MAERI: An Open-source Framework for Generating DNN Accelerators with Flexible Dataflow Support

Hyoukjun Kwon*, Michael Pellauer, and Tushar Krishna***

MAERI

*Synergy Lab, Georgia Institute of Technology
** Architecture Research Group, NVIDIA

ISCA 2018

June 3, 2018

Acknowledgement (particular for this session)

- Mr. Charlie Hauck
- Dr. Rishiyur Nikhil

For providing BSV license for hands-on exercises

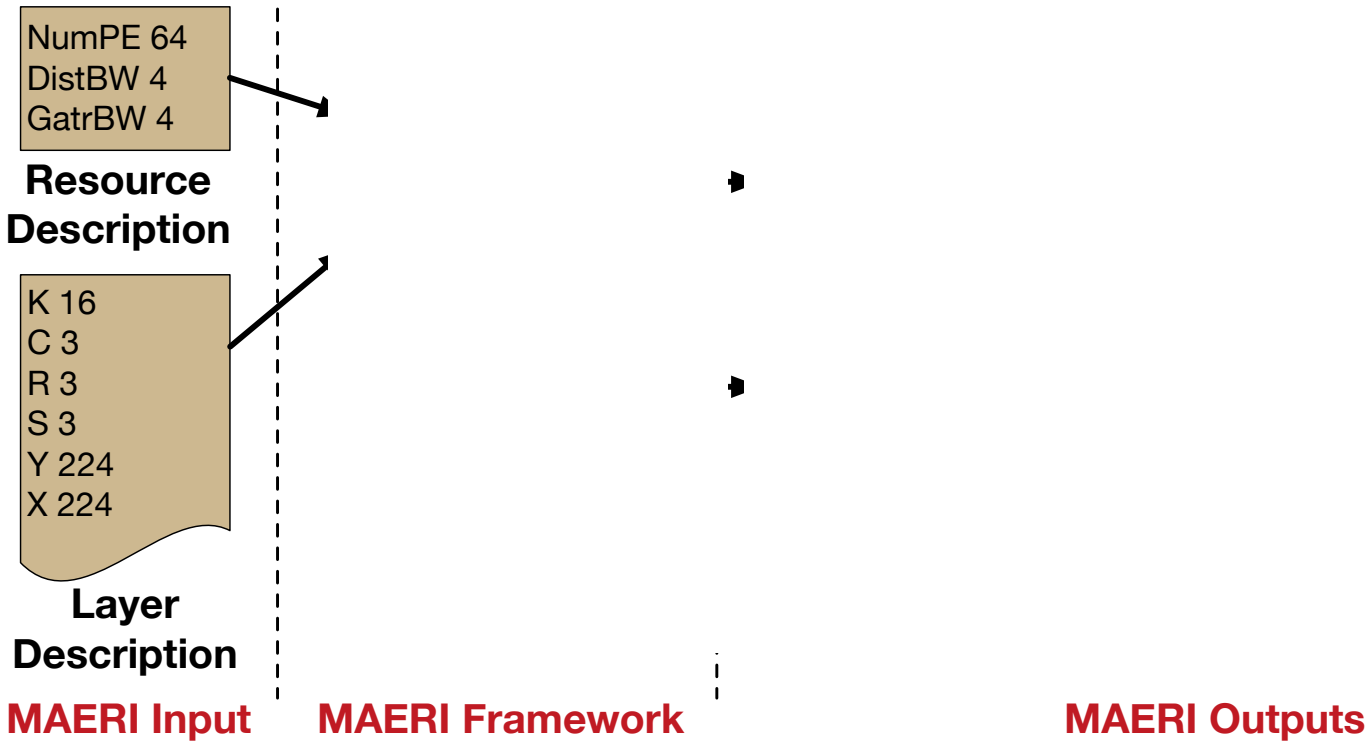
- Ananda Samajdar
- Eric Qin
- Yehowshua Immanuel

For ASIC and FPGA synthesis/PnR

Outline

- **Tool Flow of MAERI**
- **Design**
- **Architecture**
- **Source code and MAERI front-end**
- **Demo**
- **Hands-on Exercises**

Tool Flow of MAERI



Bluespec System Verilog (BSV)

- **A high-level hardware description language**
  - Generates fully synthesizable Verilog

- **Inspired by Haskell and System Verilog**
  - Strong type checking system and polymorphism

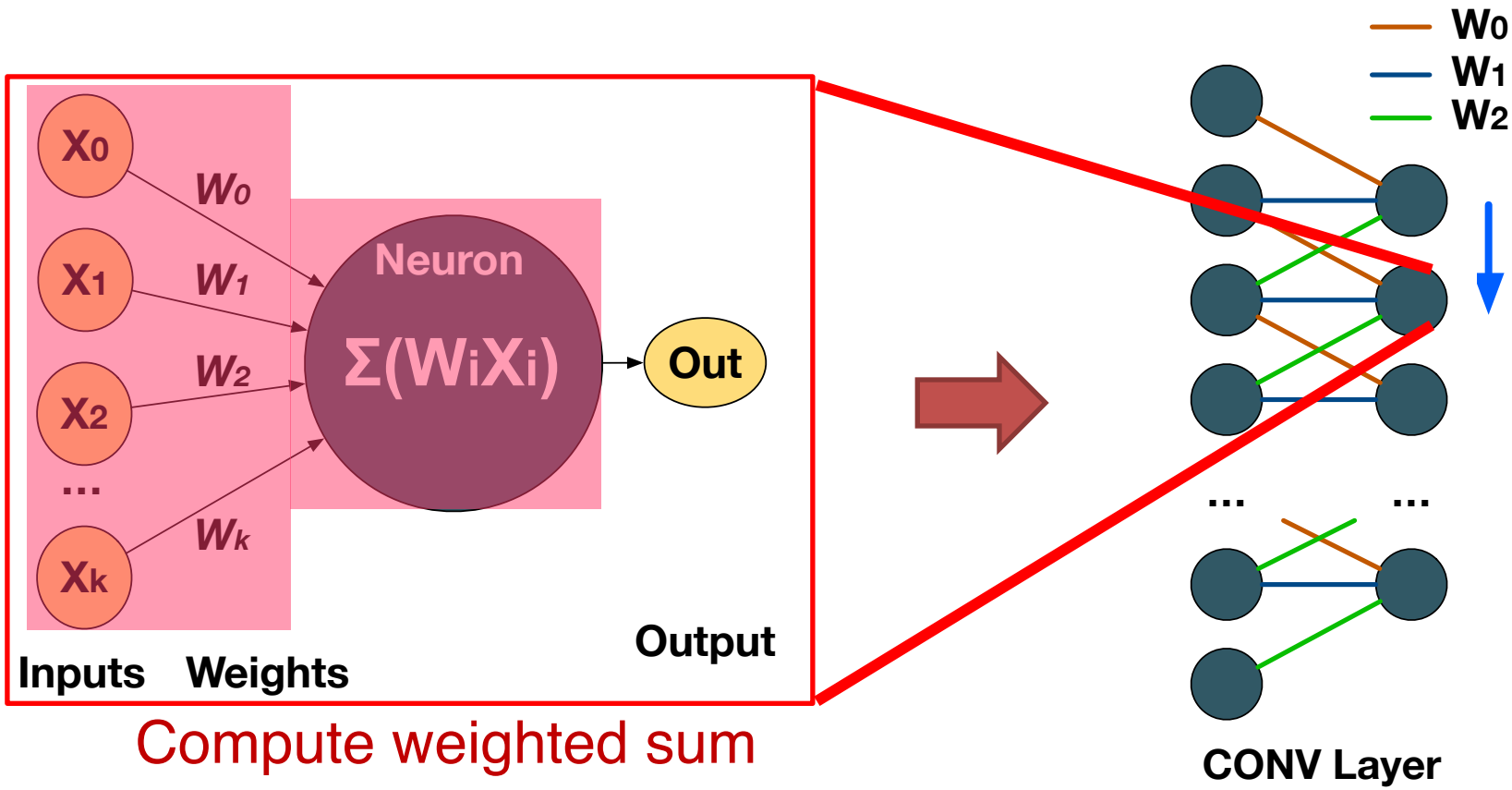
For details, please refer to "BSV by Example" (*http://csg.csail.mit.edu/6.S078/6_S078_2012_www/resources/bsv_by_example.pdf*)

- **Based on "guarded atomic action" blocks**
  - Provides coarse-grained description of parallel actions

Outline

- Tool Flow of MAERI
- Design
- Architecture
- Source code and MAERI front-end
- Demo
- Hands-on Exercises

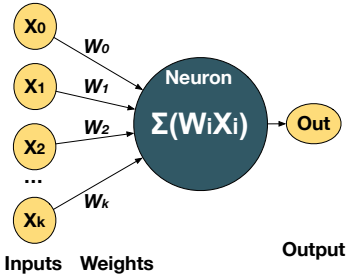
Background: Convolutional Layer in CNN



Compute weighted sum

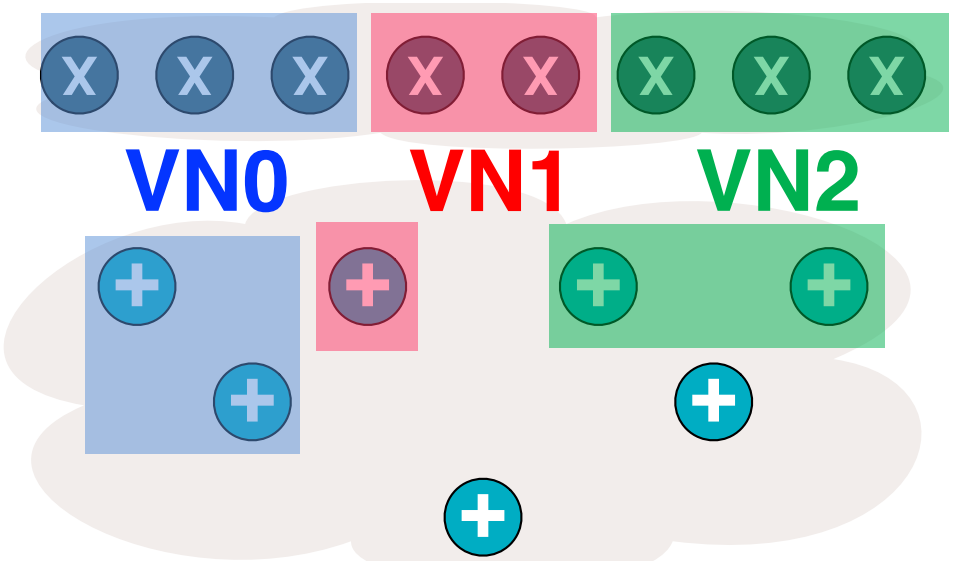
- **Independent multiplication**
- **Accumulation of multiplication results**

Fully Flexible Accelerator



"MultAlloc(3); AddAlloc(2)"

Multiplier Pool



Adder Pool

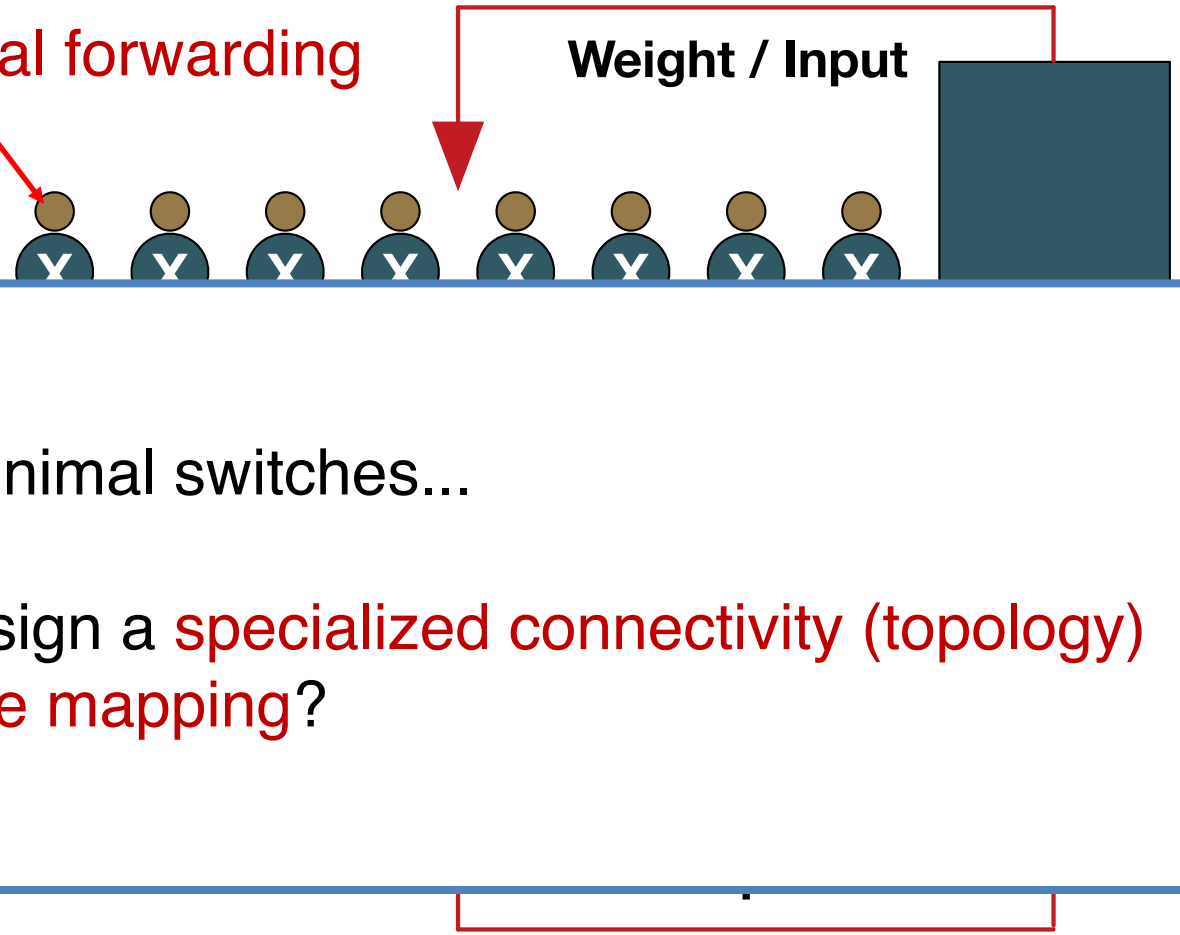
How to enable flexible grouping?

Virtua … for an ou…

Need to ***provide flexible connectivity*** among arbitrary compute units and buffer

Practical Approach: Minimal Switches

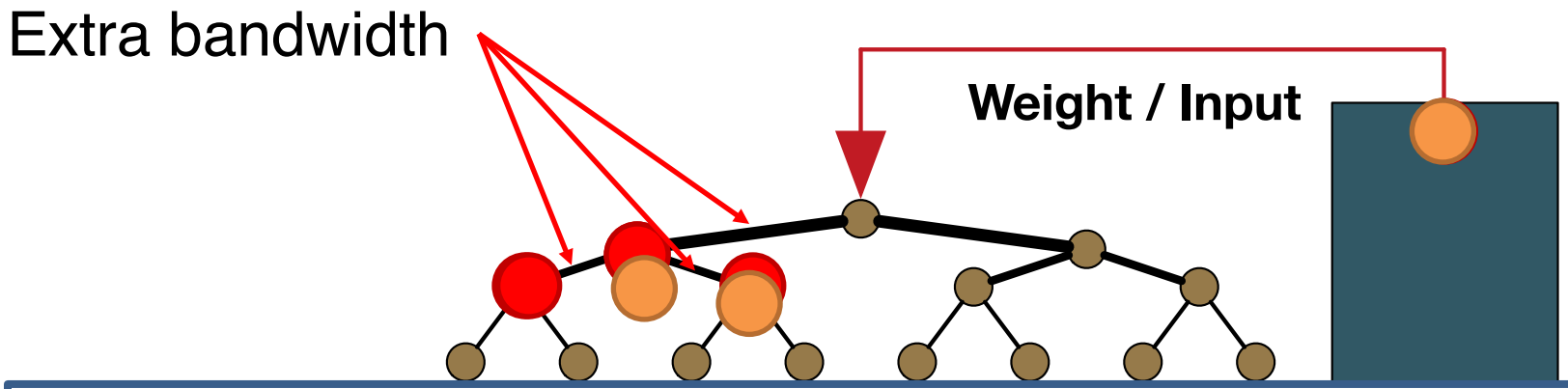
One port for distribution
One port for local forwarding



Using these minimal switches...

How do we design a specialized connectivity (topology) allowing flexible mapping?

Connectivity: Distribution of Weights/Inputs

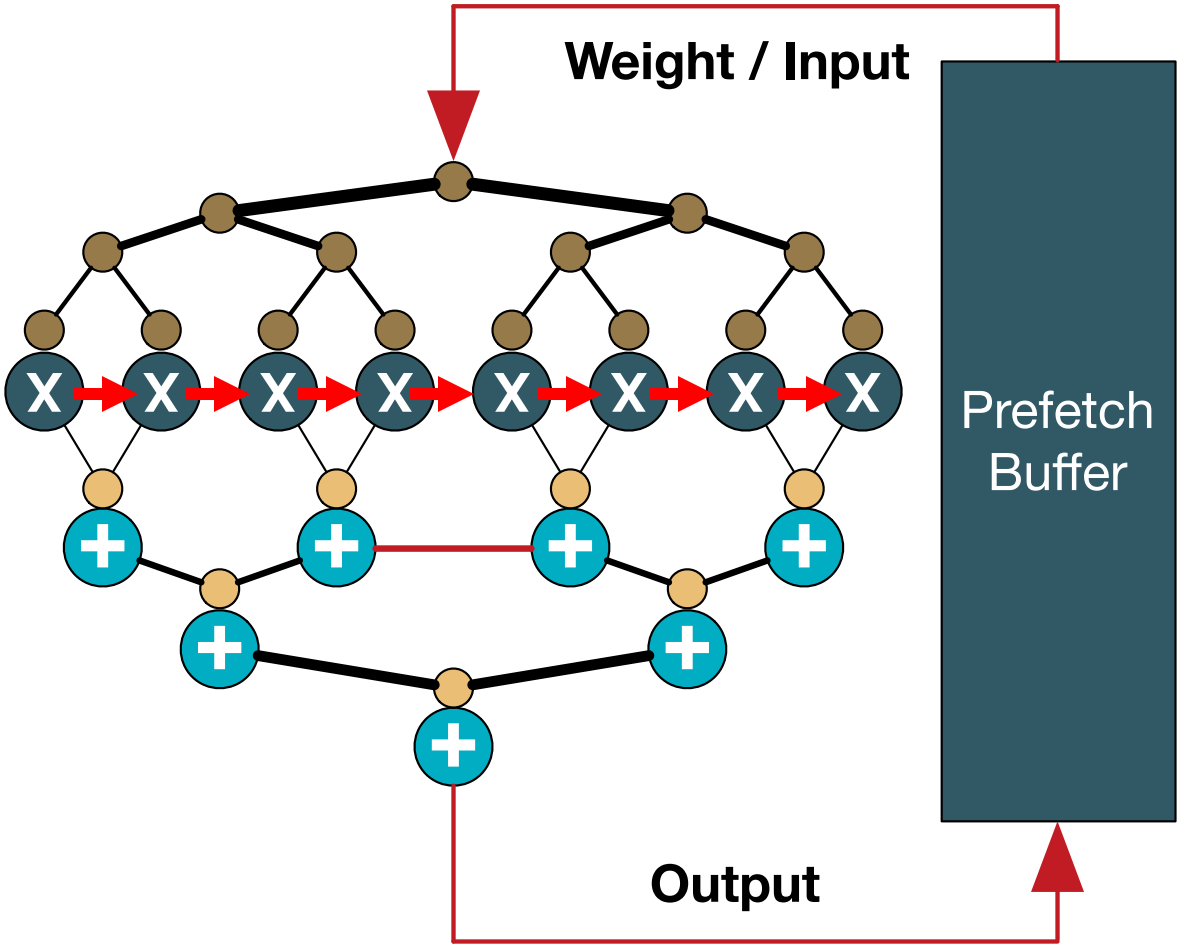


Features

1. **Fat tree-based topology**
   - Supports multicasting for **spatial data reuse**

2. **Extra bandwidth near the top**
   - Provides **high bandwidth** that enables multiple multicasting simultaneously

Connectivity: Local Communication



Enables **spatio-temporal date reuse**

Connectivity: Reduction/Collection of Outputs

Augmented Reduction Tree (ART)

Features

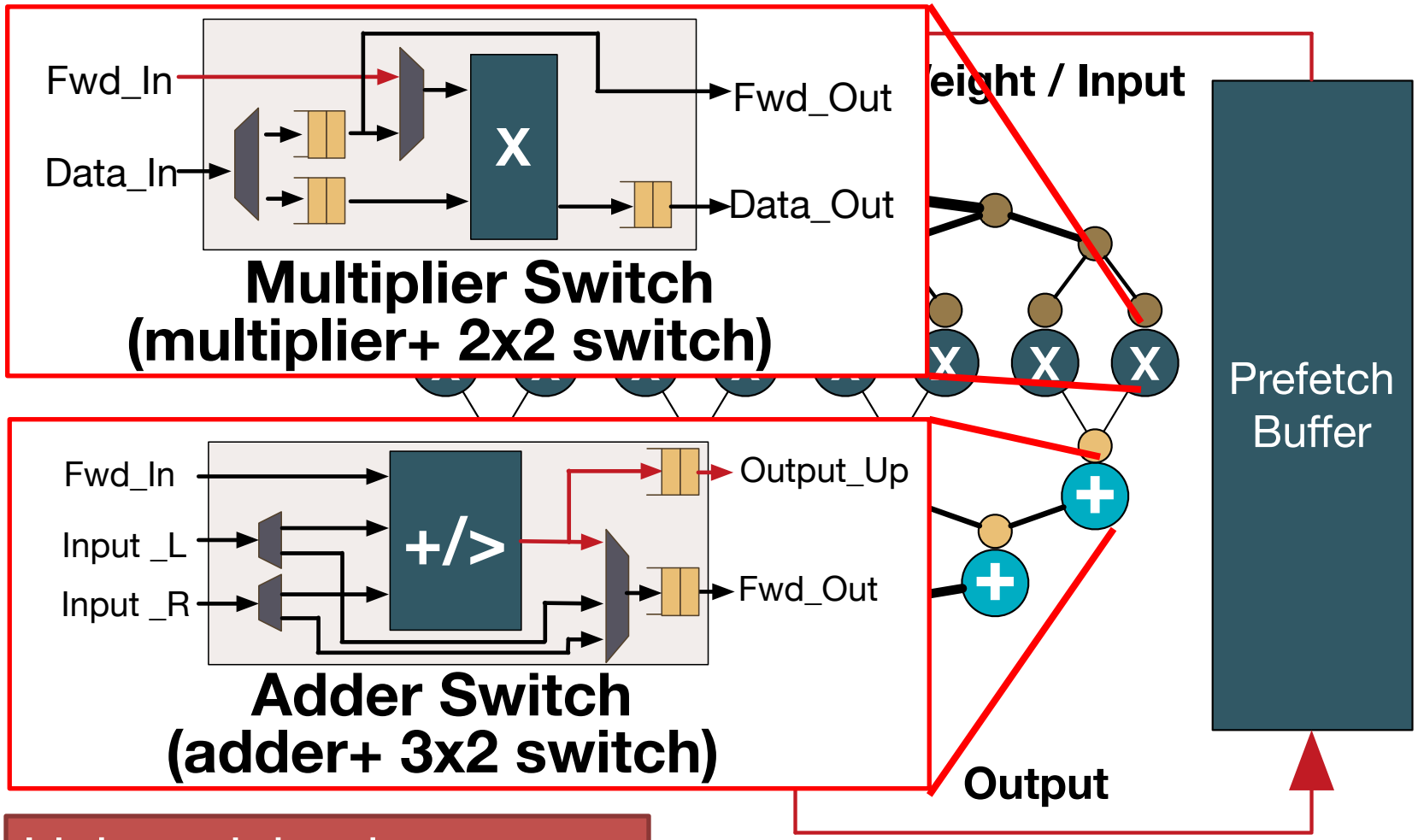
1. **Extra links**
   - Supports **non-blocking in-network reductions** for arbitrary virtual neuron sizes
   - Provides **high compute unit utilization**

2. **Extra bandwidth near the root switch**
   - Provides high bandwidth that enables **multiple reduction (accumulation) simultaneously**

Outline

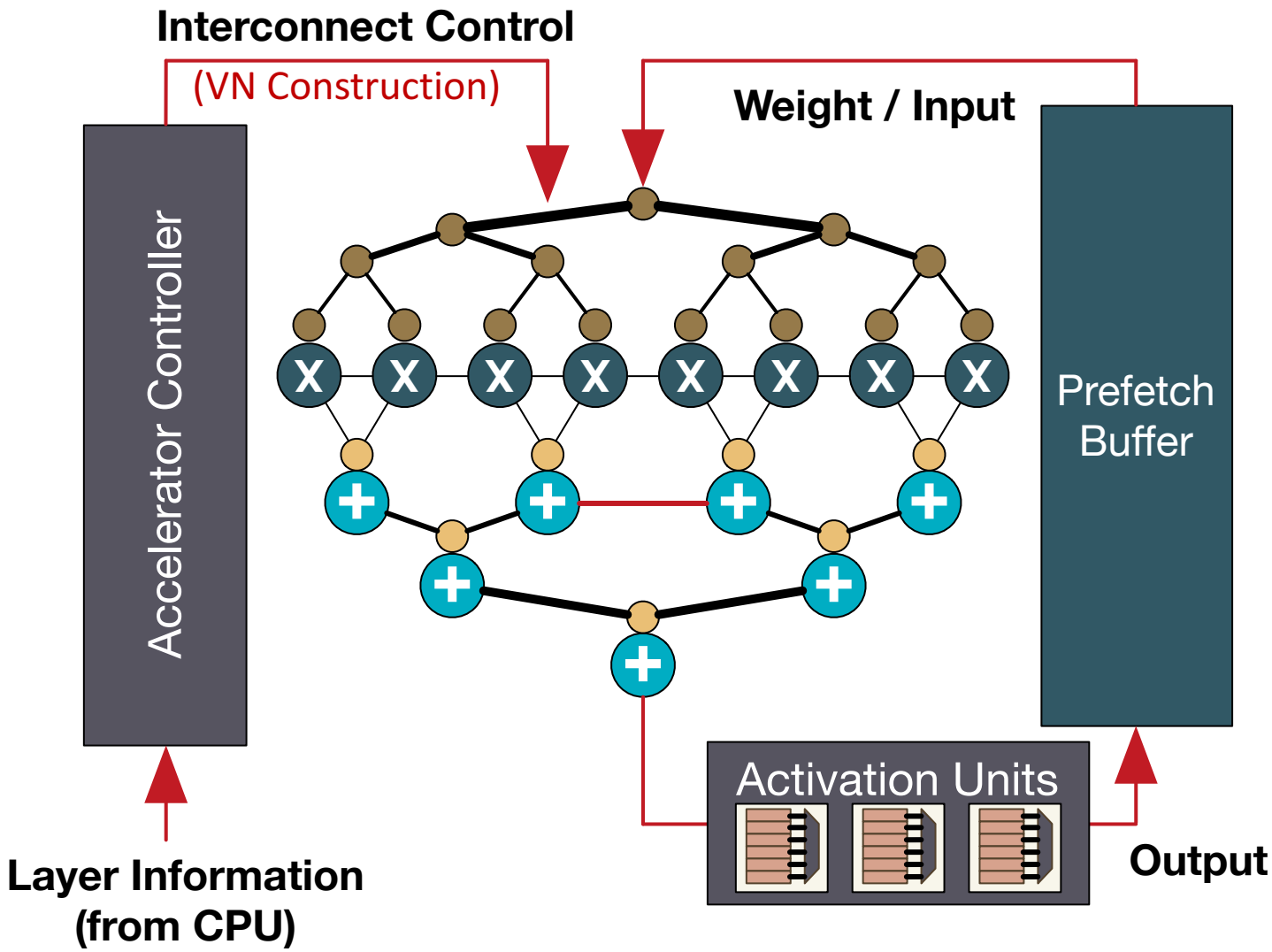
- Tool Flow of MAERI
- Design
- Architecture
- Source code and MAERI front-end
- Demo
- Hands-on Exercises

Switch Microarchitecture



Light-weighted structure

MAERI DNN Accelerator

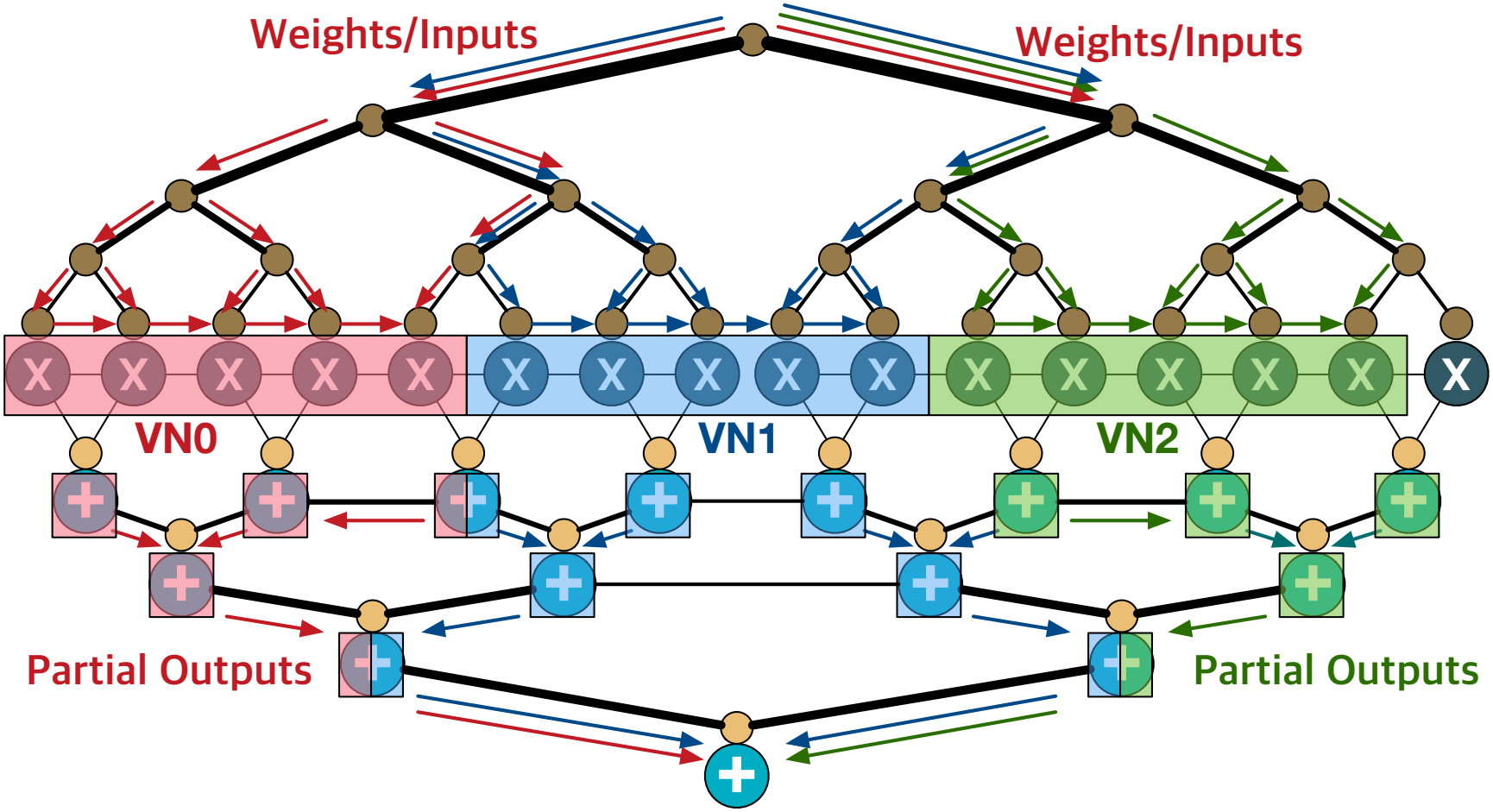


Regular and Irregular Dataflow

- **DNN Topologies**
  - Layer size / shape
  - Layer types: Convolution / Pool / FC / LSTM
  - New sub-structure: e.g., Inception in Googlenet
- **Compiler Optimization**
  - Loop reordering
  - Loop tiling size
  - Cross-layer mapping
- **Algorithmic**
  - Weight pruni

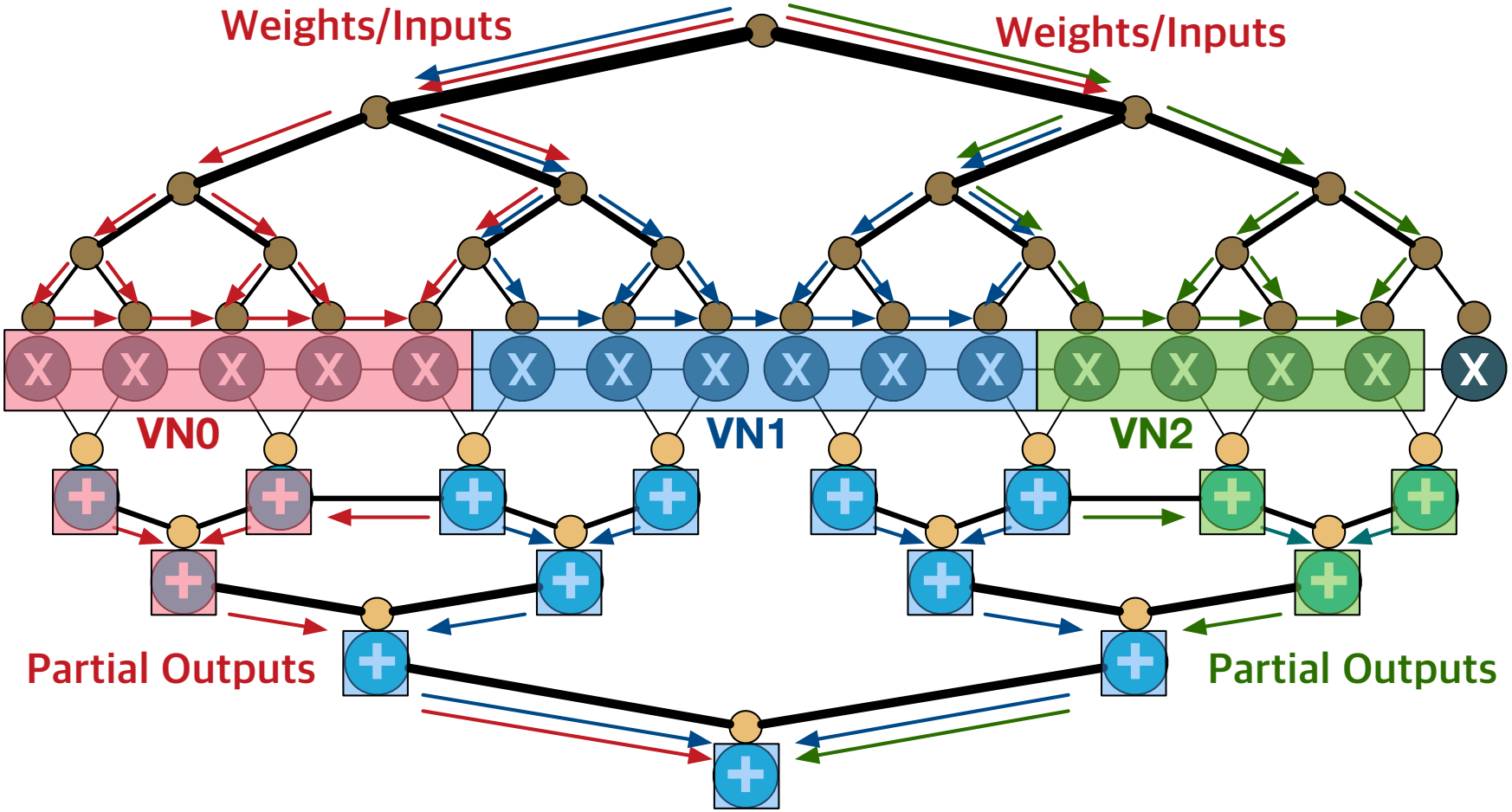
Our Key insight: Each dataflow translates into neurons of different sizes

Dense Dataflow



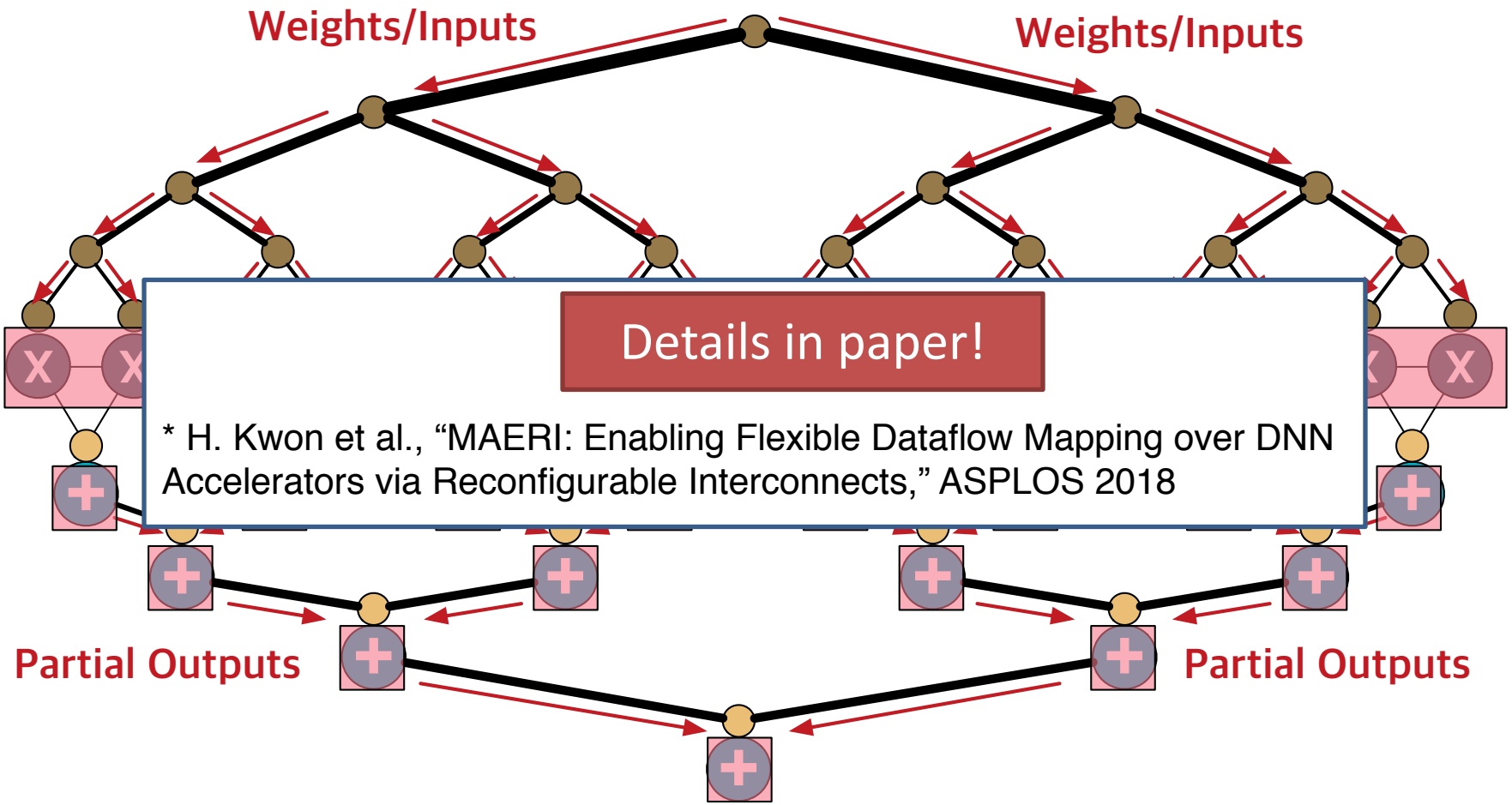
Irregular Dataflow Support Example

- **Sparse Convolutional Layer**



Irregular Dataflow Support Example

- **Fully-connected / LSTM layer**



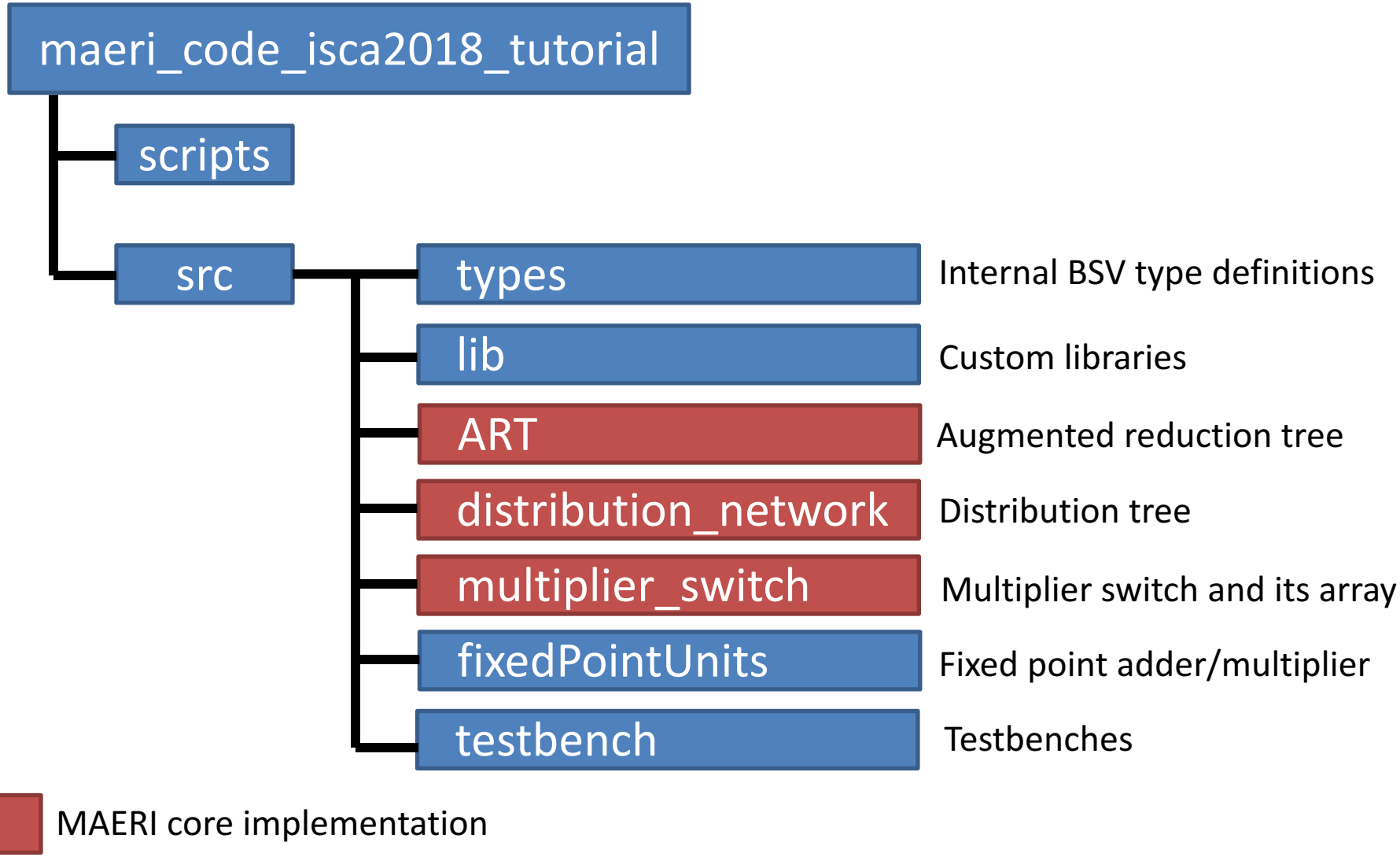
Details in paper!

* H. Kwon et al., "MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects," ASPLOS 2018

Outline

- Tool Flow of MAERI
- Design
- Architecture
- Source code and MAERI front-end
- Demo
- Hands-on Exercises

Source code directory structure



How to use MAERI front-end

- **Configuration change**
  - Modify AcceleratorConfig.bsv at the top directory
    - Distribution bandwidth
    - Reduction bandwidth
    - Number of multiplier switches

- **Verilog code generation and simulation**
  - **./Maeri –c :** Compile a simulation
  - **./Maeri –r :** Run compiled simulation
  - **./Maeri –w :** Launch GTKwave for waveform analysis
  - **./Maeri –v :** Generate Verilog code
  - **./Maeri –clean :** Clean up intermediate files

How to use MAERI front-end

- **Simulation results example**
  - Triggered with “./Maeri –r” after “./Maeri –c”

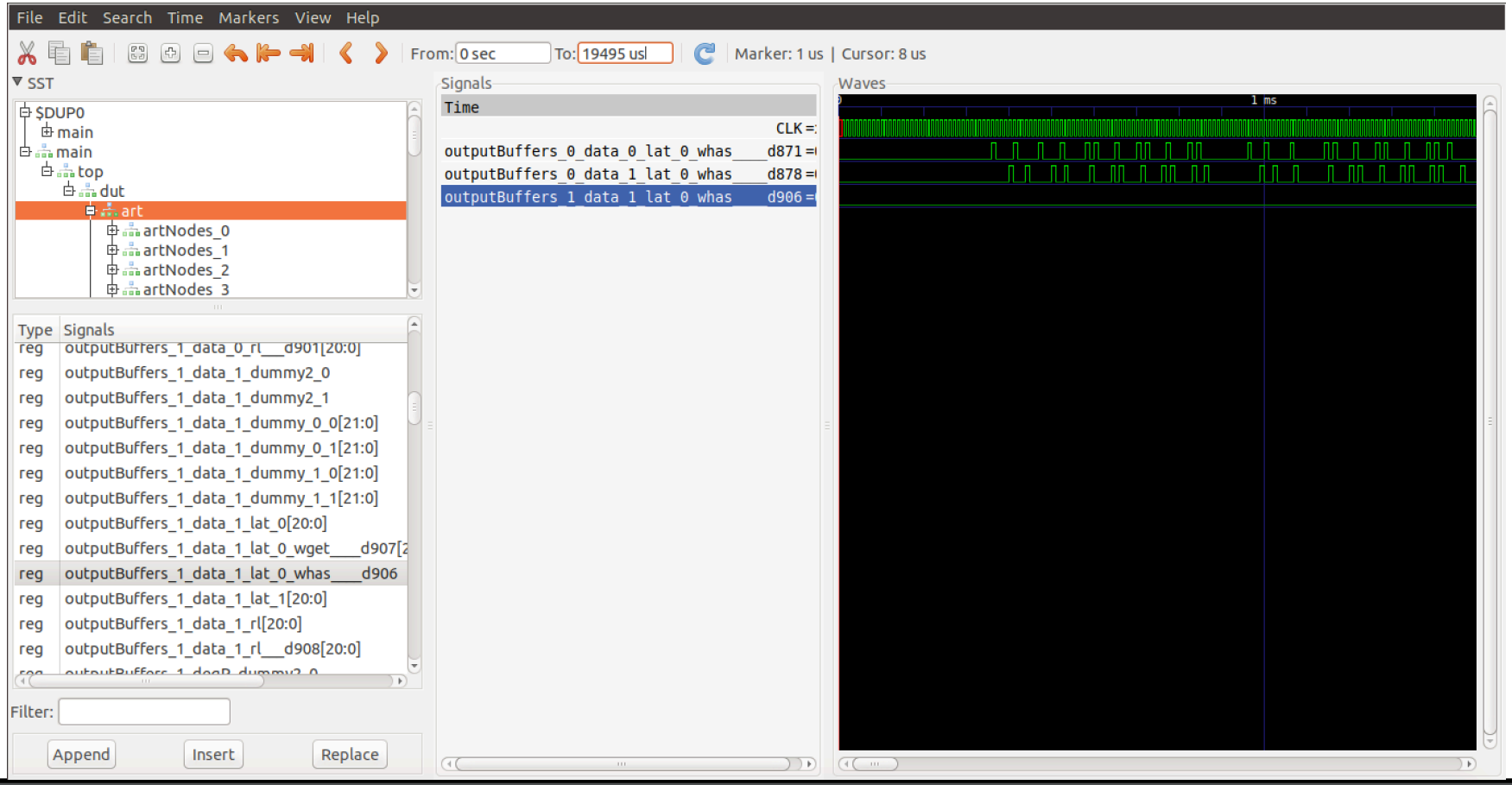
```
isca2018-maeri@isca2018maeri-VirtualBox:~/ISCA2018/MAERI$ ./Maeri -r
-----------------------------
@          0, newSetting: VN size =            9
@ Cycle        1208: Testbench terminates
 Layer dimension K =            6, C =            6, R =            3, S =
 Output dimension:            6 x            3 x            3

Number of injected weights:        324
Number of injected inputs:       1620
Number of injected unique inputs:        150
Number of input multicasting:        540
Number of generated partial sums:       2916
Number of performed MACs:       5508

Total runtime (assuming 1GHz clock):       1208 ns
```

How to use MAERI front-end

- **Waveform Analysis**
  - Triggered with “./Maeri –w” after “./Maeri –r”



How to use MAERI front-end

- **Verilog generation example**
  - Triggered with "./Maeri –v"

```
mkART.v                 mkLIFixedMultiplier.v      mkPlainTreeController.v
mkARTSelector.v         mkMAERI_Top.v              mkPlainTreeNode.v
mkAdderSwitch.v         mkMultiplierSwitch.v       mkSCFixedAdder.v
mkDistributionTree.v    mkMultiplierSwitchArray.v  mkSCFixedMultiplier.v
mkLIFixedAdder.v        mkPlainTree.v
```

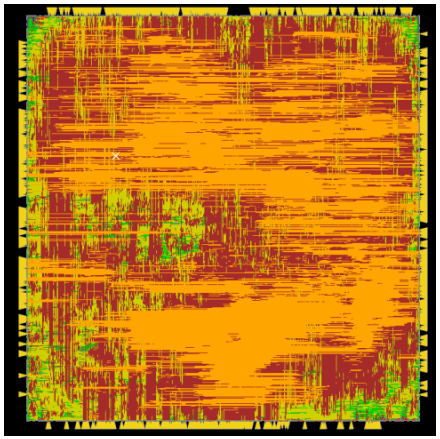
* Verilog files are generated in "(Top_Directory)/Verilog"

Generated Verilog code is synthesizable!

MAERI Synthesis/PnR

- **Synthesis/PnR Environment**
  - **Technology:** TSMC 28nm
  - **Clock frequency:** 1GHz
  - **Design:** 64 multiplier switches and 31 adder switches
  - **Distribution Bandwidth:** 16/4/1 data per cycle
  - **Gather Bandwidth:** 16/4/1 data per cycle
  - **RTL Code:** Verilog generated using MAERI code base
  - **CAD Tool Chain:** Synopsys Design compiler, Cadence Innovus, Primepower

Post-layout Area and Power



Number of PEs	Distribution BW	Reduction BW	Area (mm^2)	Power (mW)
64	16X	4X	0.440	407.861
64	4X	16X	0.255	252.326
64	4X	1X	0.219	194.912
64	1X	4X	0.239	254.603

*** Based on 28nm technology and 1GHz clock**

Outline

- Tool Flow of MAERI
- Design
- Architecture
- Source code Structure
- Using MAERI source code base
- Demo
- Hands-on Exercises

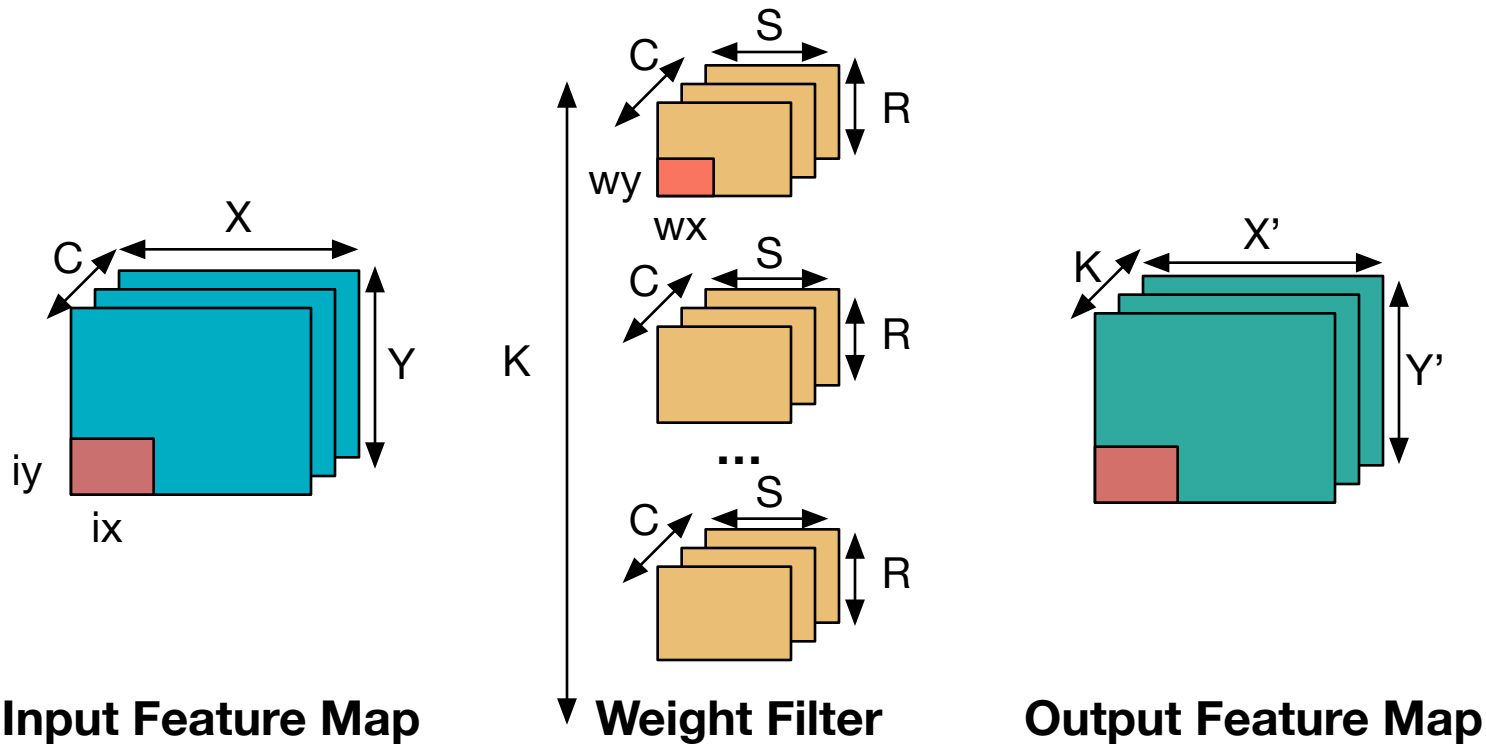
Demo

- Launching cycle-accurate simulations
- Modifying user configuration
- Compiling simulations
- Launching wave form analysis
- Generating Verilog files

Outline

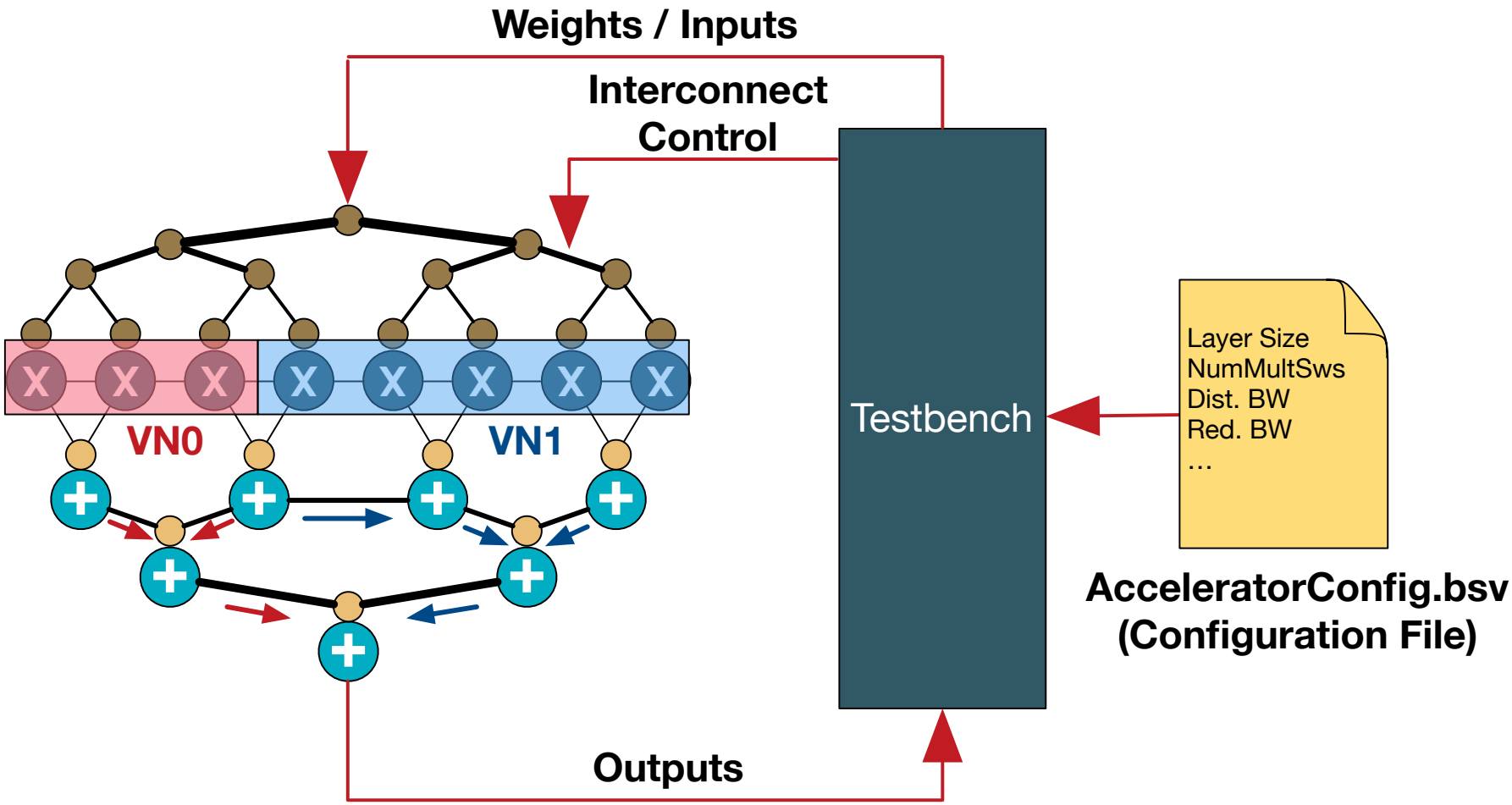
- Tool Flow of MAERI
- Design
- Architecture
- Source code Structure
- Using MAERI source code base
- Demo
- Hands-on Exercises

Revisiting Convention



- K/C: Output/Input Channel
- Y/X: Input activation height/width
- R/S: Weight filter height/width

Testbench Structure



Testbench Dataflow (MAESTRO description)

let sWindowSz = sizof(R) x sizeof(S)

let numVNs = floor(NumMultSwitches / sWindowSz)

- **Temporal_Map** (1, 1) C
- **Spatial_Map** (1, 1) K
- **Temporal_Map** (1, 1) Y
- **Temporal_Map**(sWindowSz, 1) X
- **Tile**(sWindowSz)
- **Unroll** R
- **Unroll** S

High weight filter parallelism

- **Exercise#1:** Compile a simulation with default, early, and late layers with 32 PEs (“./Maeri –c,” “-/Maeri –ctarg EARLY_SYNTHETIC,” “./Maeri –ctarg LATE_SYNTHETIC”). Run simulation and compare results.

- **Exercise#2:** Compile a simulation with 32 and 64 PEs using default seting (“./Maeri –c”). Run simulation and compare results

- **Exercise#3:** Compile a simulation with 2X/4X/and 8X distribution bandwidth (fix reduction bandwidth as 4X). Run simulation and compare results.

- **Exercise#4:** Compile a simulation with 2X/4X/and 8X reduction bandwidth (fix distribution bandwidth as 4X). Run simulation and compare results.