A Multi-Ported Memory Compiler Utilizing True Dual-port BRAMs

Ameer Abdelhadi and Guy Lemieux

Department of Electrical and Computer Engineering

University of British Columbia

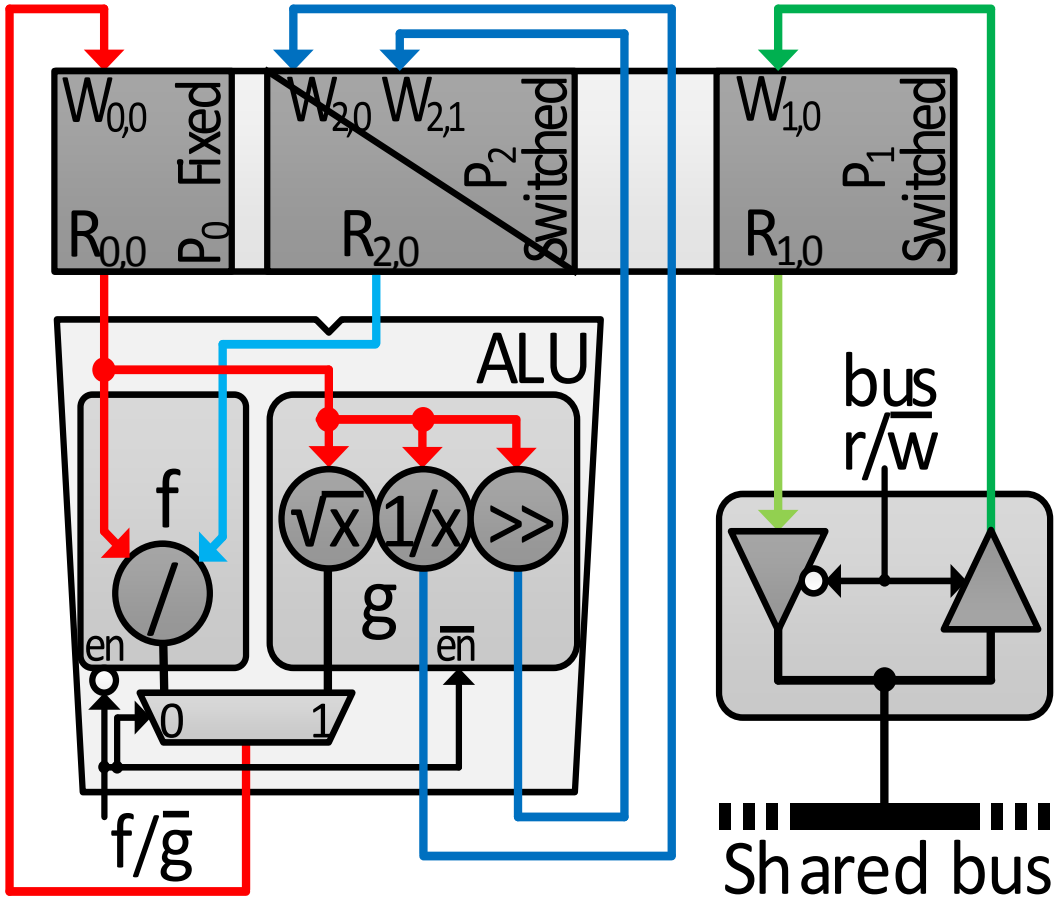
Vancouver, Canada

May 3rd, 2016

Motivation (1): FPGAs as parallel accelerators

- Used as parallel accelerators
  - ✘ Have dual-ported memories only

Motivation (2)
Mixed port requirements

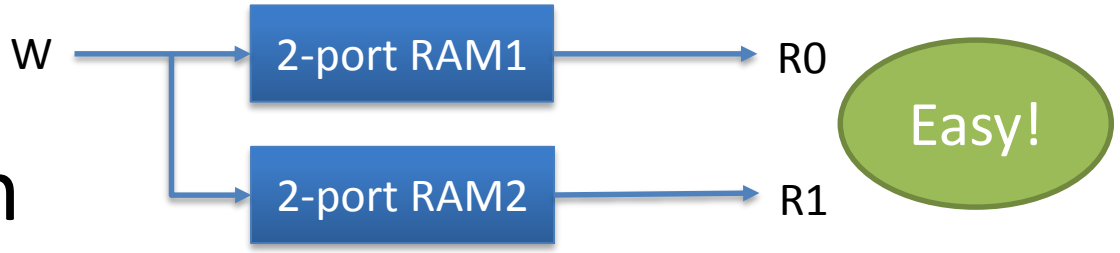


- Multi-porting approaches provide simple (fixed) ports only
- Waste of resources if these ports are not active simultaneously

Live-Value Table (LVT)

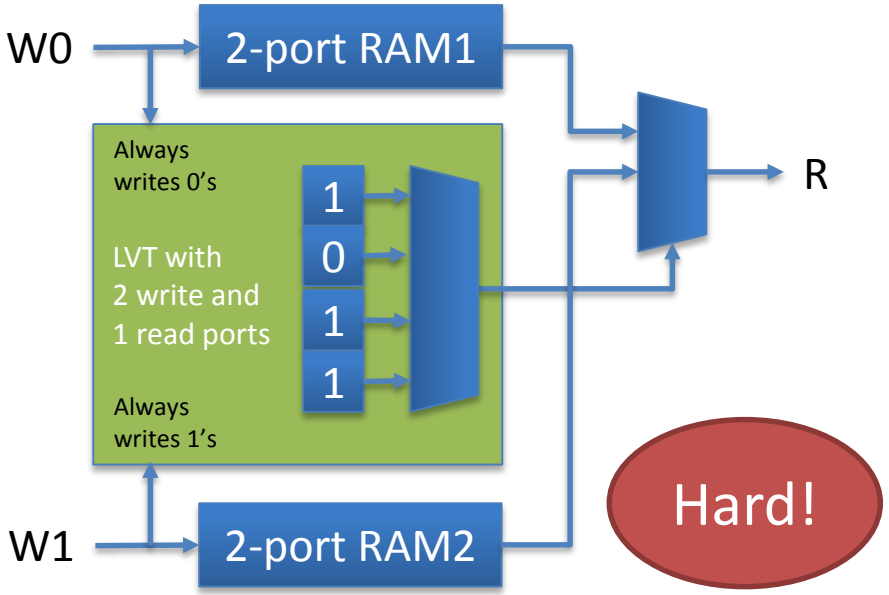
Multi-read

- Replication



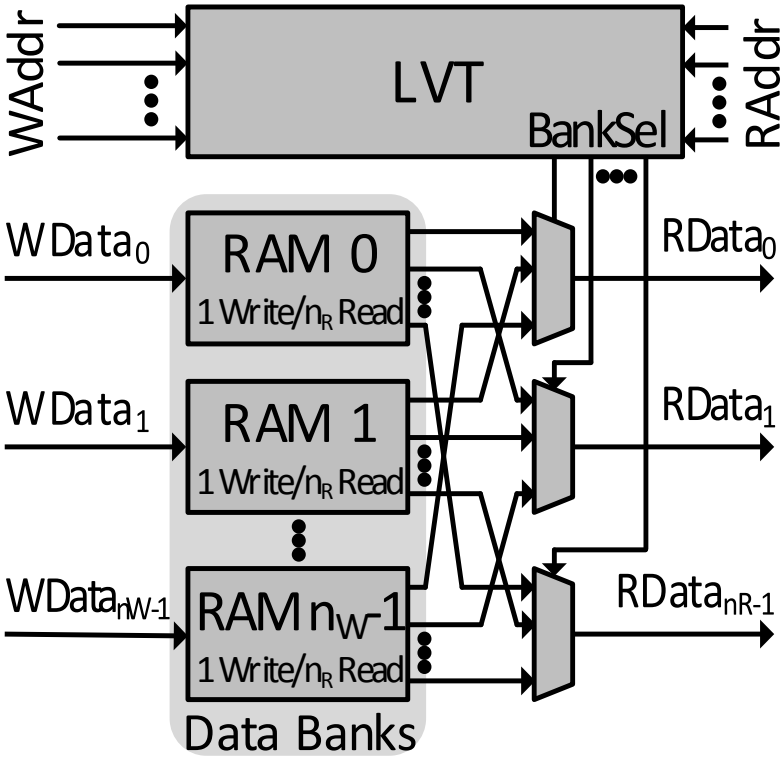
Multi-write

- LVT



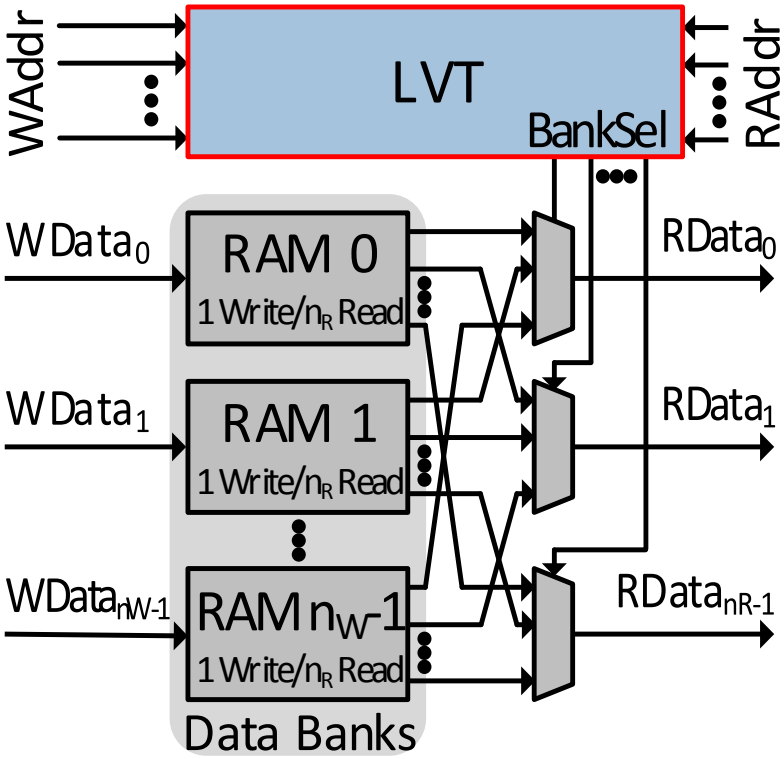
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:



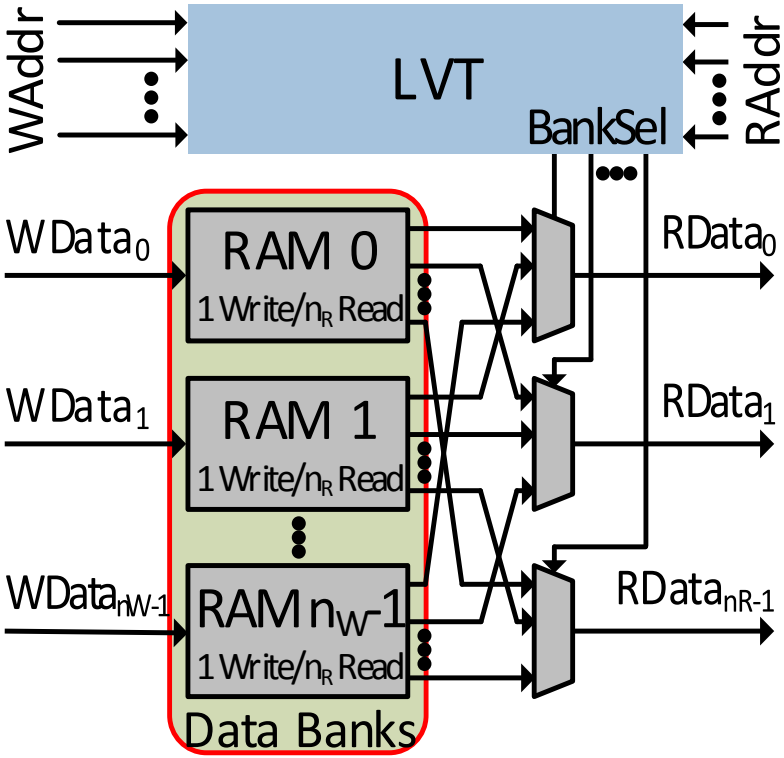
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  1) LVT - tracks changes



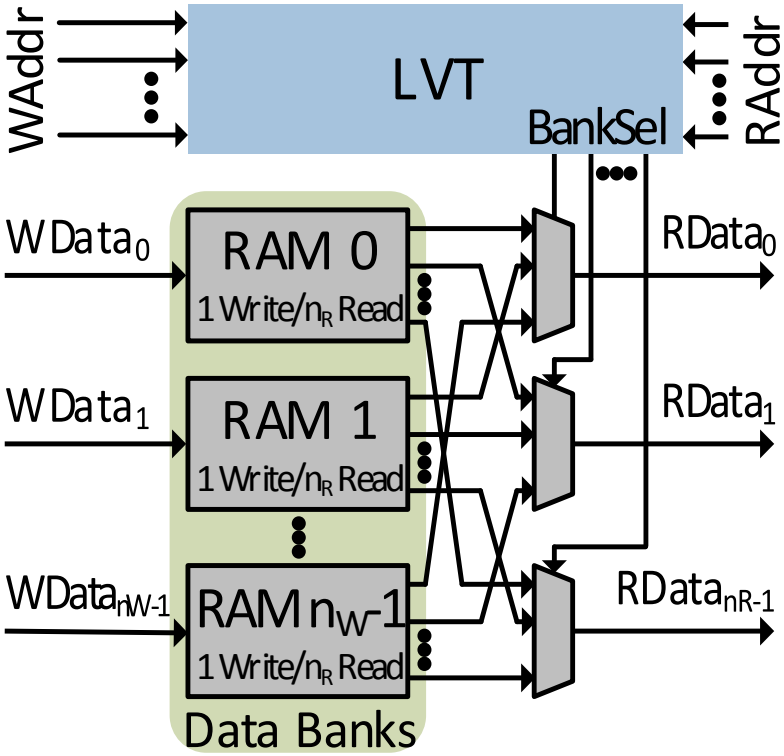
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  1) LVT - tracks changes
  2) Data banks - store data copies



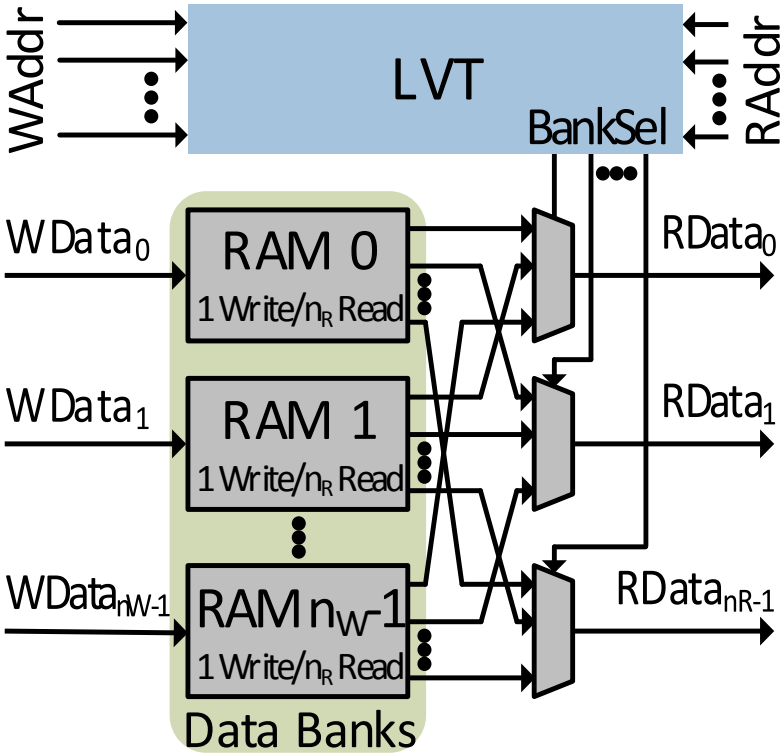
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  - 1) LVT - tracks changes
  - 2) Data banks - store data copies
- Our previous work (I-LVT/ FPGA'14) optimizes LVT only



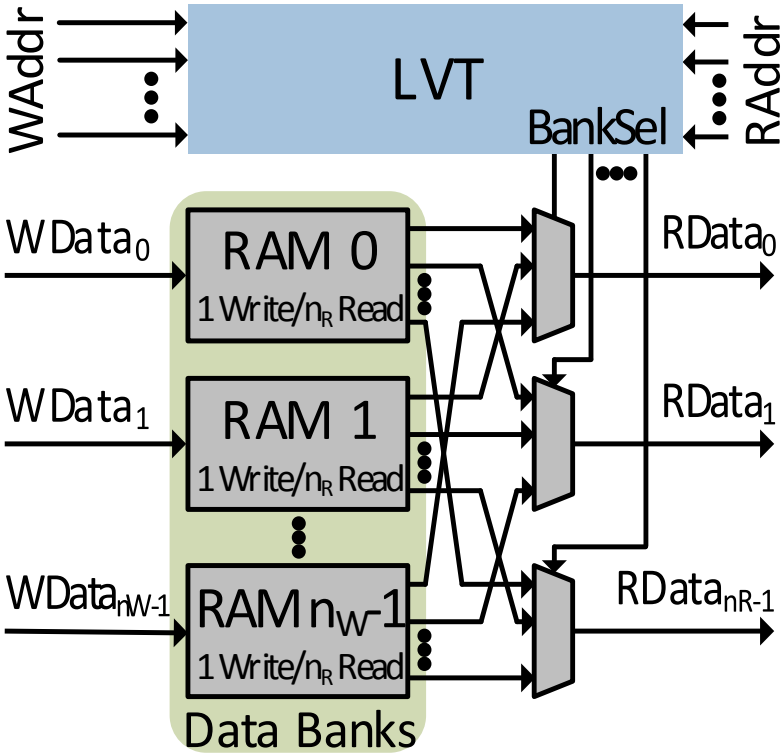
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  - 1) LVT - tracks changes
  - 2) Data banks - store data copies
- Our previous work (I-LVT/ FPGA'14) optimizes LVT only
- This work
  - Optimizes the data banks (not the LVT!)



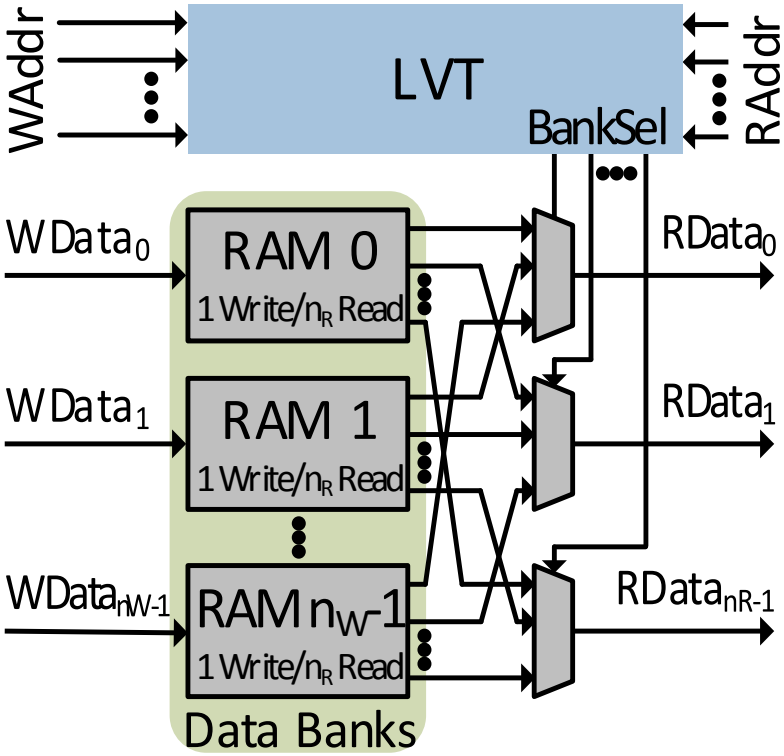
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  - 1) LVT - tracks changes
  - 2) Data banks - store data copies
- Our previous work (I-LVT/ FPGA'14) optimizes LVT only
- This work
  - Optimizes the data banks (not the LVT!)
  - The first technique that requires a CAD tool



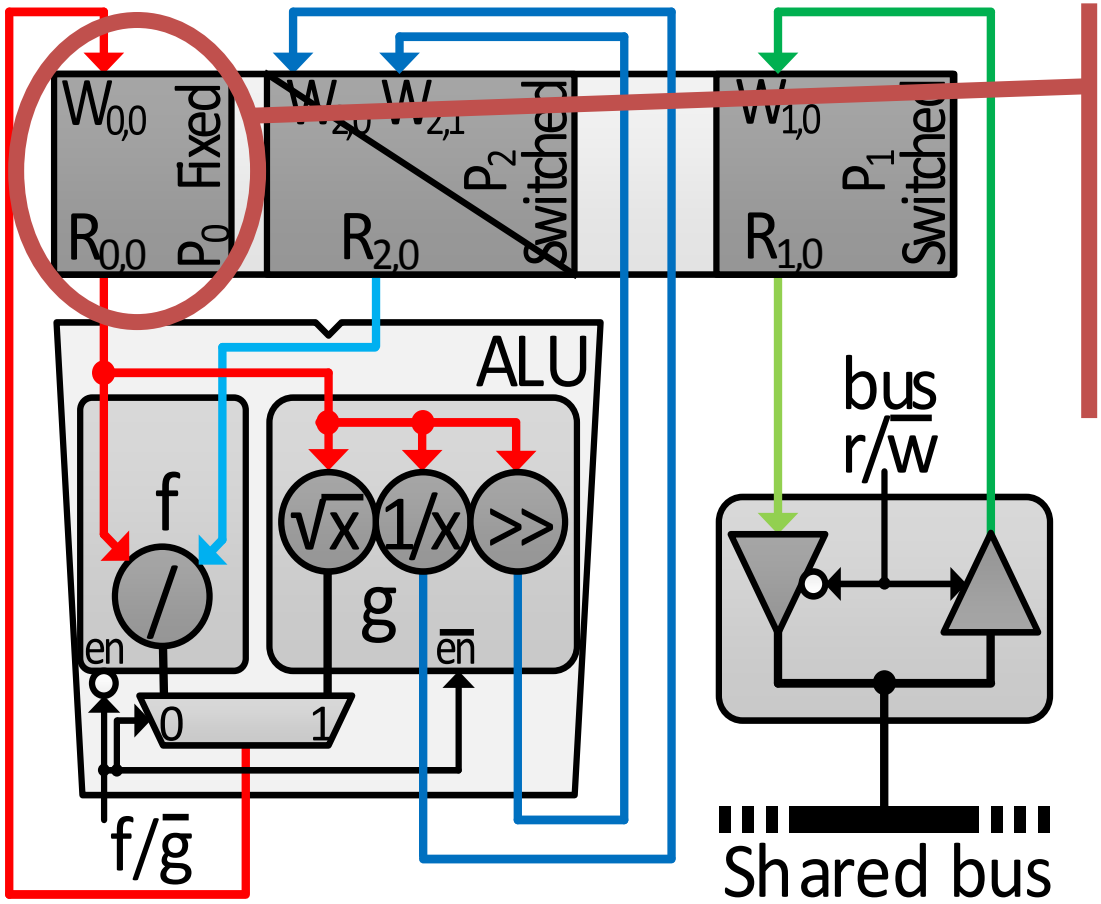
Data Banks Optimization

- LVT-based multi-ported RAM is composed of:
  - 1) LVT - tracks changes
  - 2) Data banks - store data copies
- Our previous work (I-LVT/ FPGA'14) optimizes LVT only
- This work
  - Optimizes the data banks (not the LVT!)
  - The first technique that requires a CAD tool



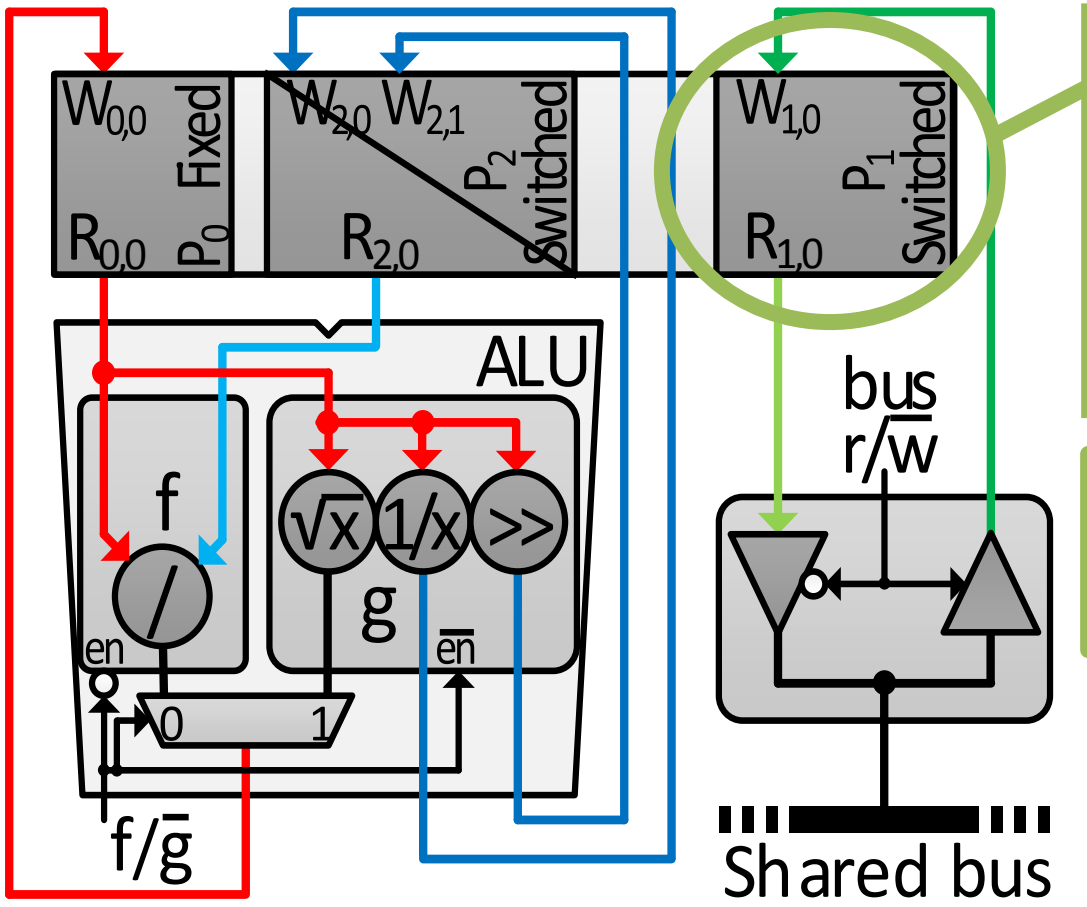
This work solves the final step and most important problem of Block RAM allocation

Mixed Port Requirements (1): Fixed ports



Fixed (simple) ports:
The majority of multi-ported memories supports fixed ports only

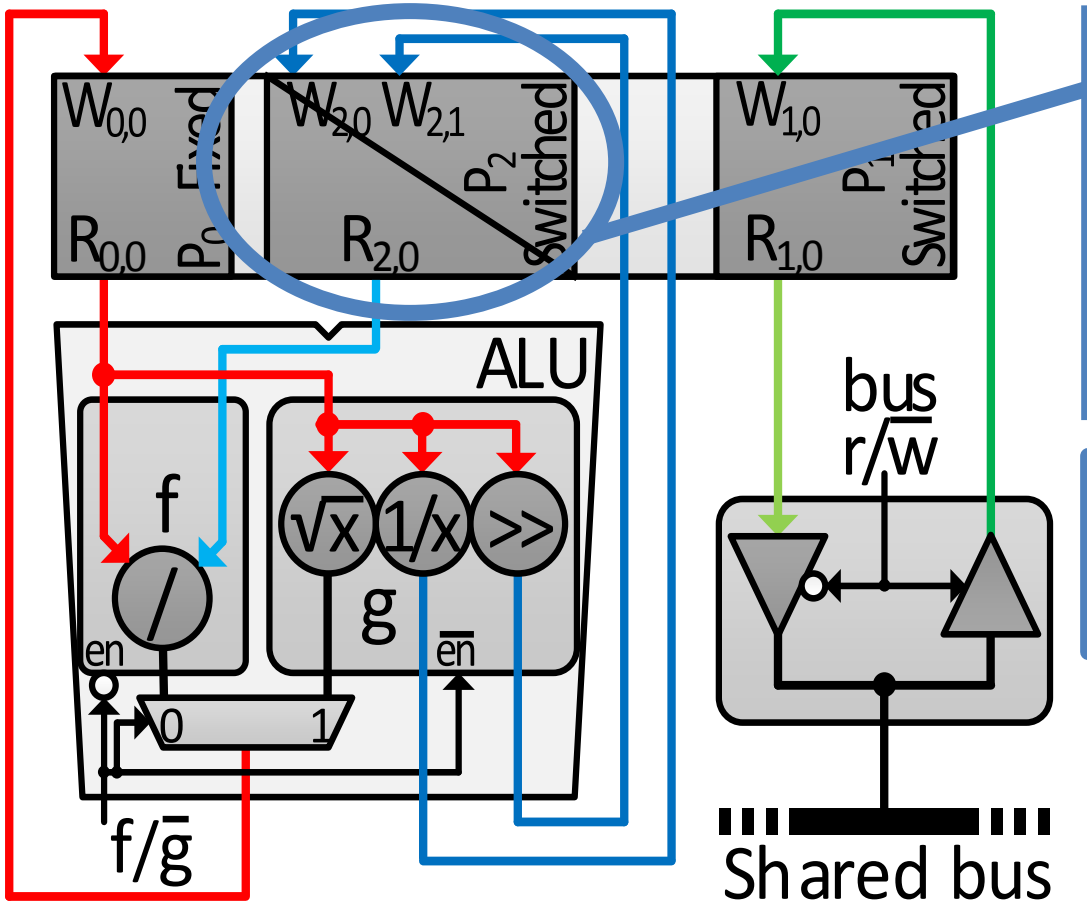
Mixed Port Requirements (2): True ports



True ports:
Some techniques support the construction of multi-true-ports

BRAMs in FPGAs are true dual-ported

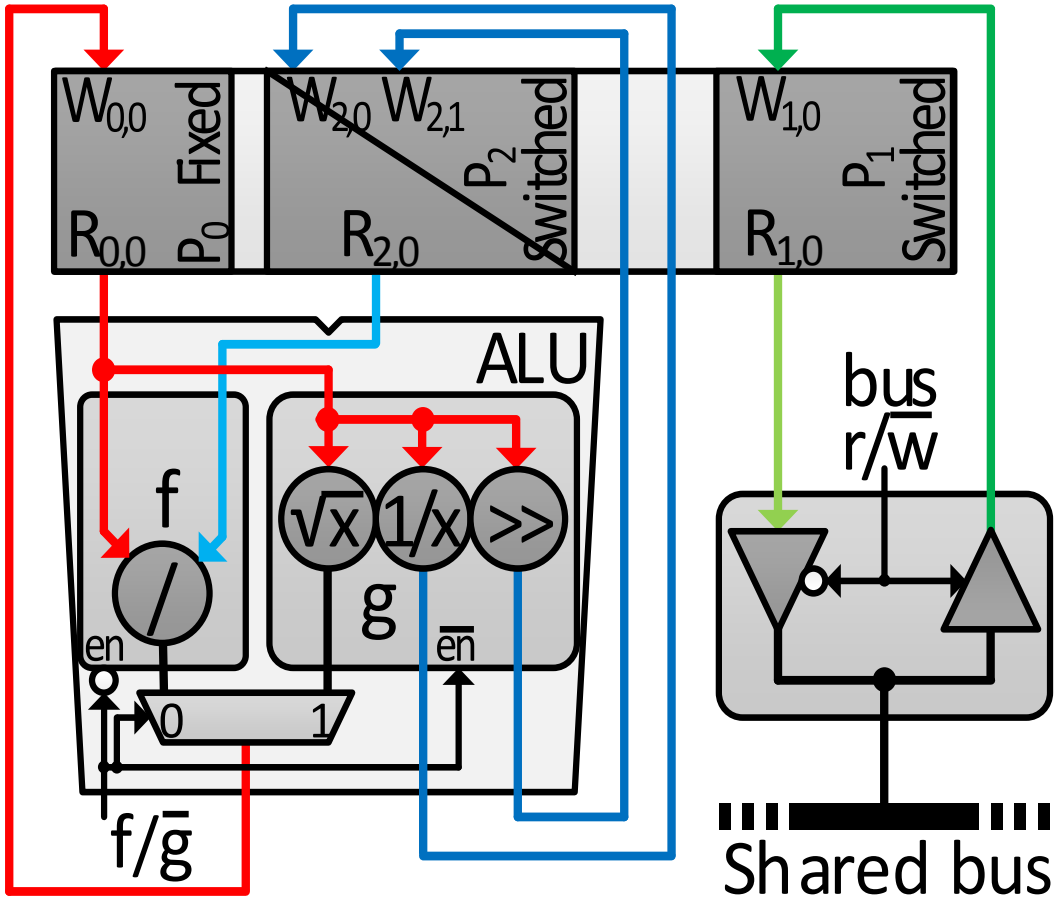
Mixed Port Requirements (3): Switched ports



Switched ports:
A number of writes are switched with a number of reads

True ports are special case of switched ports

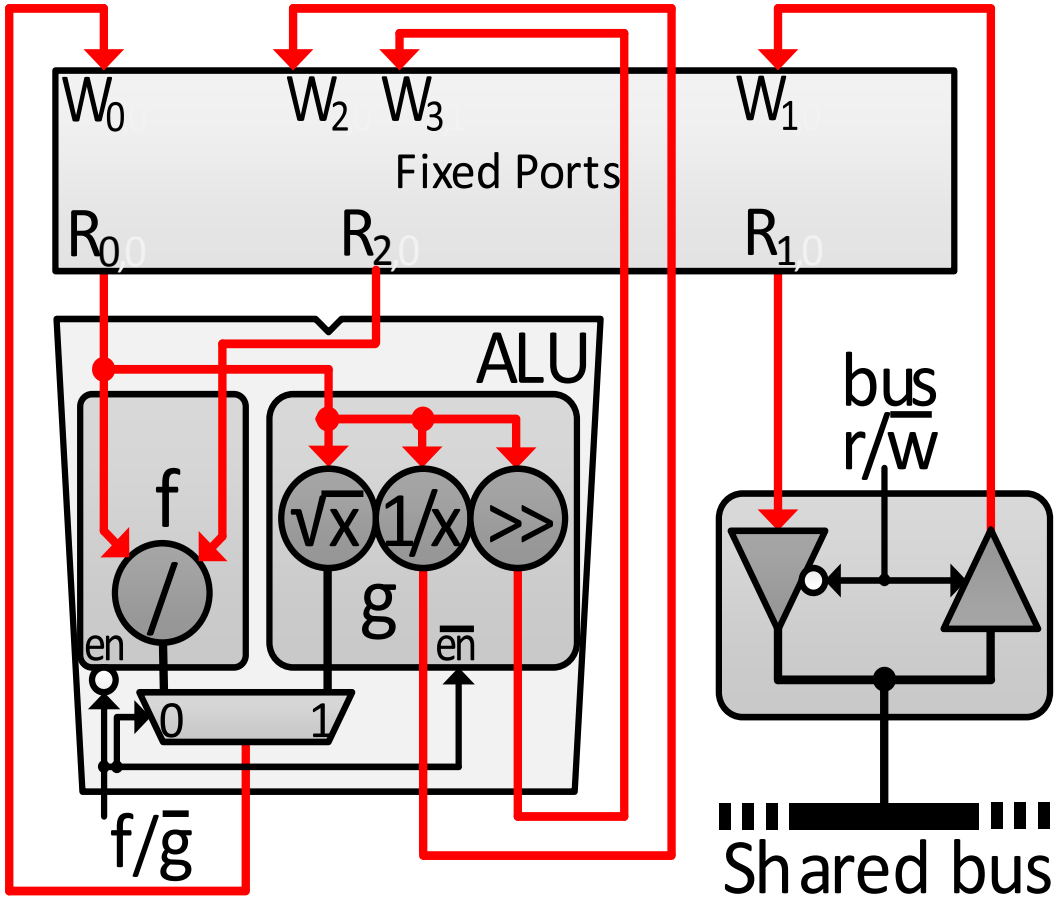
Switched Ports (1) Example



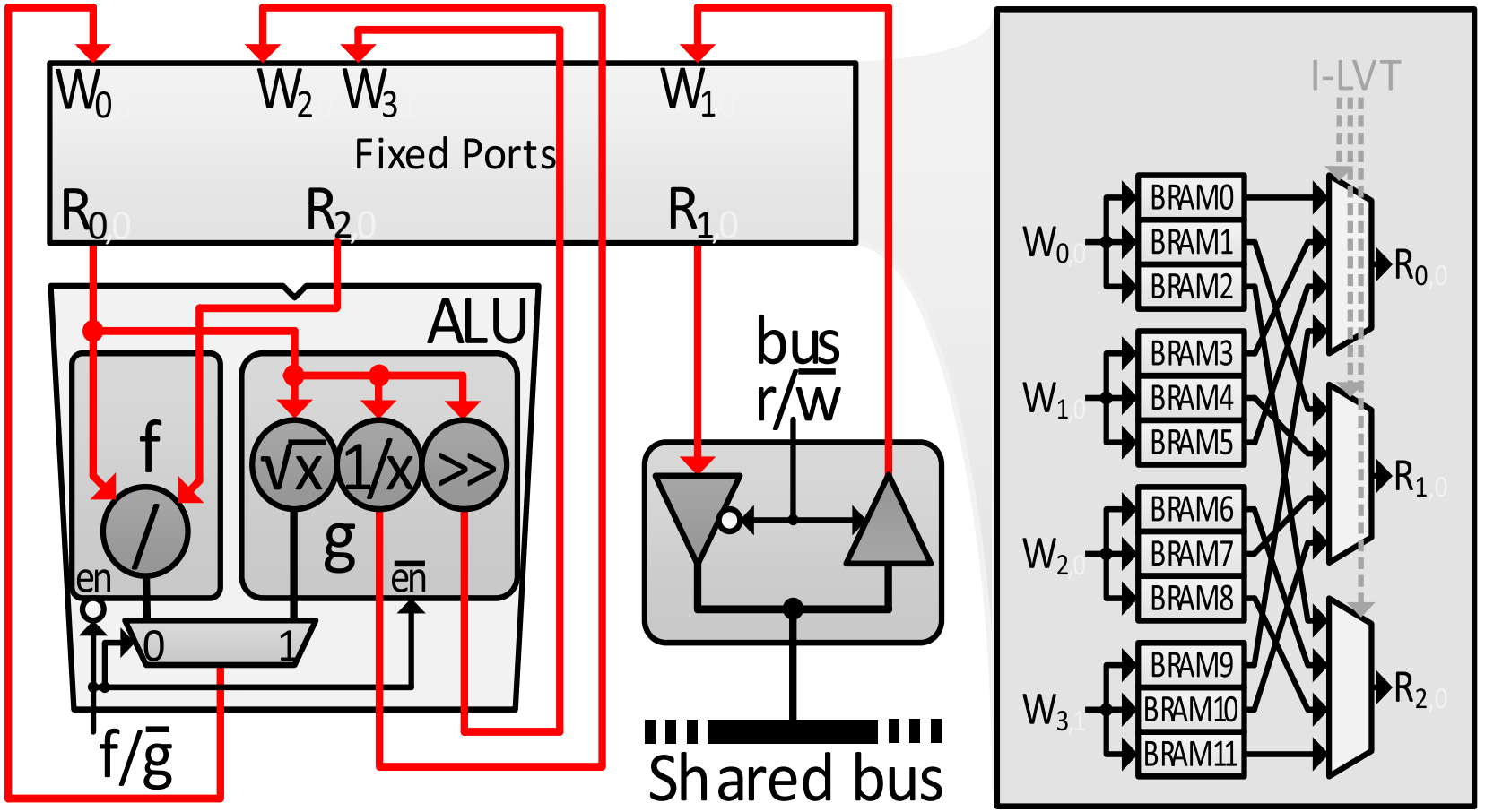
Objectives:
Optimize the construction of multi-switched ports

Key Observation:
BRAMs' **true** ports can be utilized to optimize switched ports

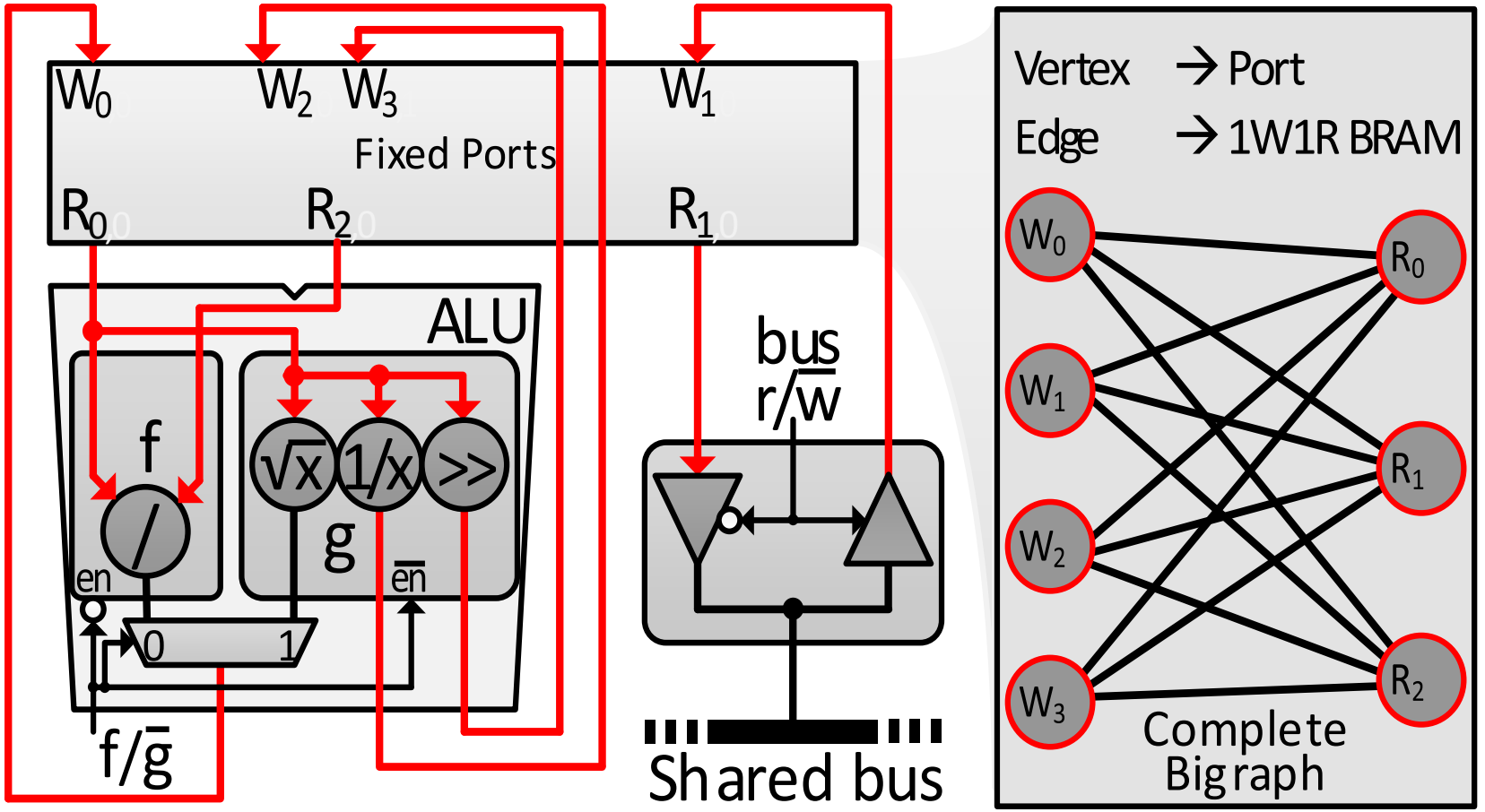
Switched Ports (2)
Fixed ports abstraction



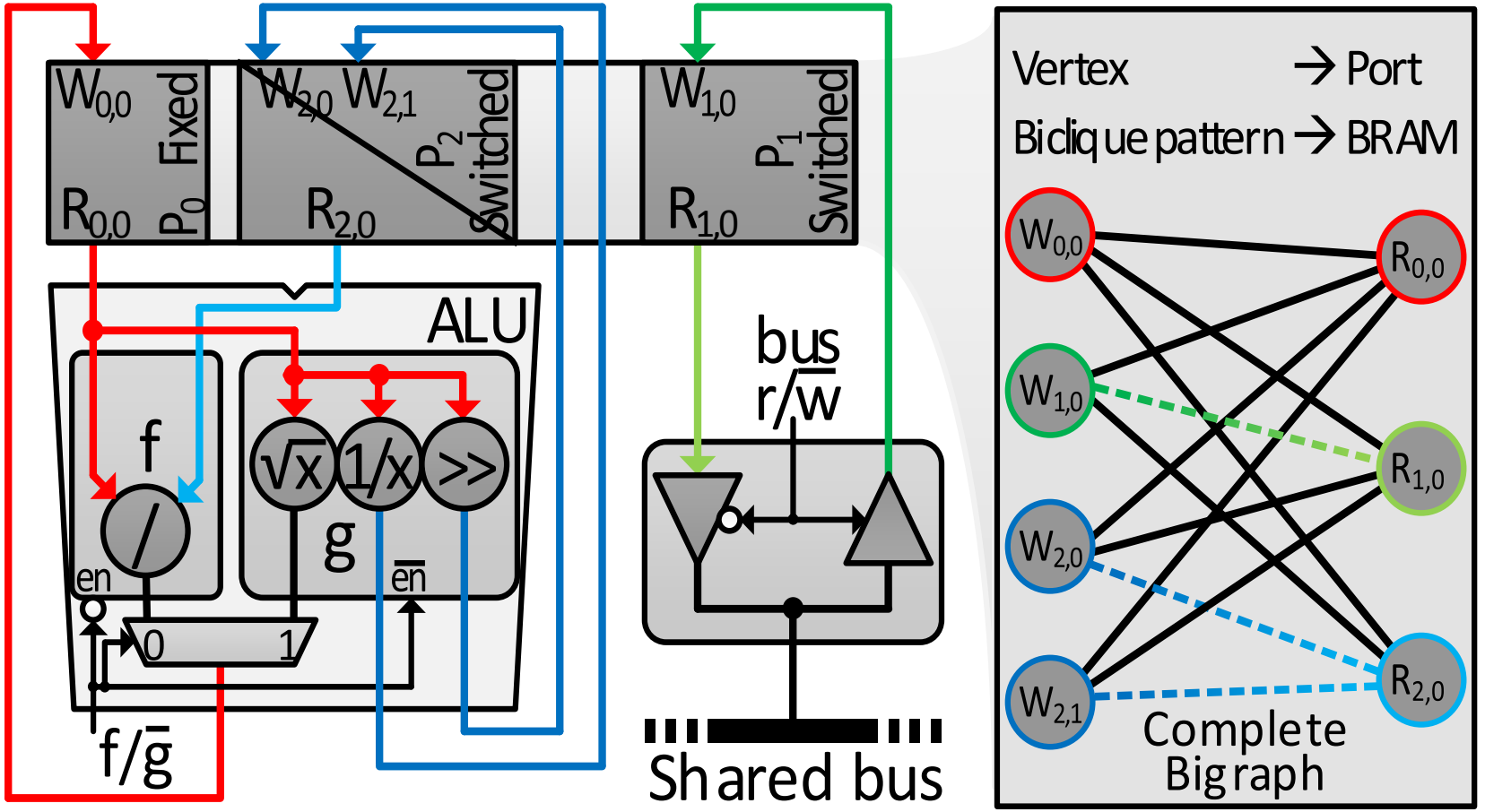
Switched Ports (3)
Fixed data banks



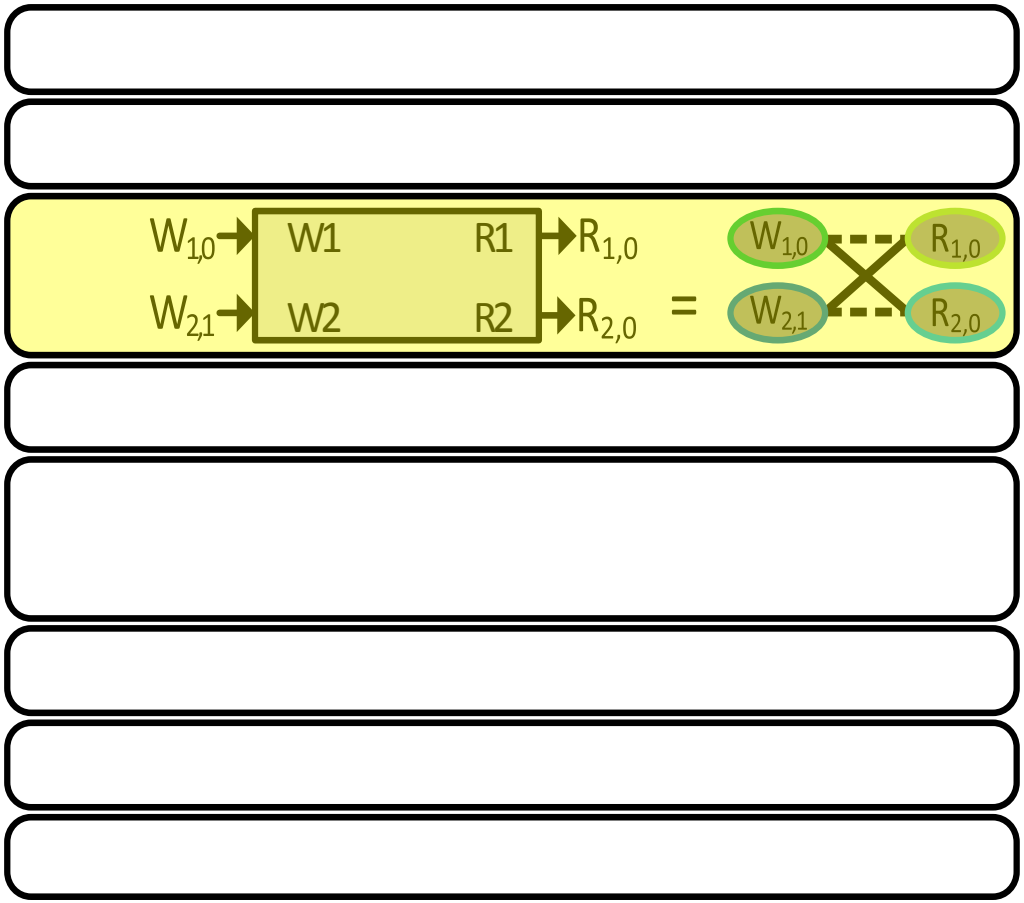
Switched Ports (4)
DFG modeling

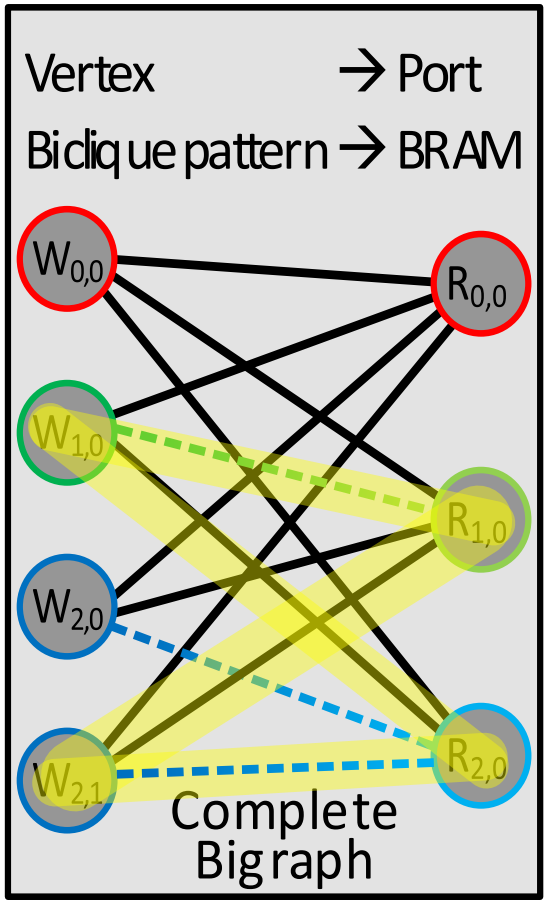


Switched Ports (5)
Switched DFG

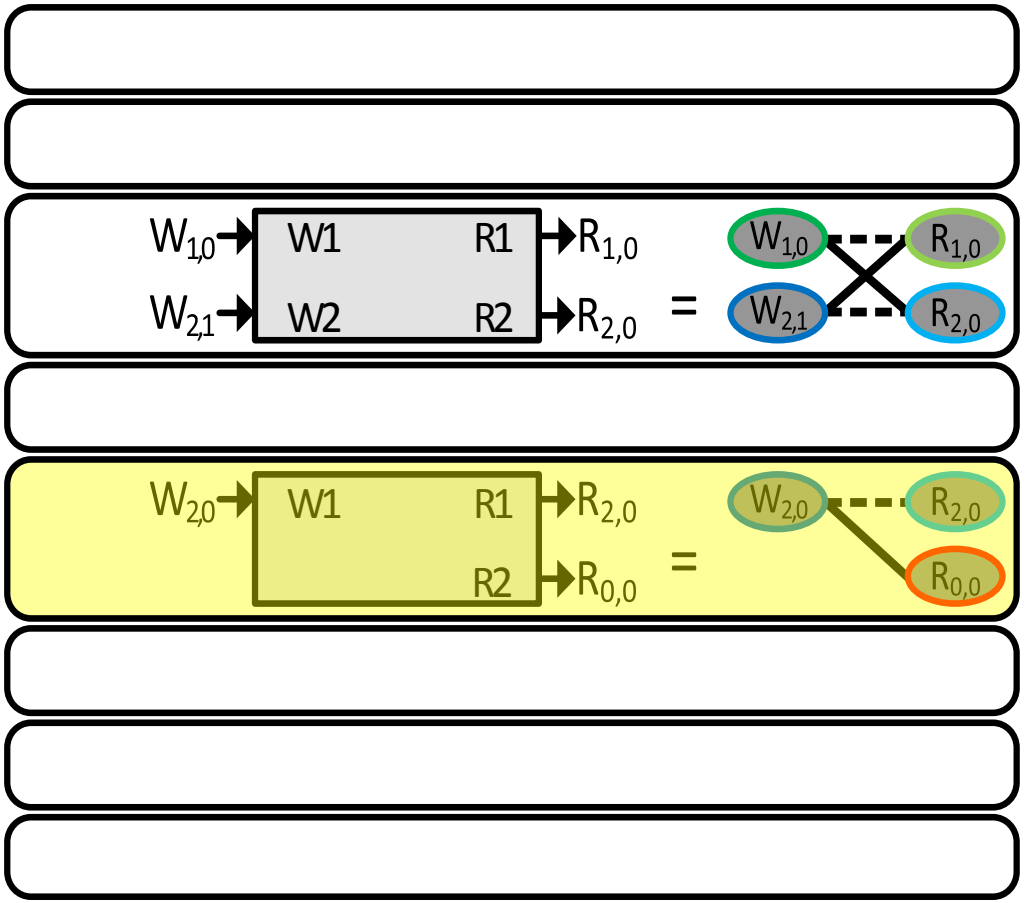


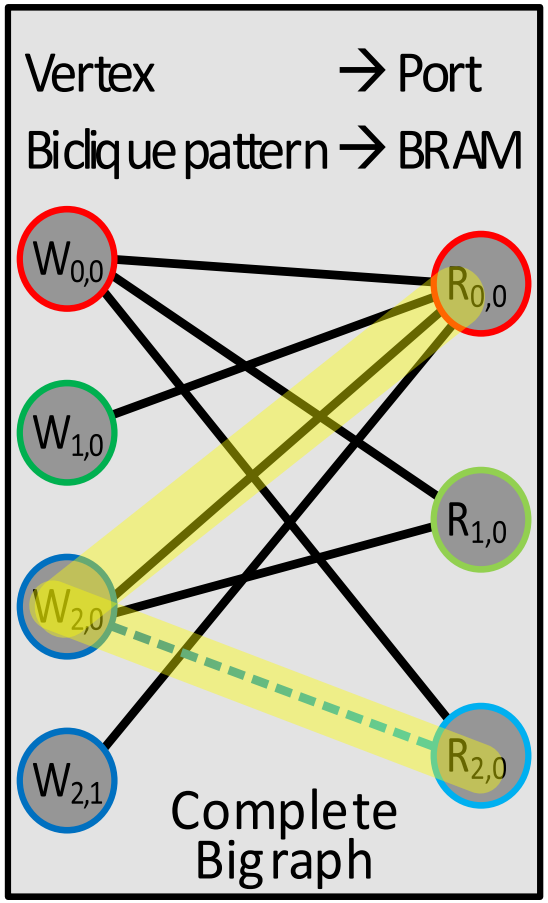
Switched Ports (6)
DFG Covering



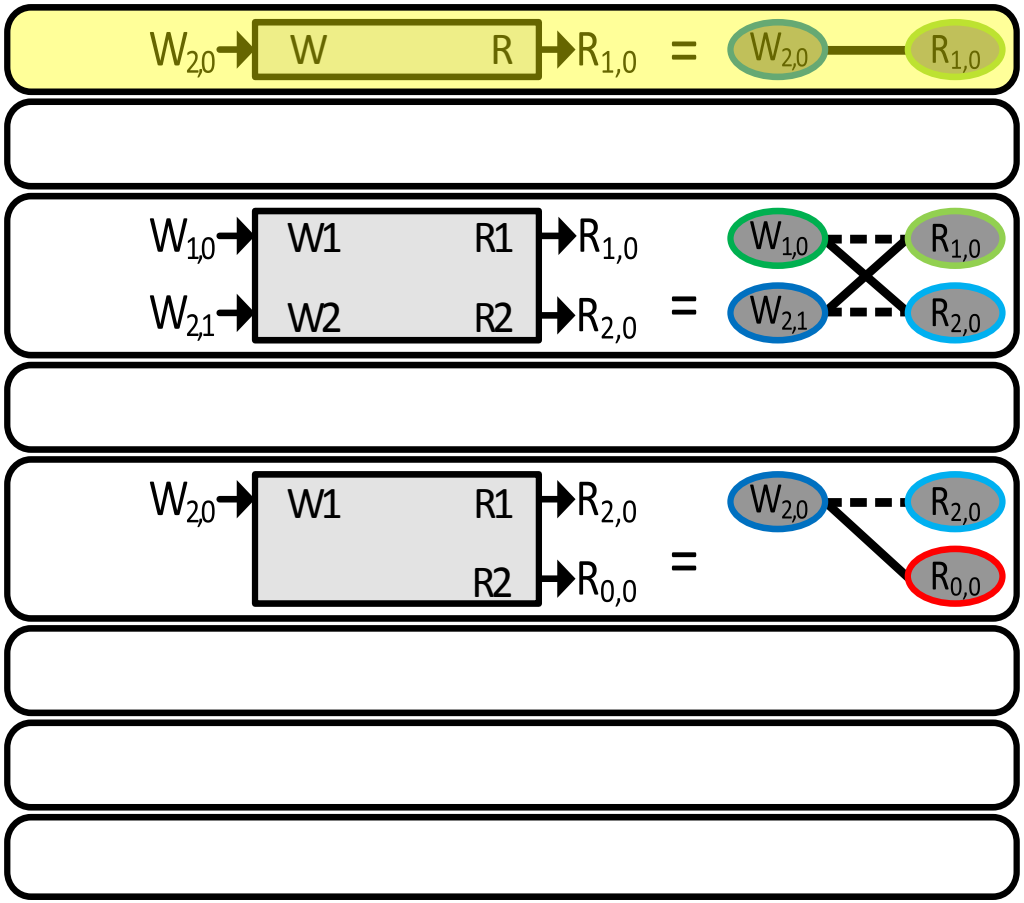


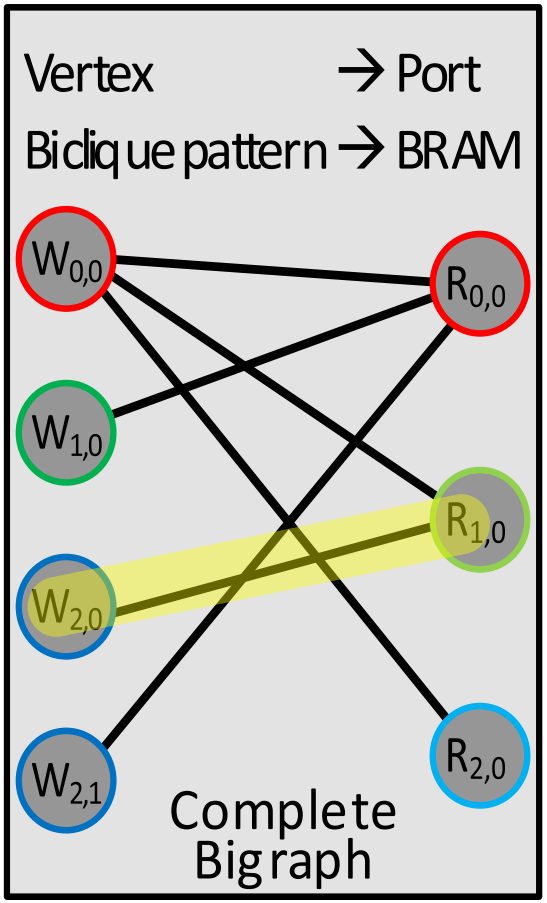
Switched Ports (6)
DFG Covering



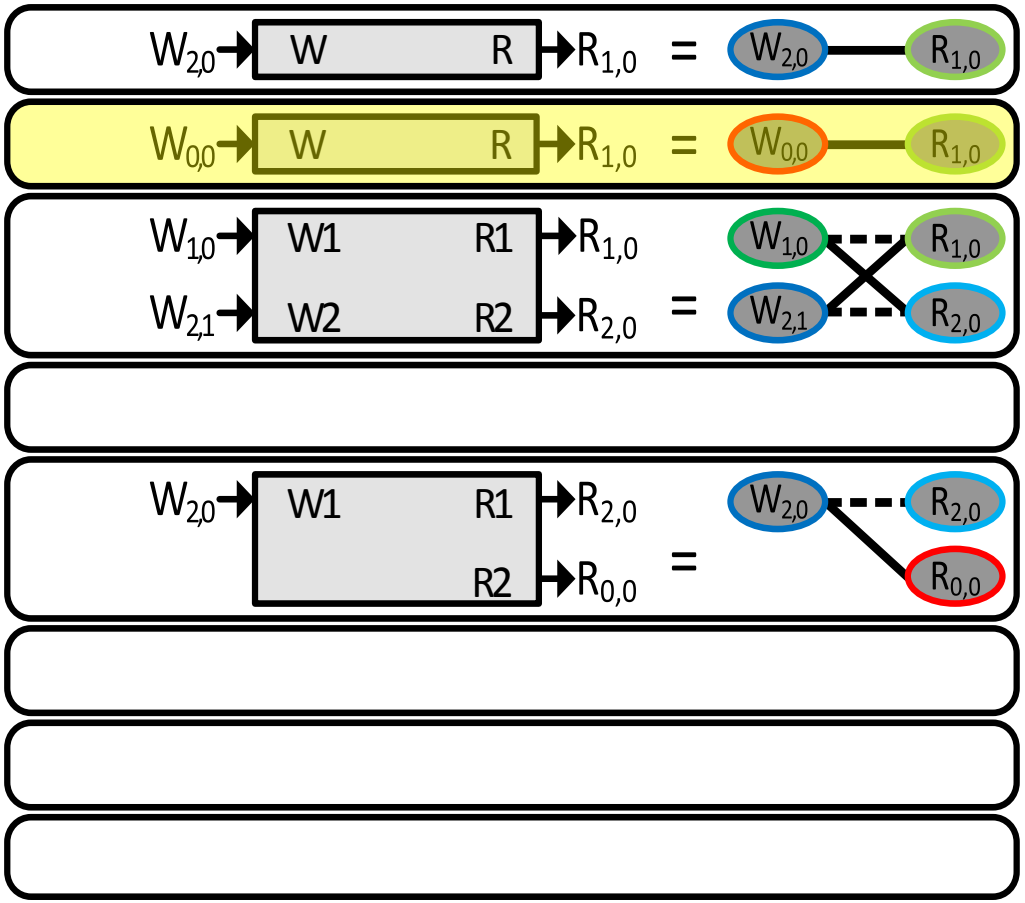


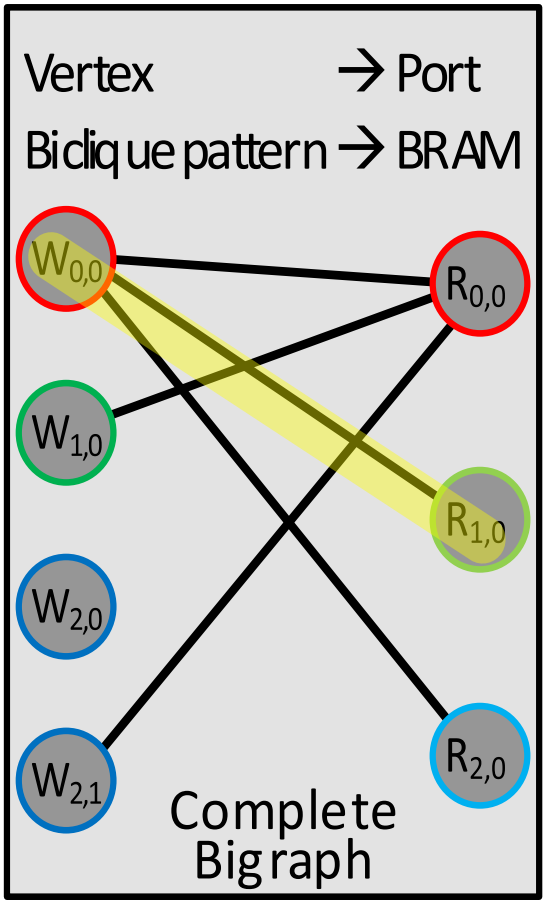
Switched Ports (6)
DFG Covering



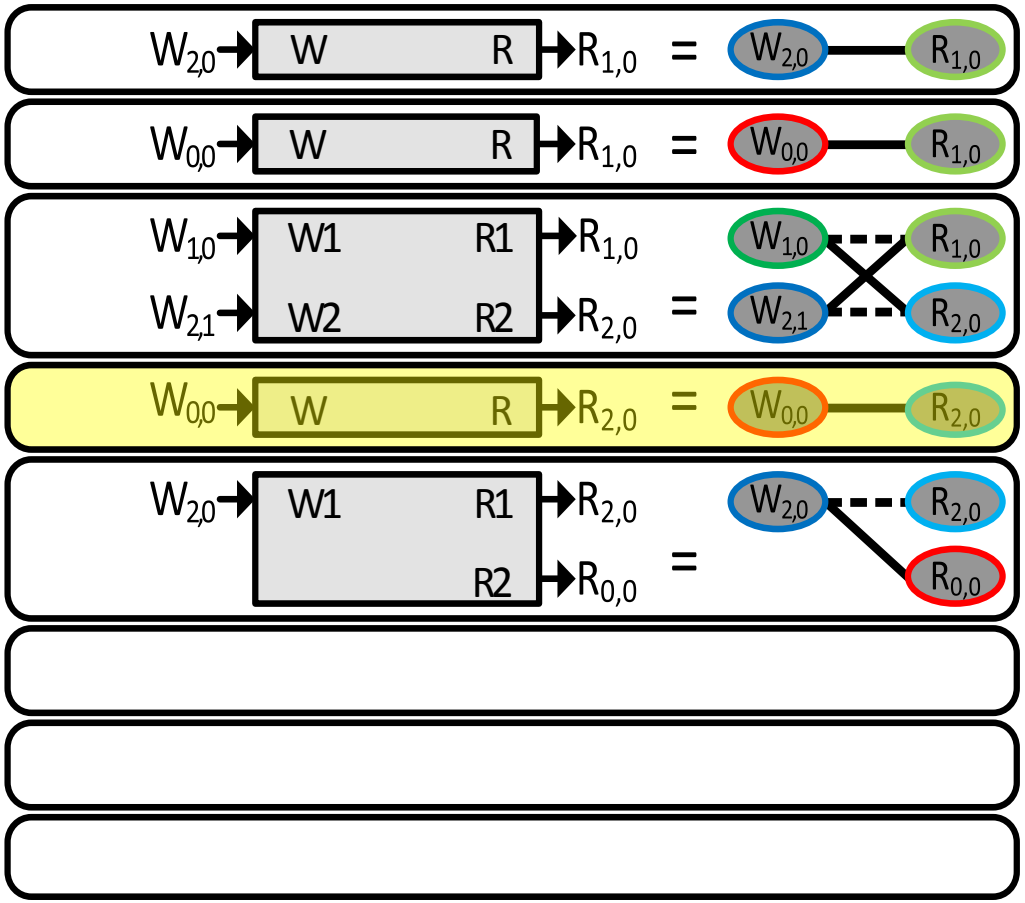


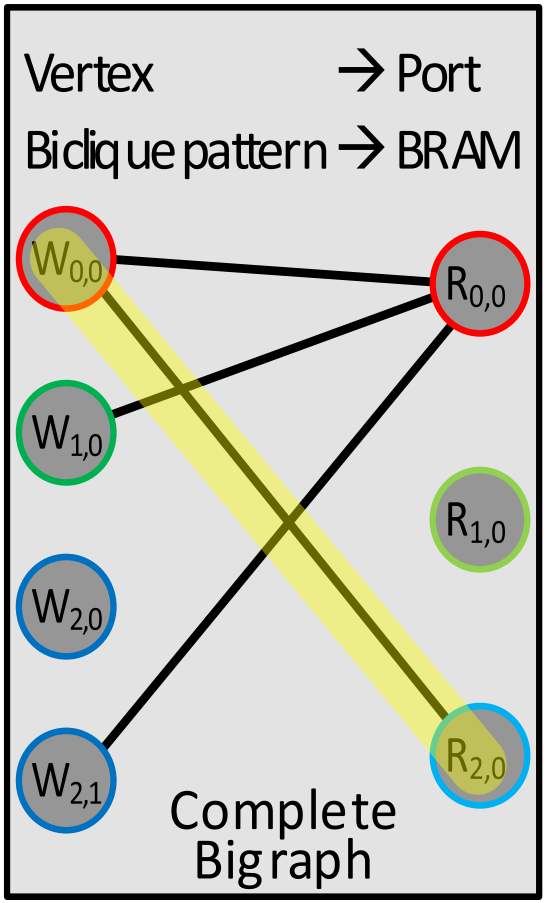
Switched Ports (6)
DFG Covering



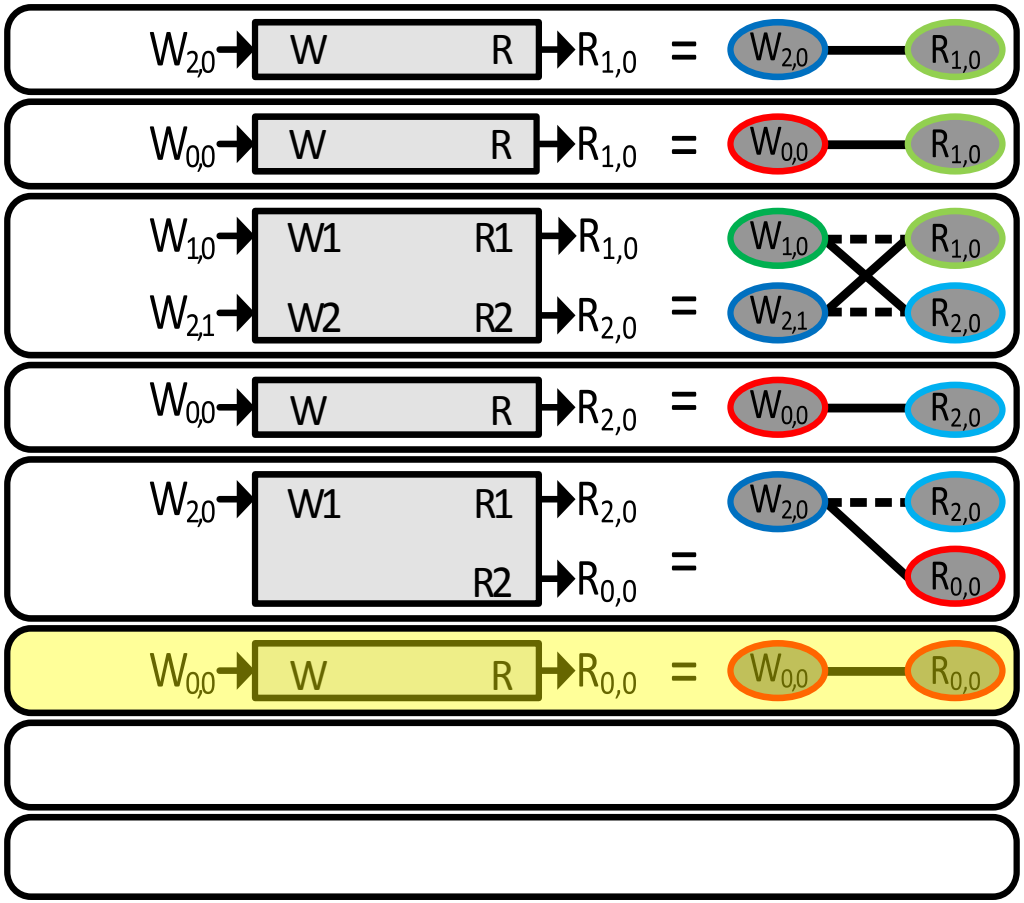


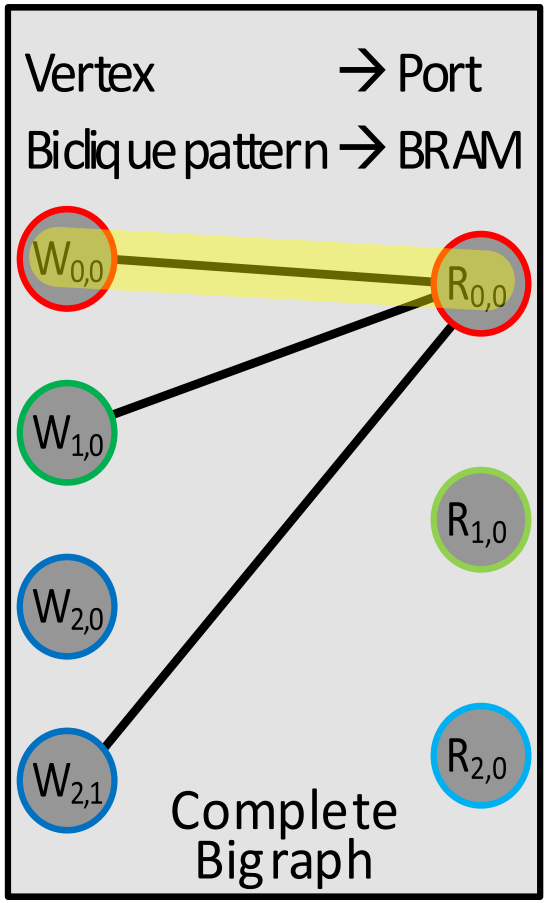
Switched Ports (6)
DFG Covering



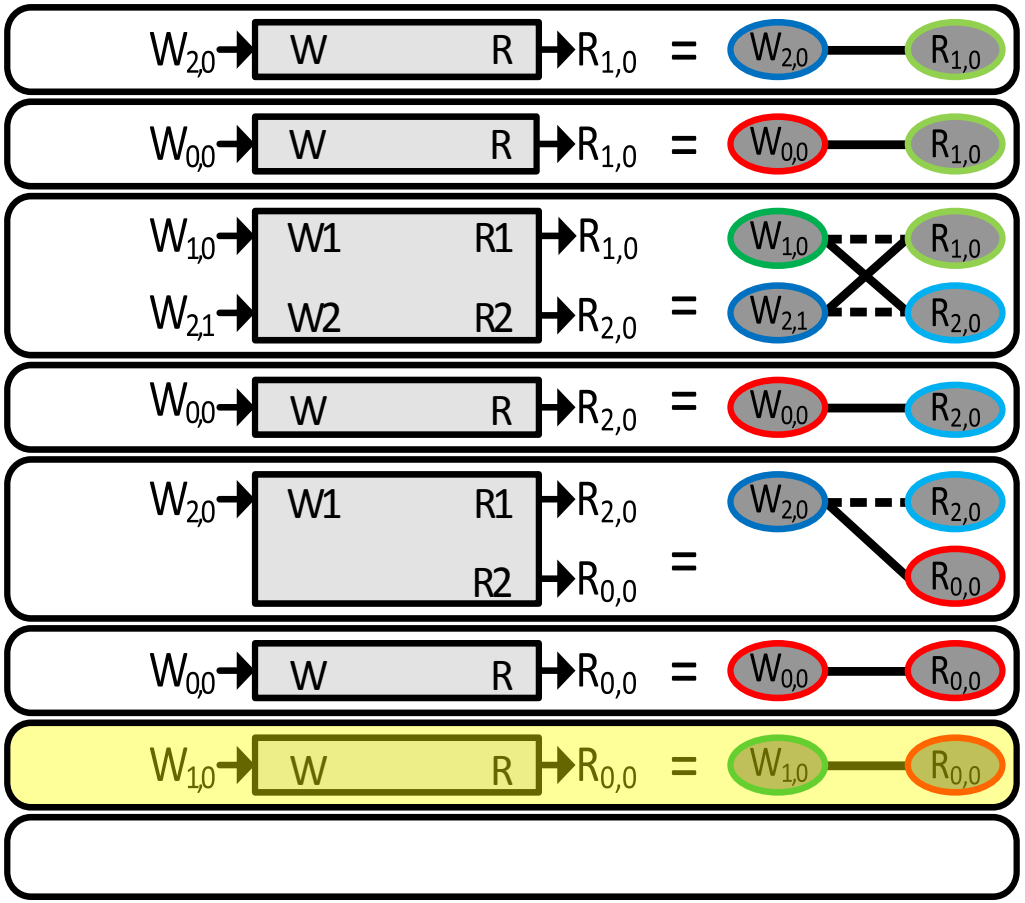


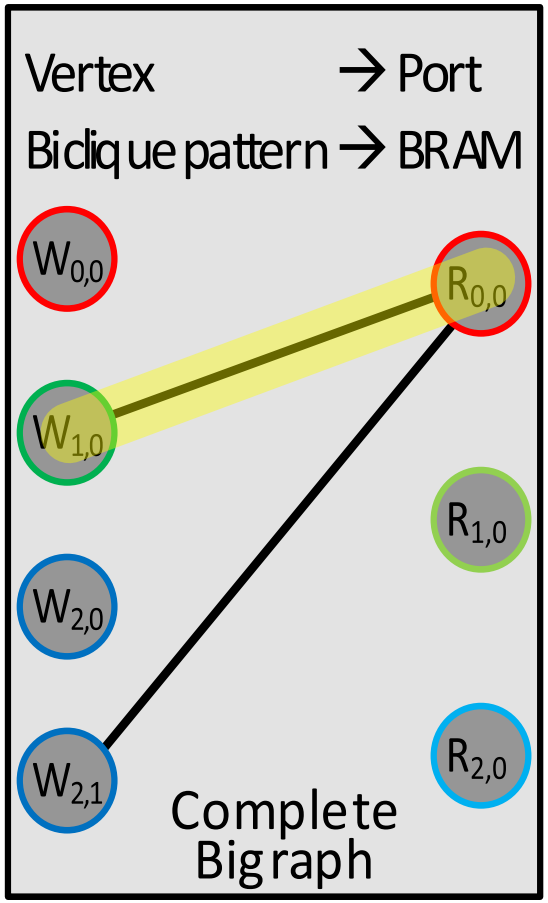
Switched Ports (6)
DFG Covering



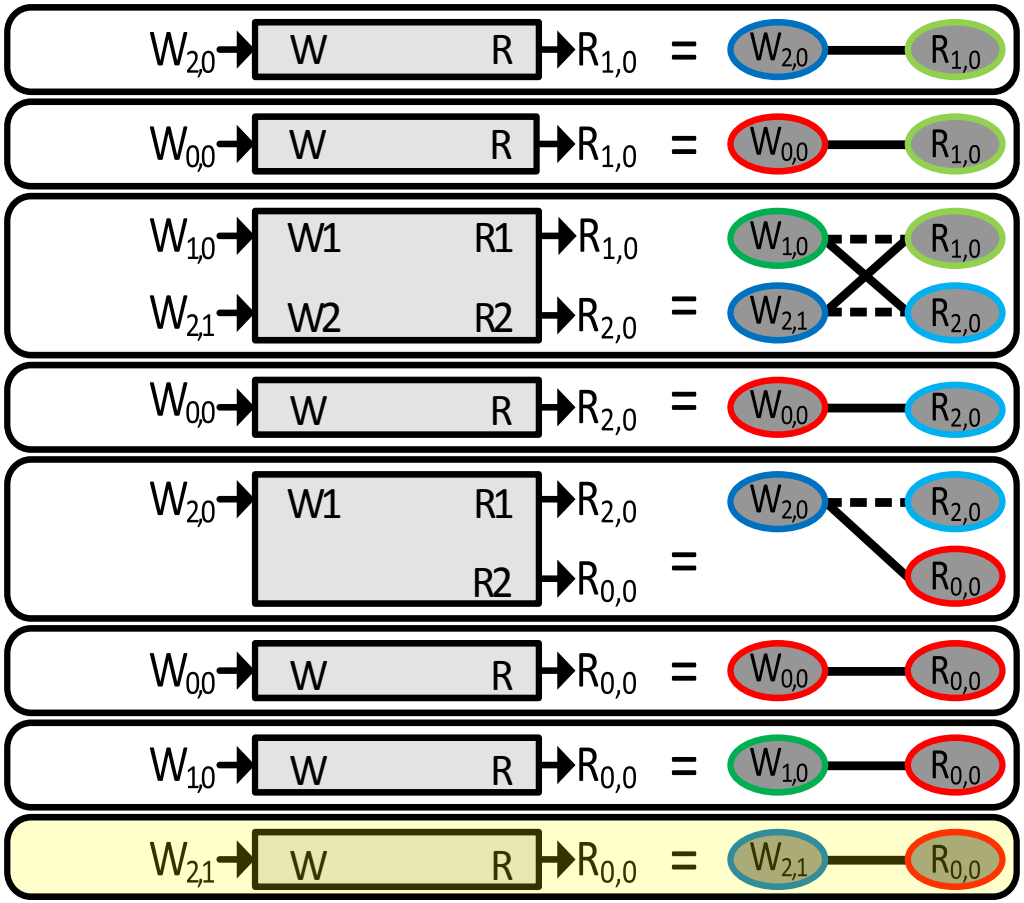


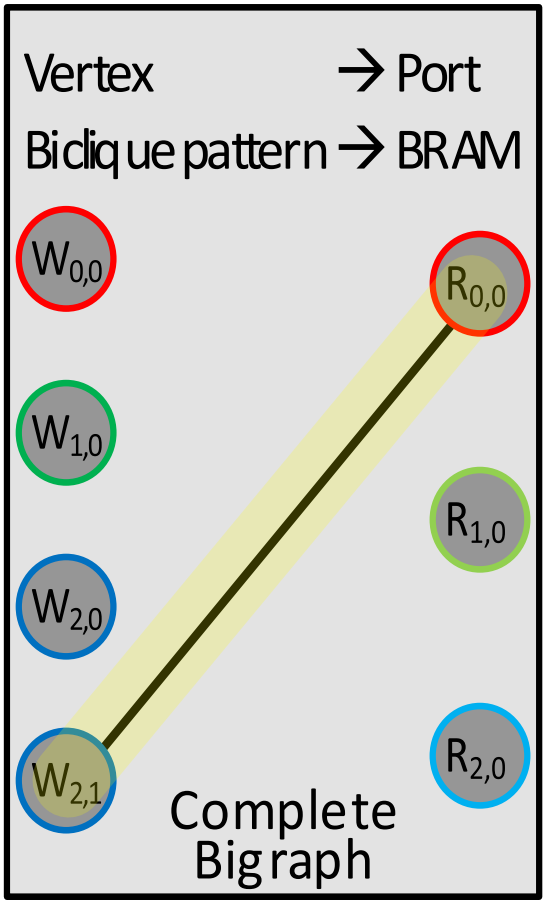
Switched Ports (6)
DFG Covering



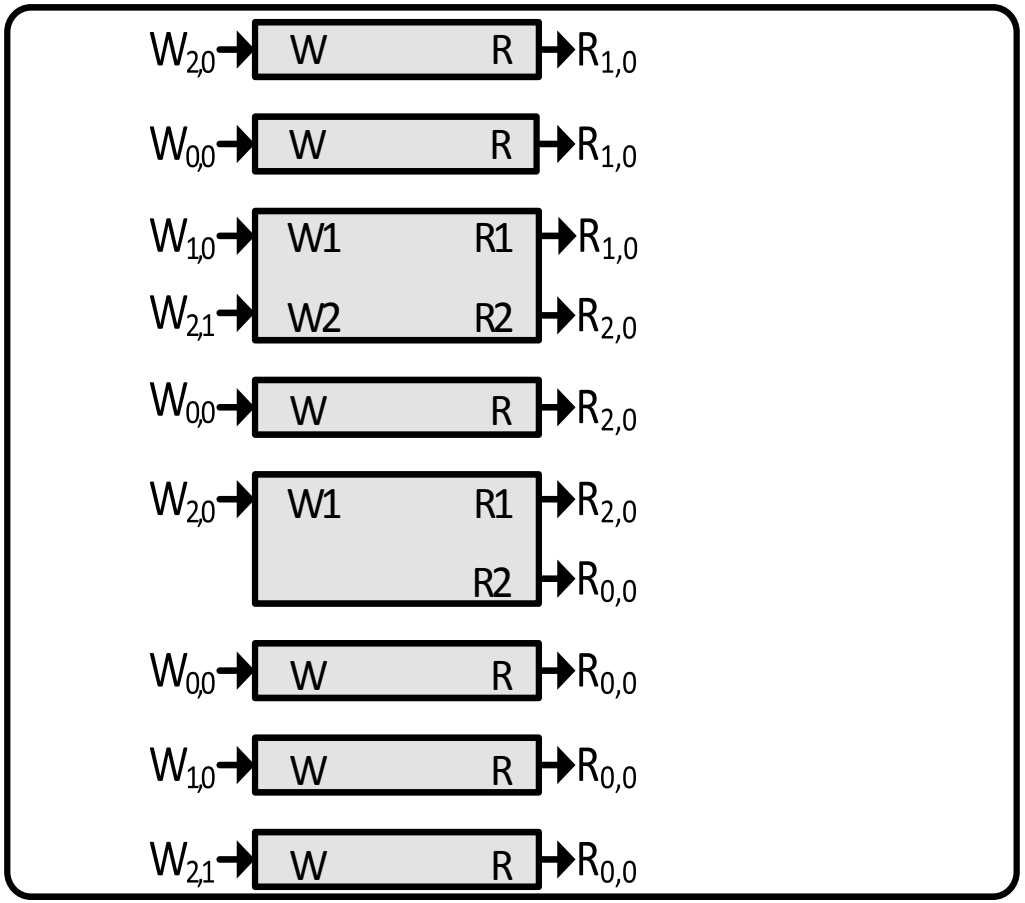


Switched Ports (6)
DFG Covering

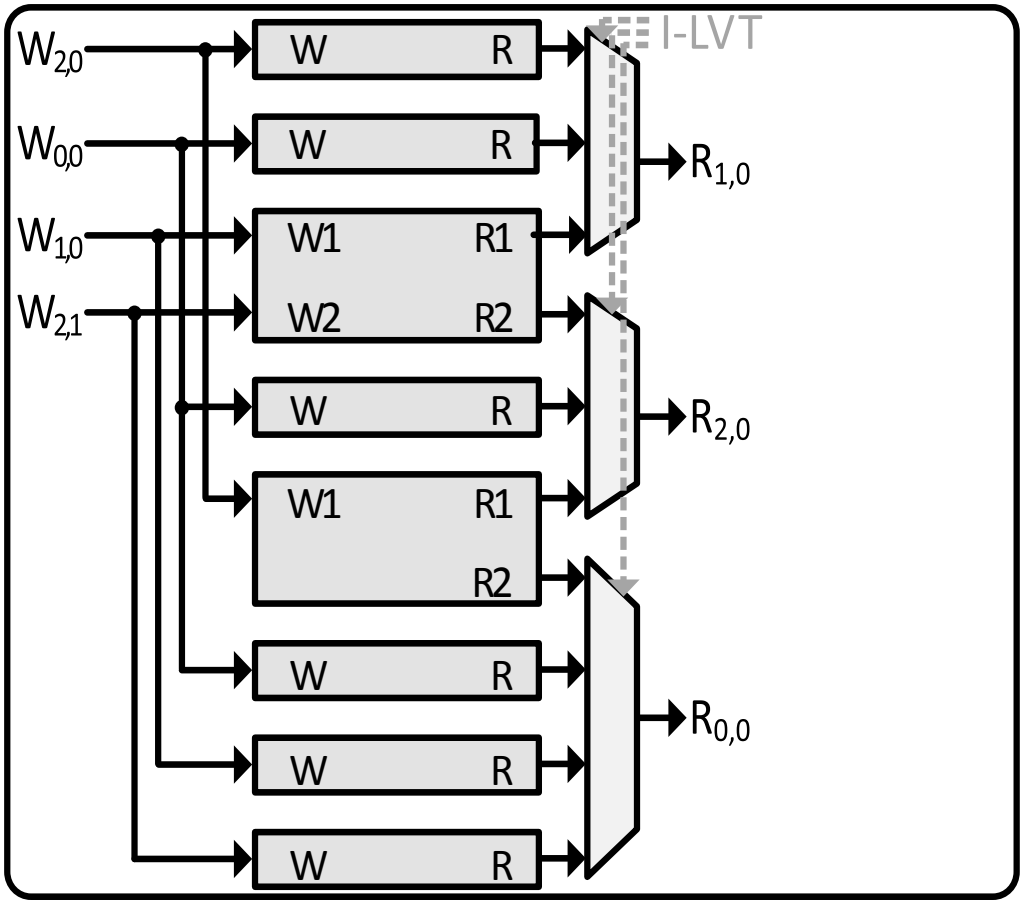




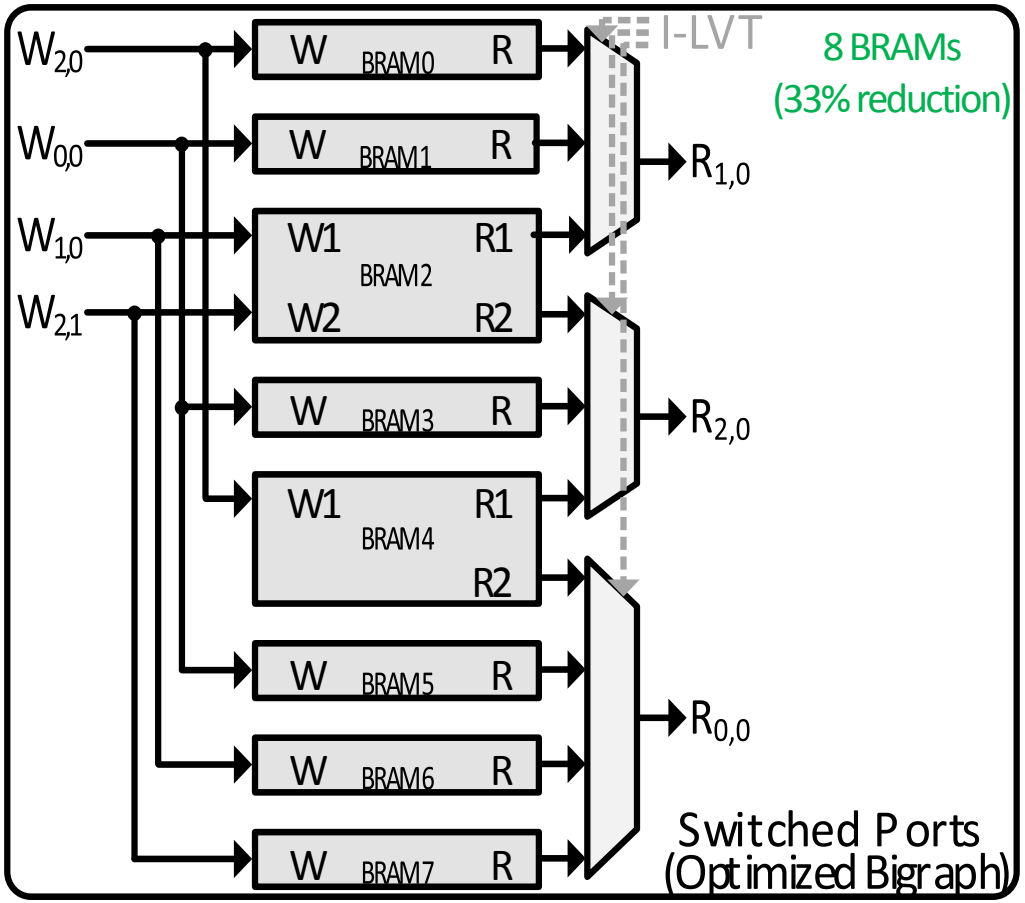
Switched Ports (7)
Switched data banks

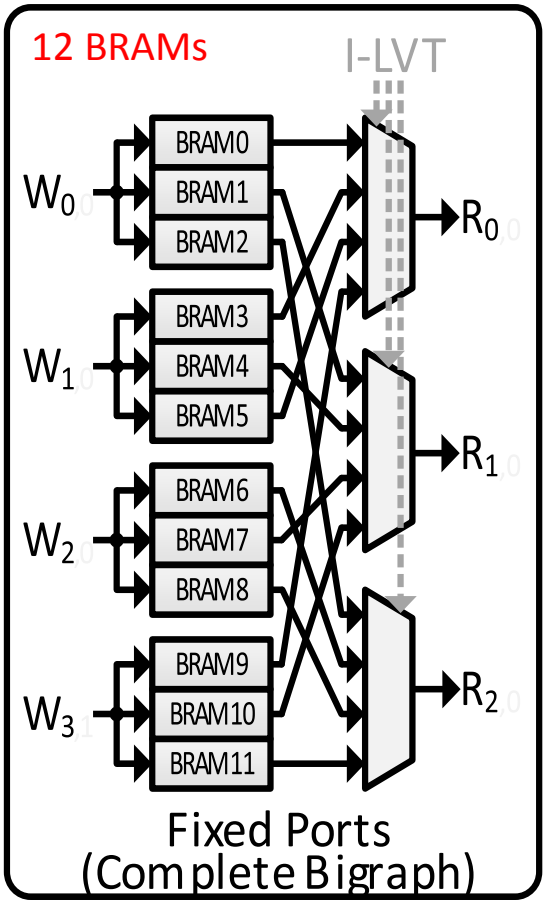


Switched Ports (7)
Switched data banks



Switched Ports (7)
Switched data banks





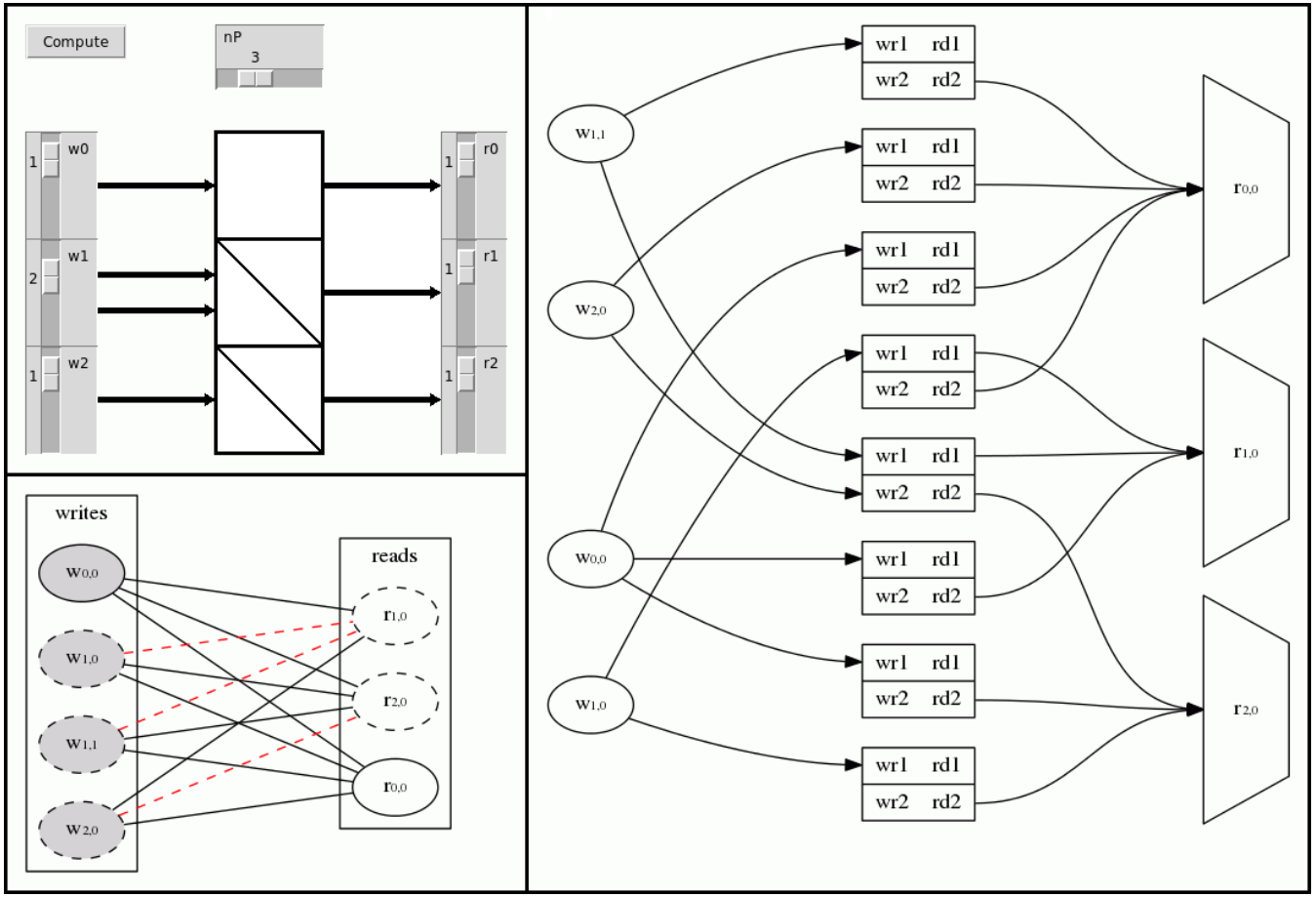
Multi-switched-ports Compiler

- A RAM compiler optimizes data banks construction
  - Generates DFG from port requirements
  - Solves set-covering problem on all edges
    - Covers are predefined biclique patterns
    - Solved as BLP problem
  - Generates Verilog modules based on optimal covering

Supports bypassing (RAW & RDW) and Initialization

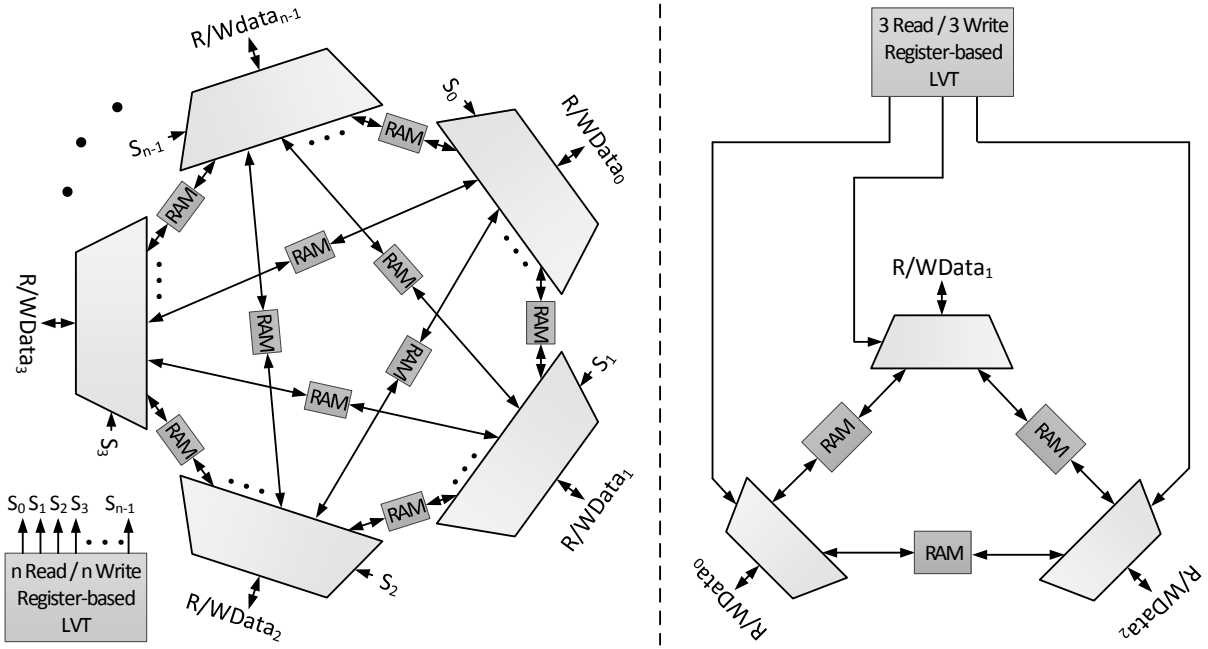
Available as open source contribution
https://github.com/AmeerAbdelhadi
http://www.ece.ubc.ca/~lemieux/downloads/

Graphical User Interface (GUI)



Source of inspiration: Multi-True-Ports by Choi *et al.* / UofT

- Provides true ports only (no simple/fixed ports)
- Is a special case of our generalized approach
- Doesn't need a CAD tools



Experimental Results

- Run-in-batch flow manager
  - Uses Altera's Quartus II for synthesis on Stratix V
  - Uses Altera's ModelSim for verification with:
    - Random vectors
    - Over a million RAM access cycles
- Results on random test-cases
  - Up to 8 switched ports
  - Up to 4 writes and 4 reads per switched port
  - Up to 28 writes/reads per test-case

	Average BRAM Reduction	Average ALMs Reduction	Average Fmax Increase
Best of Previous	18%	-3%	-1%
True Ports	42%	53%	15%

Conclusions

- A methodology to support switched write/read functionality
- True dual-ported BRAMs are utilized to optimize the RAM allocation
- A RAM compiler optimizes the problem
- An additional 18% average BRAM reduction compared to the best of other approaches
- Practical solution:
  - Initialization
  - Bypassing
  - Available as open source

Future Directions

- Applications
  - Parallel computation
  - HLS – storage binding
- Optimization of switched ports port assignment
  - Extraction of mutually-exclusive functions from HDL
- Statistical approach
  - Ports which are mutually-exclusive in most cases can use a switched port
  - Access conflicts will be rare

Thank You!