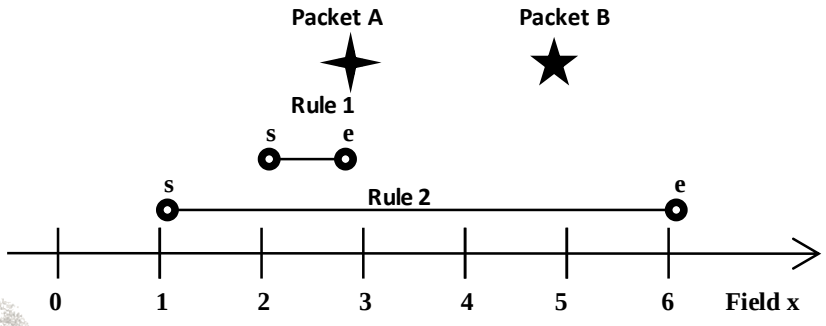
- **Is it possible to "somehow" remove these "conflicting rules"?**
  - **Yes, it turns out that it is possible to reduce memory consumption considerably by removing "certain" rules**
- **Idea:**
  - **Divide the original rule set into M groups/subset, each group containing non-conflicting rules, such that the union of all subset is the original rule set**
  - **Build a decision tree based on each group of rules**
  - **When doing a look up, traverse all trees and combine results by selecting the highest priority rule**
- **How to find these "good" groups/subset of rules?**
  - **ParaSplit: Range-Point Conversion to generate a good initial grouping + Simulated Annealing to approximate global minima**
- **Deal with multiple tree traversal by taking advantage of abundant resources and parallelism available on FPGA**

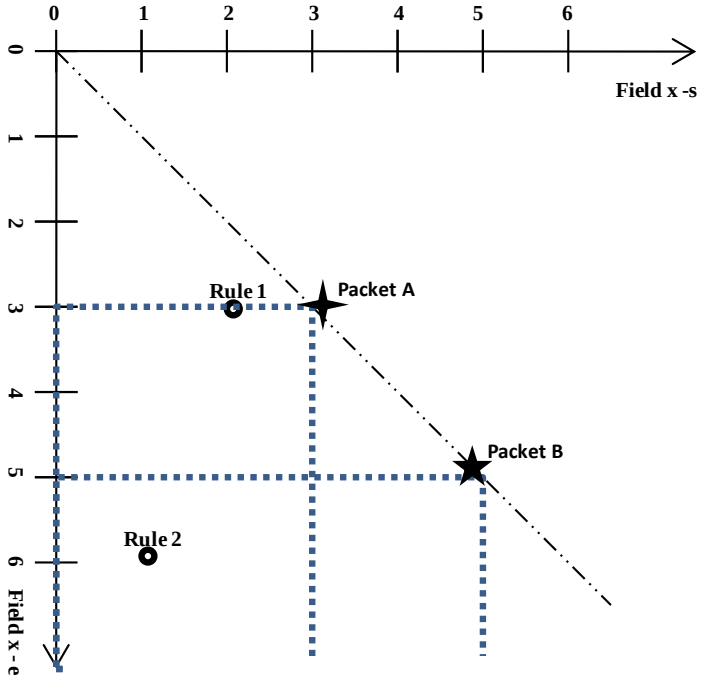
Range-Point Conversion

- **Difficult to group rules represented as objects in F-dimensional space**
- **Convert it into points in 2F dimensional space by treating starting and end point as separate dimensions and then group points together**

Field-x	Start (s)	End (e)
Rule 1	2	3
Rule 2	1	6

Packet	Field-x	Matched Rules
A	3	1, 2
B	5	2



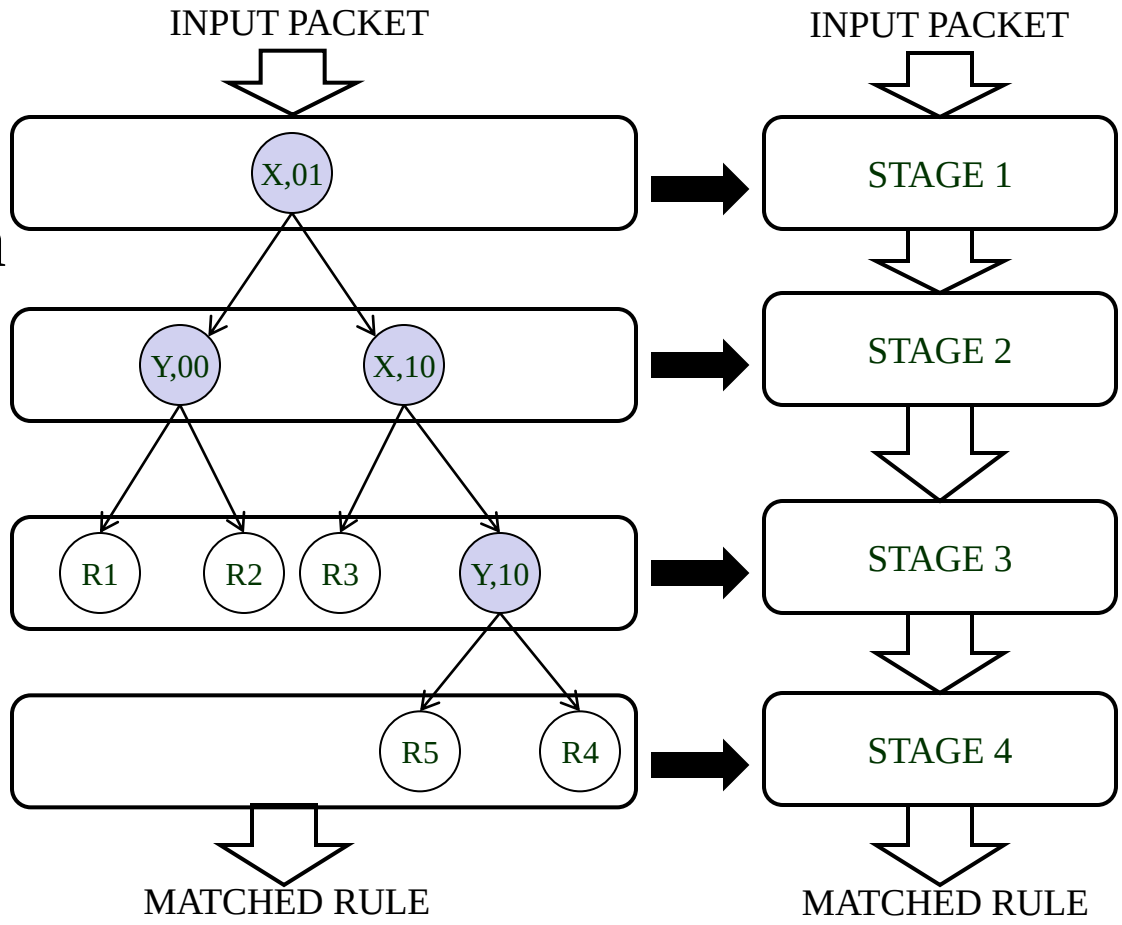


Simulated Annealing

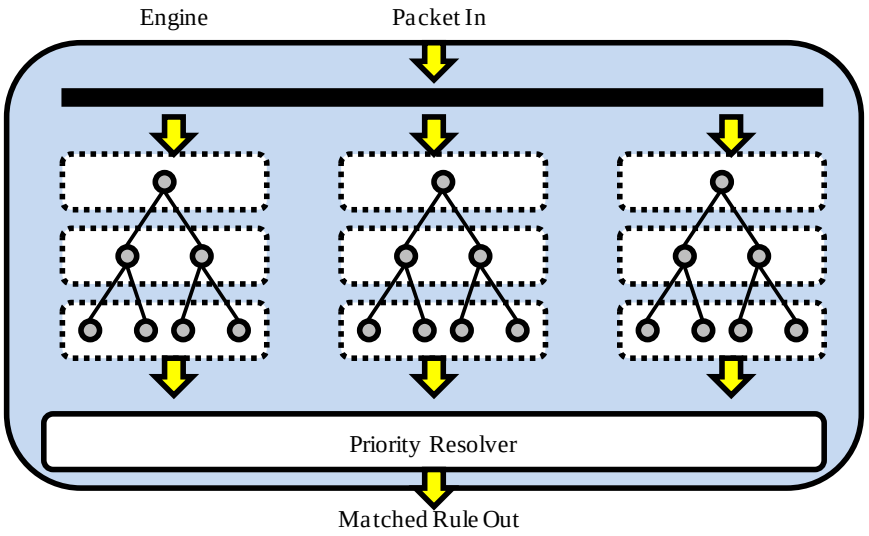
- Using the initial partitions generated by Range-Point partitioning, apply Simulated Annealing to approx. global minima
- Goal is to further reduce memory usage (cost = mem. consumption)
- Randomly select 2 subsets, S_i and S_j, and perform one of three possible action:
  1. Move rule, R_i, from S_i and S_j
  2. Swap rule, R_i, from S_i with rule, R_j, from S_j
  3. Move rule, R_j,from S_j to S_i
- With a probability of $P_{accept} = e^{-\Delta D/T}$ accept the new state
- T = temperature of system = (initial cost)/(50*ln(2))
- $\Delta D = final\ cost - initial\ cost = \#leaf_{final} - \#leaf_{initial}$

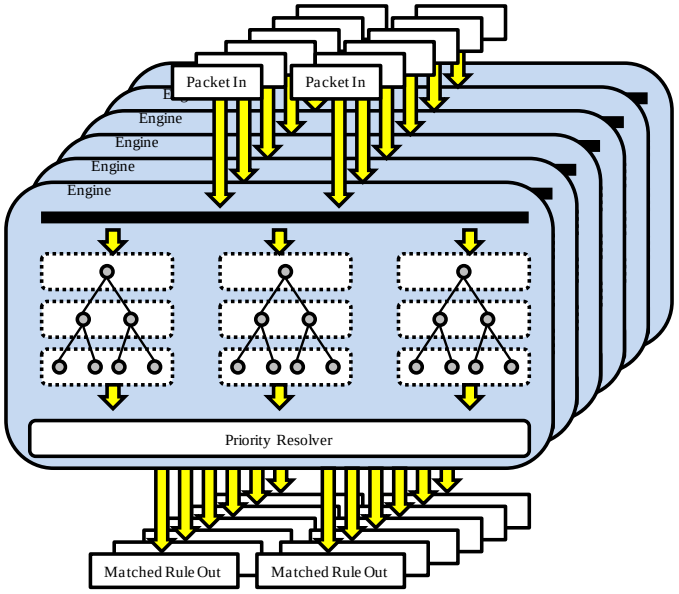
Decision Tree & Hardware Mapping

- HyperSplit is then applied to build a decision tree for each group
- Group nodes within the same level into one stage
- Build a pipeline



Hardware Implementation





Each rule subset maps into a separate pipeline

Priority resolver to find the best matching rule

Dual-port BRAM for double performance without extra memory usage

Multi-engine on a single FPGA for higher throughput

Outline

- Background and Motivation
  - The Packet Classification Problem
- ParaSplit
  - Range Point Conversion Set Partitioning
  - Simulated Annealing
  - Hardware Design Implementation
- **Performance Evaluation**
- Conclusion

Test Bed

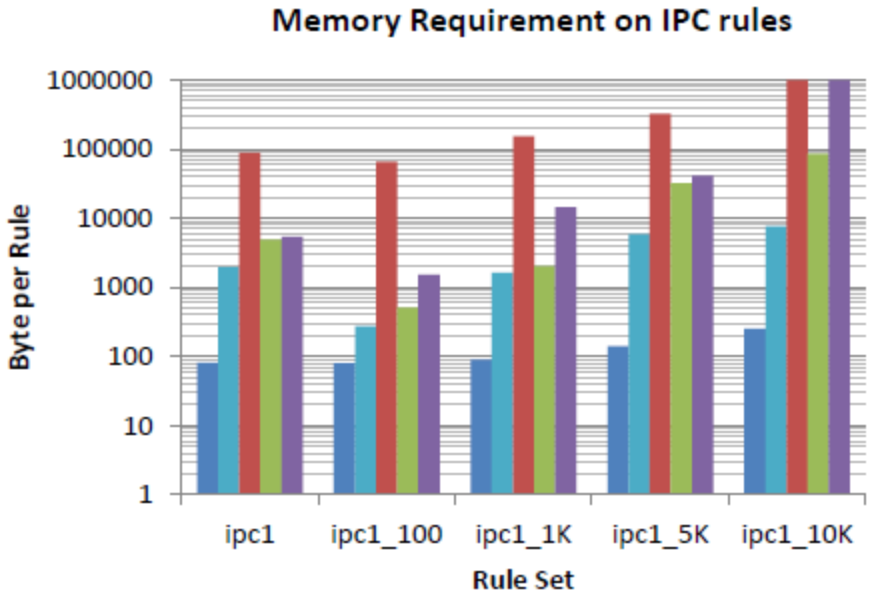
Tested with:

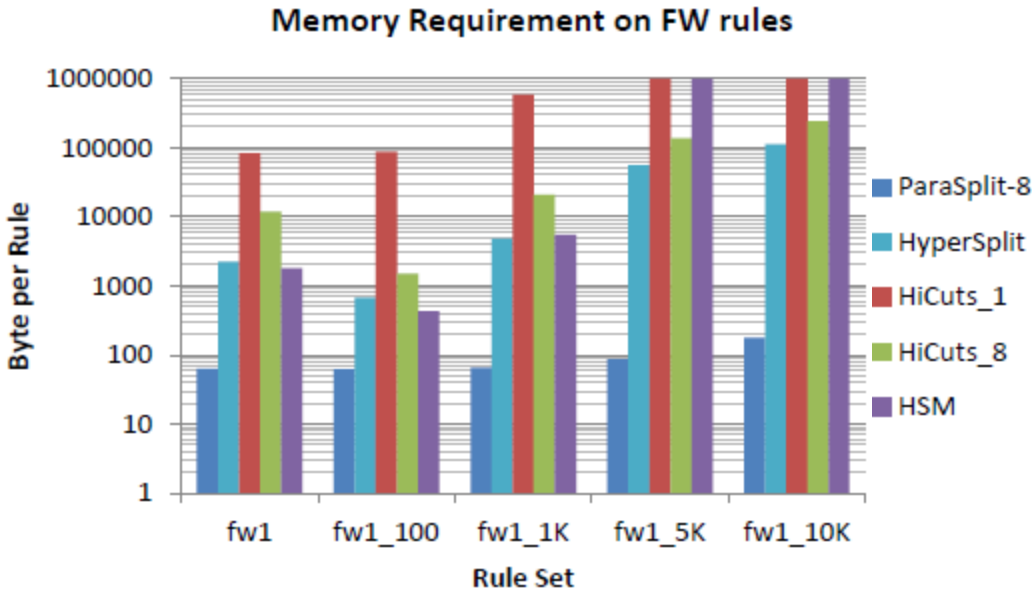
- a publicly available rule set from Washington University
  - Used the IPC & FW 100, 1K, 5K, 10K rule sets
- OpenFlow-like 11-tuple rule set generated based on 216 real-life 11-tuple rules from enterprise customers

Design is implemented on a Xilinx Virtex-5

- Model: VC5VSX240T
- Containing 4,200Kb Distributed RAM and 18,576Kb Block RAM

ParaSplit vs Well-known Algor.

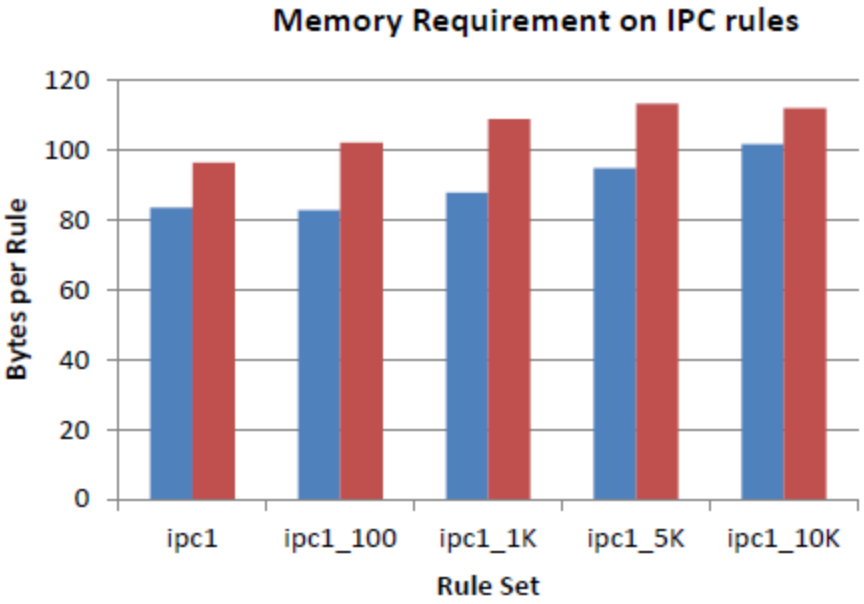


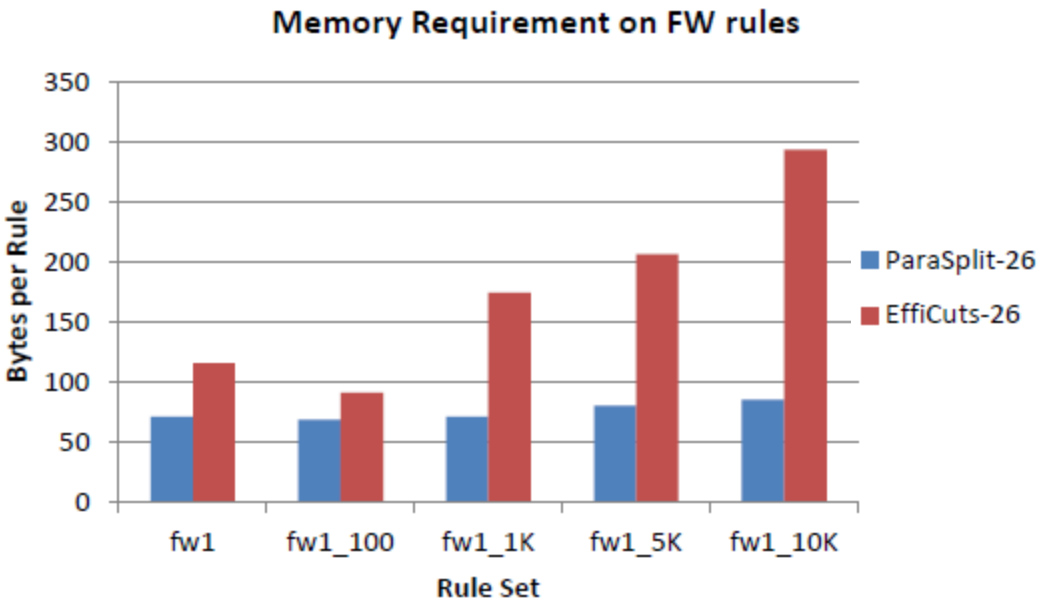


- Memory consumption reduction by an average of 150x!
- Rule sets that used to consume 1GB of memory can now fit within the 2MB BRAM of the FPGA

Note: HSM and HiCuts_1 fails to generate data structure for ipc1_10K and fw1_10K due to exhaustion of memory (over 4GB)

ParaSplit vs EffiCuts Scheme





- ParaSplit requires 20% to 500% less memory than EffiCuts scheme

OpenFlow-like Complex Rules

PERFORMANCE WITH 11-TUPLE RULE SETS

# of rules	ParaSplit-8		HyperSplit	
	Bytes/rule	Tree height	Bytes/rule	Tree height
400	1.25	4	8.75	16
800	2	11	196.25	23
1200	3.625	15	559.5	23
1600	5.625	16	10141.5	29
2000	17.625	20	14401.5	30

- Up to 3 orders of magnitude lower memory consumption than HyperSplit
- Worst-case tree height is also reduced by at least 30%

Hardware Performance

PERFORMANCE AND RESOURCE USAGE WITH A SINGLE ENGINE

Rule set	Max freq. (MHz)	Max thrupt (Gbps)	Tree height	# slice	# BRAM
$fw1\_100$	120.86	123	12	7270	48
$fw1\_1K$	118.02	120.8	16	10274	151
$fw1\_5K$	105.52	108.0	20	13834	253
$fw1\_10K$	100.23	102.6	25	12095	399

PERFORMANCE WITH MULTIPLE ENGINES

Rule set	BRAM usage per engine	# Engines	Aggregated Throughput (Tbps)
$fw1\_100$	1.6%	60	7.38
$fw1\_1K$	5.2%	19	2.29
$fw1\_5K$	8.7%	11	1.18
$fw1\_10K$	13.9%	7	0.72

Outline

- Background and Motivation
  - The Packet Classification Problem
- ParaSplit
  - Range Point Conversion Set Partitioning
  - Simulated Annealing
  - Hardware Design Implementation
- Performance Evaluation
- **Conclusion**

Conclusion and Future Work

ParaSplit is optimized software-hardware combined solution with the following contributions:

- Set partitioning that achieves 100x memory reduction compared to algorithms without rule set partitioning
- Pipelined decision tree to hardware mapping with parallel pipeline and engines that can provide over 100Gbps sustained throughput per engine
- Due to low memory consumption, multiple engines can be used to provide up to and over 1Tbps

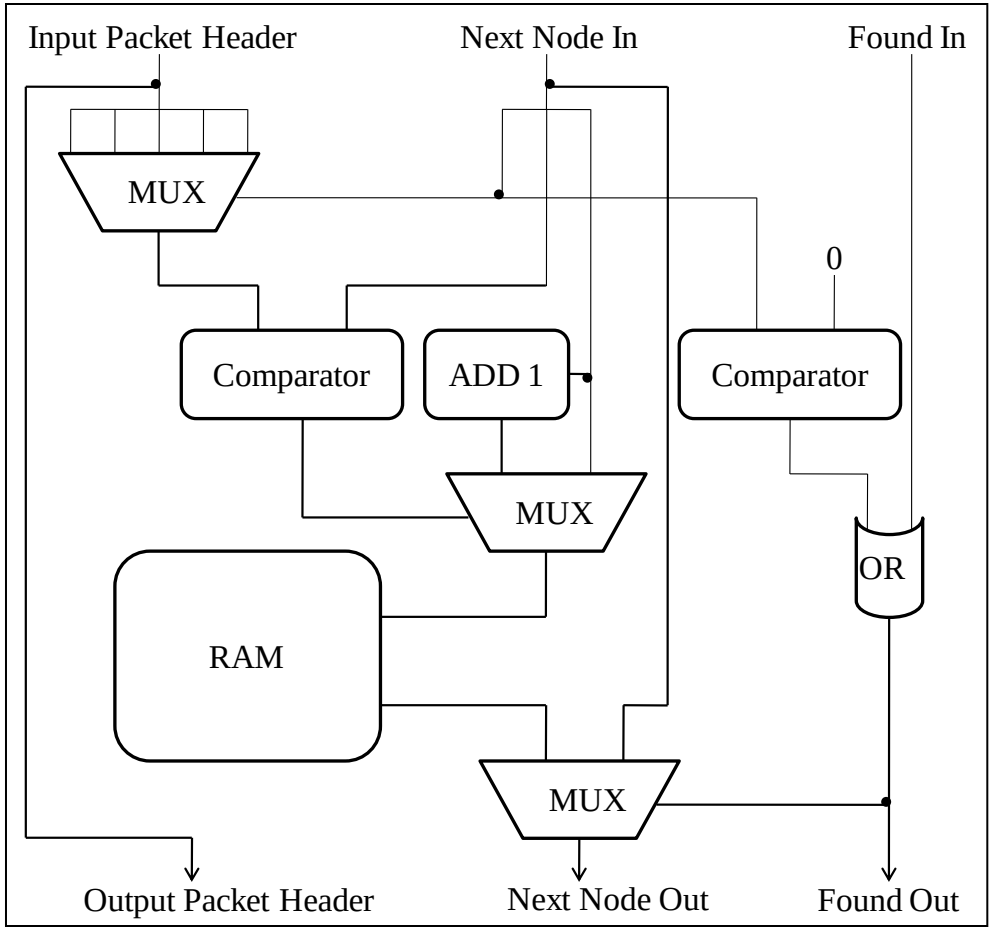
Future Works

- Heterogeneous Engine to support various algorithms
- Better set partitioning heuristics

Thank you!

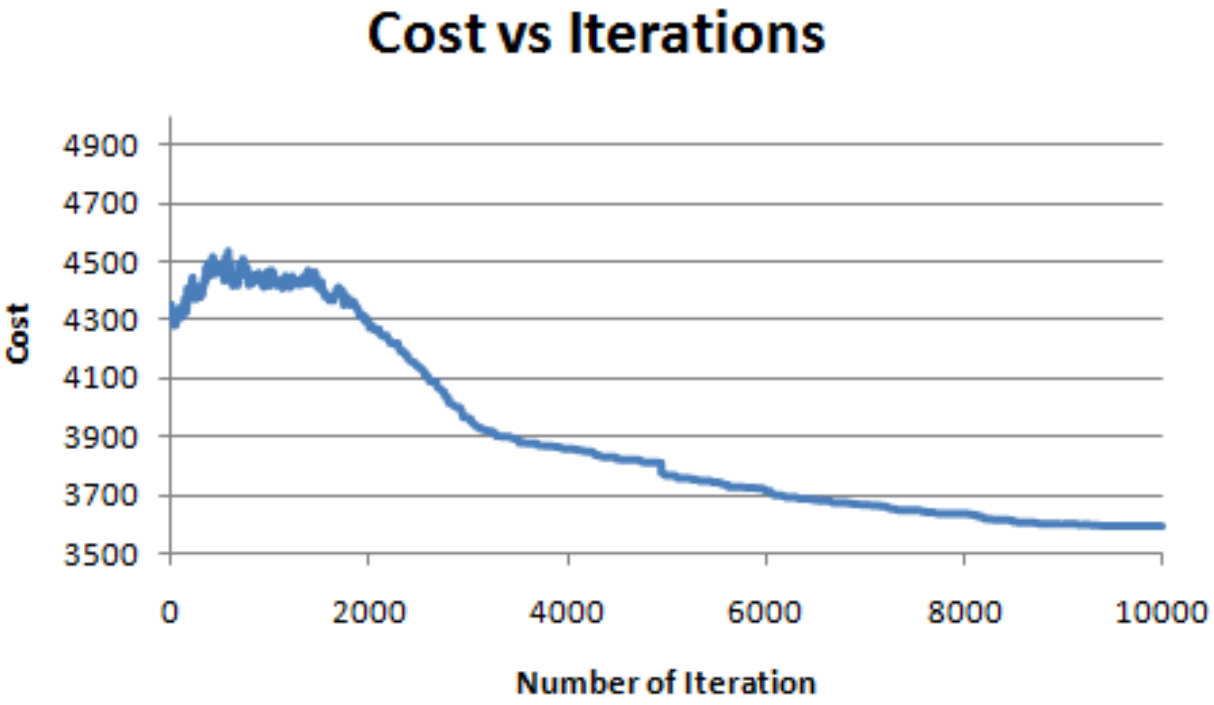
Hardware Implementation

STAGE n

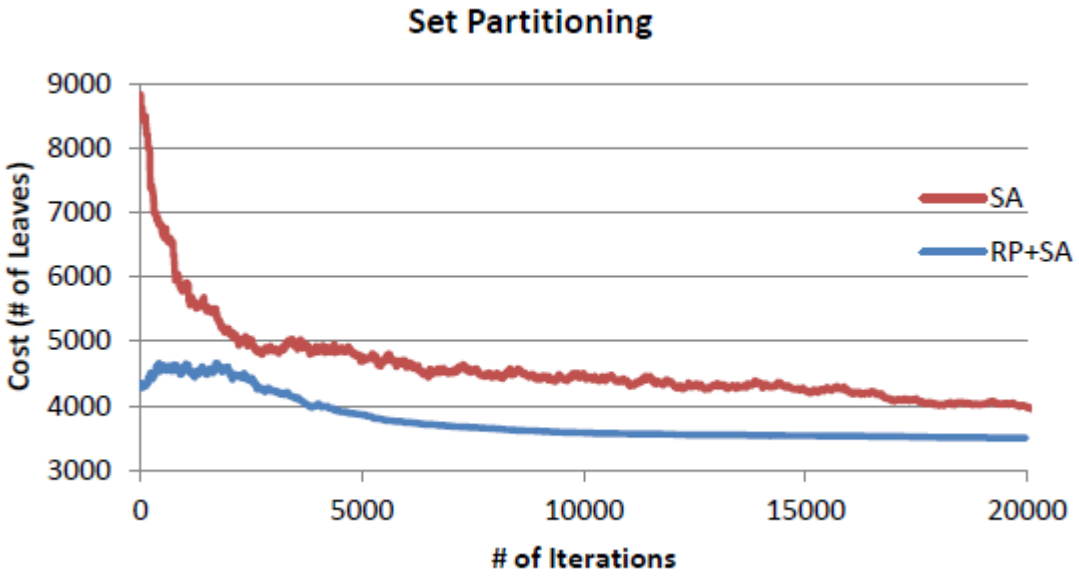


Implemented with Verilog Hardware Description Language (HDL)

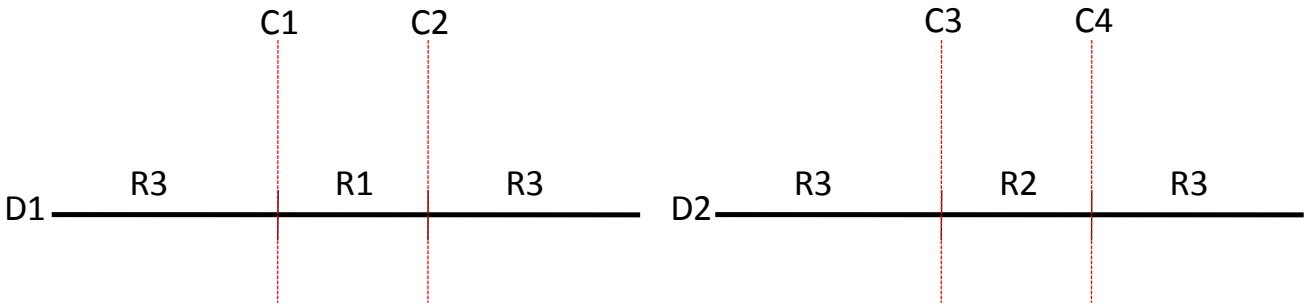
Simulated Annealing

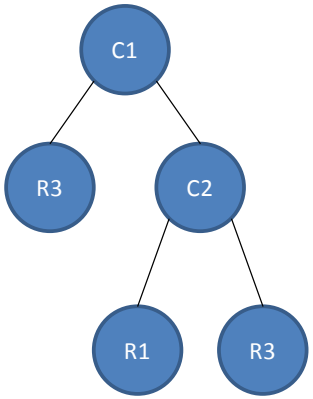


RA + SA

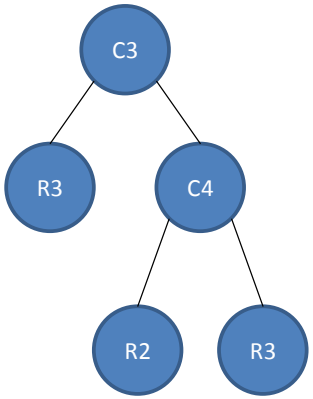

Set Partitioning
9000
8000
7000
6000
5000
4000
3000
Cost (# of Leaves)
0
5000
10000
15000
20000
# of Iterations
SA
RP+SA

Set Partitioning on Example Rules





Tree 1
Nodes = 5
Leaf = 3

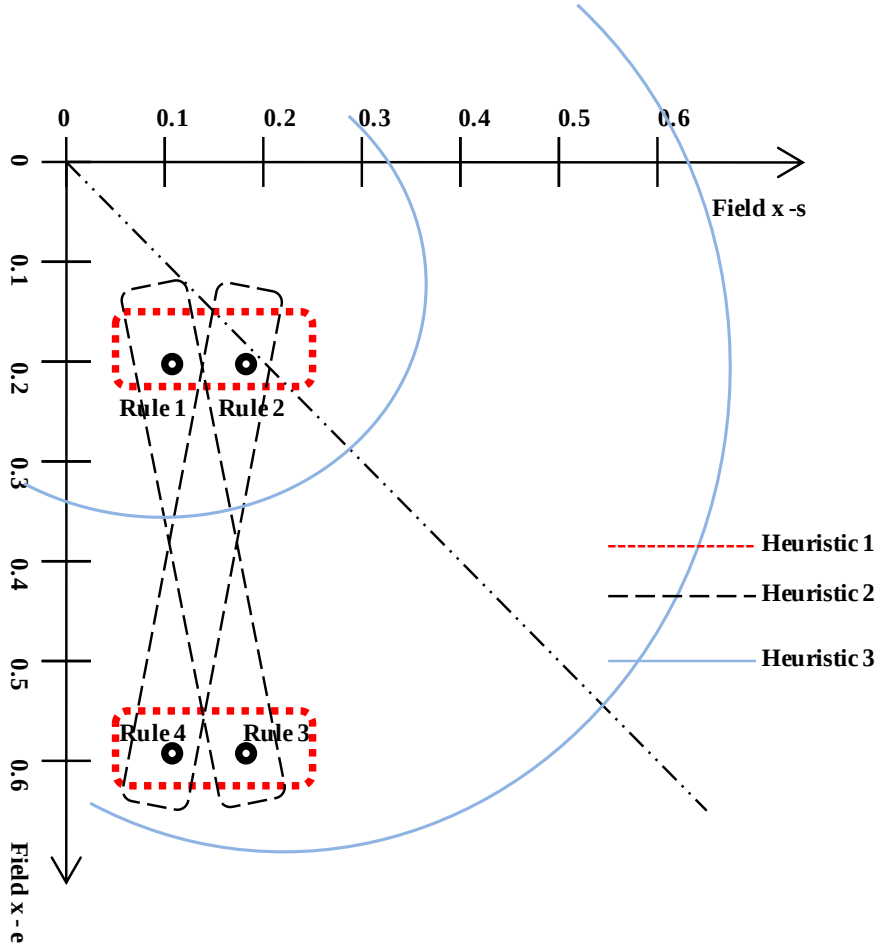


Tree 2
Nodes = 5
Leaf = 3

Grouping Heuristics

Heuristics:

1. Minimum Distance: group similar rules
2. Maximum Distance: group dissimilar rules
3. Distance from origin: mixture of similar and dissimilar



Reduced Complexity

Mathematically:

- Divide rule set into K groups
- Assume each rule subset has n/K rules
- Complexity becomes:

$$\Theta(K*(n/K)^F)=\Theta(n^F/K^{(F-1)}) < \Theta(n^F)$$

Complexity is reduced (by a factor of $K^{(F-1)}$)