

---

# A middle ground between CAMs and DAGs for high- speed packet classification

---

Amit Prakash, Adnan Aziz  
University of Texas at Austin

---

## Roadmap

- Introduction
    - classification: map rules to reconfigurable HW
    - formalization: logic function on packet header
    - previous data-structures and algorithms
  - BDD-based classification
    - partitioning rule-database
    - architecture
  - Experiments
-

## Motivation

- Packet classification
  - QoS
  - accounting
  - security
- Classification rule-database  $\Rightarrow$  Boolean function
  - dynamic rule-database (reconfigurable)
- Custom hardware
  - few 10s of nanoseconds per packet
  - amortized preprocessing time
  - small cycle time  $\Rightarrow$  room for more pipeline stages

## Formalization

User-specifies condition on a packet field: prefix

- example: 011\*

Predicate: a 5-tuple of conditions on fields

- example:  $\pi = *01*, 11*, 1*, *, **$

Action: an integer encoding some operation

Rule: predicate-action pair

- example:  $R_1 = (\pi, a)$

Rule-database:

- a collection of prioritized rules

Classification problem:

- given a header find the highest priority applicable rule

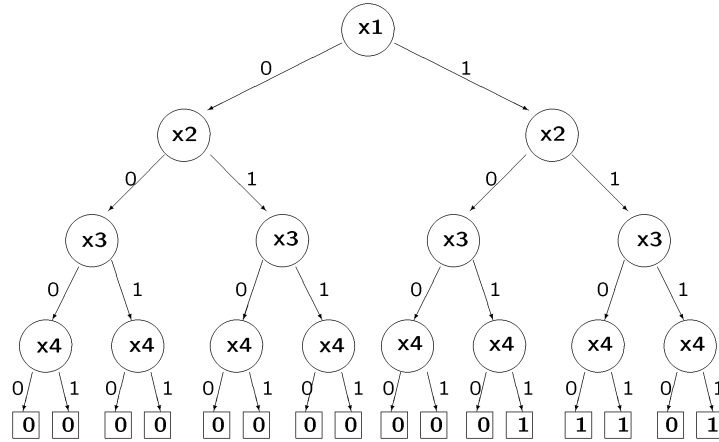
## Previous Work

- BDD-based forwarding [PA-HotI-01]
  - works at the speed of one lookup per SRAM access
  - BDD blowup for rules on multiple fields (linear size for single field)
- RFC [GM-AS-99]
  - isomorphic to BDD-trees [M-CAV-94]
  - same problem as BDDs
- Hi-Cuts [GM-HotI-99]
  - isomorphic to free-BDDs [B-ICCAD-96]
  - cannot pipeline
- Ternary CAM [LN]
  - slower than SRAMs
  - power hungry, area-inefficient
  - very inefficient for certain rule-sets

## IP forwarding using BDDs

- Forwarding table  $\Rightarrow$  Boolean function
  - input: 32 bits (destination IP address)
  - output: k bits for  $2^k$  output ports (usually  $k < 8$ )
- Looks like logic synthesis!
  - function keeps changing, special structure in function
- FPGAs do not work well
  - synthesis took one day to map one output-bit
  - delay: 85ns
- Our earlier work
  - construct the BDD for the forwarding function  $F_T: 2^{32} \Rightarrow 2^k$
  - implement function by an array of cascaded lookup-tables corresponding to BDD

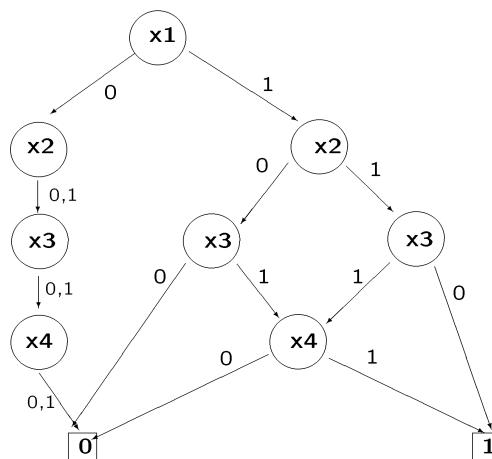
## Shannon tree



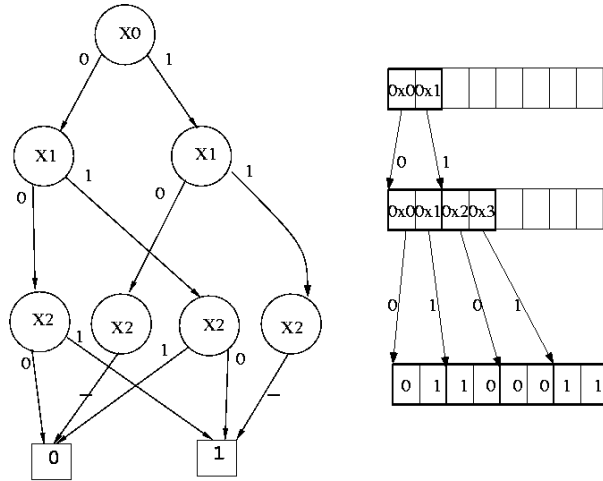
$$F = x_1 \cdot (x_2 \cdot (x_3' + x_4) + x_2' \cdot x_3 \cdot x_4)$$

## Binary Decision Diagram

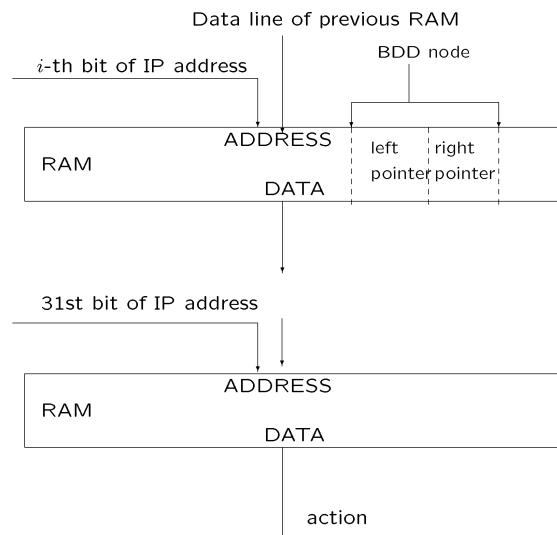
- Merge isomorphic children
  - can lead to exponential reduction in BDD size



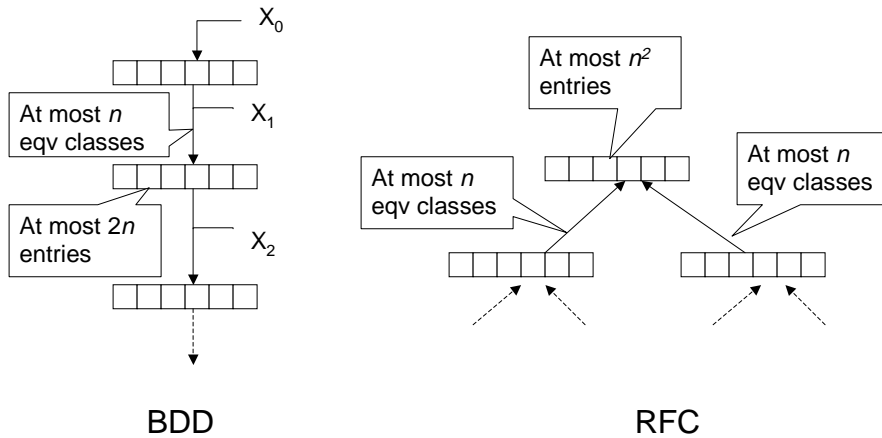
# Mapping BDDs to memory



# Architecture



## Quadratic factor in RFC



## Problem with multiple fields

- Prefix \* interval on number line
  - $n$  1-D rules \* at most  $(2n+1)$  intervals
- Prefixes on multiple fields \* cubes in Cartesian space
  - $n$  rules on  $d$  fields \* at most  $(2n+1)^d$  cubes

Rule-database:

$R_0 = *, *$   
 $R_1 = *01*, 01*$

R0	R0	R0
R0	R1	R0
R0	R0	R0

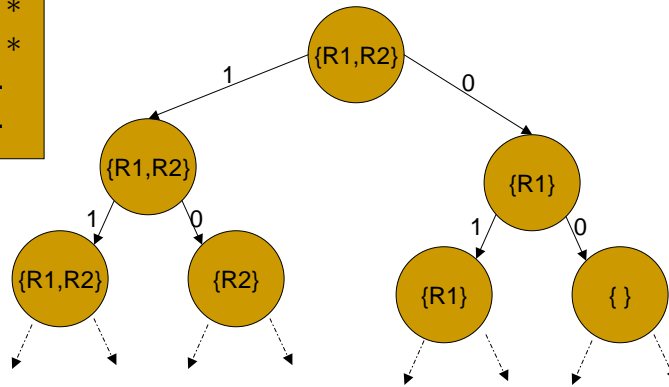
## BDD-blowup

Rule-database:

R1 = \* , 1\* , ... \*

R2 = \*1\* , \* , ... \*

.....  
.....



## Our approach

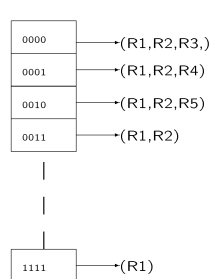
- Partition the rule-database into multiple databases
  - use a small number of classification engines operating in parallel
  - use a priority encoder to merge results
- Benefits
  - worst case goes roughly from  $\Theta(2^N)$  to  $\Theta(k * 2^{N/k})$
  - can use different variable order
  - intelligent partitioning essential
- Optimum partition: hard combinatorial problem
  - need heuristics

# Partitioning algorithm

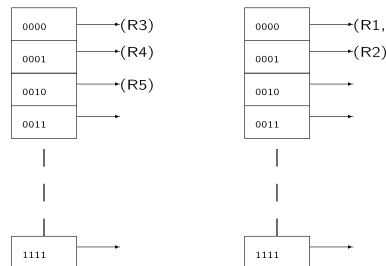
- Ideal problem (**P1**)
  - find a partition:  $D = D_1 * D_2 .. * D_p$
  - find a variable ordering for each database
  - such that the sum of sizes of BDD representing each database is minimized
- Restricted problem (**P2**)
  - find a partition:  $D = D_1 * D_2 .. * D_p$ , and
  - a small index set  $I_k$  for each  $D_k$ , such that,
    - for any pair of rules in any database  $D_k$  there is at least one integer  $i * I_k$  such that the two rules differ at the  $i$ -th bit

# Example

Rule	source IP	Destination IP
R1	****	0000
R2	00**	0001
R3	0000	0***
R4	0001	01**
R5	0010	****



(a) Indexing on Source IP



(b) 2 Partitions, one indexing on source IP, the other on destination IP

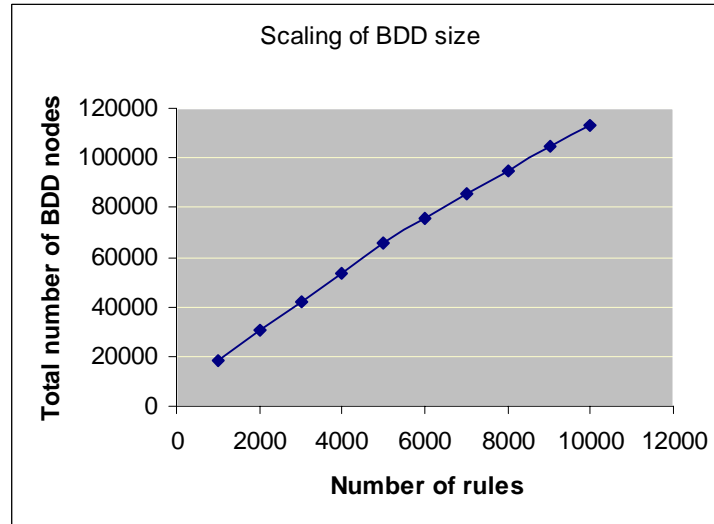
## Heuristics for *P2*

- Limit choice of index bits to contiguous blocks of variables
- Greedily construct databases
  - try all possible index bits
  - no database gets a rule with more than  $c$  \*s in the index positions
  - no more than  $m$  rules in a partition agree upon any assignment to index bits
  - guess a small value for  $c$  and  $m$
- If some rule remains unassigned, increase either  $c$  or  $m$

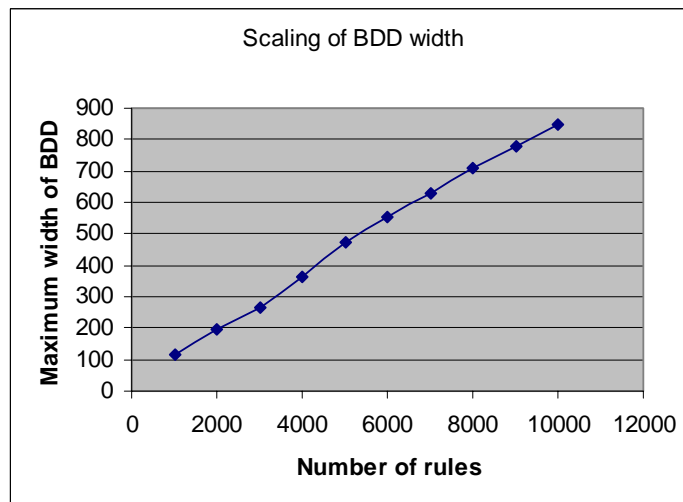
## Results

- Number of classifiers
  - between 5-10 for up to 25000 rules
- Total number of BDD nodes
  - approximately linear in number of rules
- Maximum number of nodes at any level
  - approximately linear in number of rules
- Throughput
  - one lookup per SRAM access: 500 million lookups per second

## BDD size



## Maximum BDD width



---

## Hardware

- For 1024 rules
    - 5 pipeline units
    - 110 banks of 4KB SRAM (440 KB)
    - 5 input 10-bit priority encoder
  - For 10000 rules
    - 10 pipeline units
    - 220 banks of 20KB SRAM (4.4MB)
    - 10 input 14-bit priority encoder
- 

---

## Other Issues

- Level Compression
    - Dynamic programming
  - Updates
    - amortizing merge process
  - VLSI issues
    - layout, signaling, power
  - Software classification
-

---

## Questions for audience

- Rule database
  - Other applications
    - MAC layer lookups
    - String search
  - Expressive power of rules
  - Integration in a network processor
- 



Questions!

---