FPL 2010

# Test Compression for Dynamically Reconfigurable Processors

INOUE, Hiroaki
Junya Yamada, Hideyuki Yoneda,
Katsumi Togawa, and Koichiro Furuta

NEC Corporation, and Renesas Electronics

August 31st, 2010

# Today's Key Message

# 2.7

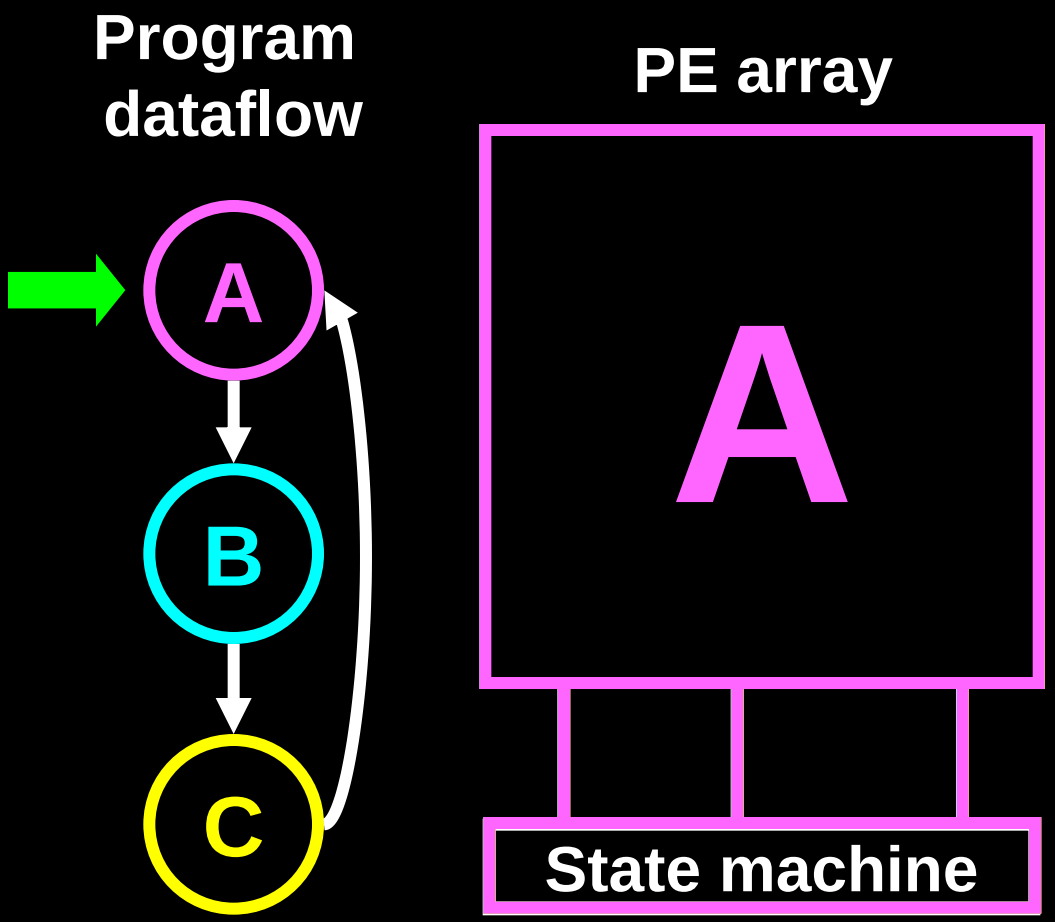
times shorter test time

# Outline

- **Background**
- **Our solution: DPC**
- **Evaluation**
- **Conclusion**

# What is DRP?

## Dynamically switching multiple contexts every cycle




## Benefits:

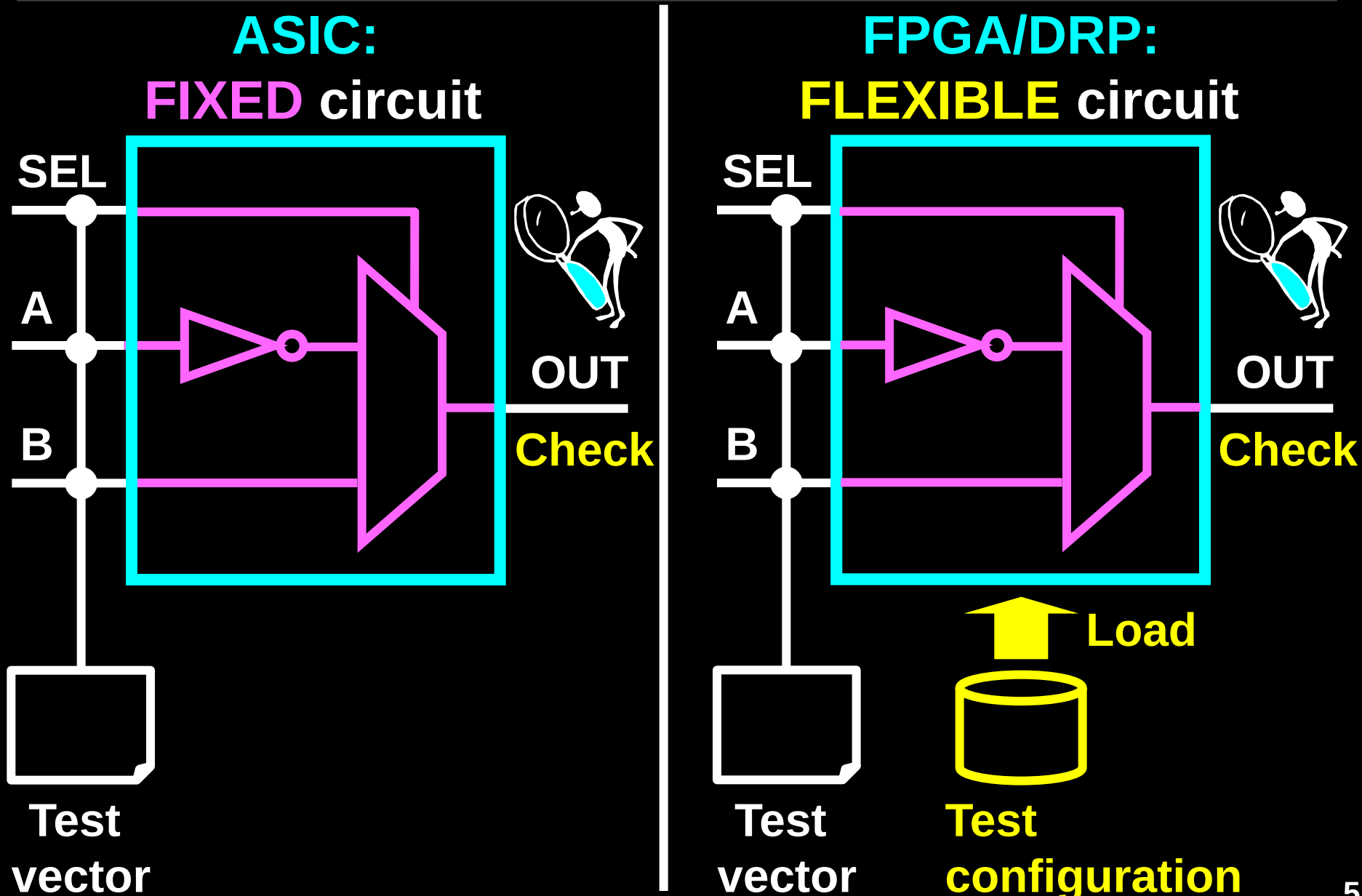
- Full programmability
- High performance
- Small area occupation

## Product examples:

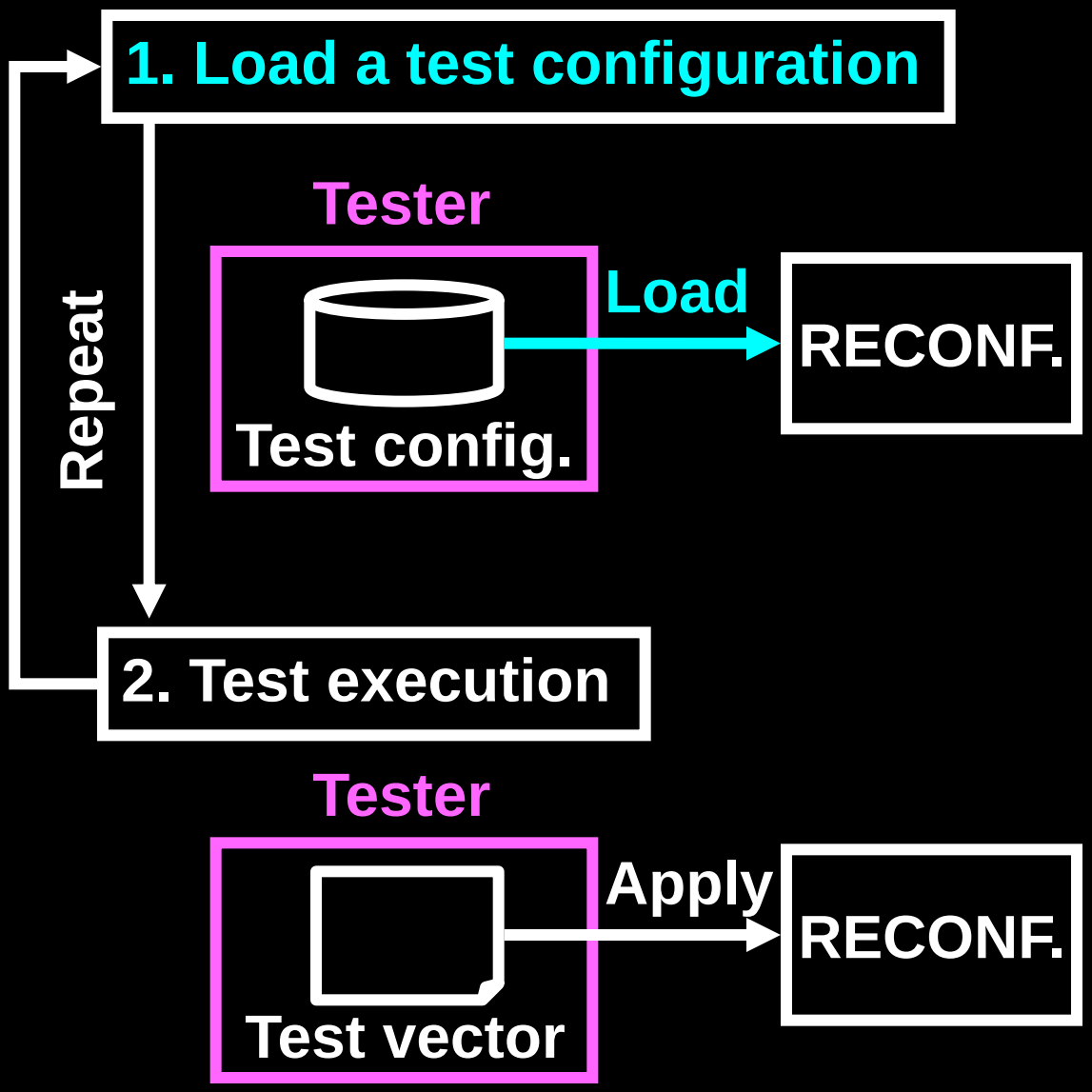
Printers

Cameras

# Testing Reconfigurable Devices



# What is the Problem?




2. Test execution (1%~)


1. Load test configurations (~99%)

**Big problem!**

# Test Time Reduction Techniques

Already done

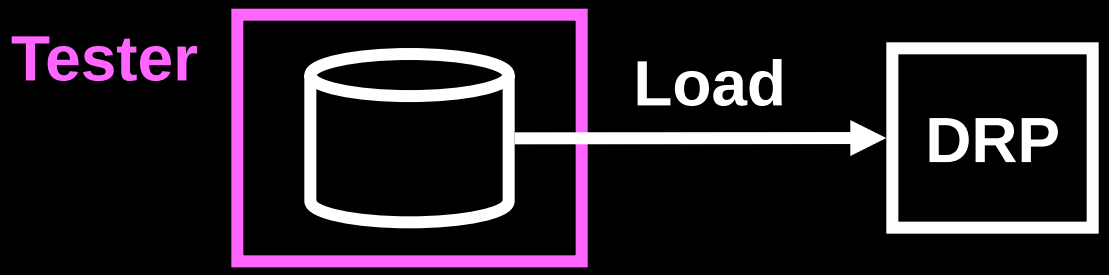
- High clock frequency
- A number of test pins
- Simultaneous testing

Our approach ➡

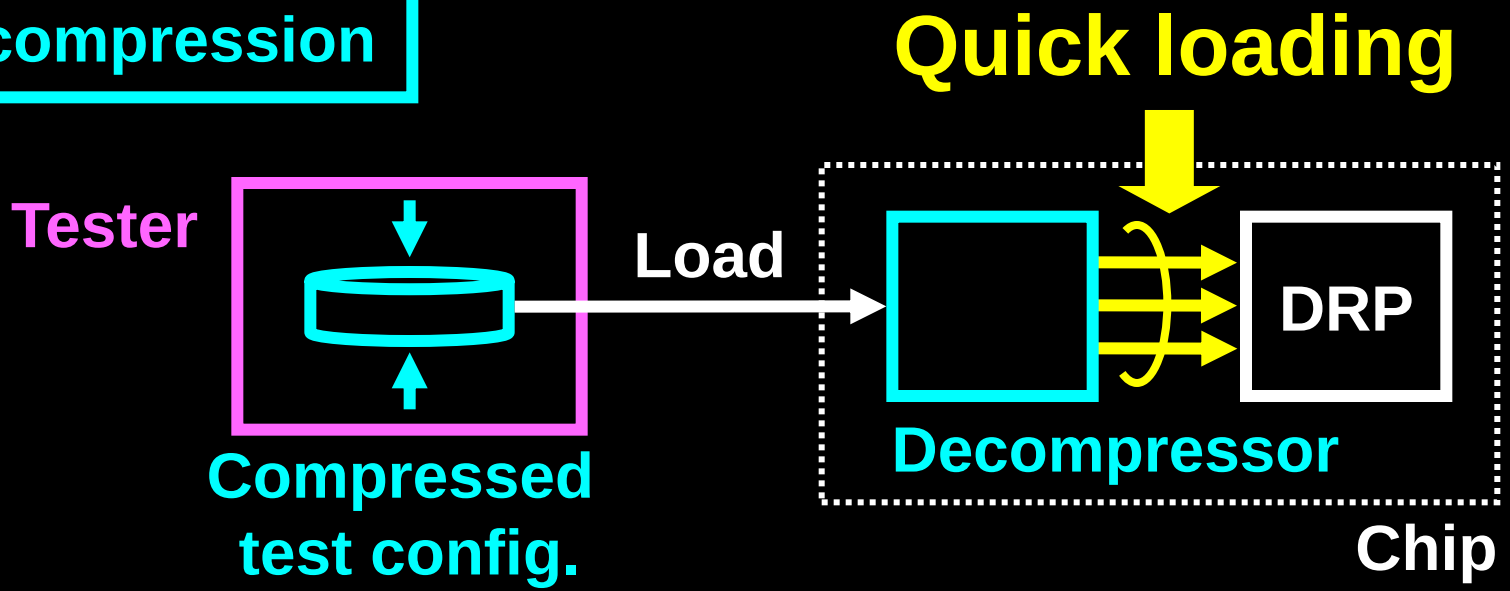
- **Test compression**

# What is Test Compression?

**Normal testing**




**Test compression**

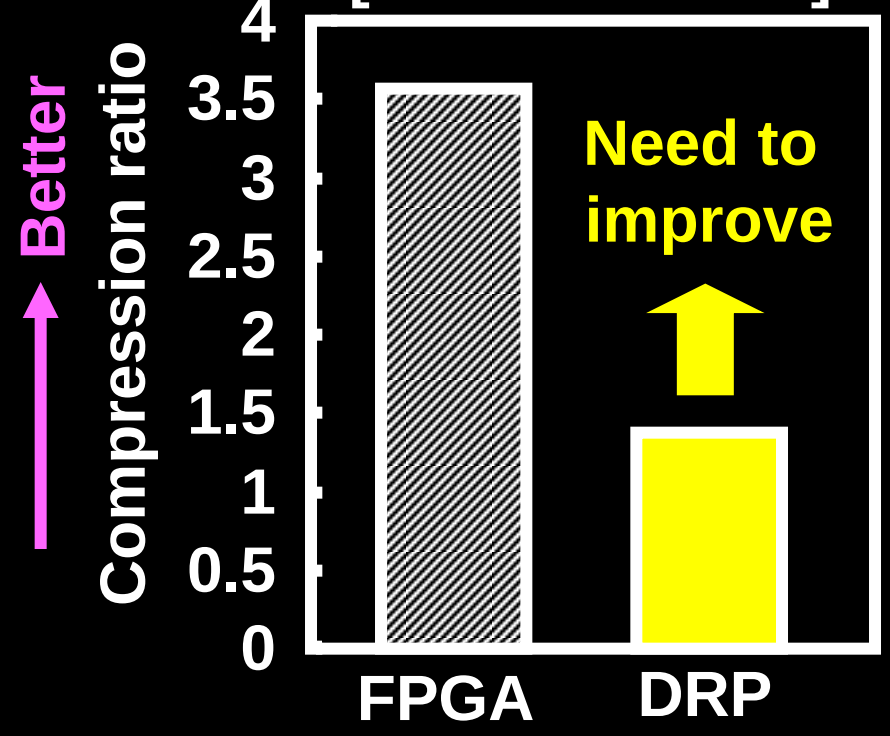


# Requirements

For DRP

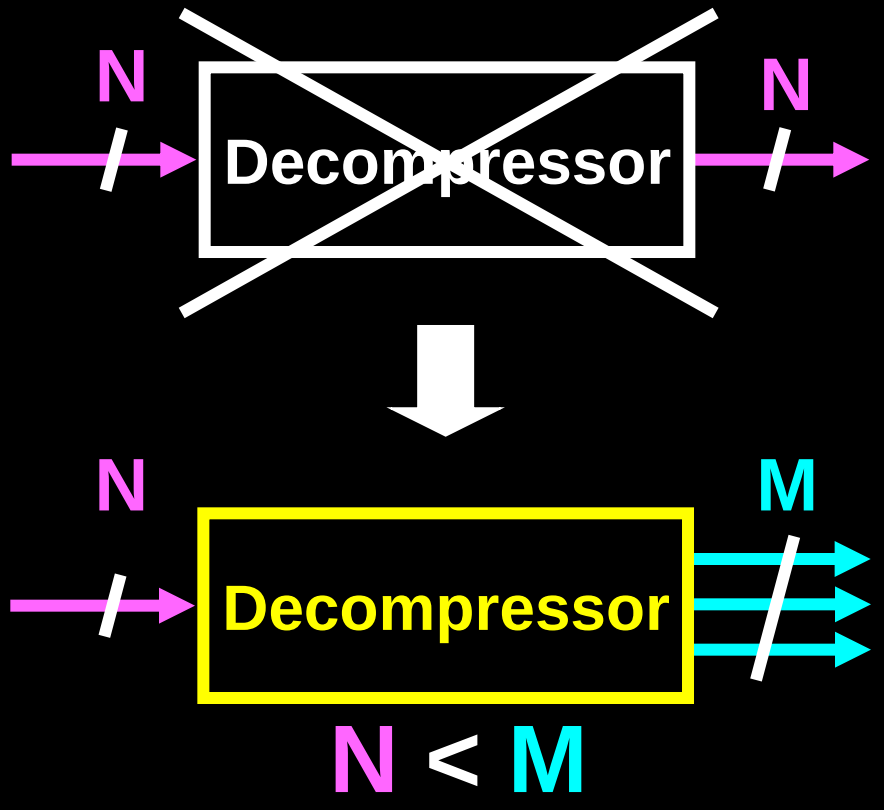
**High compression ratio**

[Kitaoka FPL03]




For test

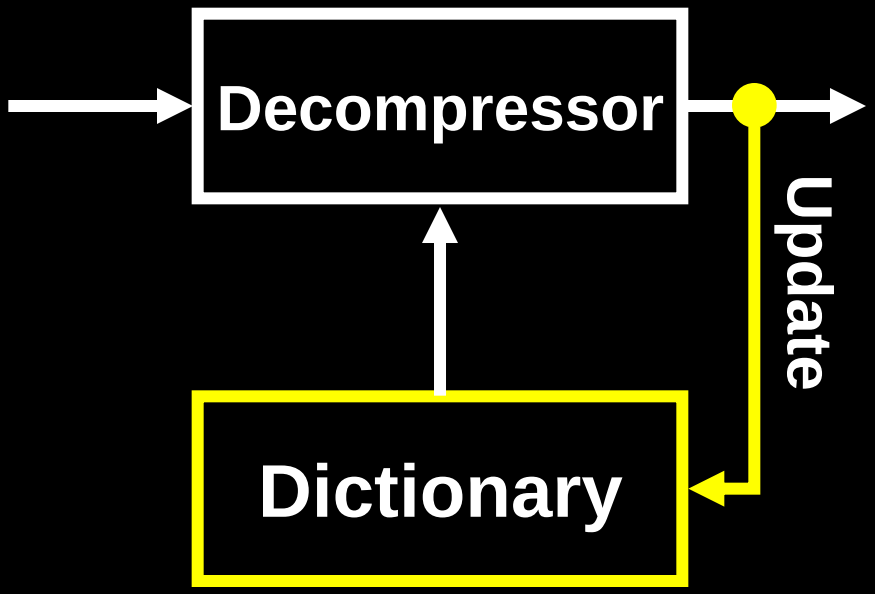
**High decompression bandwidth**



# Conventional Approaches

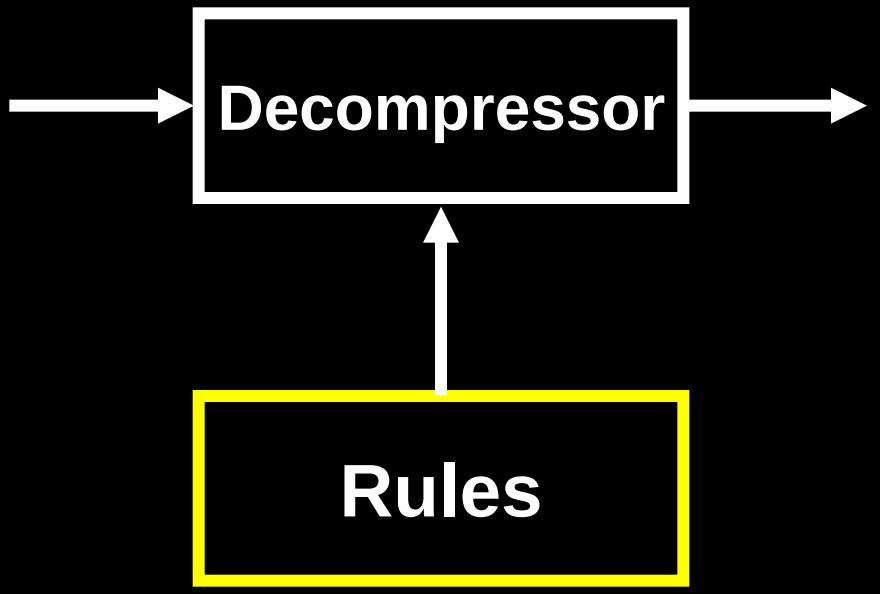
## Dictionary-based (ex. LZ [17])

Use a dictionary updated by data




## Rule-based (ex. FPC [20])

Use pre-defined rules



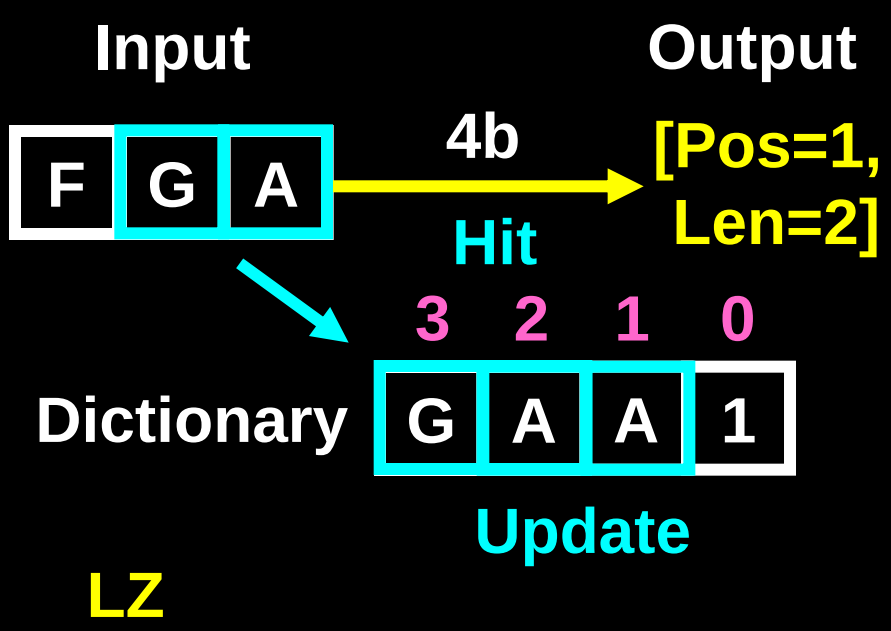
# Dictionary-Based Approach

**Compression ratio: FAIR 😐**

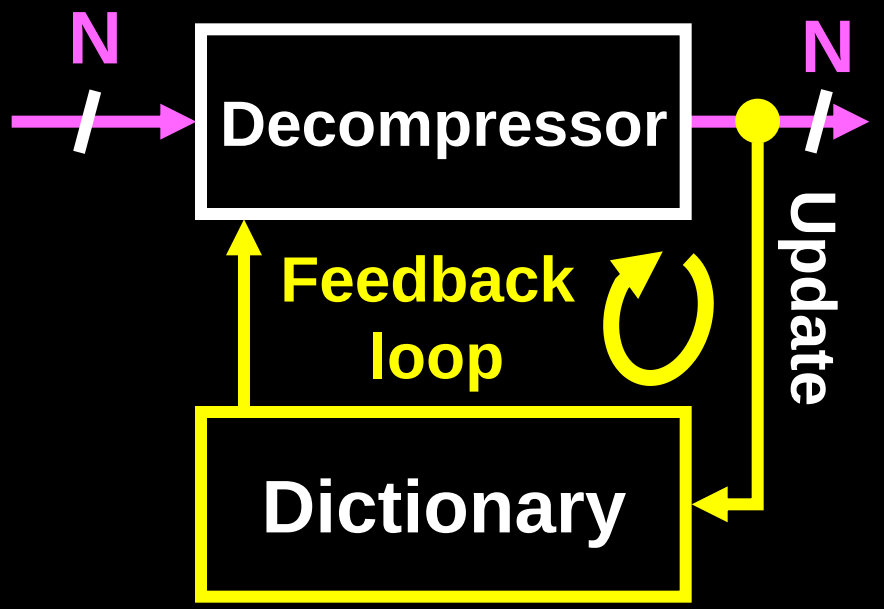
**Decompression bandwidth: BAD ☹**

Adaptively match various symbols

Slow due to symbol dependence




LZ



# Rule-Based Approach

**Compression ratio: BAD ☹**

**Decompression bandwidth: GOOD ☺**

Dependent on application domains

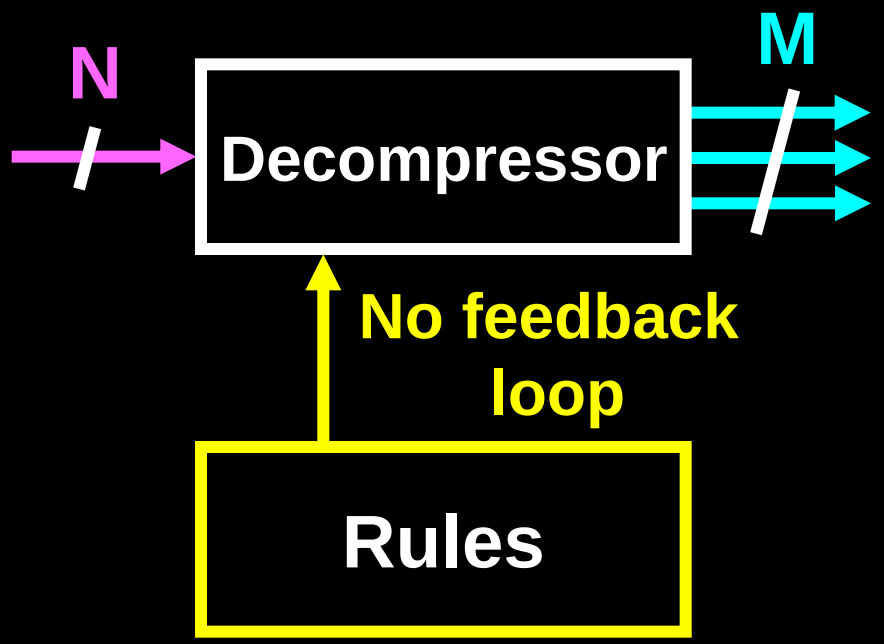
Fast because of symbol independence

Input → Output

0x0000 —2b→ 0b00

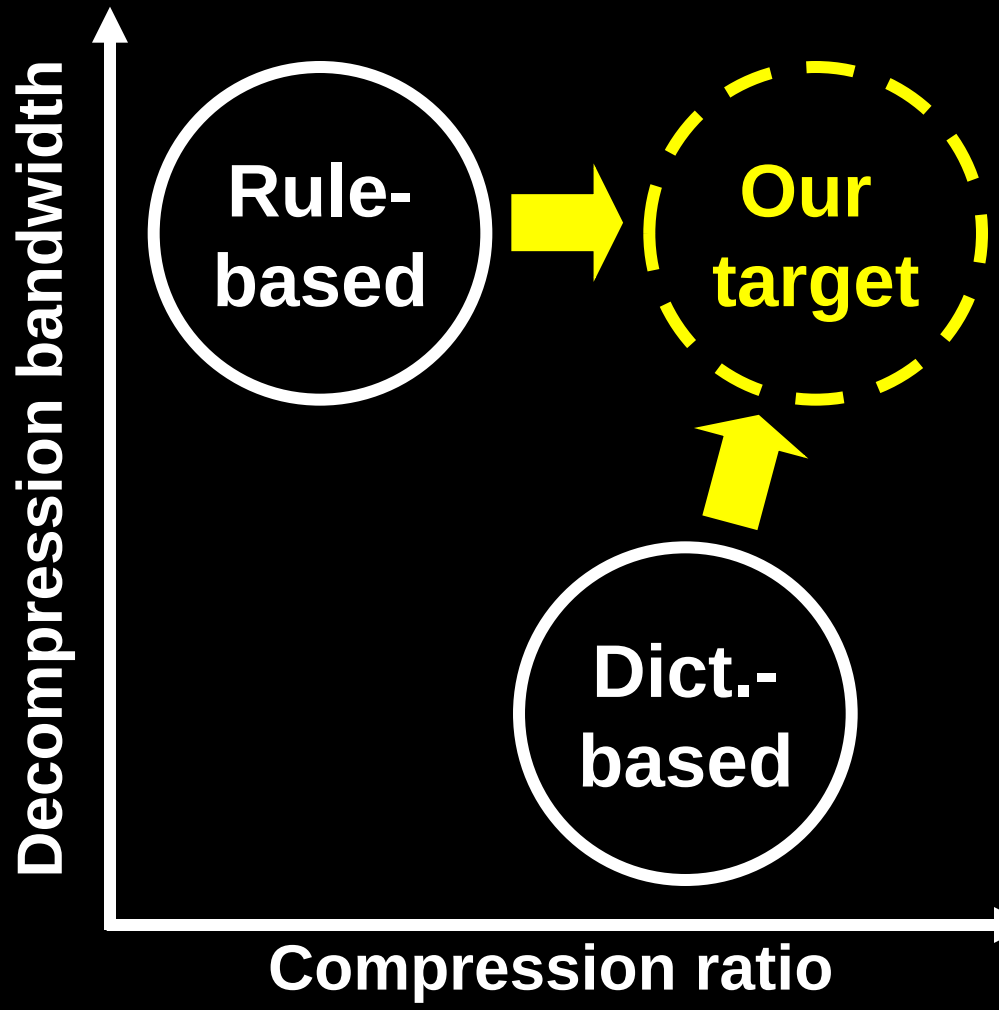
0xFF80 —10b→ 0b10+0x80

| 16bit rule | Prefix | Data |
|---|---|---|
| All zero | 0b00 | N/A |
| Sign ext. | 0b01 | 4bit |
| Sign ext. | 0b10 | 8bit |
| Default | 0b11 | 16bit |



# Summary of Background

## A new compression approach is required

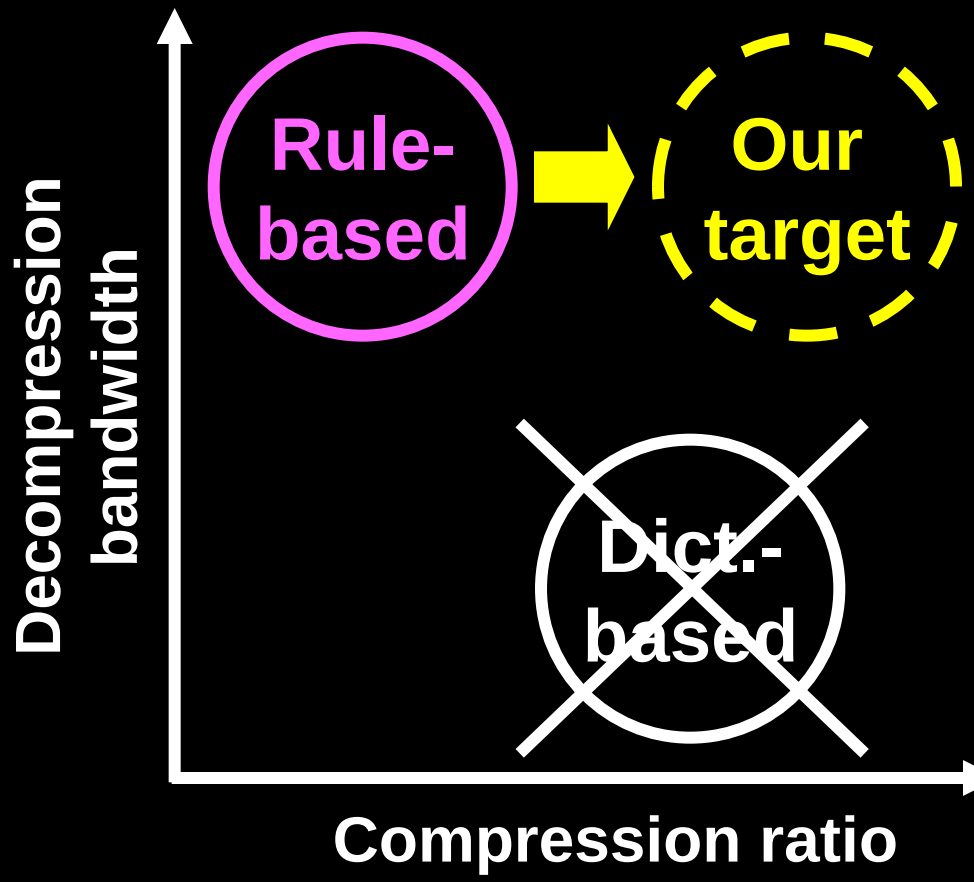


# Outline

- ✓ Background
- ➡ Our solution: DPC
- Evaluation
- Conclusion

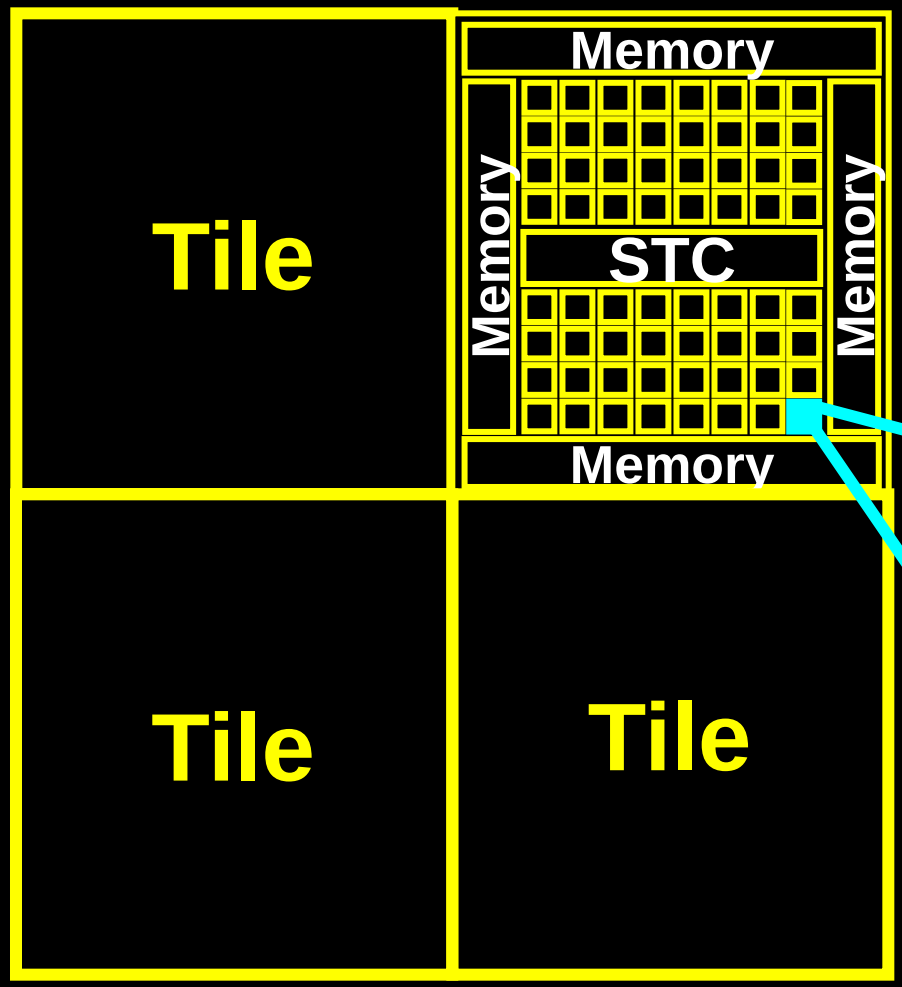
# Our Approach

**Improving compression ratios, exploiting the features of test configs.**



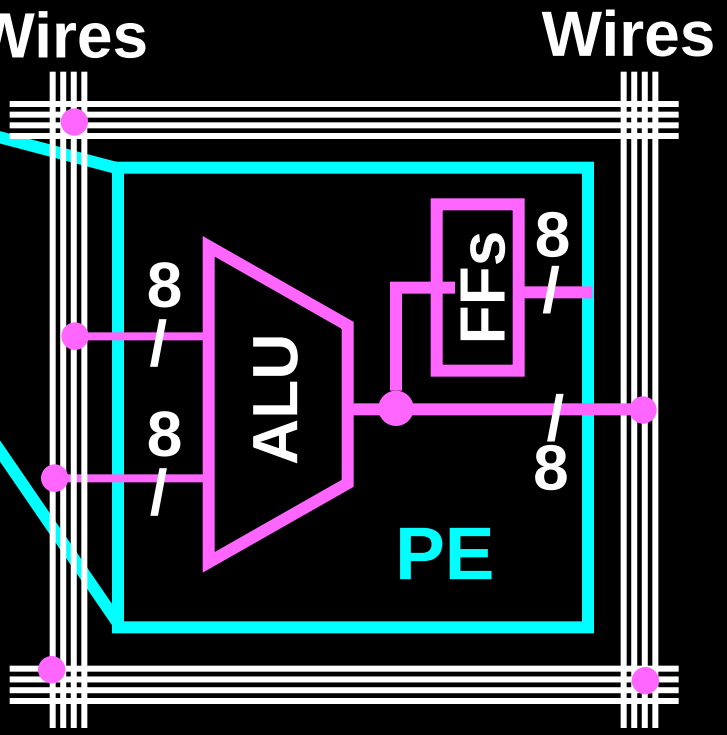
# Target Architecture

## Tile-based architecture with 8-bit PEs




**Per tile**

| | |
|---|---|
| # of PEs | 64 |
| # of states | 128 |



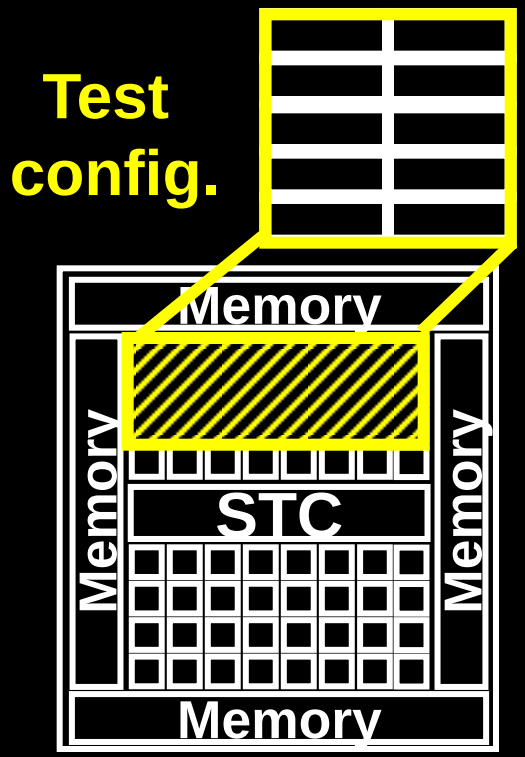

STC=State Transition Controller

# Test Configurations

**DRP: 64-bit, byte-addressing format**

| PE positions | PE instructions |
|---|---|
| Address (32bit) | Data (32bit) |

(c.f. FPGA: a bit-stream format)




**Address feature:**

- High spatial locality
  (=Tested PEs are tightly-packed)

**Data feature:**

- Fine-grained pattern locality
  (=Tested PEs use similar instructions)

# Double Pattern Compression

| Address1 | Data1 |
|---|---|
| Address2 | Data2 |

Data split → 16b | 16b

Address differentiation

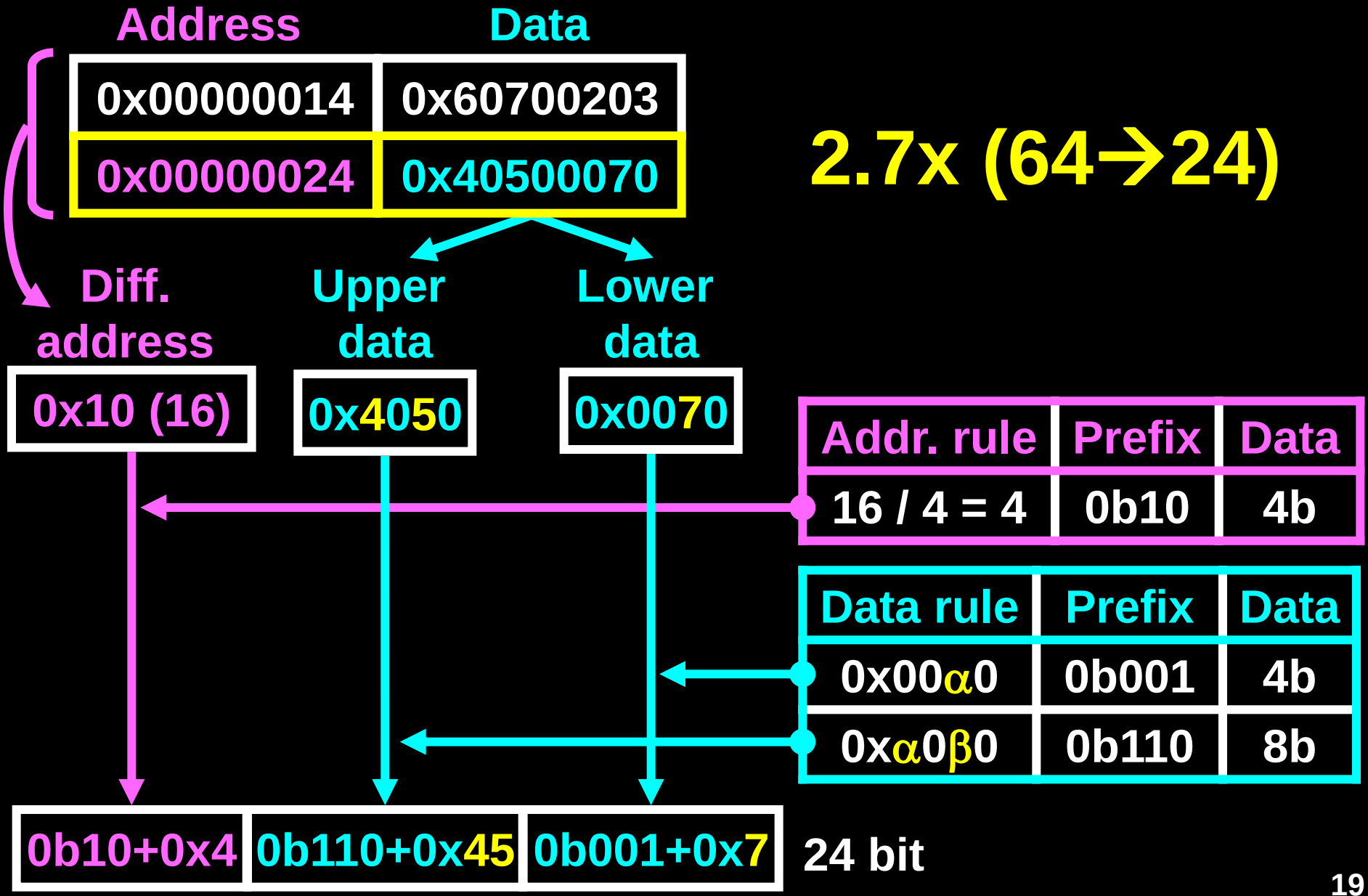
Diff. addr. (= Address2 – Address1)

| Addr. rule | Prefix | Data |
|---|---|---|
| Address divided by 4 | 0b00 | 1b |
| | 0b01 | 2b |
| | 0b10 | 4b |
| | 0b11 | 30b |

| Data rule | Prefix | Data |
|---|---|---|
| 0x0000 | 0b000 | N/A |
| 0x00α0 | 0b001 | 4b |
| 0x000α | 0b010 | 4b |
| 0x00αβ | 0b011 | 8b |
| 0xαβαβ | 0b100 | 8b |
| 0xαβ00 | 0b101 | 8b |
| 0xα0β0 | 0b110 | 8b |
| Default | 0b111 | 16b |

α, β = any hex. values

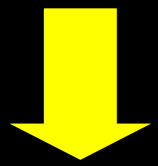
# Example


Address
Data
0x00000014
0x60700203
0x00000024
0x40500070
2.7x (64→24)
Diff. address
Upper data
Lower data
0x10 (16)
0x4050
0x0070
Addr. rule
Prefix
Data
16 / 4 = 4
0b10
4b
Data rule
Prefix
Data
0x00α0
0b001
4b
0xα0β0
0b110
8b
0b10+0x4
0b110+0x45
0b001+0x7
24 bit

# Question

## How can we obtain the rules?

- Patterns
- The number of rules
- Split data width

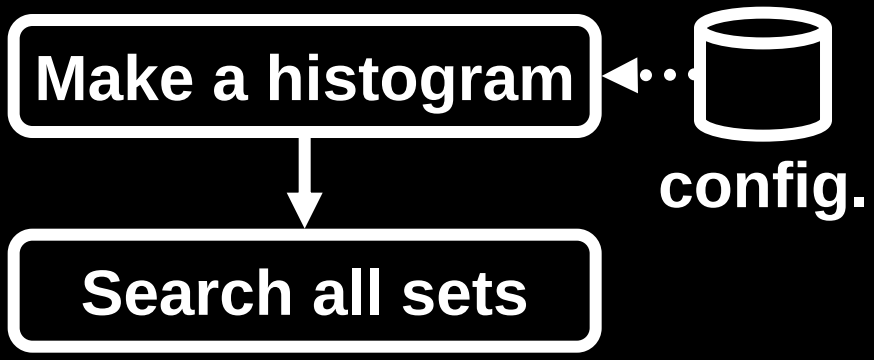


## Rule automation

# Rule Automation - Algorithms

## Address rule automation

A few hours



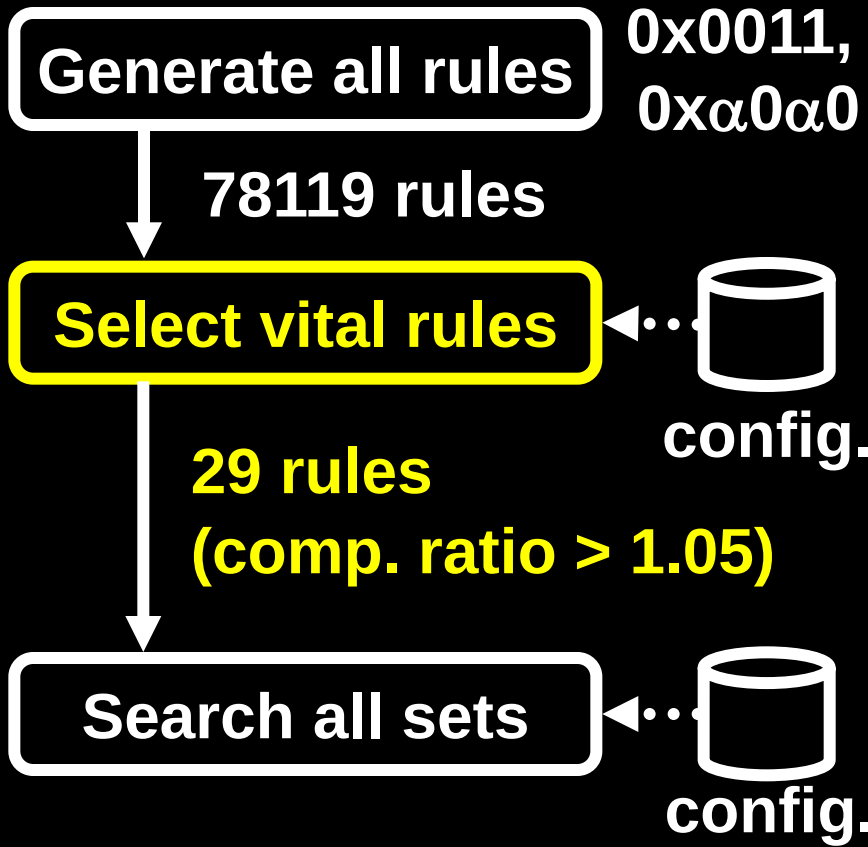

| Diff. addr. | Count |
|---|---|
| 1 bit | 80900 |
| 2 bit | 20050 |
| ... | ... |
| 32 bit | 1021 |

Select

## Data rule automation

One day

Ex.: split data width=16b



# Summary of DPC

## Fully-automated, rule-based approach

**Requirements:**

- High compression ratio
- High decompression bandwidth

Exploit features

Rule-based

**Our approach:**

Double pattern compression

Improve comp. ratio

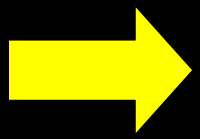
**Technique:**

Address/data rule automation

# Outline

- Background

- Our solution: DPC

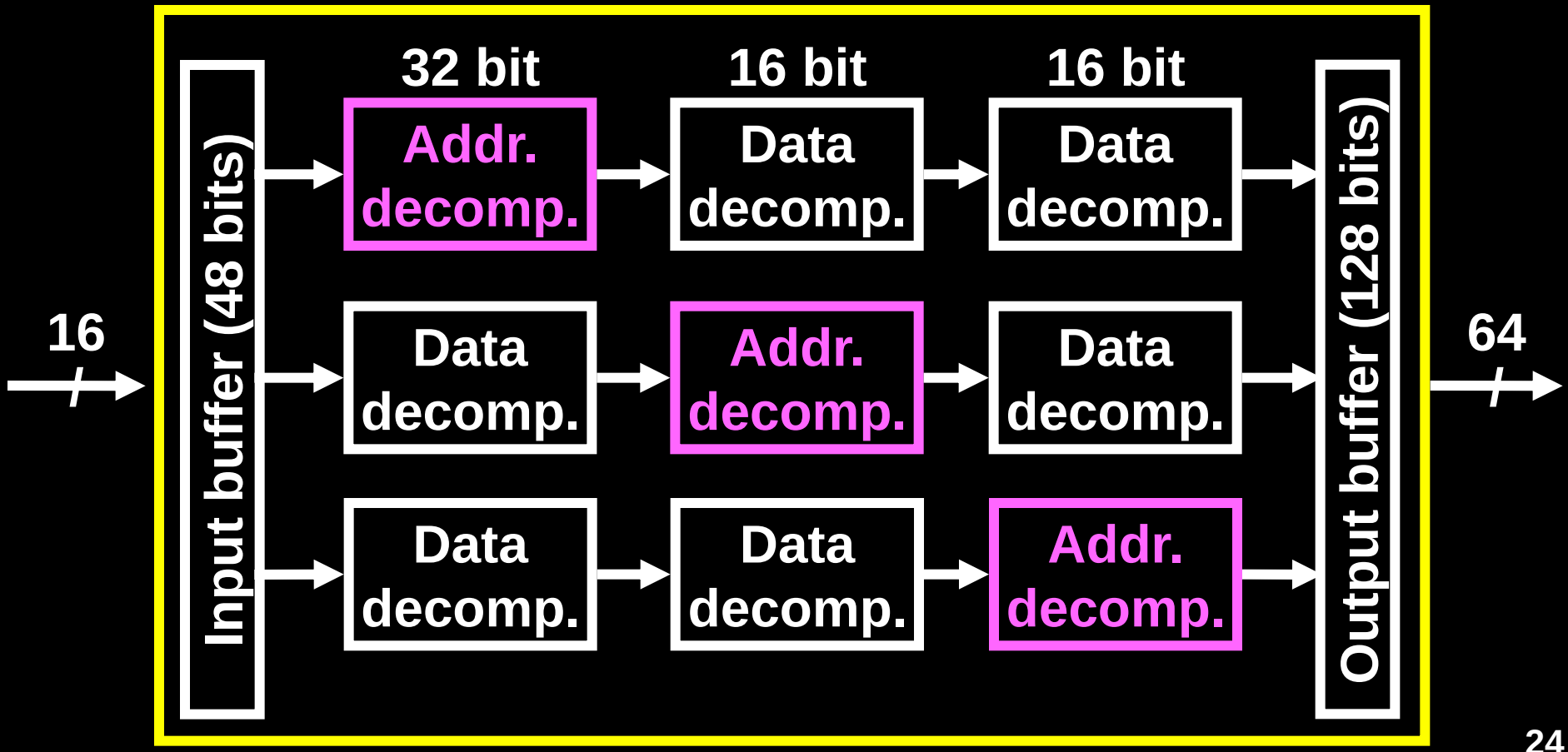


- **Evaluation**

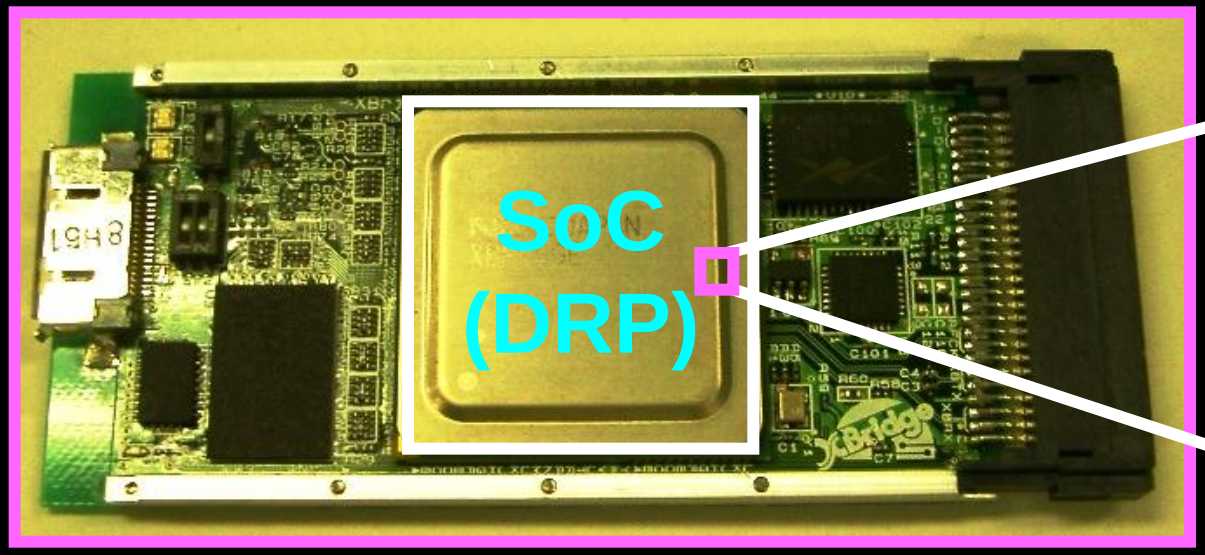
- Conclusion

# Decompressor Circuit Design

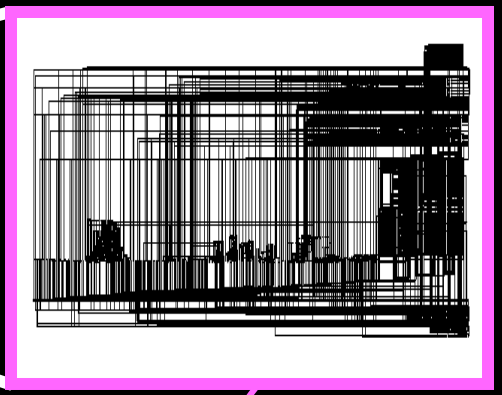
**Three sets of sub-decompressors, decompressing symbols in any order**
**(speed-optimized design)**



# Circuit Specifications




Schematic



| Process | 40nm |
|---|---|
| Max clock frequency | 200MHz |
| Decomp. bandwidth | 1.6GB/s |
| Power dissipation | 1.8mW (active) / 19.4 μW (idle) |
| Area | 0.046 $mm^2$ |

# Evaluation Items

Focus

- Test compression ratio
- Test time reduction

- Application compression

(See our paper)

# Evaluation Conditions

**Test configurations:**

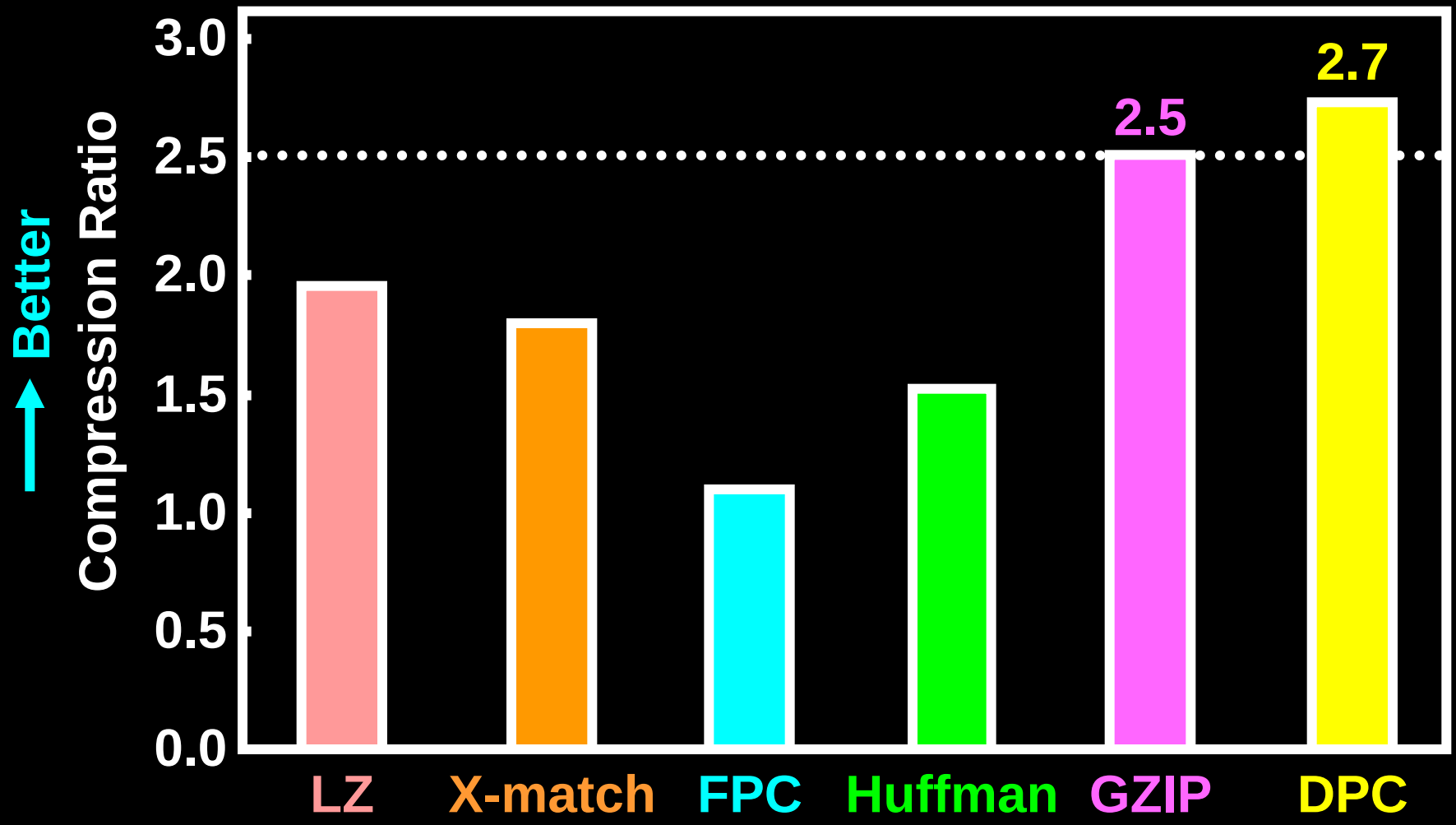
- **Size:** 100s of mega-bits
- **Fault model:** stuck-at, transition, etc.
- **Fault coverage:** product-level

**Five compared techniques:**

- LZ (dictionary-based)
- X-match (dictionary-based; HW-friendly)
- FPC (rule-based for CPU memory)
- Huffman (coding)
- GZIP (LZ and Huffman)

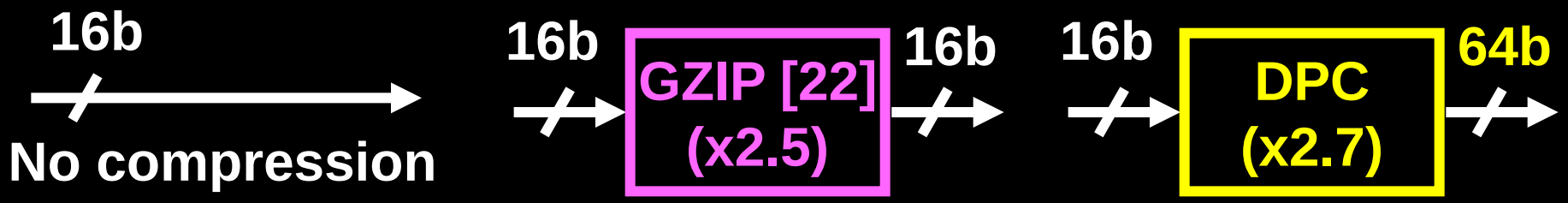
# Test Compression Ratio

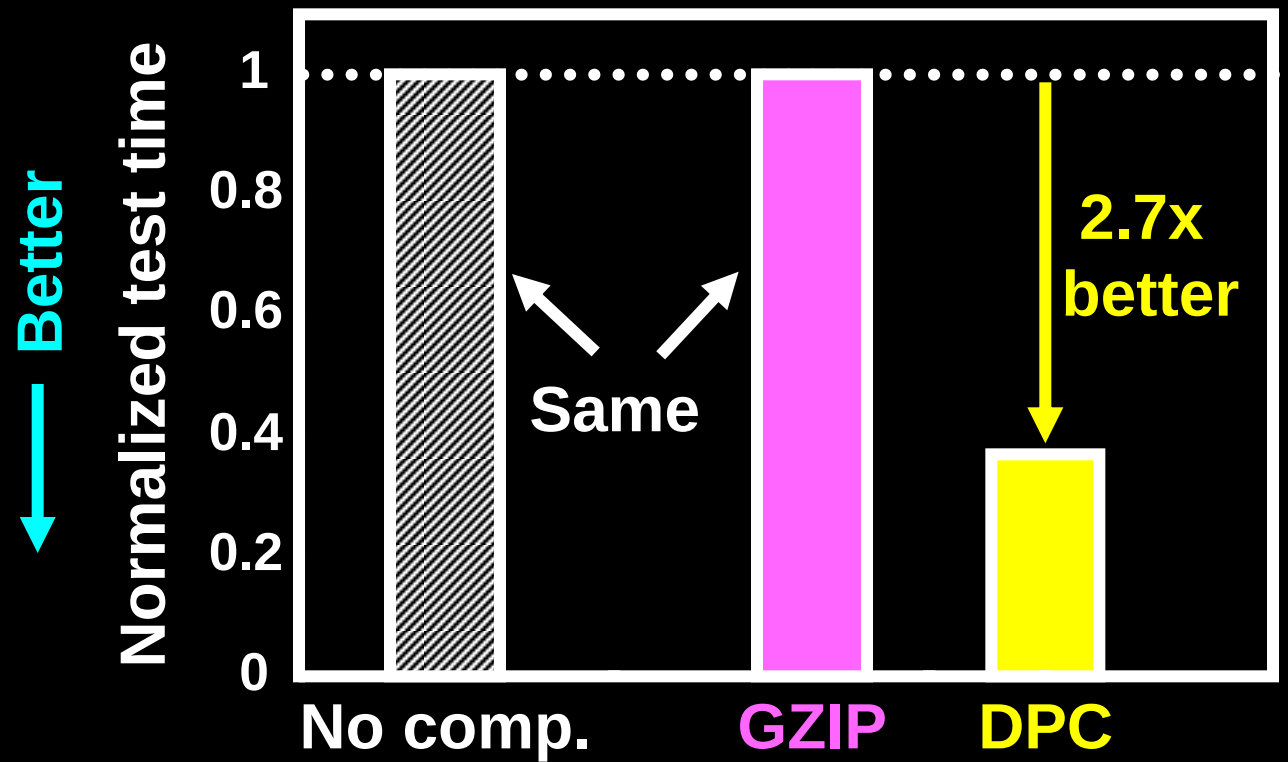
## 2.7x better for original test configurations (slightly better than GZIP)



# Test Time Reduction

## DPC achieves 2.7x smaller test time



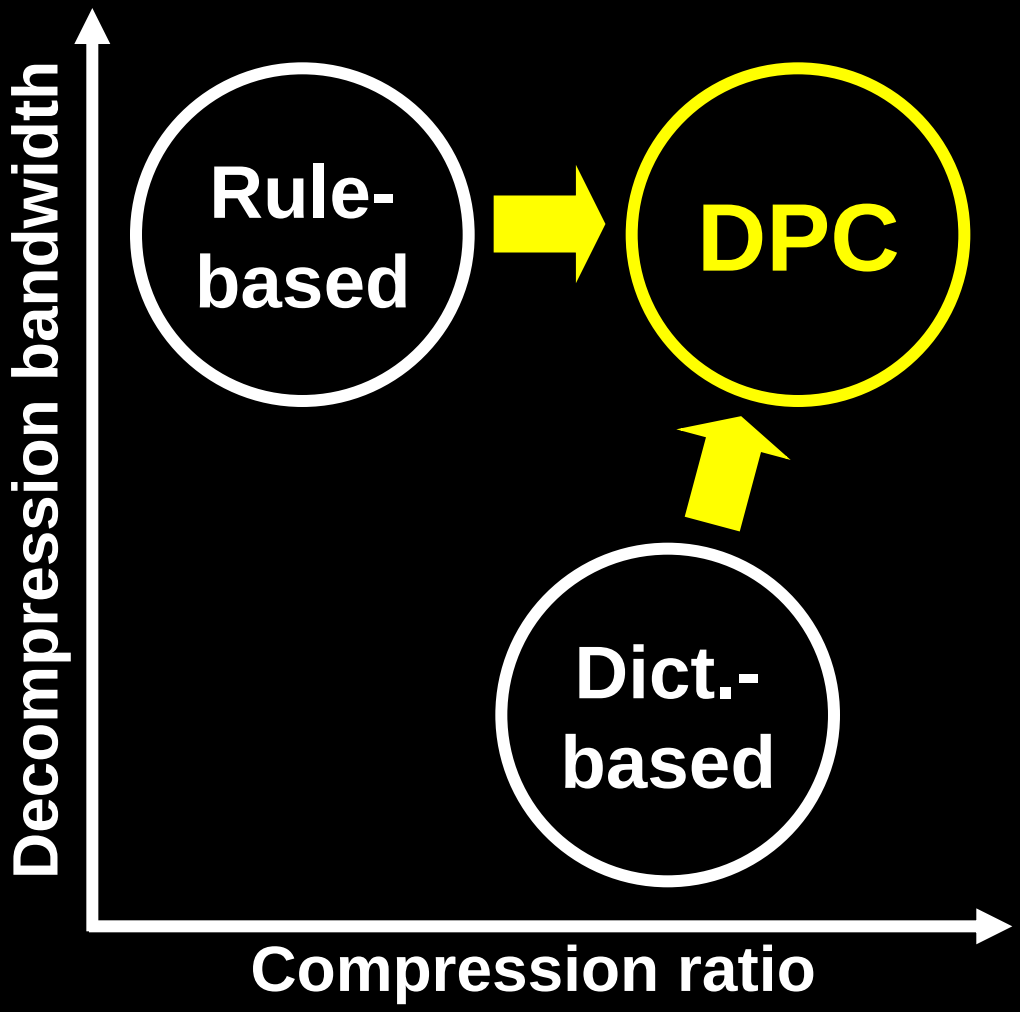
# Outline

- ✓ Background
- ✓ Our solution: DPC
- ✓ Evaluation
- ➡ Conclusion

# Conclusion

## The world's first test compression for DRPs

- High comp. ratio: better than GZIP
- High decompression bandwidth: 16b to 64b
- 2.7x shorter test times



# Frequently Asked Questions

- **Do you need to modify the compression rules every time test configurations are updated?**

  → **Practically, never after the early-design stage.**

- **Why can the compression rules be used for application configurations?**

  → **Rule automation extracts architecture features**

- **Is this approach also effective for other DRP architectures?**

  → **I guess so. Any collaboration is welcome.**