ICCAD 2004

Configuration Bitstream Compression for Dynamically Reconfigurable FPGAs

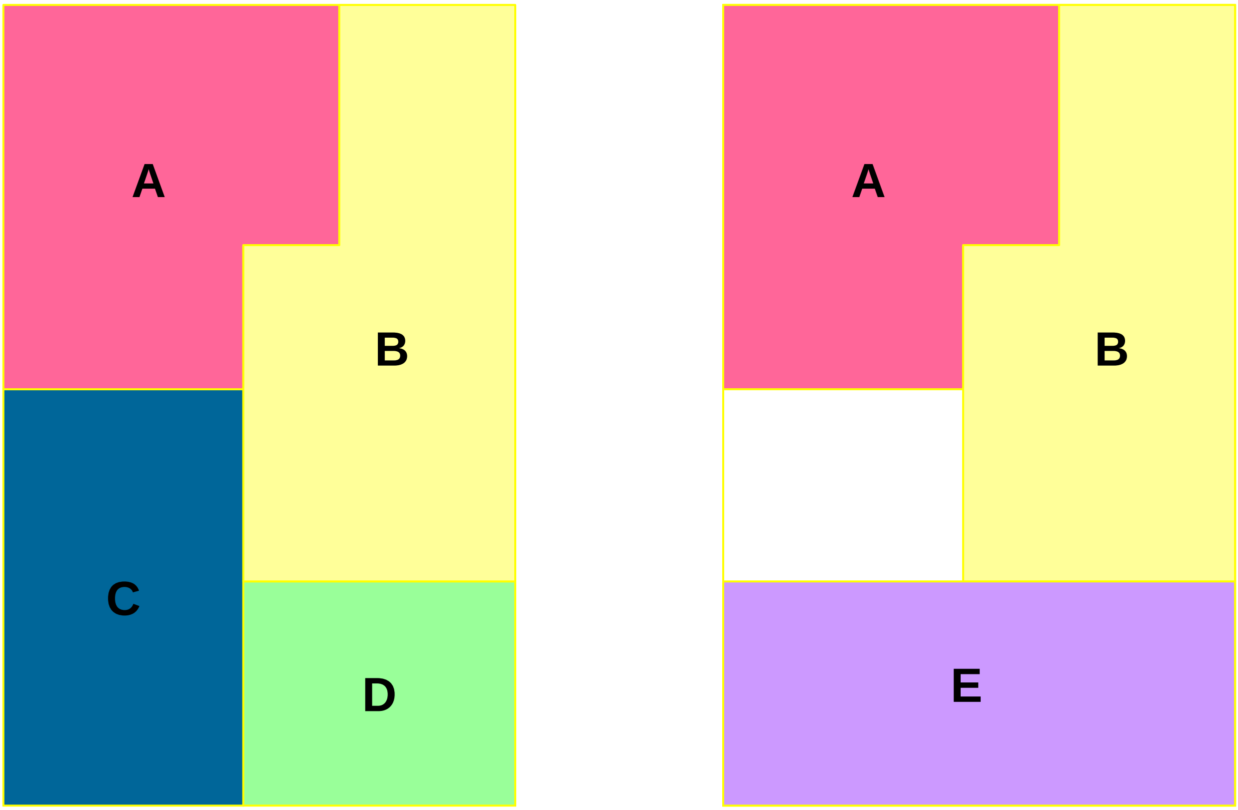
Ju Hwa Pan * Tulika Mitra * Weng Fai Wong

NUS
National University of Singapore

Dynamically Reconfigurable FPGA

- Field Programmable Gate Array (FPGA)
  - Hardware realization of application-specific functionalities
- Dynamically reconfigurable FPGA
  - Reconfiguration during execution of the application
  - Sharing of resource among multiple computational kernels

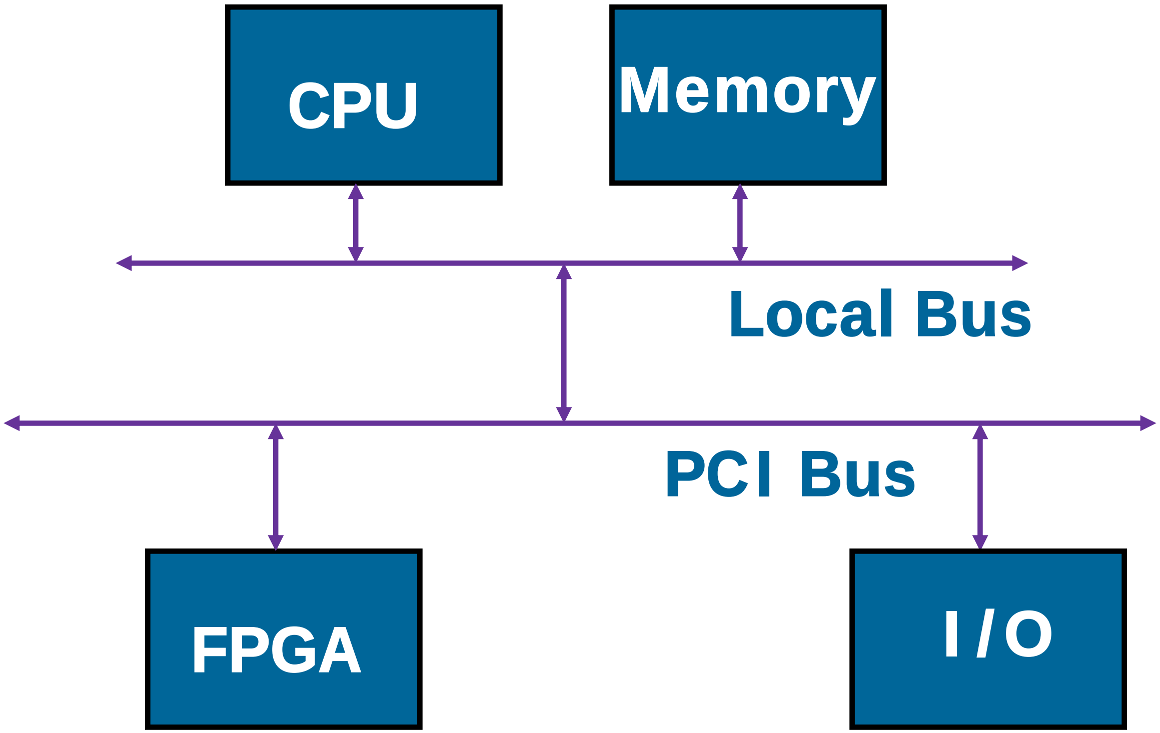
Dynamically Reconfigurable FPGA (contd.)



Reconfiguration Overhead

- Multi-million gate FPGAs require considerable reconfiguration time
  - 23.6 ms for RC1000-PP with XCV1000 device
- Reconfiguration time
  - Transfer the configuration bitstream to FPGA
  - Write the bitstream into FPGA
- Transfer time dominates for I/O system bus based FPGA cards
  - Most common form of commercial FPGA

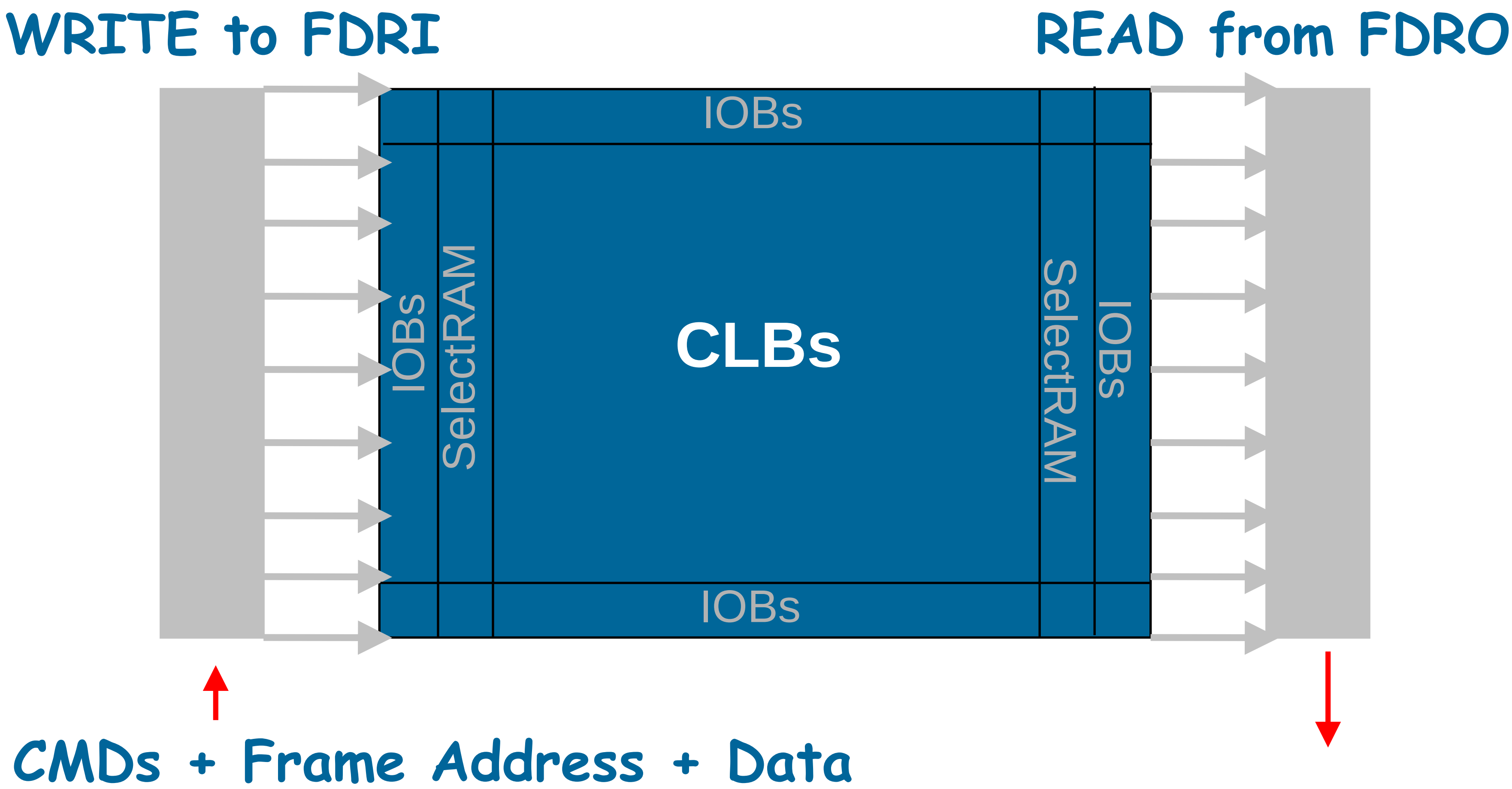
FPGA Coupled to I/O System Bus



Solution: Partial Reconfiguration ?

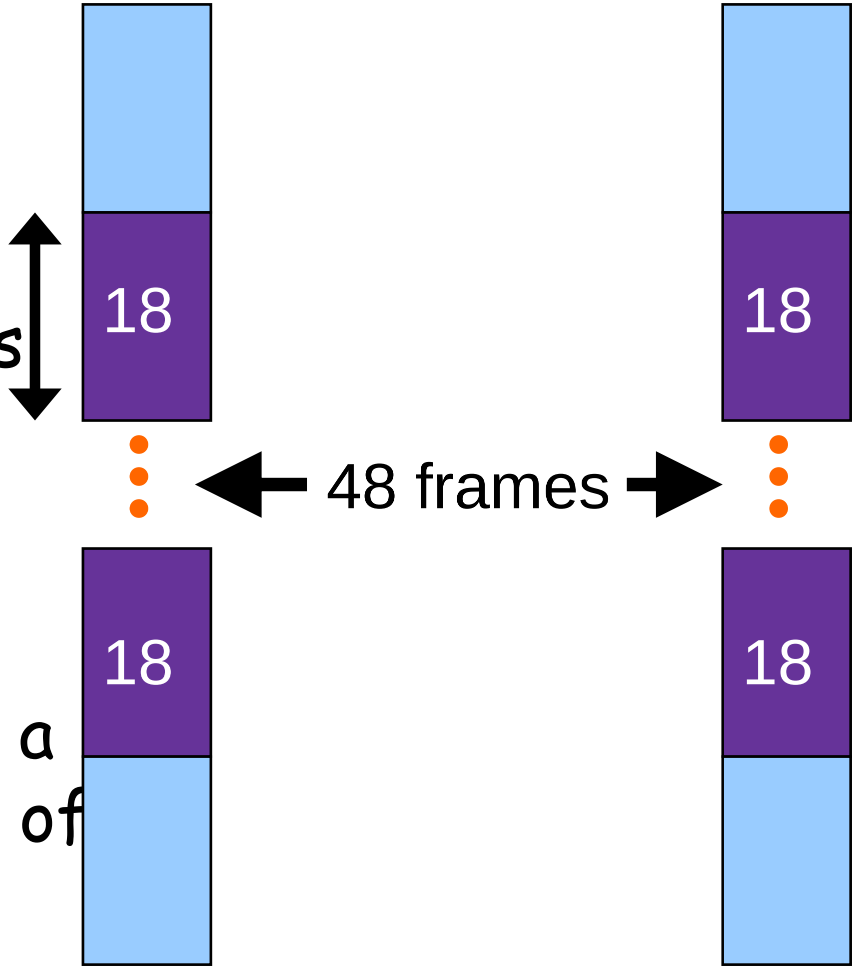
- Observation: Different bitstreams (configurations) typically share some static kernels
- For two consecutive bitstreams, transfer and configure only the portions that differ in the new bitstream
- Reconfiguration time proportional to the difference between two bitstreams
- However, partial reconfiguration requires appropriate placement of static kernels

Xilinx Virtex FPGA Overview



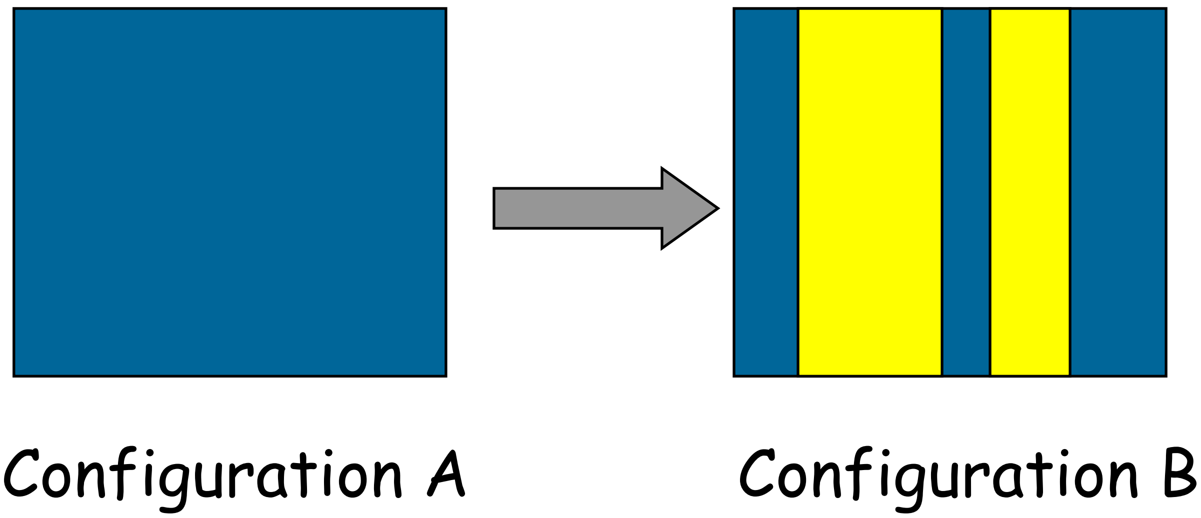
Virtex FPGA Configuration Memory

- Basic unit – 1 frame
  - 1 bit wide
  - Spans vertically across SRAM array
- Each CLB column
  - 48 frames wide
  - Each 18-bit section of a frame configures part of a CLB row



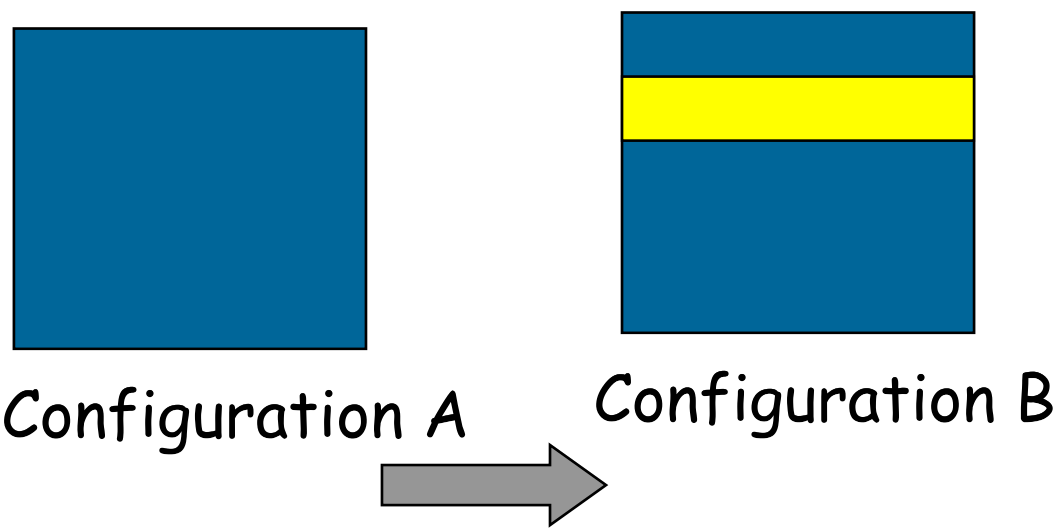
Partial Reconfiguration

- Only load modified frames in new configuration



Partial Reconfiguration: Limitations

- Restriction in on-chip placement of computational kernels
- Possible decrease in computational efficiency due to restricted placement



No partial reconfiguration

Alternative Solution: Compression

- Intra-bitstream compression
  - Exploit redundancies across frames in a bitstream
  - Exploit special features of FPGA
    - Huack'99, Park'99
  - Dictionary-based compression algorithms such as LZSS, LZW, and LZ77
    - Dandalis'01, Huack'01, Khu'01
- Cannot exploit static kernels in dynamically reconfigurable FPGA
- Inter-bitstream compression

Bitstream Compression Framework

Intra-Bitstream

Redundancy between frames in new bitstream

Inter-Bitstream

Partial Reconfiguration

Redundancy between frames in old and new bitstream

Redundancy between frames in new bitstream

Roadmap

Establish compression scheme between pair of frames

Roadmap

Establish compression scheme between pair of frames

Discover "similar" pair of frames in bitstream

Roadmap

Establish compression scheme between pair of frames

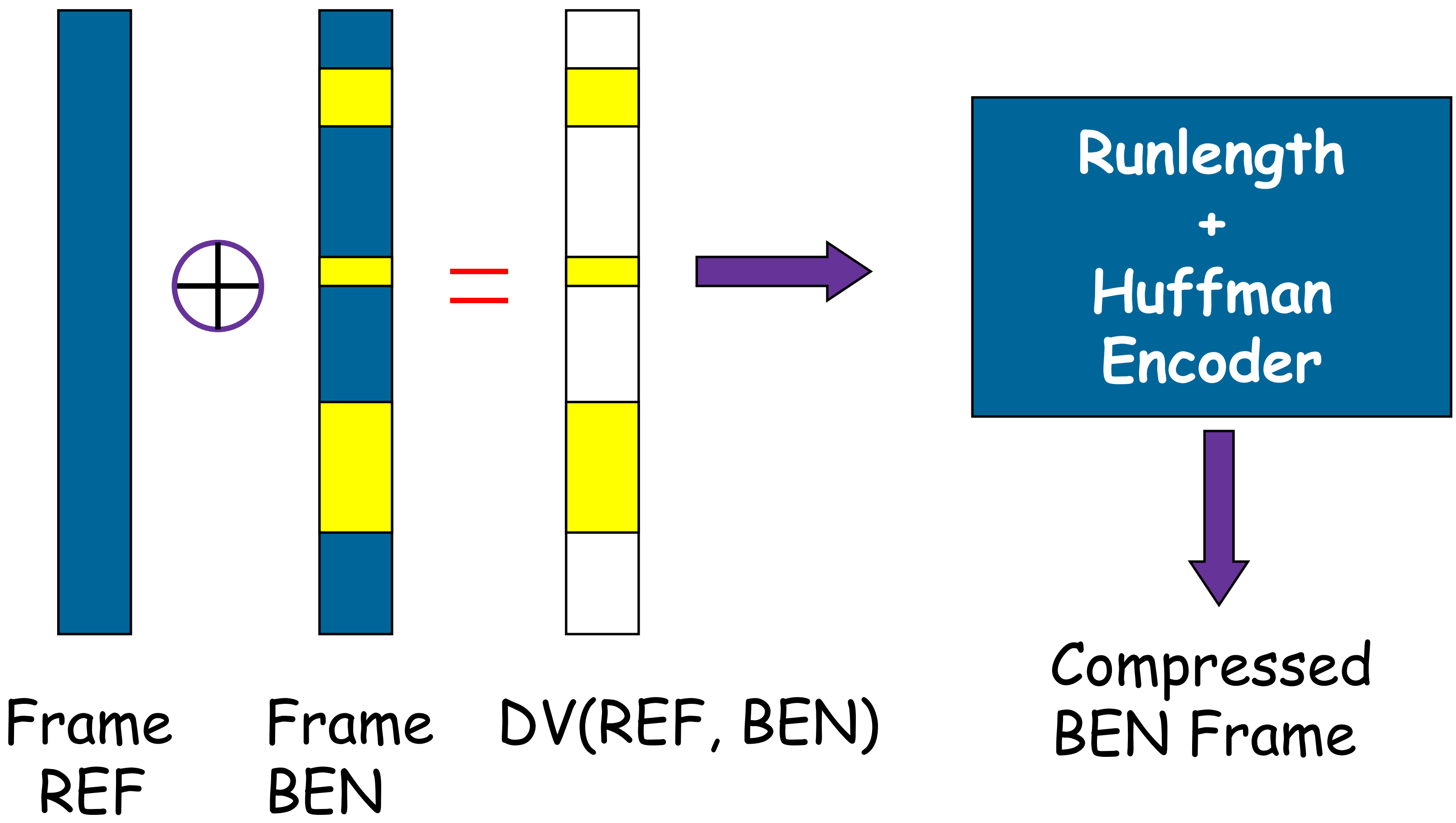
Discover "similar" pair of frames in bitstream

Discover "similar" pair of frames across bitstreams

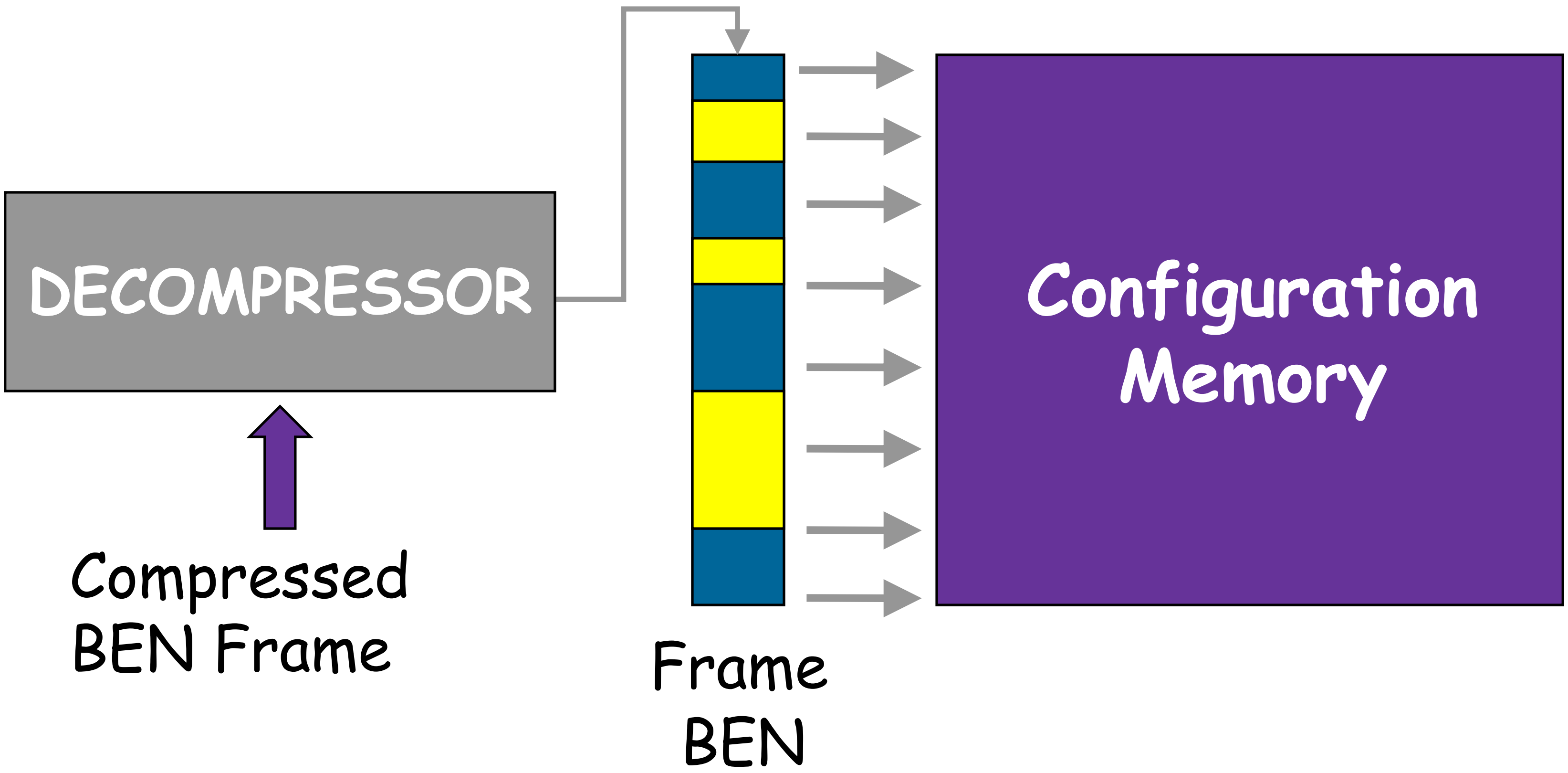
Compression Scheme between Frames

- Best known existing scheme: LZSS [Hauck'01]
  - Relies upon high frame regularity with data matches longer than threshold value
  - Fine grained regularity cannot be captured

Difference Vector (DV) Compression



Difference Vector Decompression



Difference Vector Compression: Intuition

- Repetitive structures across CLBs
  - Gives rise to regular bitstreams
- Frame differences are either
  - Few and scattered
  - Clustered in groups of 18-bit bands

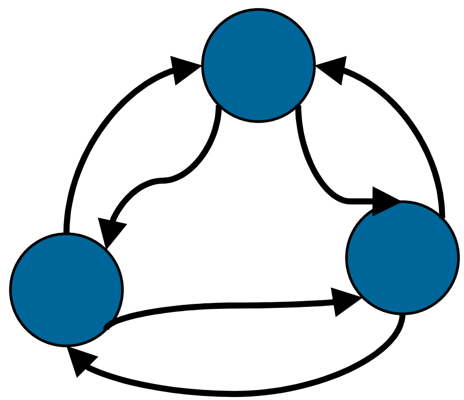
Determining Frame Sequence

- Configuration sequence of frames must be such that
  - REF frame used to compress a BEN has already been configured
    - Best case: REF frame in FDRI, i.e., it is the last configured frame
  - Overall sequence achieves as close to optimal compression as possible

Frame Sequencing Schemes

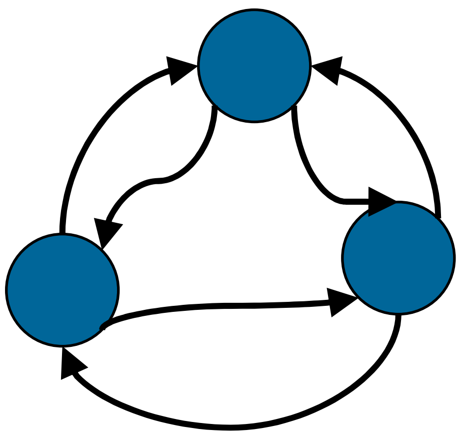
	Frame Readback	No Frame Readback
Intra-Bitstream	Intra-CWR	Intra-CWOR
Inter-Bitstream	Inter-CWR	Inter-CWOR

Inter-frame Regularity Graph (IRG)



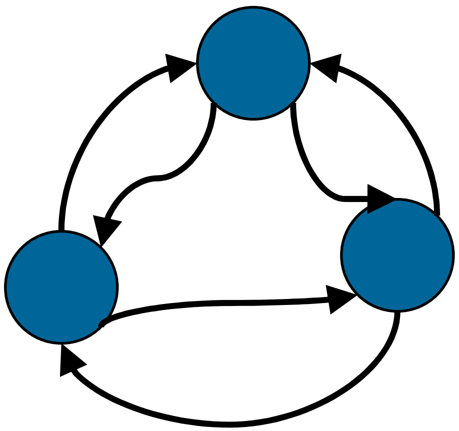
- Complete directed graph
- Each node represents a frame
- An edge u → v represents the size of the compressed encoding of v if u is REF frame
- For DV, we use the number of 0 → 1 and 1 → 0 transitions as the edge weight

Intra-CWOR



- Optimal configuration sequence without readback is equivalent to the shortest Hamiltonian path
- Traverse path using predecessor as the REF frame for successor

Intra-CWR

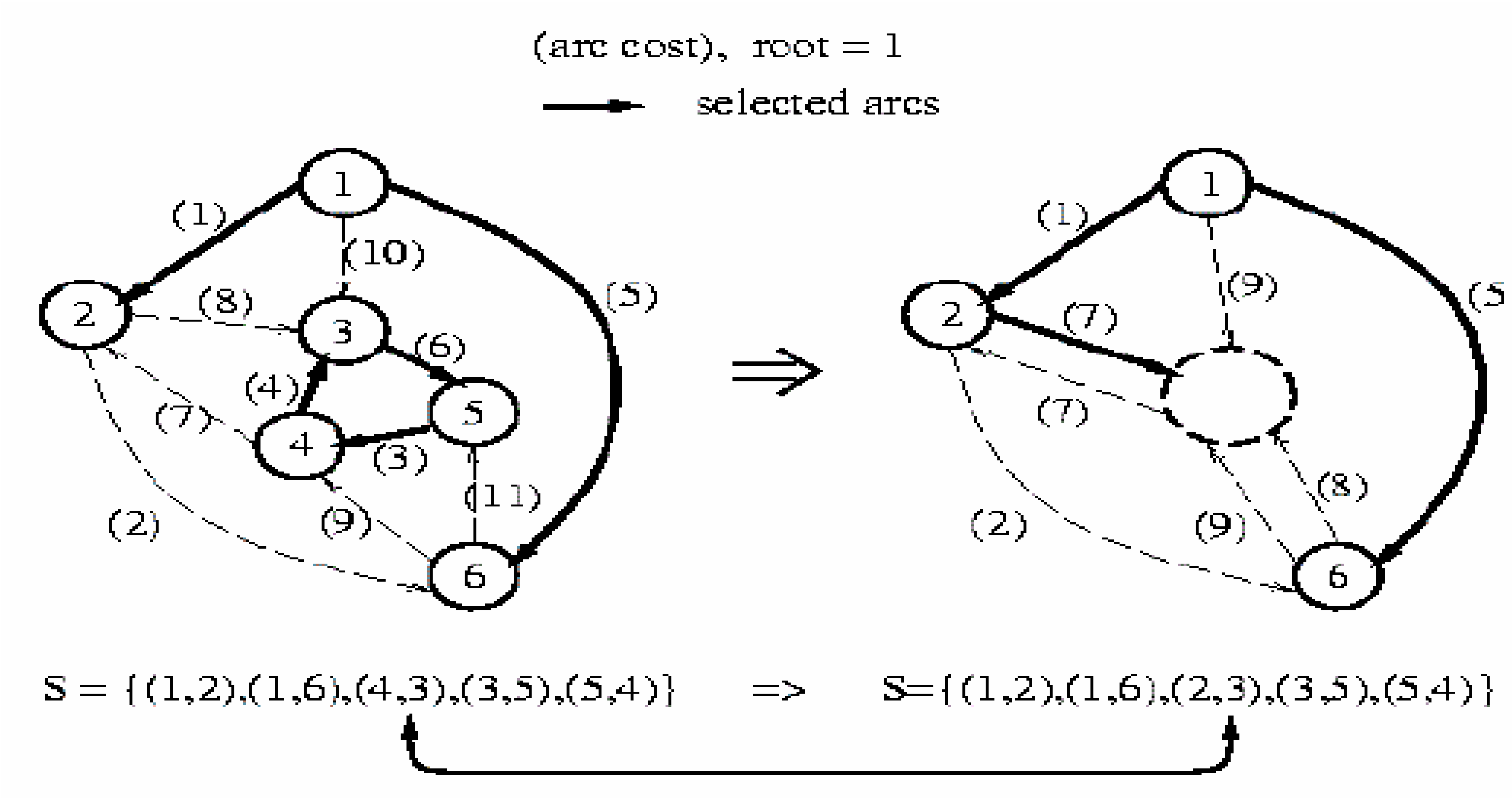


- Allow readback of configured frame if not in FDRI
- Corresponds to finding the directed minimum spanning tree (MST) in the IRG
- Generate configuration sequence through pre-order traversal of the MST

Directed MST Algorithm

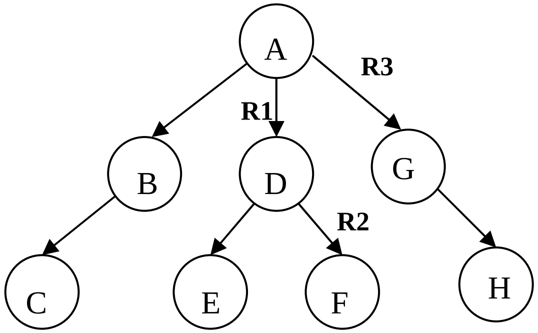
- Choose a node as the root
- For every other node, select the incoming edge with minimum weight
- We get the directed MST if the set of selected edges does not contain any cycle
- Intuitively, cycle is broken by replacing a cycle edge with an edge incident on the cycle such that there is minimum extra cost
- Continue till there are no cycles

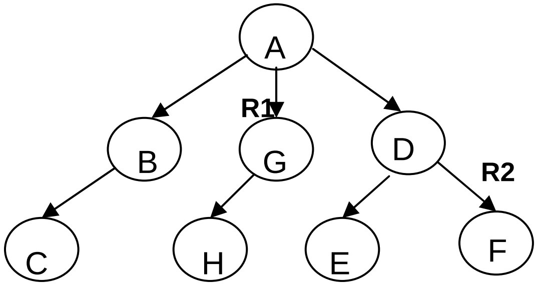
Example


(arc cost), root = 1
selected arcs
1
2
3
4
5
6
(1)
(10)
(8)
(5)
(6)
(4)
(7)
(3)
(11)
(2)
(9)
⇒
1
2
6
(1)
(9)
(7)
(5)
(7)
(8)
(2)
(9)
S = {(1,2),(1,6),(4,3),(3,5),(5,4)}
=>
S={(1,2),(1,6),(2,3),(3,5),(5,4)}

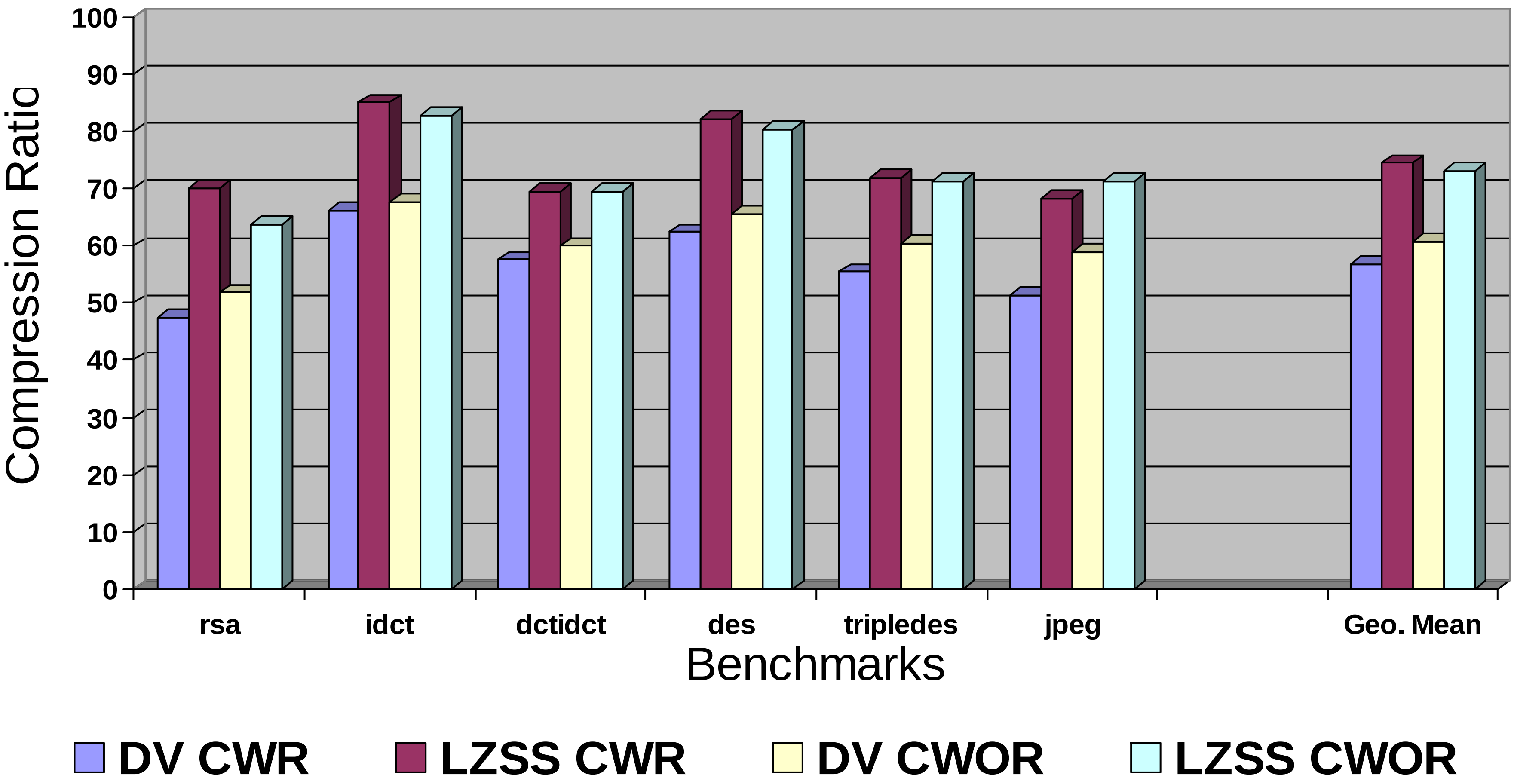
Optimization: Reusing FDRO register

- Once a frame is readback into FDRO register, it can be reused by other frames
- Transform the MST to exploit this

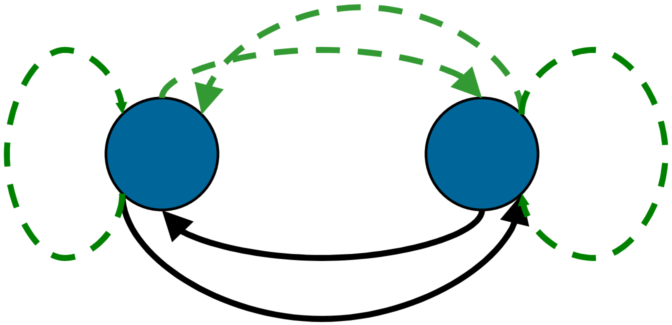




Results: Intra-Bitstream Compression



IRG for Inter-bitstream Compression



- Multi-digraph
- Each node u has a pair of directed edges Intra(u→v) and Inter(u→v)
- Self-loop Inter(u→u) for each node u
- Partial reconfiguration
  - Retained nodes
    - Nodes for which frames are identical in old and new bitstreams

Inter-CWOR

- Allow restricted readbacks of the form

  REF = u_{old} BEN = u_{new}
- Exploits static kernels
- Self-referenced node
  - If for a node u_{new}, the best choice for reference frame is u_{old}

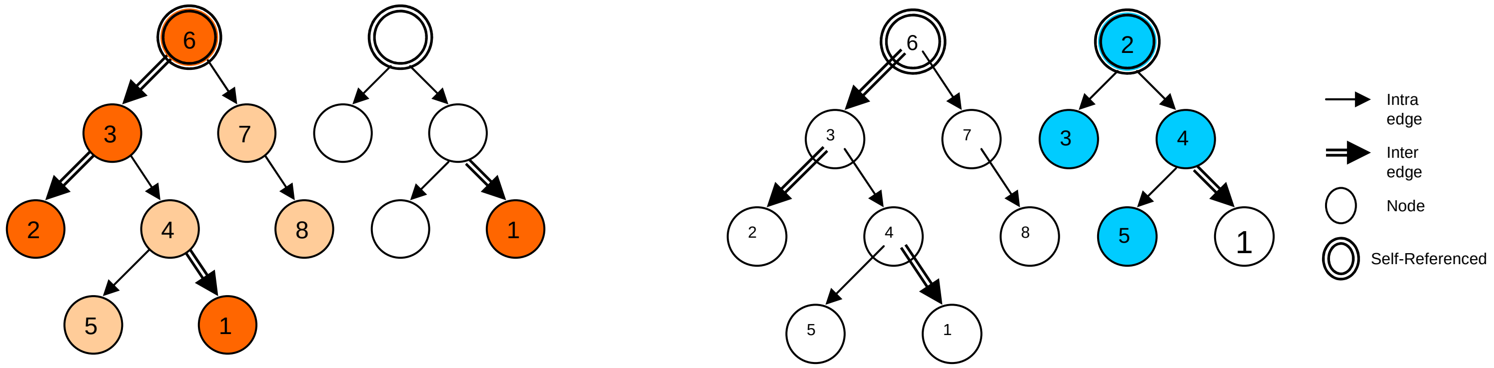
Configuration Sequence for Inter-CWOR

- Remove retained nodes from IRG
- Delete Inter(u→v) if u ≠ v
- Find all the self-referenced nodes
- Greedy heuristic finds disjoint paths where each path is headed by a self-referenced node
- Configuration requires readback only for self-referenced nodes

Inter-CWR

- Divided into 2 phases
- Phase 1
  - Modification of directed MST algorithm
  - Find a best matching reference frame for all non-retained frames
  - Result is a set of disjoint trees s.t. the roots are either self-referenced nodes or retained nodes
- Phase 2
  - Traverse the trees to ensure minimum number of readbacks and that reference frames are not overwritten

Tree Traversal



Results: Inter-Bitstream Compression

Benchmark 1

Observations

No static kernels

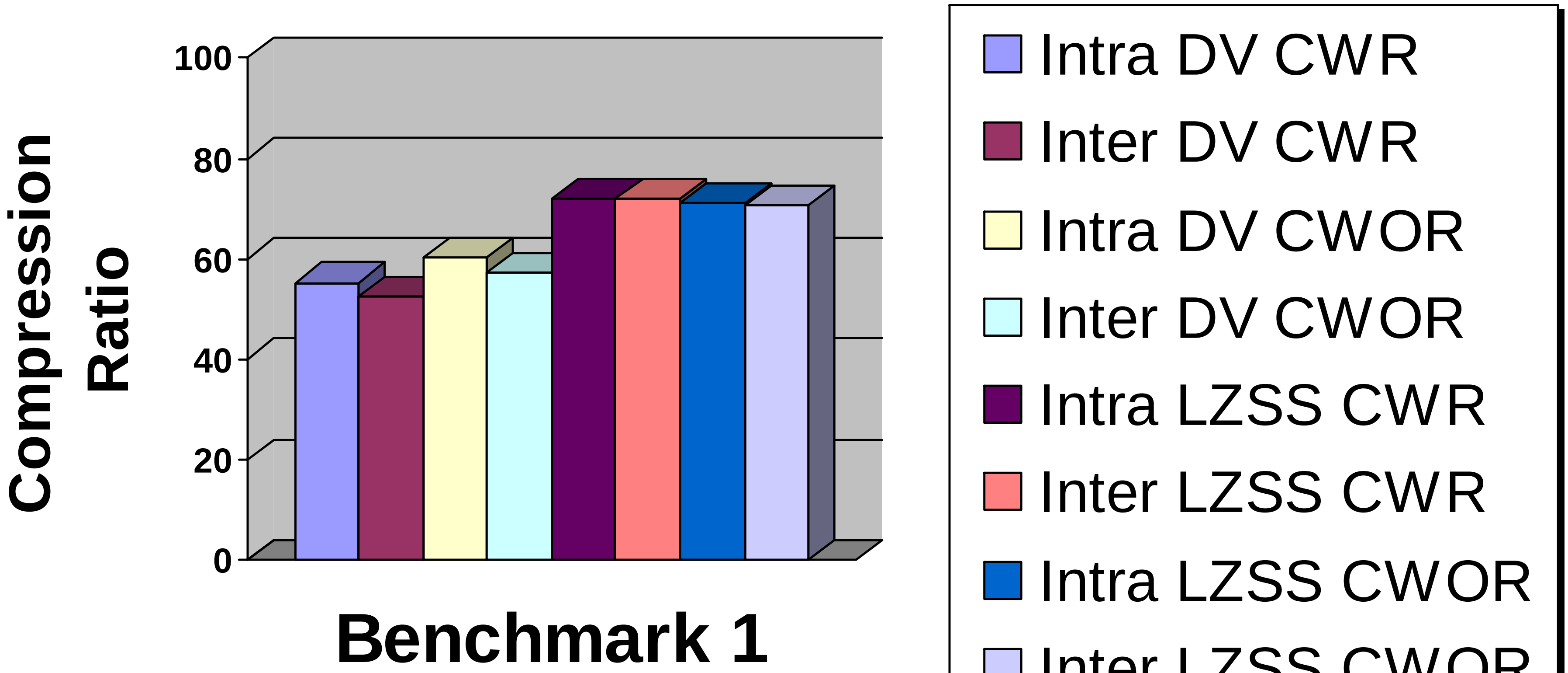
Inter-bitstream compression performs marginally better than Intra-bitstream

P - Preceding Bitstream

S - Succeeding Bitstream

■ - Dynamically swapped in kernel

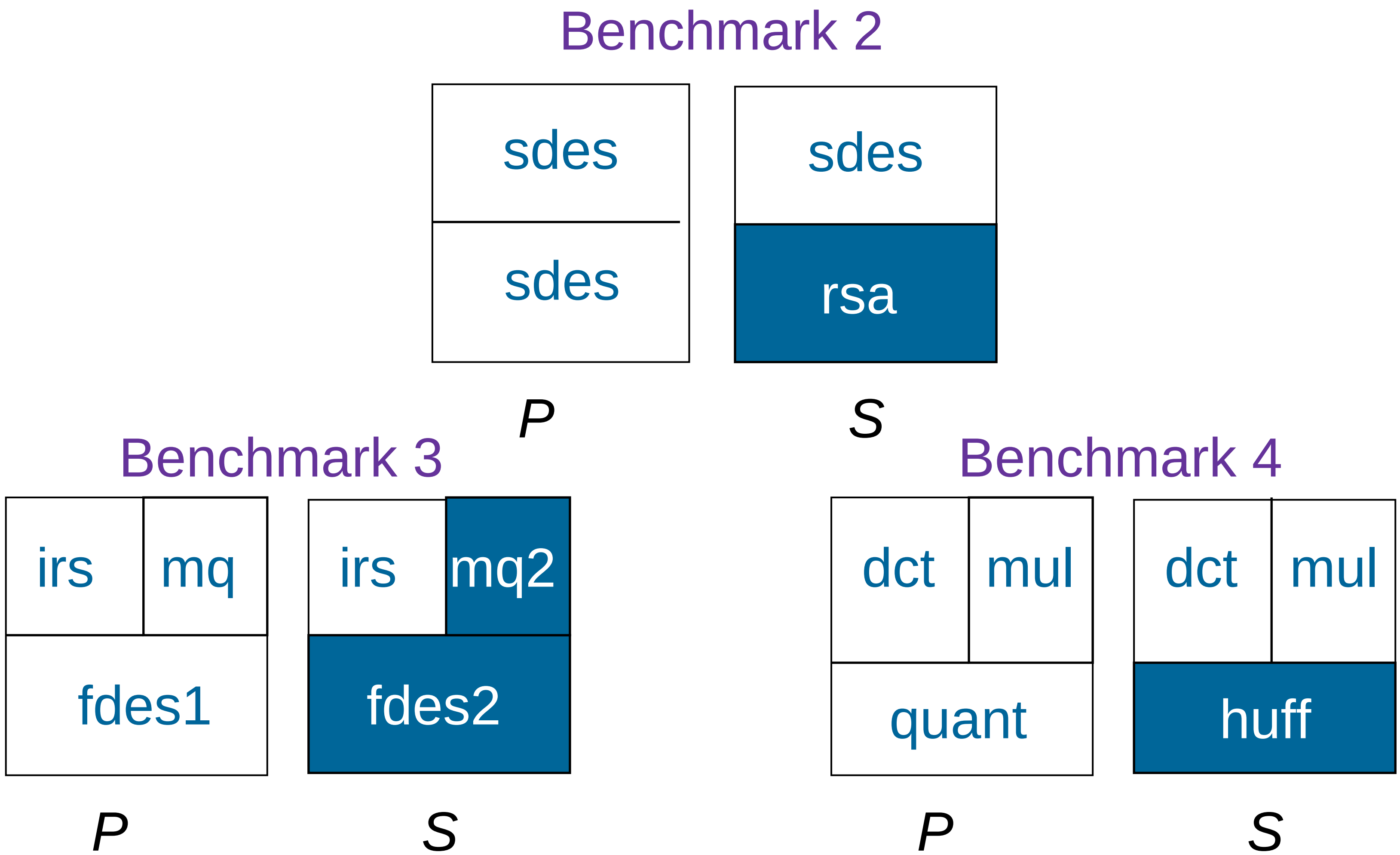
Results: Inter-Bitstream Compression



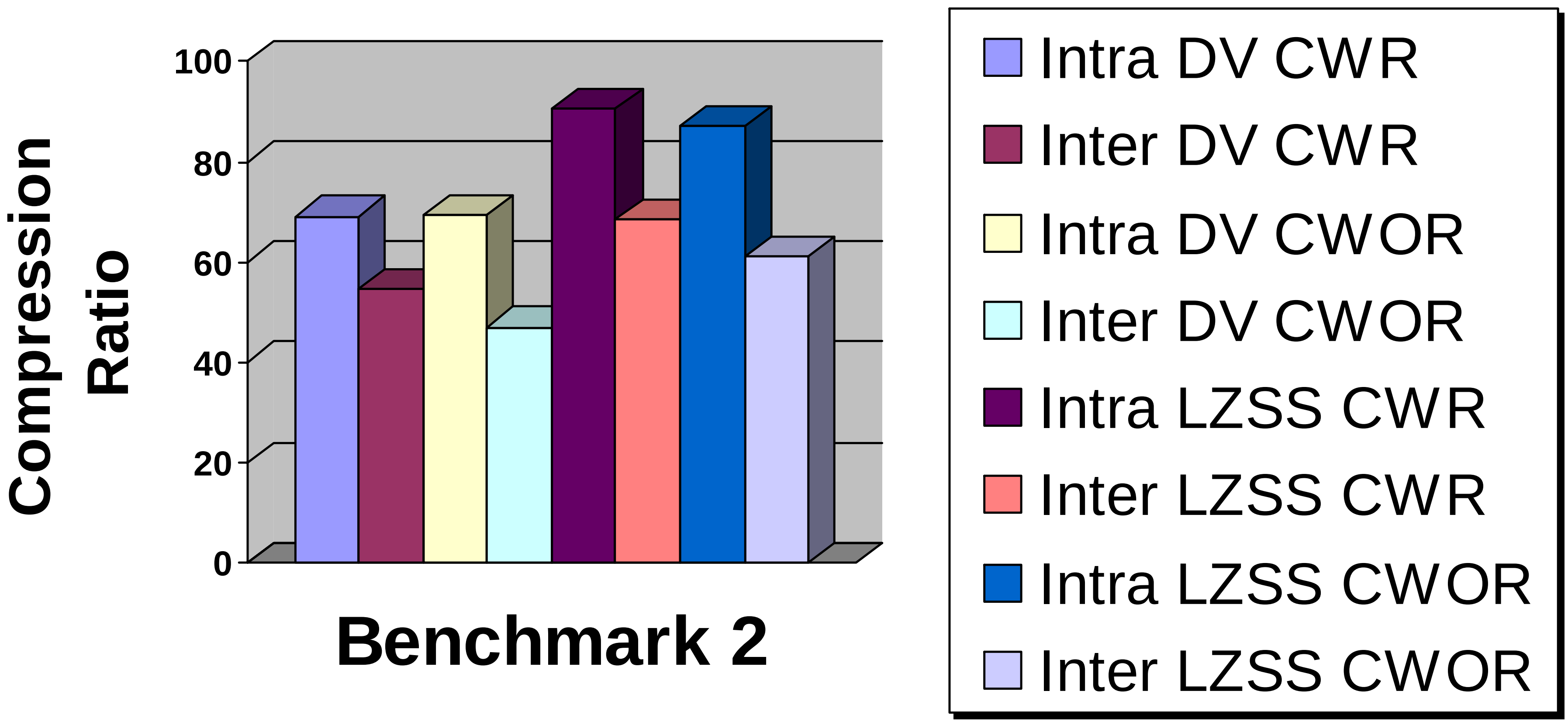
P

S

Results: Inter-Bitstream Compression



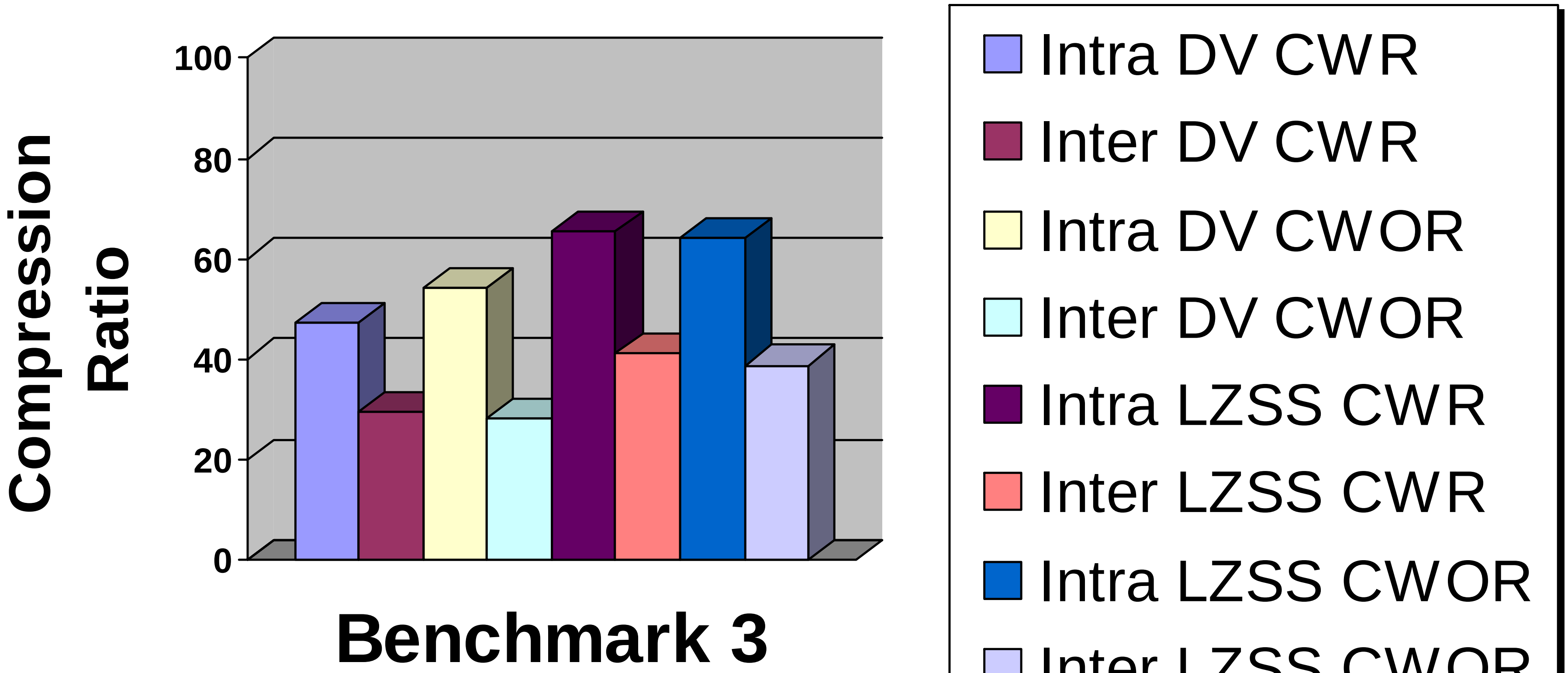
Results: Inter-Bitstream Compression

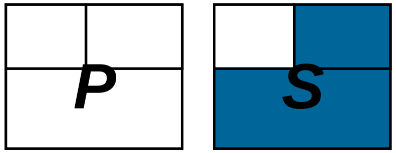


P

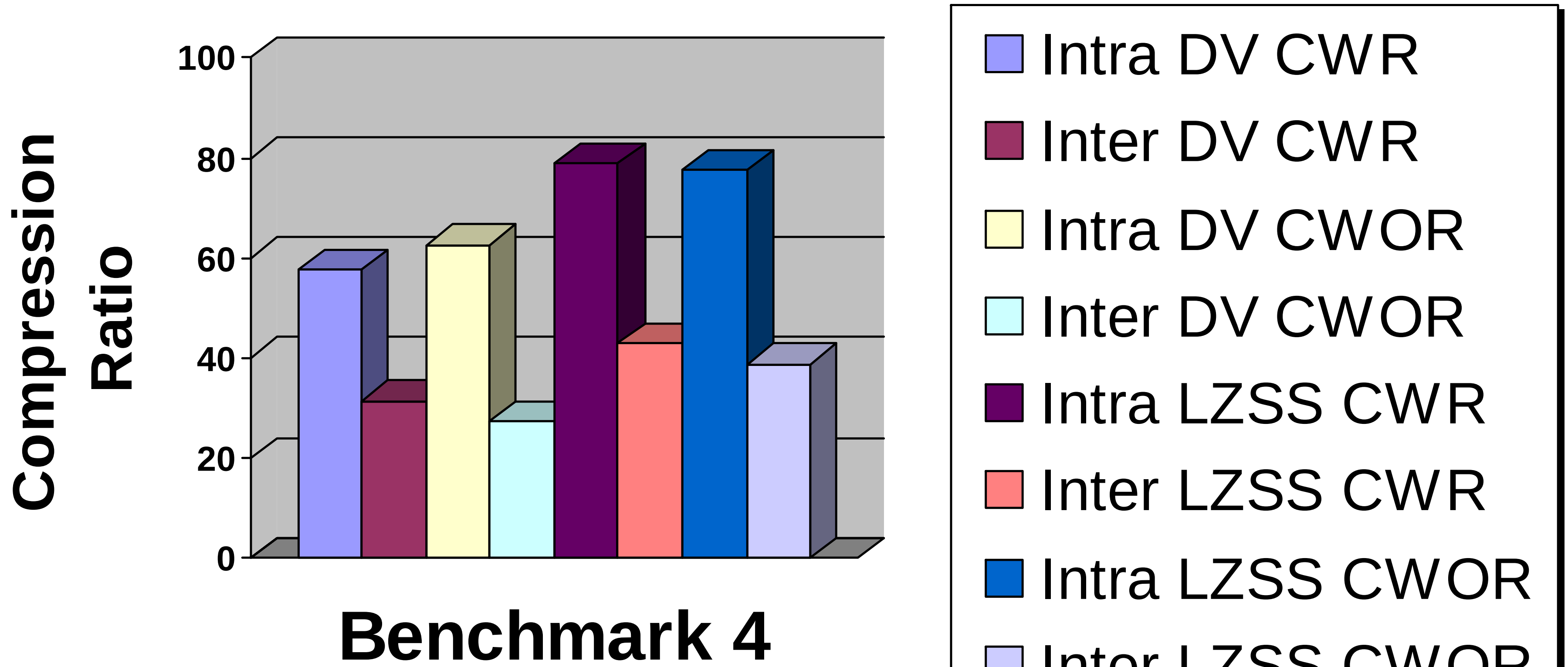
S

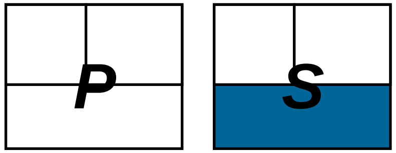
Results: Inter-Bitstream Compression





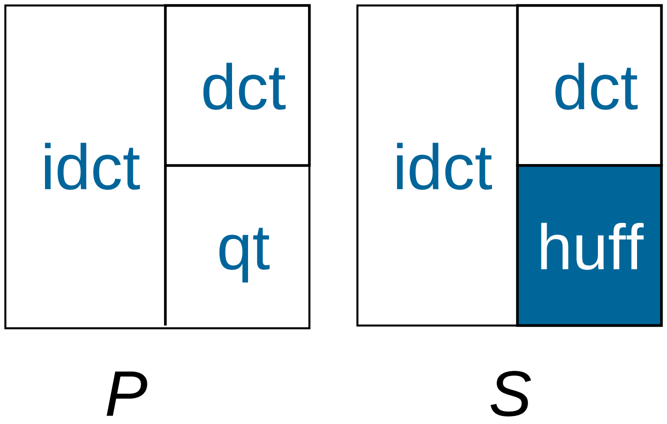
Results: Inter-Bitstream Compression





Results: Inter-Bitstream Compression

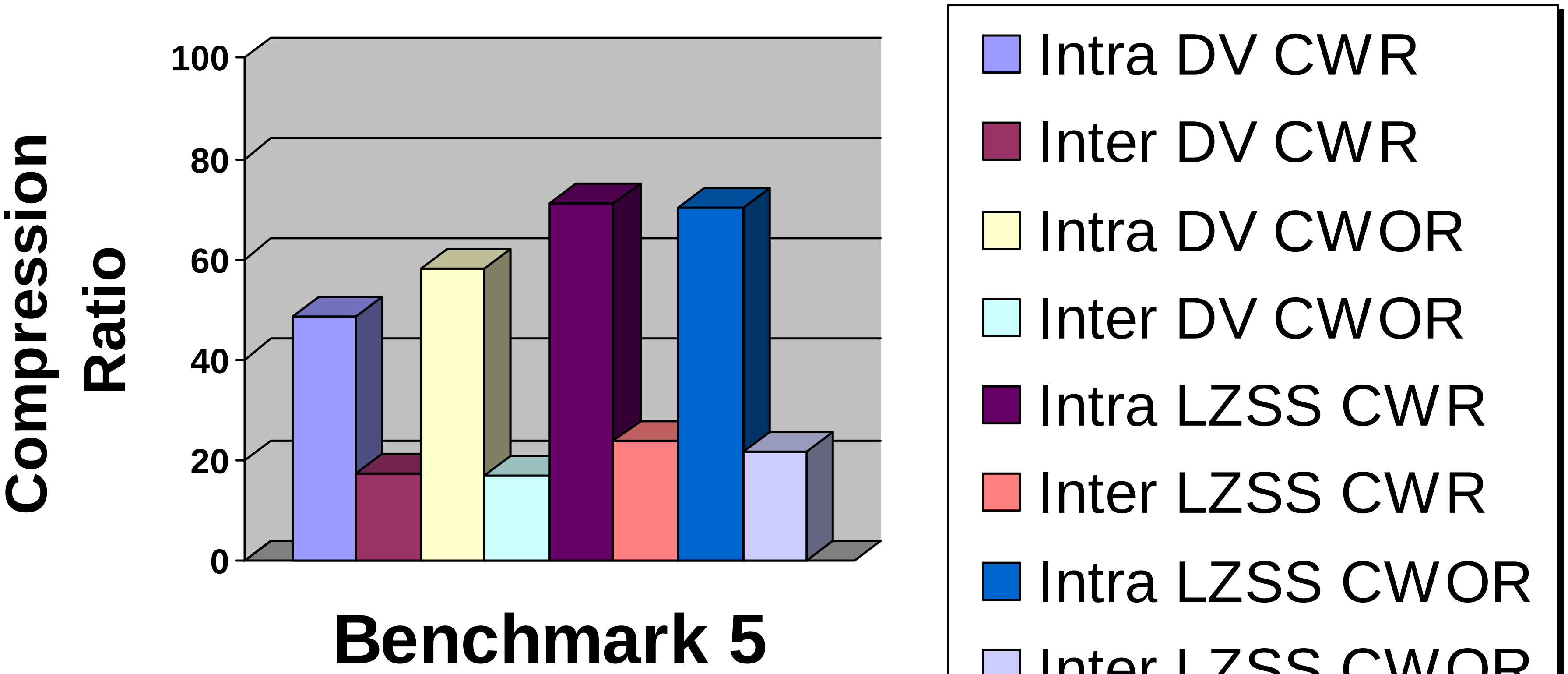
Benchmark 5

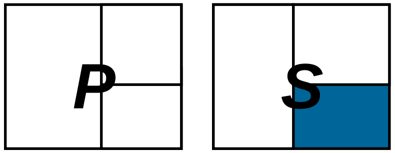


Observations

- Harness both Partial Reconfiguration and DV/LZSS techniques.
- Reduction in compress ratio reached average of ~34% (of original bitstream) over corresponding intra-bitstream compression techniques

Results: Inter-Bitstream Compression





Conclusion

- Developed novel DV compression technique
  - Compares favorably with best reported techniques on intra-bitstream compression models
- Developed inter-bitstream compression techniques
  - Performs better than both intra-bitstream compression for both DV and LZSS

Future Work

- Appropriate kernel placement to maximize reuse across multiple configurations.
- Optimal temporal and spatial partitioning of computational kernels to as to maximize the amount of configuration data preserved across bitstreams