# Reducing BDD Creation Time for Dynamic Trace Information


Graham Price

Manish Vachharajani

John Giacomoni

UNIVERSITY OF COLORADO AT BOULDER
ELECTRICAL & COMPUTER ENGINEERING

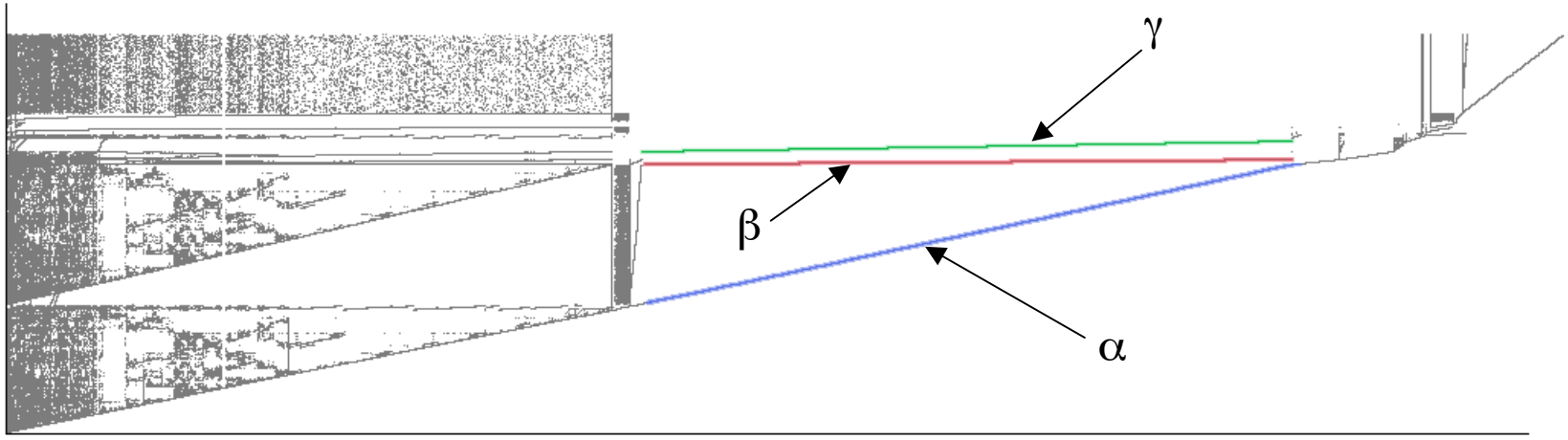
UNIVERSITY OF COLORADO AT BOULDER
CORE RESEARCH LAB

# Motivation

- Chip Multiprocessor (CMP) solutions are becoming more prevalent
- Dynamic program analysis can aid multicore development
- Dynamic trace data can be large and cumbersome

# Dynamic Program Analysis Tools

- Bug Squashing
- Program Execution Visualization



# Dynamic Program Trace Size

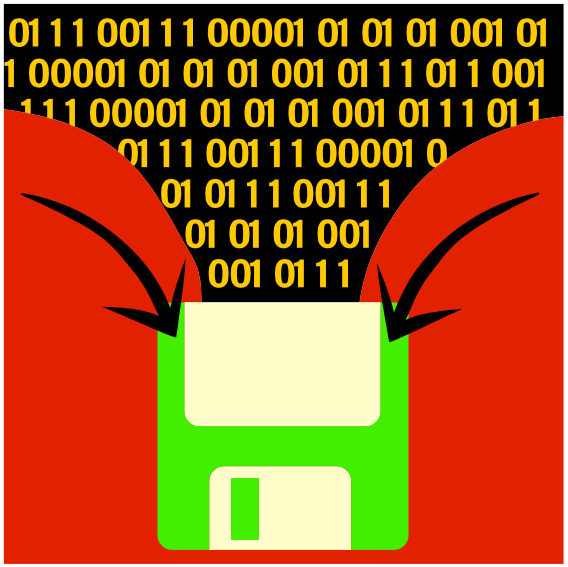
Instruction Type

Instruction Operands

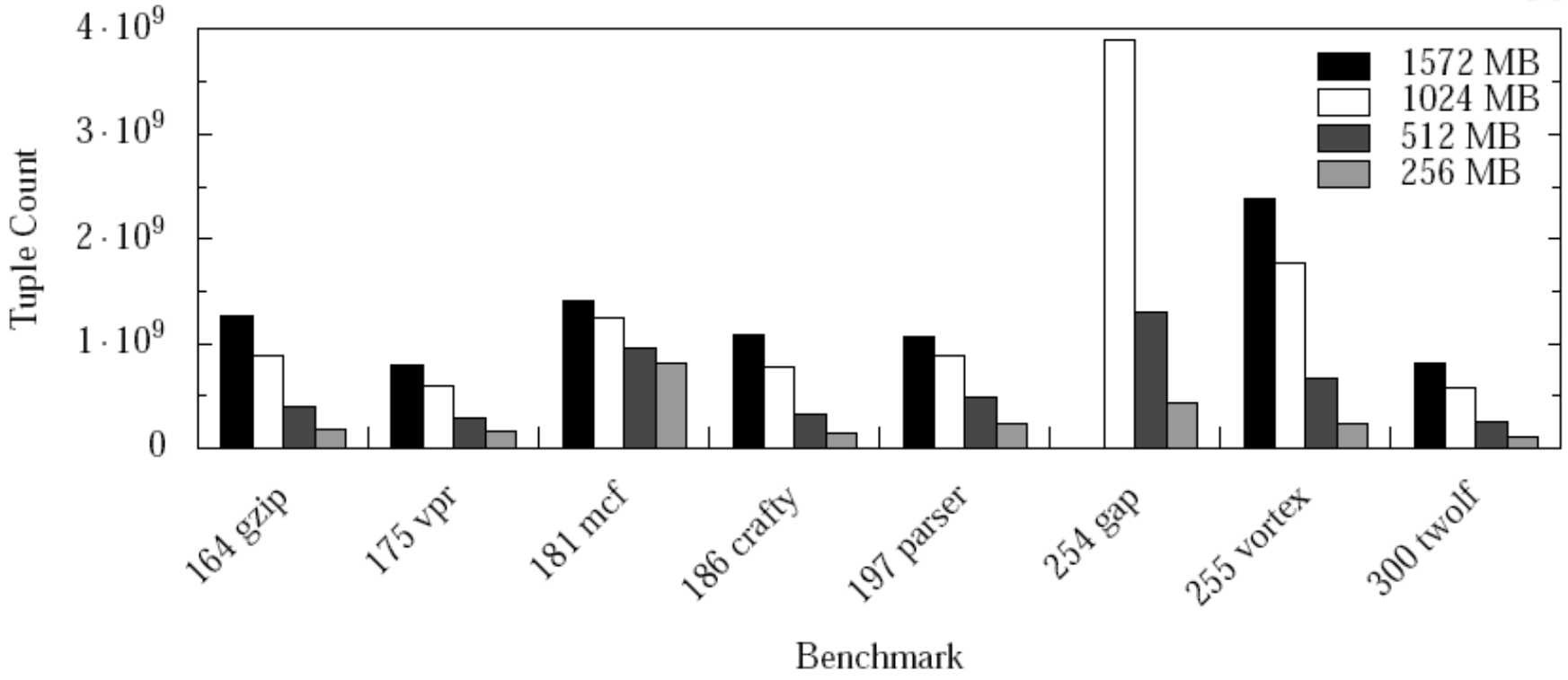
Branch?

Memory Locations

Load/Store

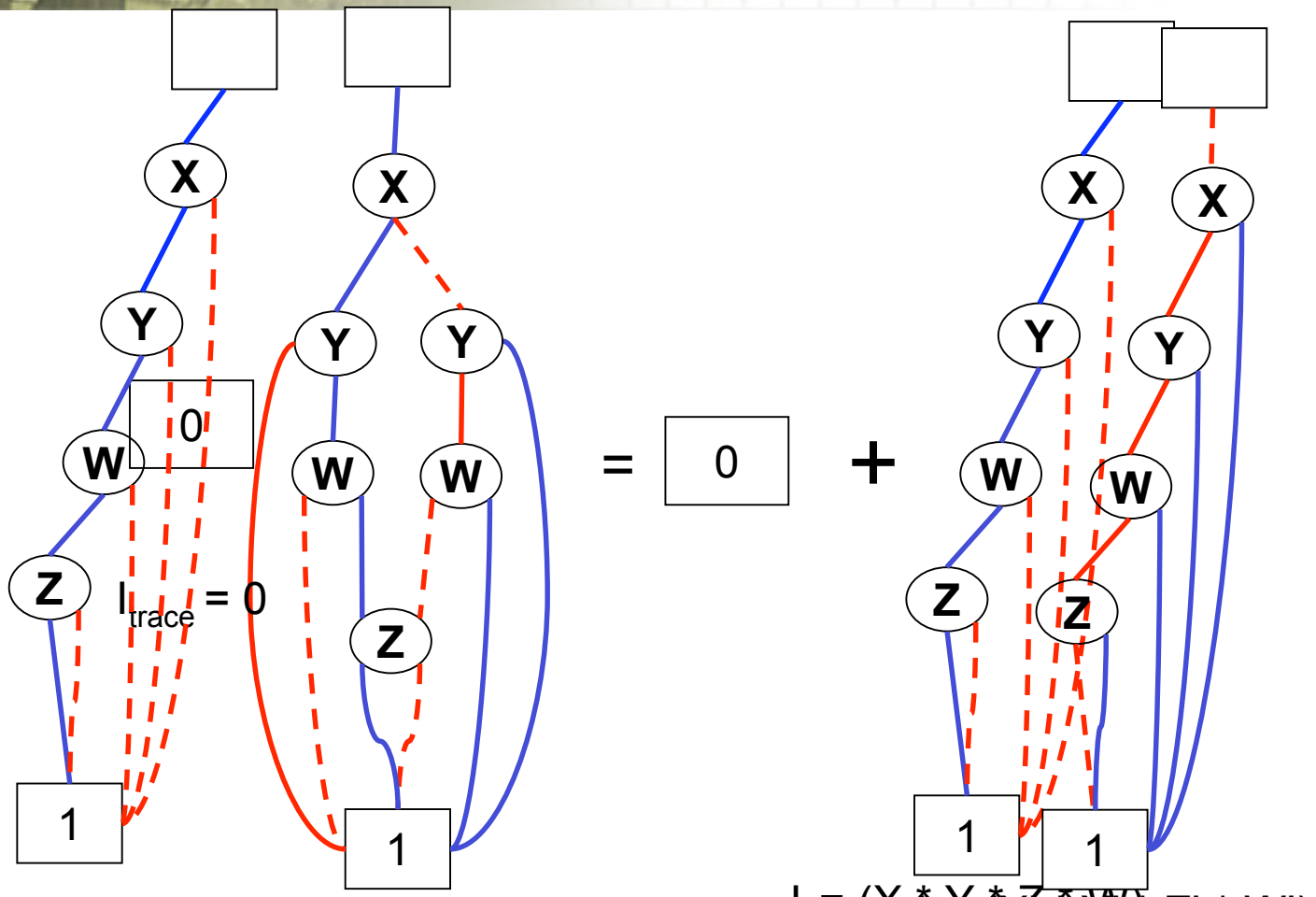


# BDD Compression Numbers




- Compression ratios range from 16 to over 60

# Old BDD Construction



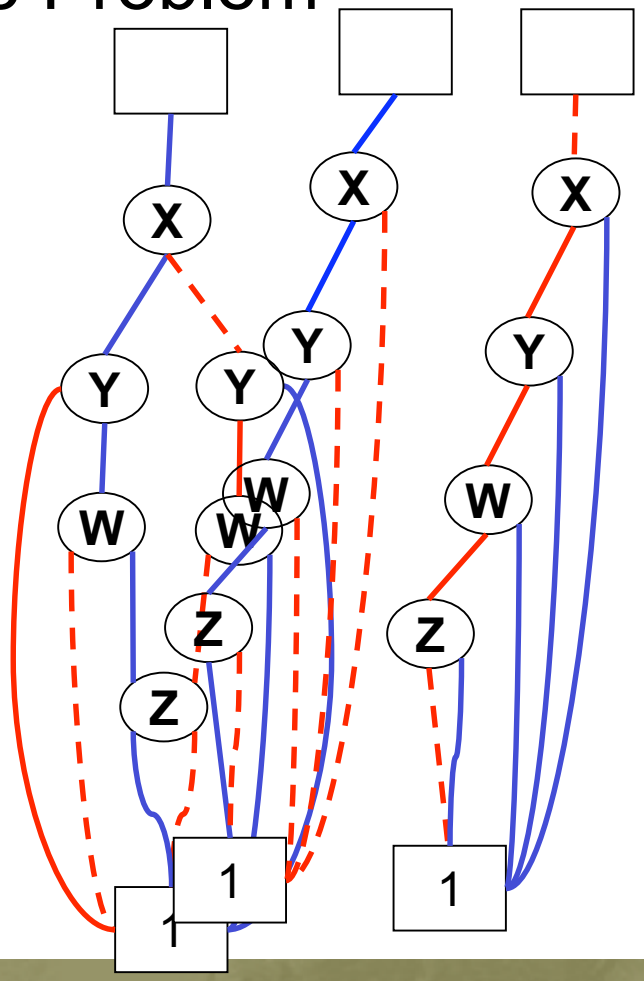

$I_{trace} = 0$

$I_{trace} = (X * Y * Z * W) + (X' * Y' * Z' * W')$

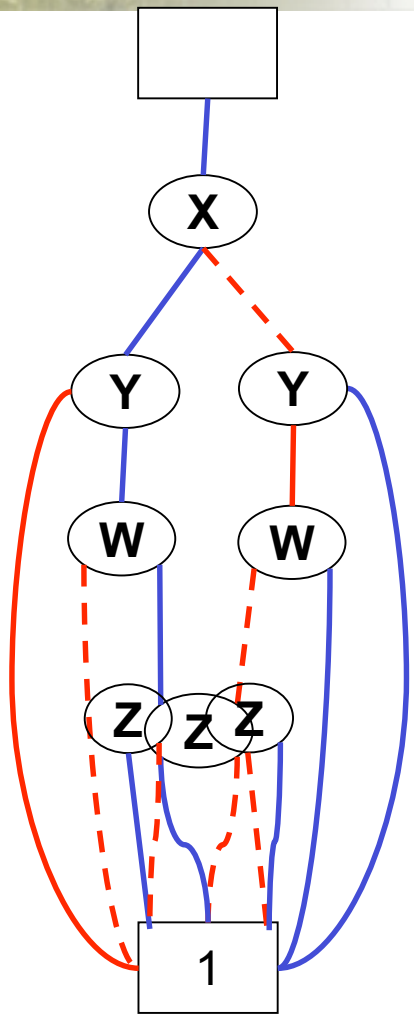
$I_t = (X * Y * Z * W) + (X' * Y' * Z' * W')$

# Old BDD Construction Cont..

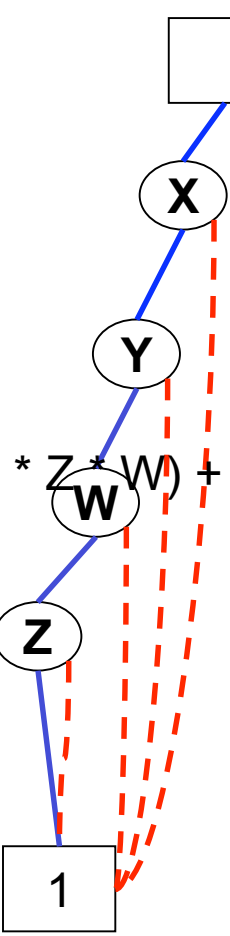
- Garbage Problem



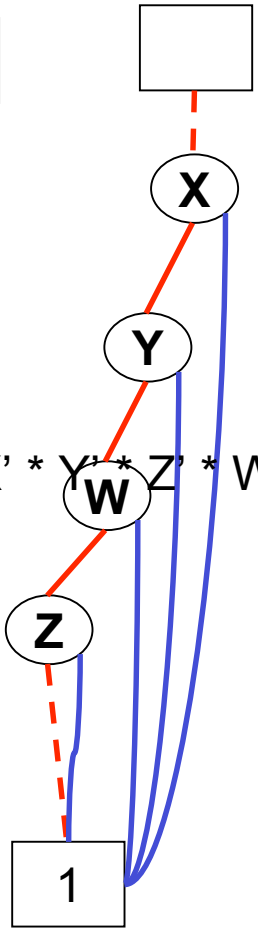

# New BDD Construction




$I_{trace} = (X * Y * Z * W) + (X' * Y' * Z' * W')$

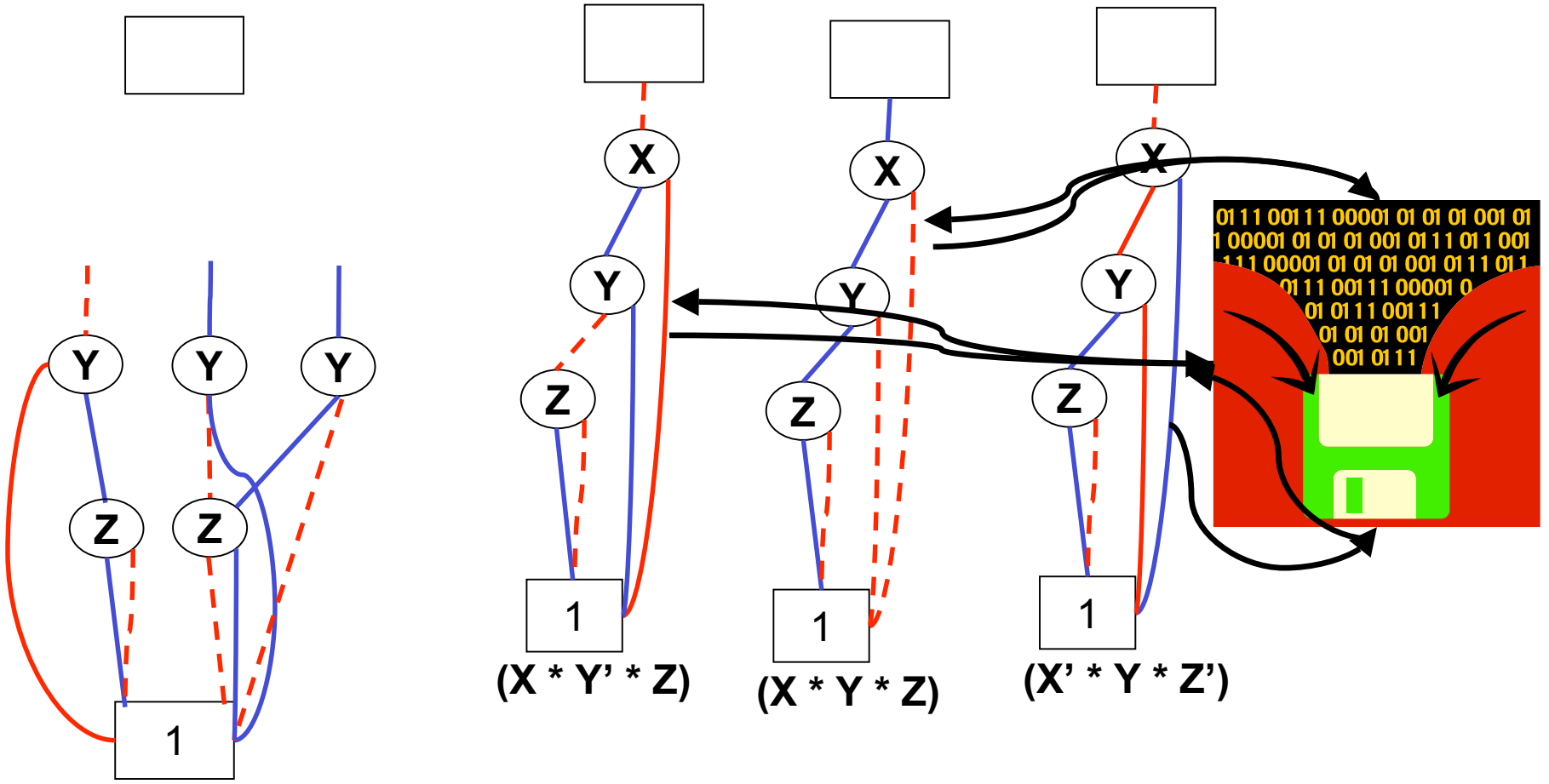



$I_t = (X * Y * Z * W)$

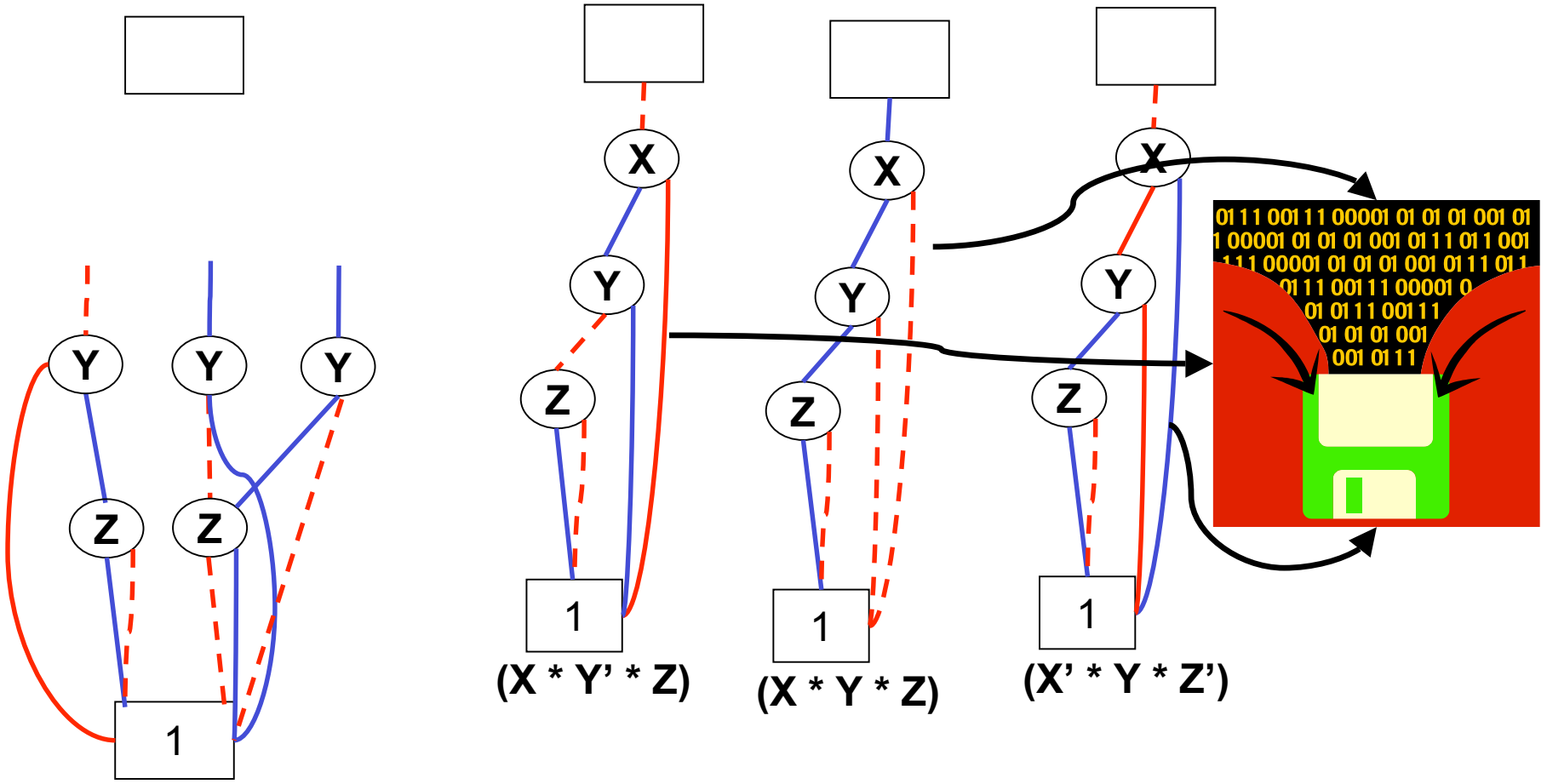



$I_t = (X' * Y' * Z' * W')$

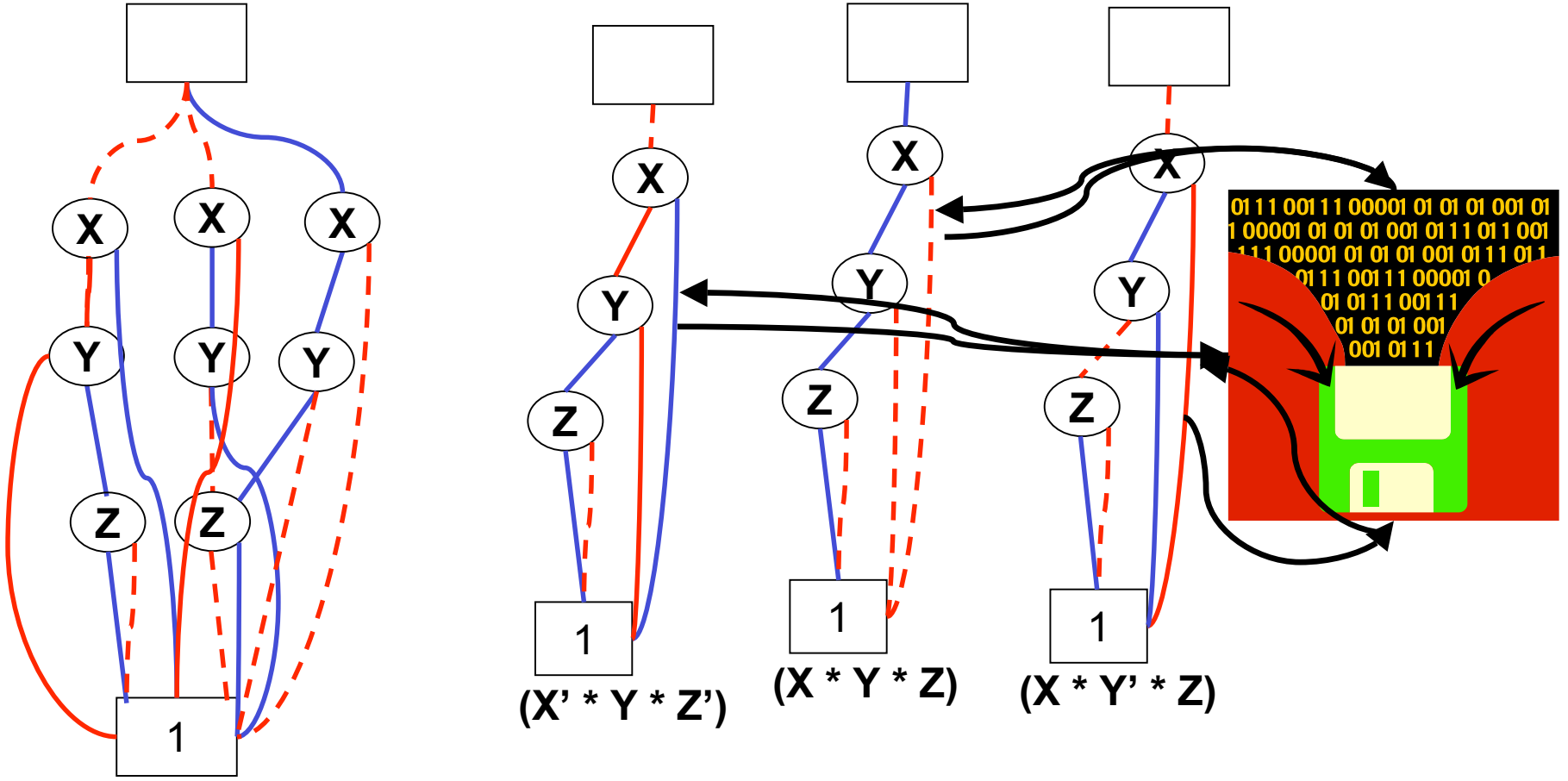
# Construction Implementation



# Radix Sort Search Reduction



# Radix Sort Continued

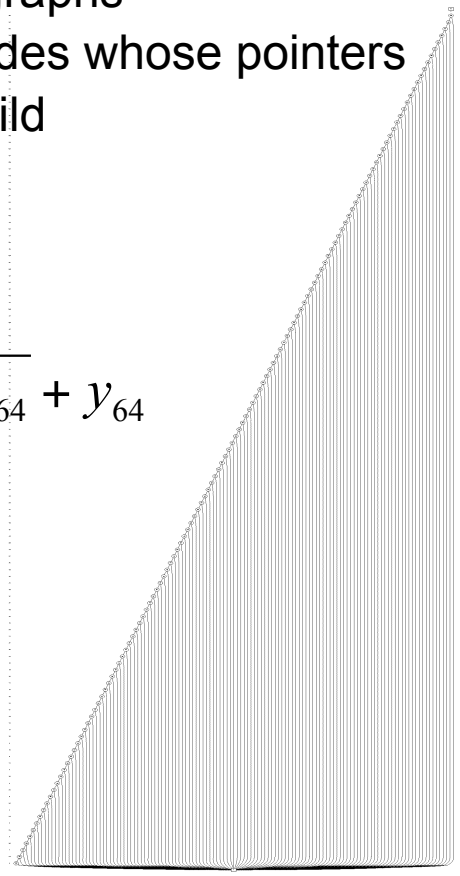


# ZDDs

## BDDs

1. Remove isomorpic graphs
2. Skip and remove nodes whose pointers point to the same child

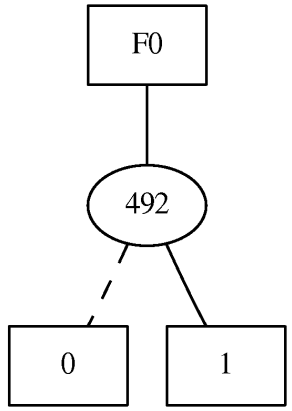
$$\overline{x_0} + \overline{y_0} + \overline{x_1} + \overline{y_1} \ldots + \overline{x_{64}} + y_{64}$$



## ZDDs

1. Remove isomorpic graphs
2. Remove “1” nodes whose pointers point to “0”

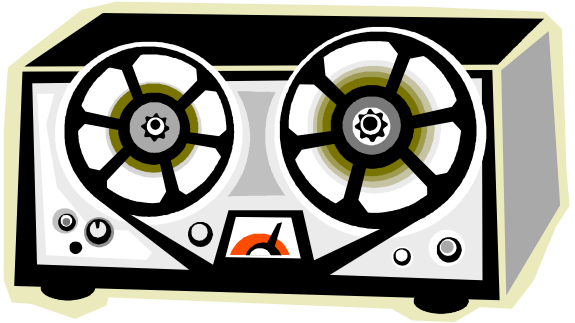
64



# Questions?

# Bonus Slides

# Prior Compression Methods



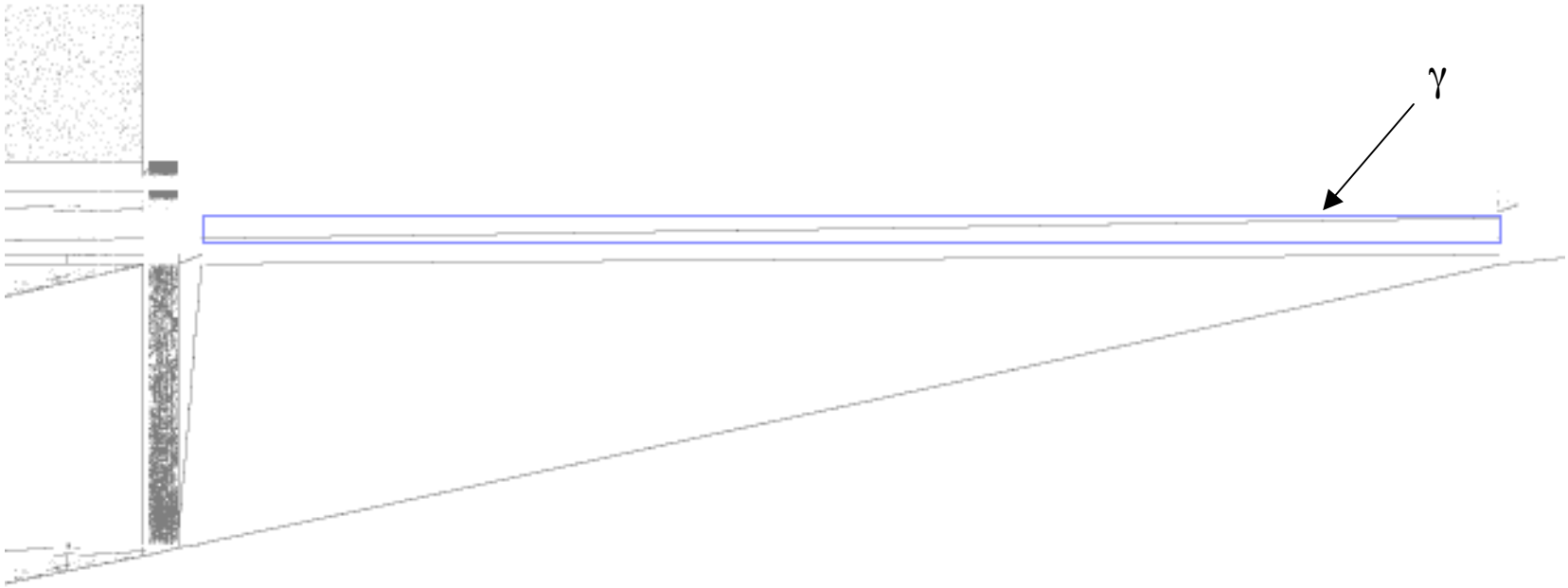
- Run Time Compression
  - Requires compression/decompression of the trace (even if it does not all reside on disk at any time)
  - Much like searching through a tape drive
- SEQUITOR compression

# Global Analysis Techniques

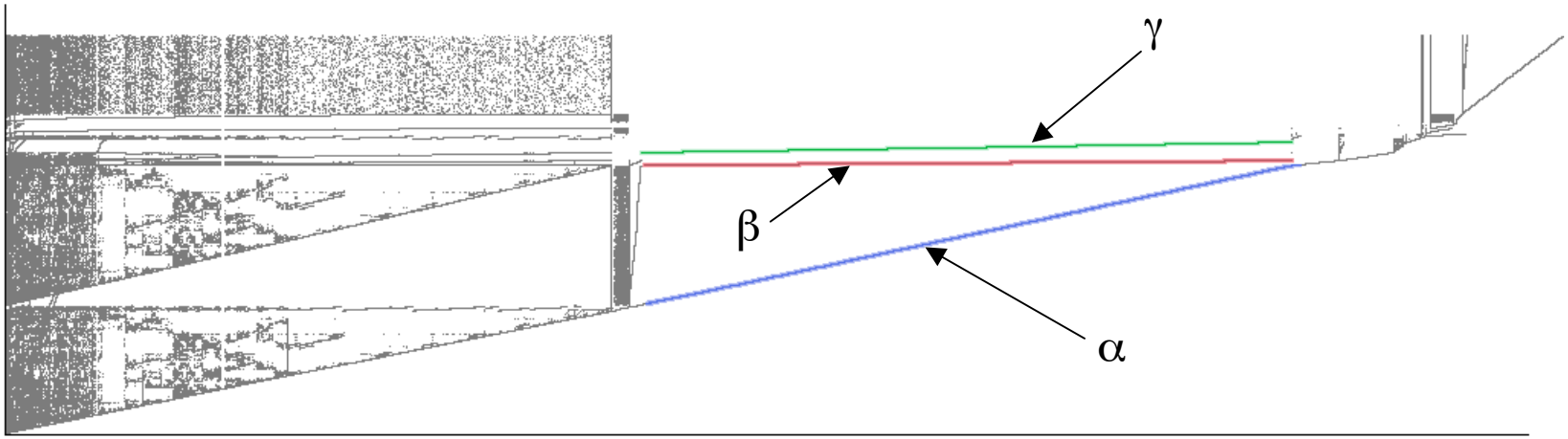
- Visualization
  - Visualize dependence chains
  - Move about a program quickly while viewing trace information
  - Requires fast recovery of data from different locations in the trace
- Trace Slicing
  - Moving forward or backwards in a trace based on a flow (control, dependences, etc)
  - Requires information from disparate trace locations
- Variable Liveness Analysis

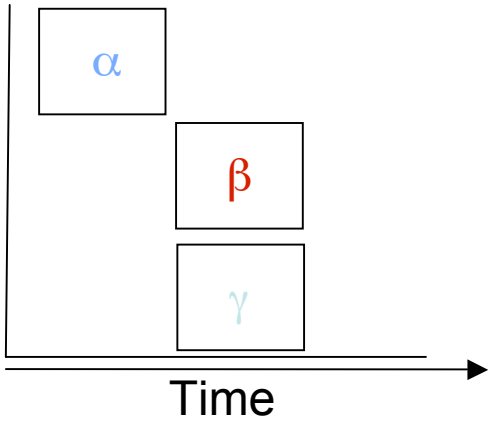
# Case Study - Region Select

- Zoom in on interesting region
- BDD created by (X > Y)*(DINxRDY)

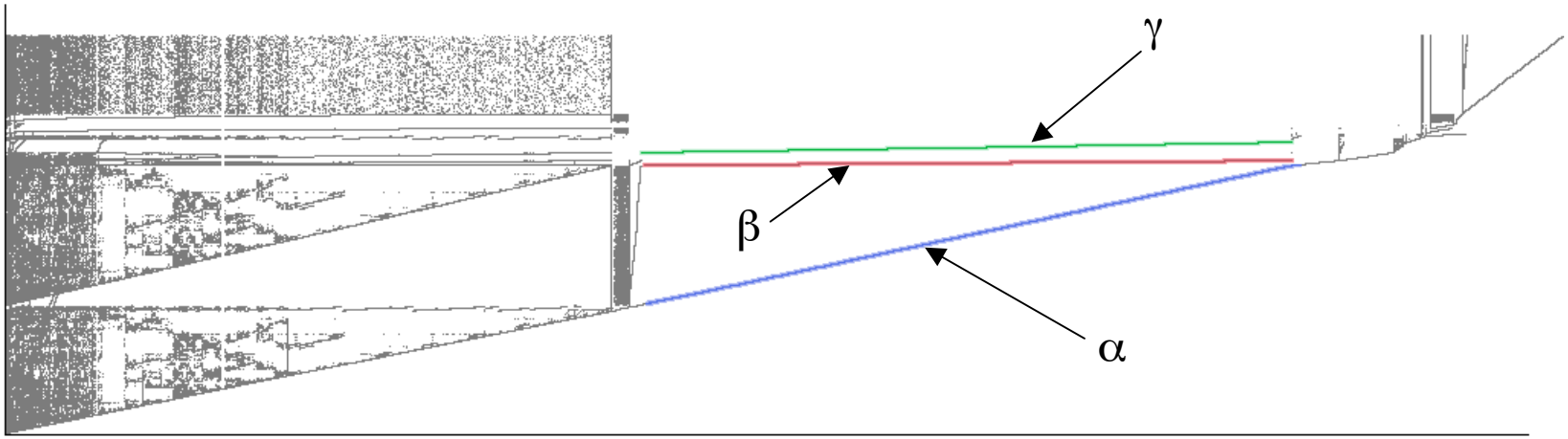


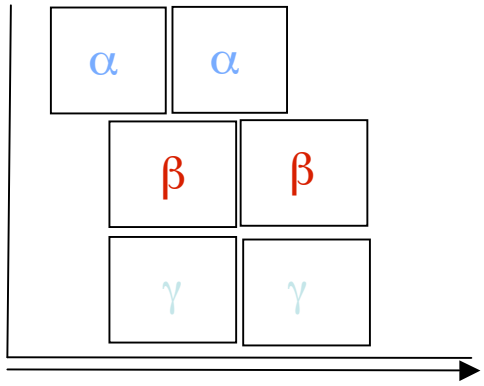
# Case Study - TLP



# Case Study - TLP



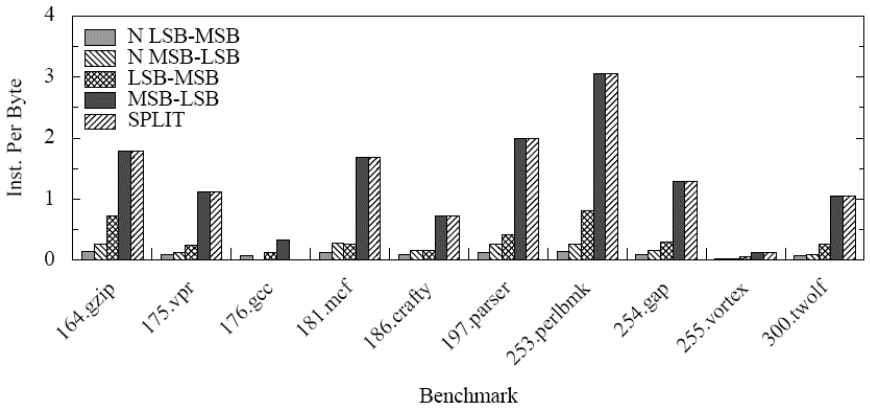



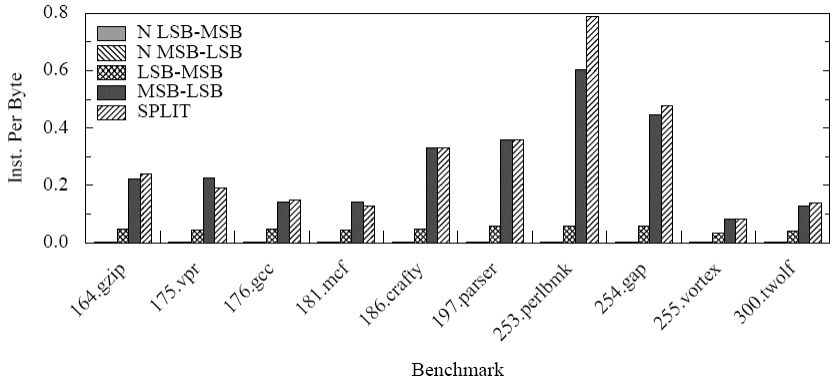
# BDD Creation - Variable Order

| Index | L-M | M-L | I L-M | I M-L | Split |
|---|---|---|---|---|---|
| 0 | A0 | A3 | A0 | A3 | A1 |
| 1 | A1 | A2 | B0 | B3 | B1 |
| 2 | A2 | A1 | A1 | A2 | A0 |
| 3 | A3 | A0 | B1 | B2 | B0 |
| 4 | B0 | B3 | A2 | A1 | A3 |
| 5 | B1 | B2 | B2 | B1 | B3 |
| 6 | B2 | B1 | A3 | A0 | A2 |
| 7 | B3 | B0 | B3 | B0 | B2 |

# BDD Creation Size - Variable Order

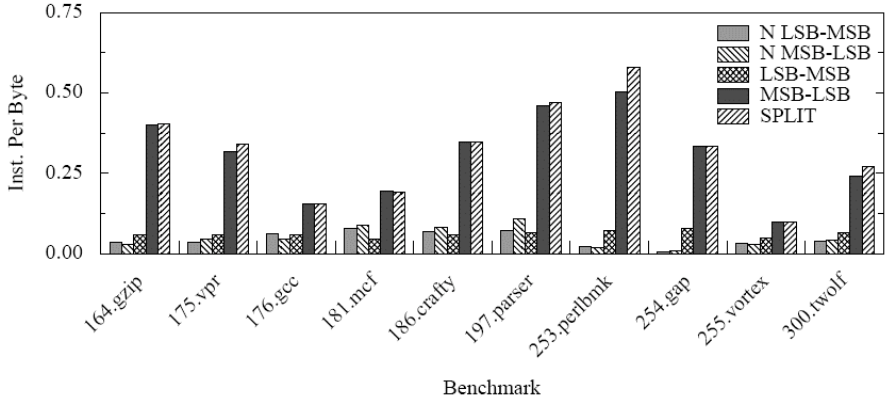
- DIN x DIN



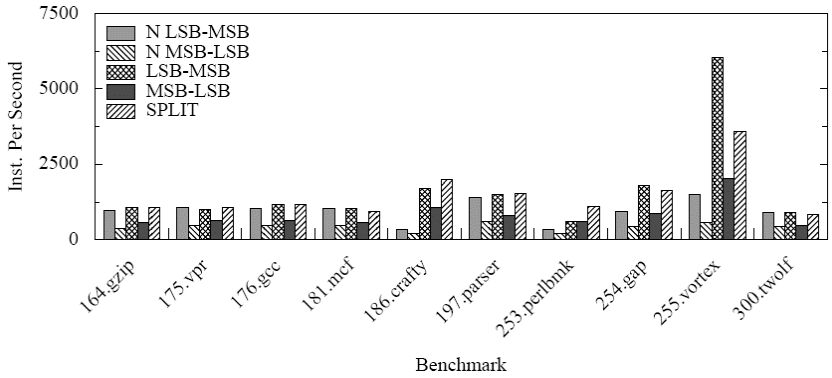

- DIN x PC

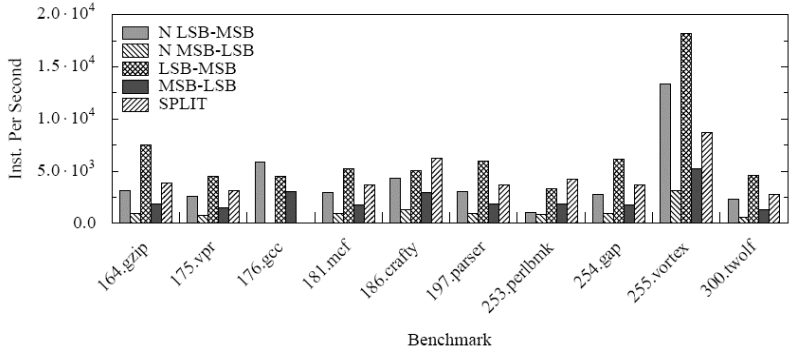
- DIN x RDY



# BDD Creation Time - Variable Order

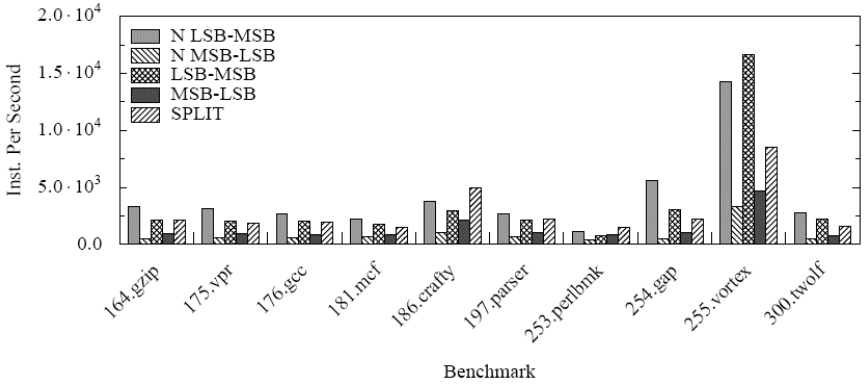
- DIN x DIN




- DIN x PC

- DIN x RDY

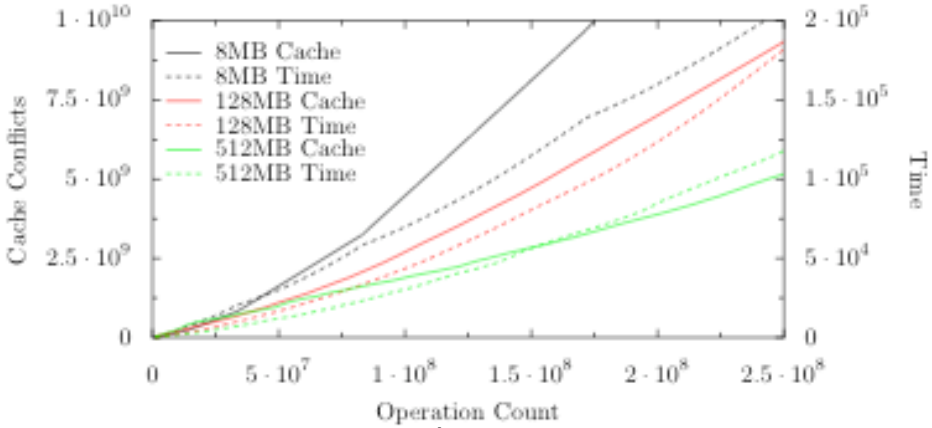


# BDD Creation - RAM

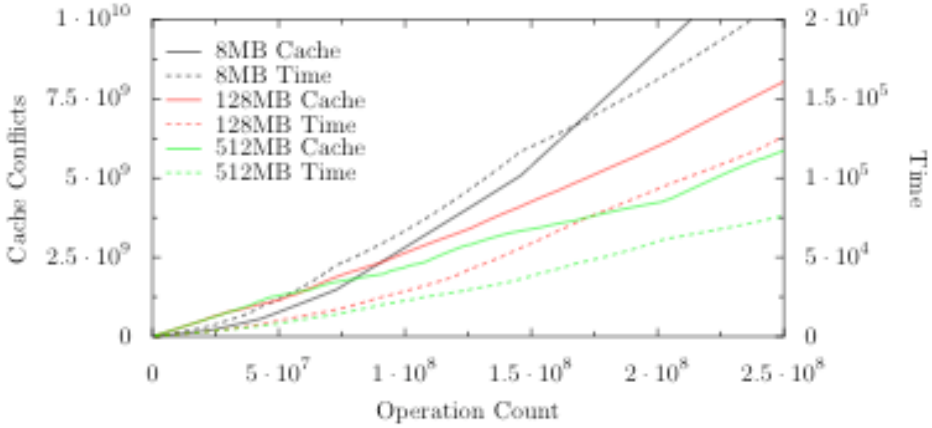
BDD Creation System Memory

- System uses CUDD
- CUDD caches unique nodes in a unique table
- More memory helps reduce cache conflicts

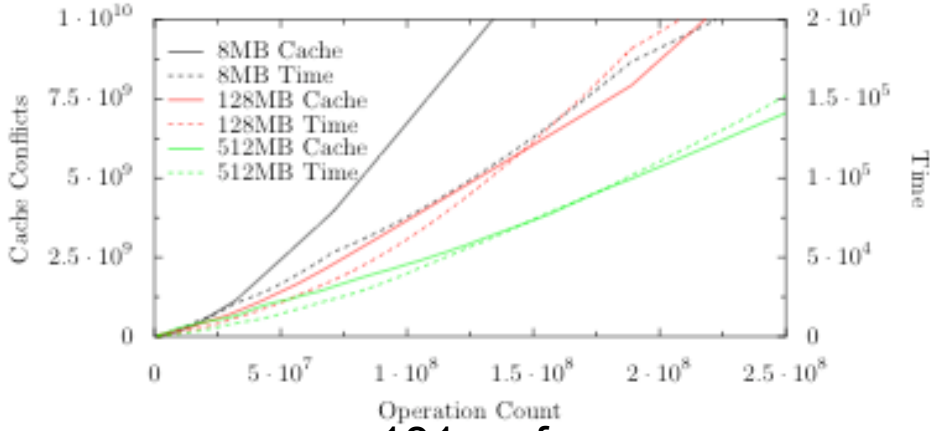
# BDD Creation - RAM




175.vpr




186.crafty

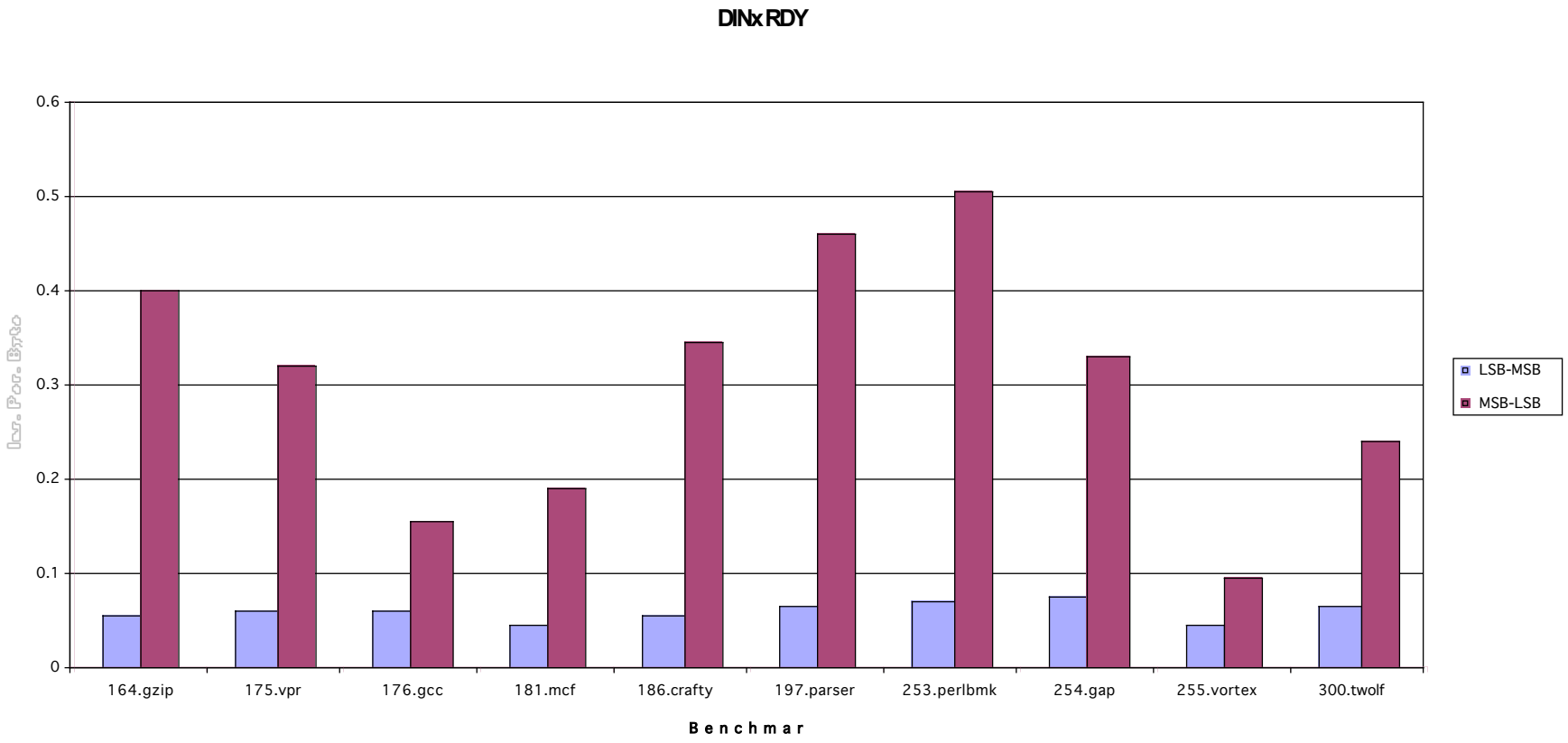



181.mcf

# Current Compression Approaches

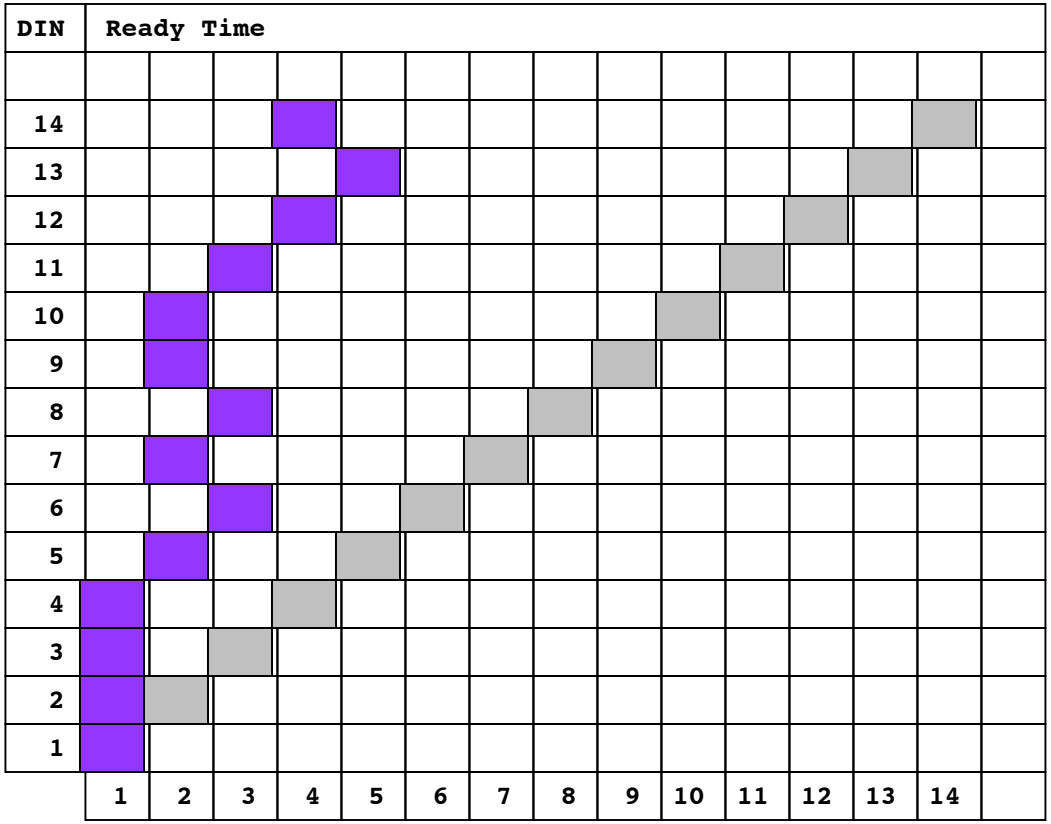
- Run Time Compression
  - Requires compression/decompression of the trace (even if it does not all reside on disk at any time)
  - Much like searching through a tape drive
- SEQUITOR compression (grammar-based)
  - C.G. Nevill-Manning and I.H. Witten. Compression and Explanation using Hierarchical Grammars. *The Computer Journal 40.*
  - Larus et al., “Whole Program Paths”
  - Great for sequence matching

# BDD Creation Size - Variable Order



# Traces as Tuple Sets

| DIN | Instruction | DIN,RDY |
|---|---|---|
| | | |
| 14 | mov r3,r4 | (14,4) |
| 13 | st 0(r1),r5 | (13,5) |
| 12 | add r5,r3,r4 | (12,4) |
| 11 | bnz r2, loop | (11,3) |
| 10 | addi r1,r1,4 | (10,2) |
| 9 | addi r2,r2,-1 | (9,2) |
| 8 | mov r4,r5 | (8,3) |
| 7 | mov r3,r4 | (7,2) |
| 6 | st 0(r1),r5 | (6,3) |
| 5 | add r5,r3,r4 | (5,2) |
| 4 | mov r3, 1 | (4,1) |
| 3 | mov r4, 1 | (3,1) |
| 2 | mov r2, 8 | (2,1) |
| 1 | li r1, &fibarr | (1,1) |



# Tuples as Boolean Functions

- Boolean Functions can represent arbitrary sets of data, if the elements are (or can be) encoded in binary
- Lets make a Boolean function for {(2,4),(3,5)}

| | X | | | Y | | |
|---|---|---|---|---|---|---|
| | $X_2$ | $X_1$ | $X_0$ | $Y_2$ | $Y_1$ | $Y_0$ |
| (2,4) | 0 | 1 | 0 | 1 | 0 | 0 |
| (3,5) | 0 | 1 | 1 | 1 | 0 | 1 |

# Tuples as Boolean Functions

| | X | | | Y | | |
|---|---|---|---|---|---|---|
| | $X_2$ | $X_1$ | $X_0$ | $Y_2$ | $Y_1$ | $Y_0$ |
| (2,4) | 0 | 1 | 0 | 1 | 0 | 0 |
| (3,5) | 0 | 1 | 1 | 1 | 0 | 1 |

$I_T(X,Y) = 1$ For (2,4) and (3,5) and 0 otherwise

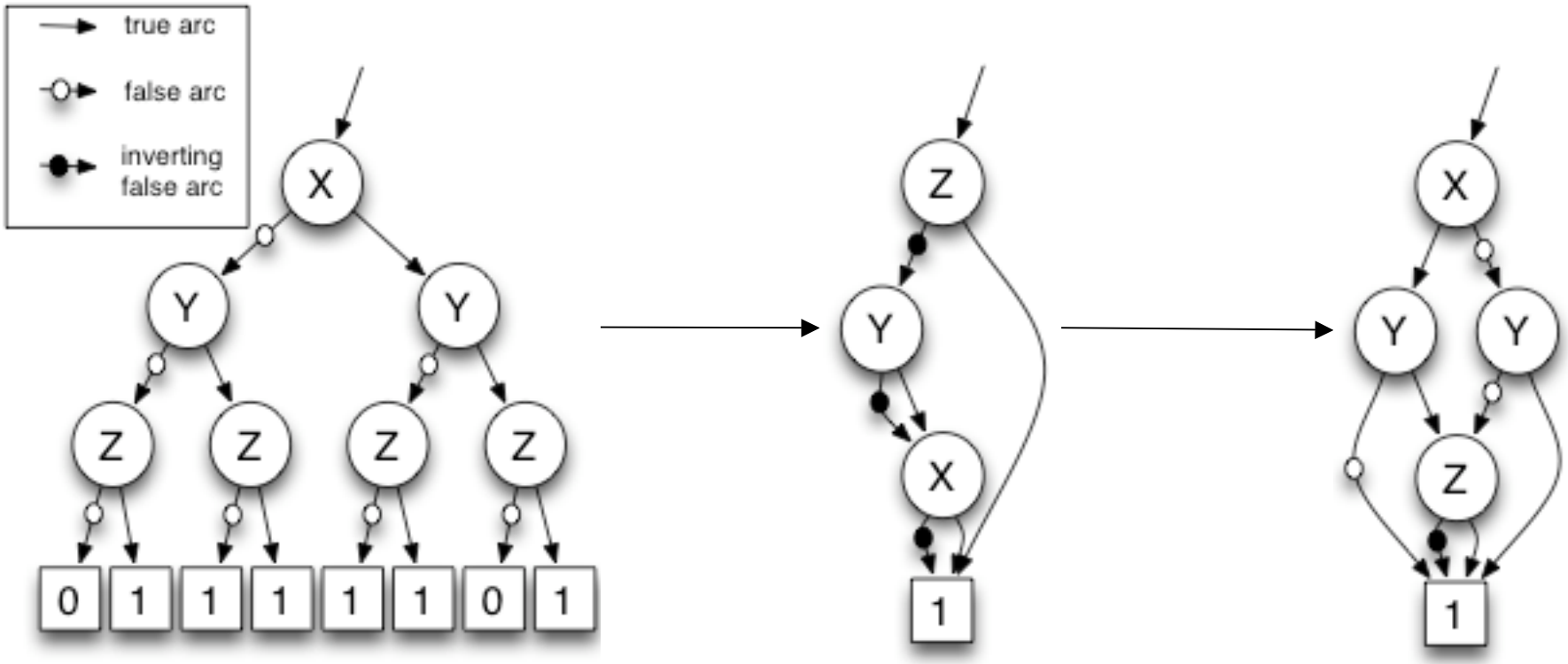
$$I_{\{(2,4)\}} = (\neg X_2 \wedge X_1 \wedge \neg X_0 \wedge Y_2 \wedge \neg Y_1 \wedge \neg Y_0)$$

$$I_{\{3,5\}} = (\neg X_2 \wedge X_1 \wedge X_0 \wedge Y_2 \wedge \neg Y_1 \wedge Y_0)$$

$$I_T = I_{\{2,4\}} \vee I_{\{3,5\}}$$

# The BDD Data Structure

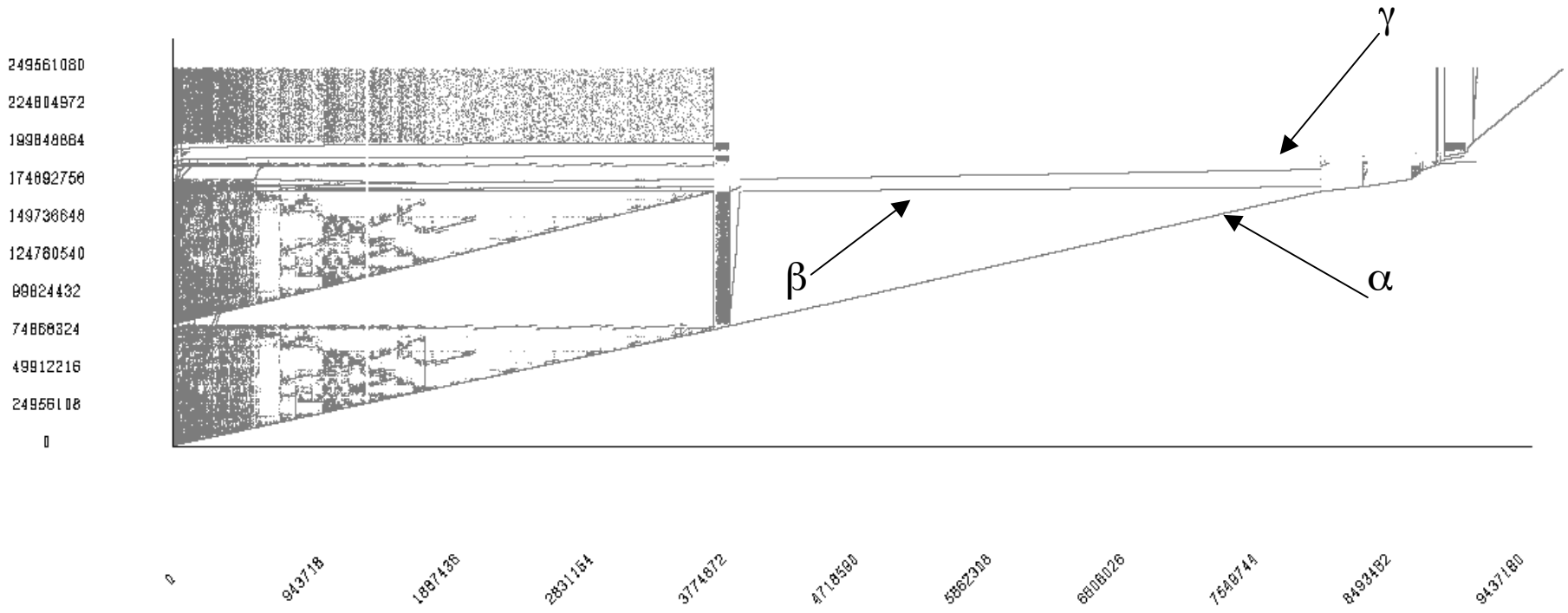
$$F(X,Y,Z) = x'y + xy' + z$$



# A Case Study

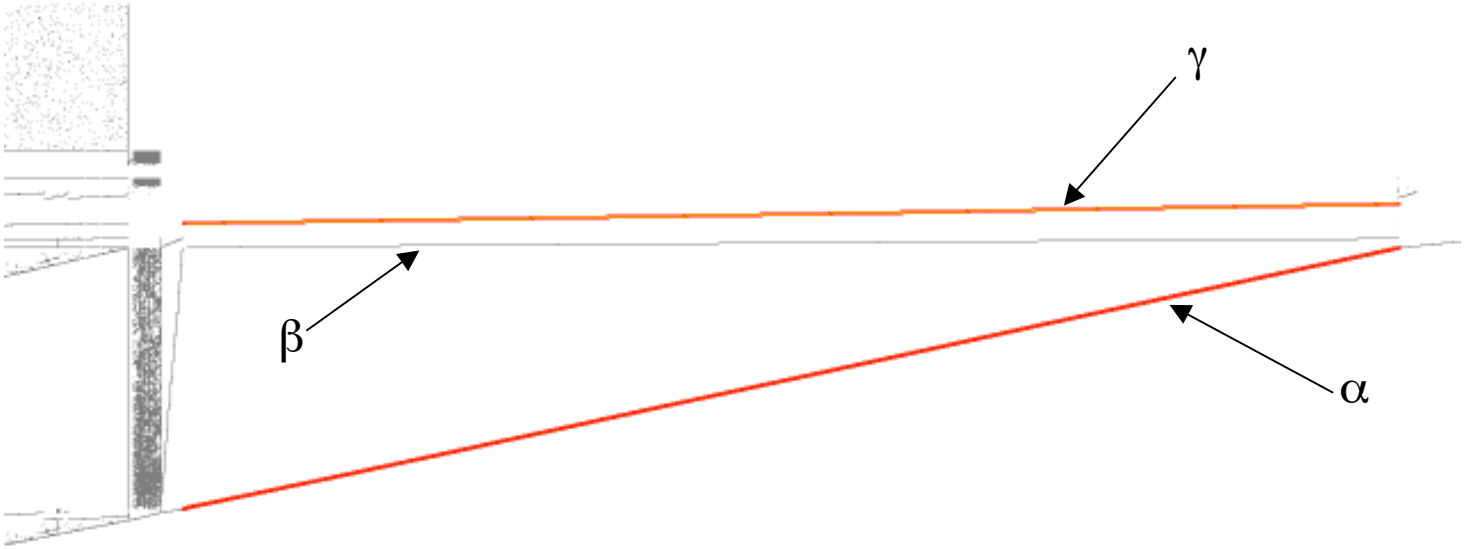
- Examine 175.vpr from SPEC INT 2000

# Initial Visualization

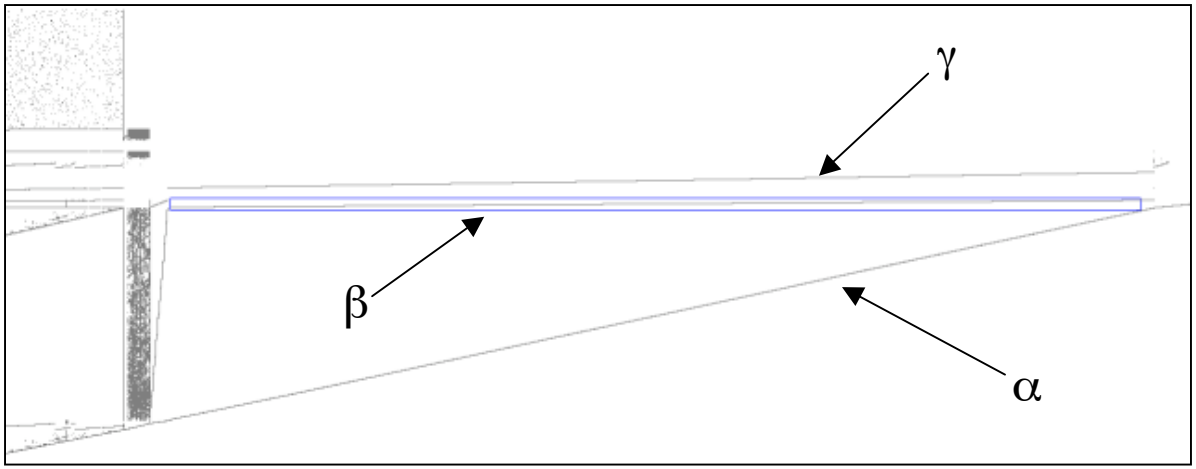


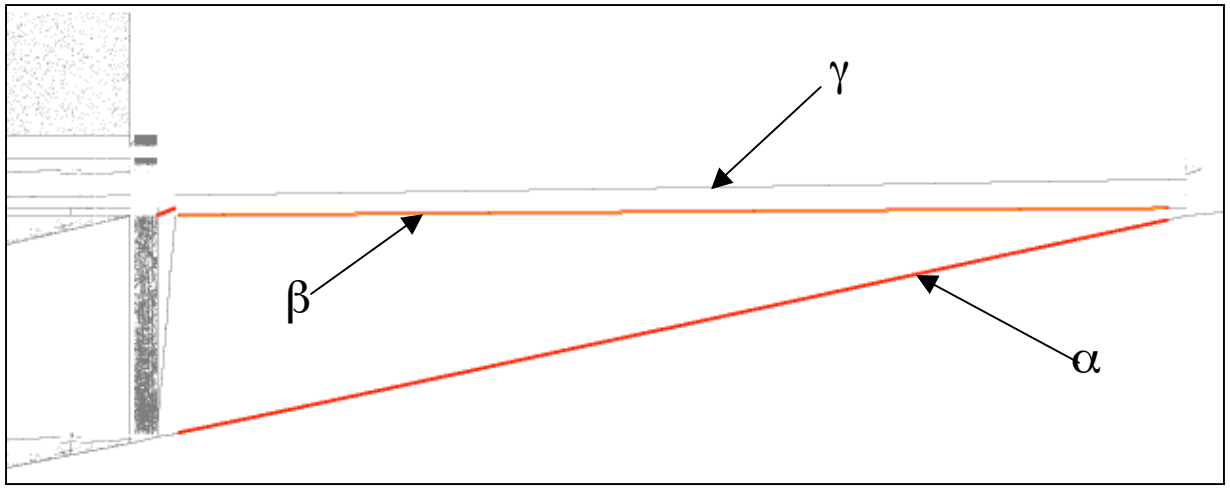
# Case Study - Reverse Slice

- Perform a single reverse data dependence slice on the selected region

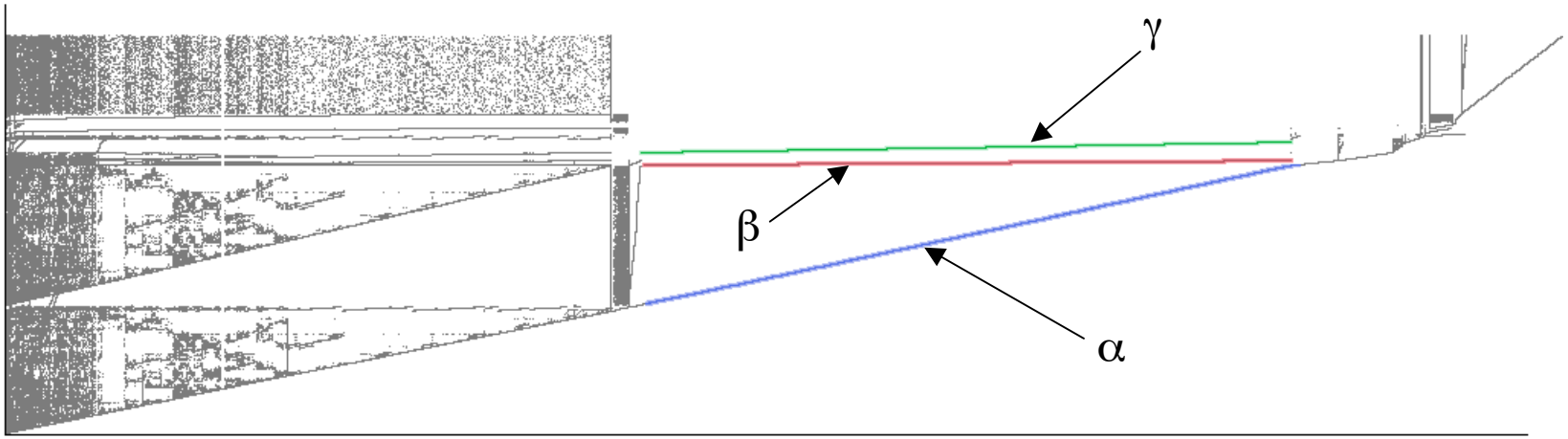


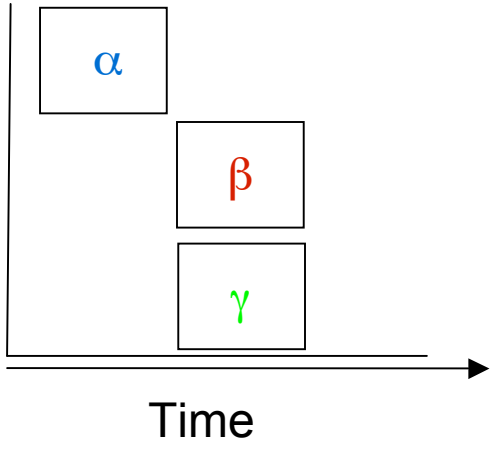
# Case Study - Select and Slice β

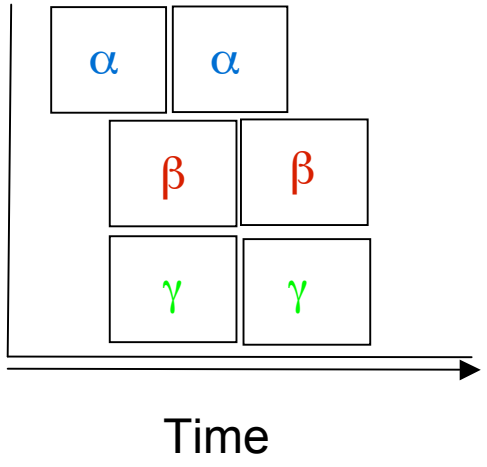


# ParaMeter - Extracting TLP

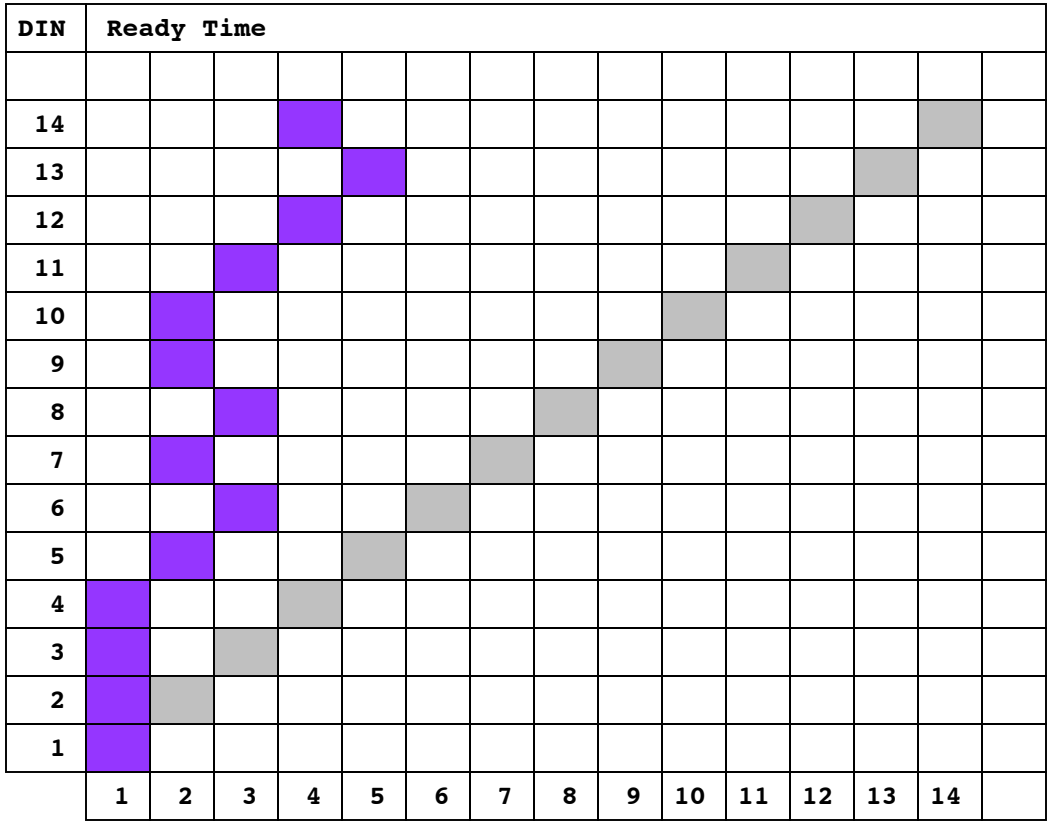


# DIN vs. Ready Time

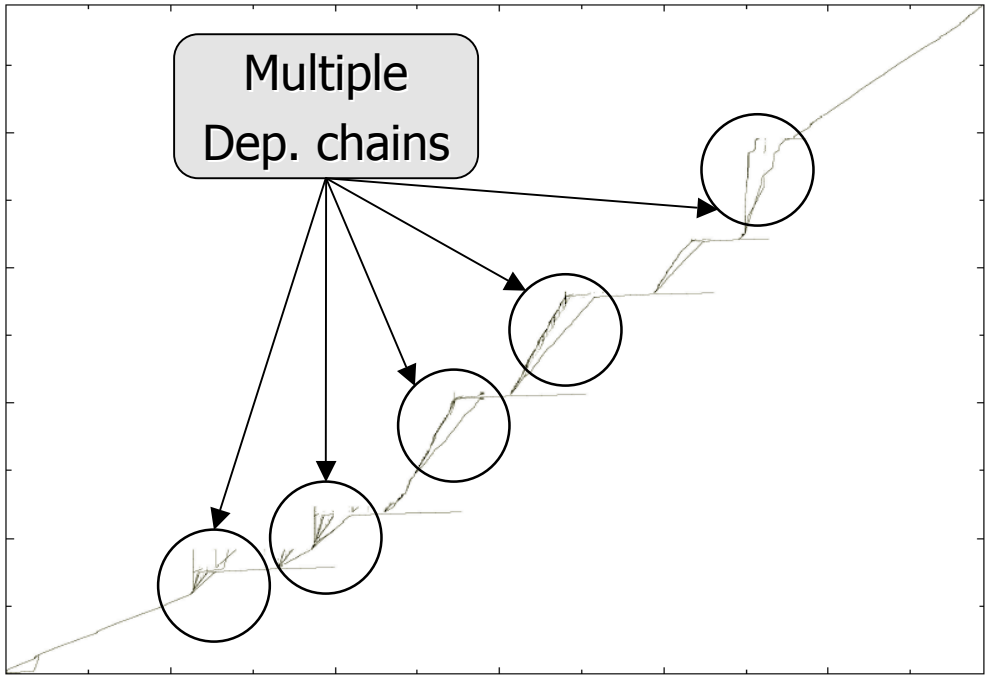
| DIN | Instruction | Mem Addr |
|---|---|---|
| | … | |
| 14 | mov r3,r4 | |
| 13 | st 0(r1),r5 | 0xbee4 |
| 12 | add r5,r3,r4 | |
| 11 | bnz r2, loop | |
| 10 | addi r1,r1,4 | |
| 9 | addi r2,r2,-1 | |
| 8 | mov r4,r5 | |
| 7 | mov r3,r4 | |
| 6 | st 0(r1),r5 | 0xbee0 |
| 5 | add r5,r3,r4 | |
| 4 | mov r3, 1 | |
| 3 | mov r4, 1 | |
| 2 | mov r2, 8 | |
| 1 | li r1, &fibarr | |



# Distant ILP Characteristics

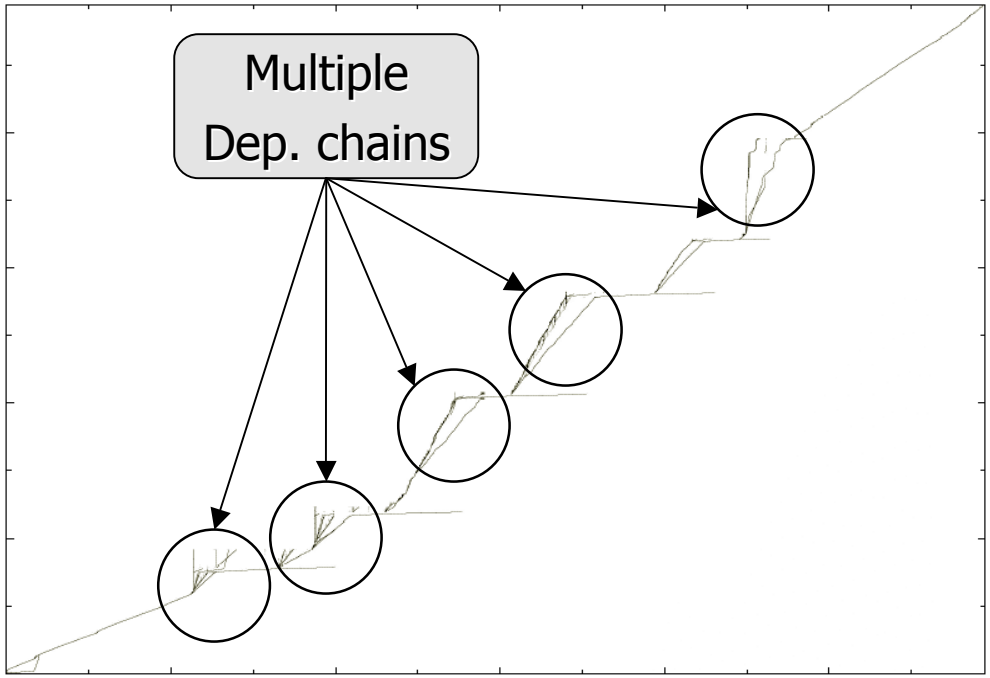
DIN plot for 254.gap (IA64,gcc,inf)

- Multiple divergent lines indicate parallel "threads" of execution
- Independent chains of dependences (DDCs)
- CMP/CMT systems
  - Execute chains on separate cores



# Trace Analysis Issues

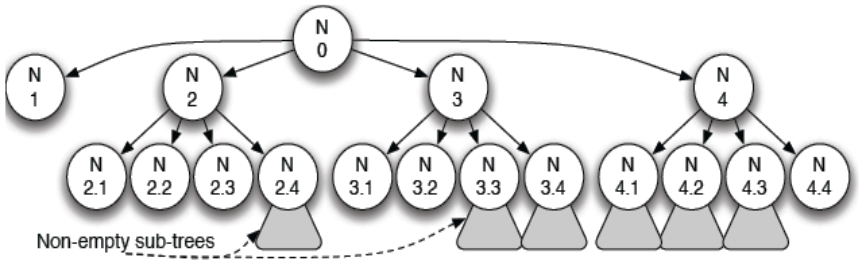
- Trace Size
- With a trace of significant size global analysis is impractical
- Prior compression methods do not provide rapid global analysis

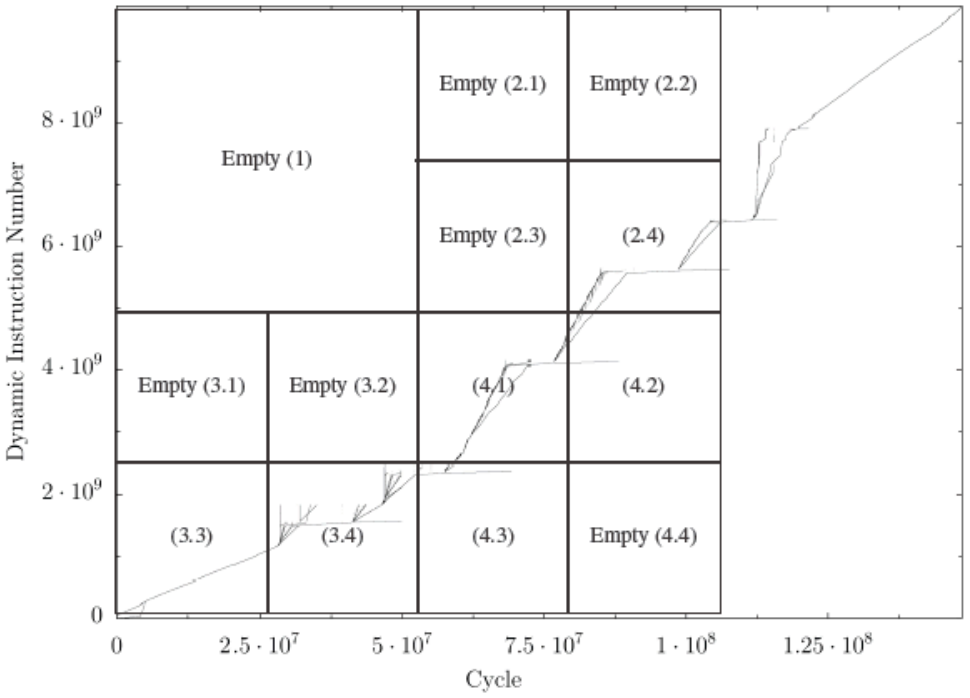


# DIN x RDY Visualization

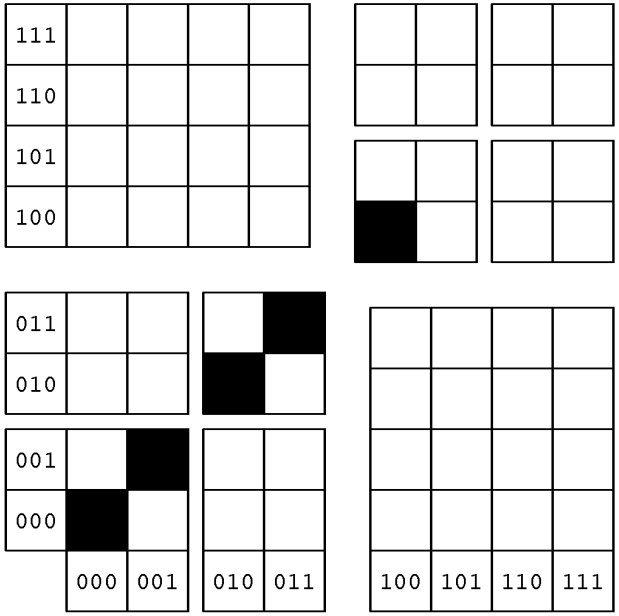
- Depth First Traversal of BDD
  - Quad - Tree Optimization
  - Depth Clipping
  - Viewport Clipping

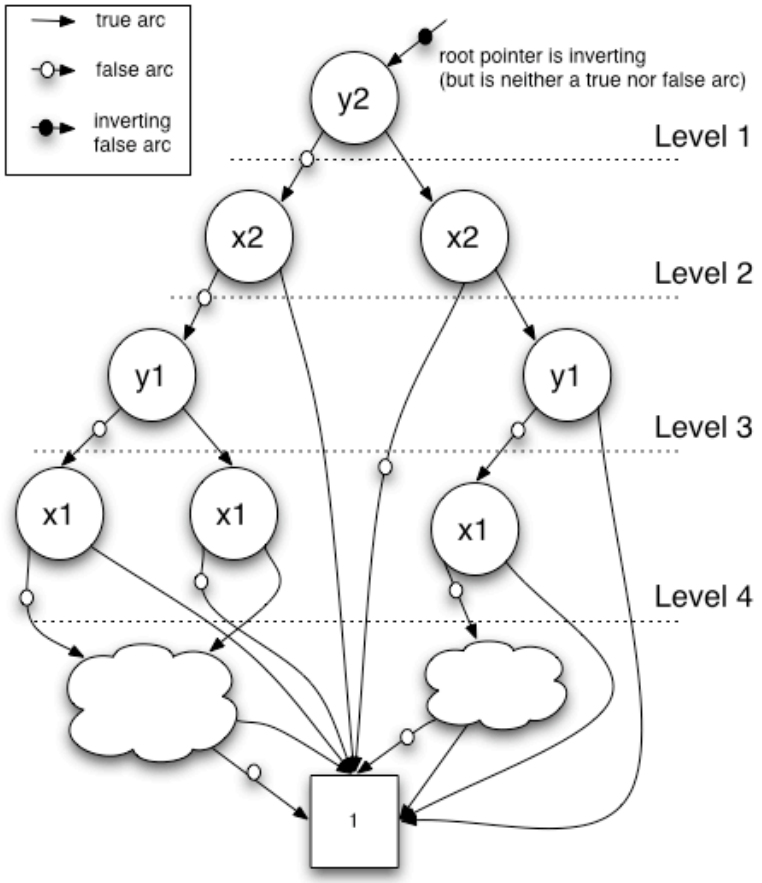
# Quad-Tree Optimization



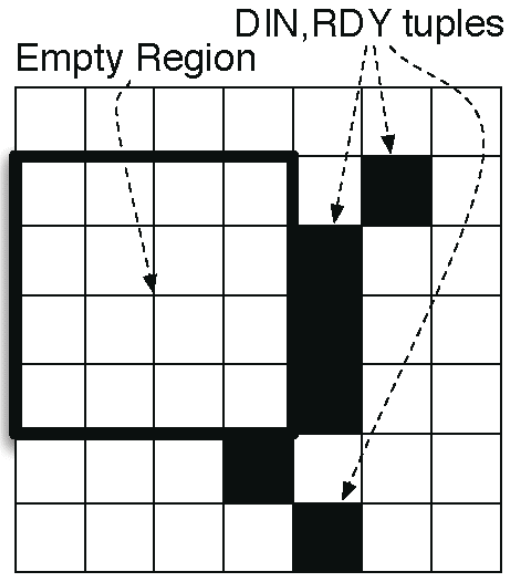



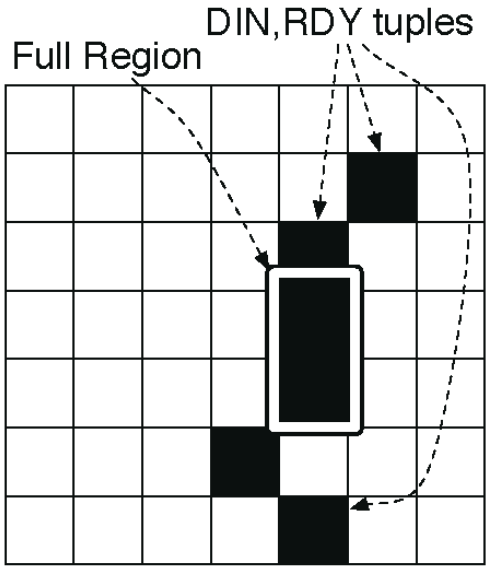
# Quad-Tree Optimization

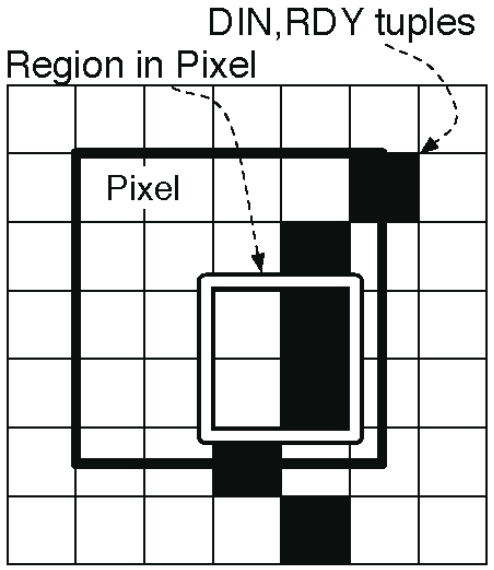


# Depth Clipping



# ParaMeter vs. Adamantium

- Adamantium uses Burtscher[CGO ‘05] bzip2 streamed compression
- DINxRDY plot data extraction

# ParaMeter vs. Adamantium

| Benchmark | Adamantium (s) | ParaMeter (s) | Perf. Gain |
|---|---|---|---|
| **164.gzip** | 198.922 | 0.00552 | 36,036 |
| **175.vpr** | 196.484 | 0.01595 | 12,318 |
| **176.gcc** | 220.791 | 0.00676 | 32,661 |
| **181.mcf** | 208.383 | 0.00542 | 38,447 |
| **186.crafty** | 9.05 | 0.00012 | 75,416 |
| **197.parser** | 212.408 | 0.00357 | 59,498 |
| **253.perlbmk** | 192.637 | 0.01095 | 17,592 |
| **254.gap** | 161.067 | 0.00423 | 38,077 |
| **255.vortex** | 2.511 | 0.00006 | 41,850 |
| **256.bzip2** | 3.993 | 0.00009 | 44,366 |
| **300.twolf** | 194.133 | 0.00842 | 23,056 |
| **254.gap(500)** | 342.729 | 0.00716 | 47,867 |

# Forward Slice

- BDD Based Analysis [Whaley and Lam, PLDI 2004]
- Forward slice algorithm:

**function** forward_slice($e$,$I_D$)
  $s := I_D$
  **do**
    $s_{\text{old}} := s$
    $e' := e \wedge s$
    $s' := \exists \mathbf{d^1}.e'$
    $s := s \vee \text{rename}(\mathbf{d^2}, \mathbf{d^1}, s')$
  **while**($s \neq s_{\text{old}}$)
  **return** $s$

# Video