Global Buffer Power Optimization for Onyx

Taeyoung Kong

Amber Architecture Diagram

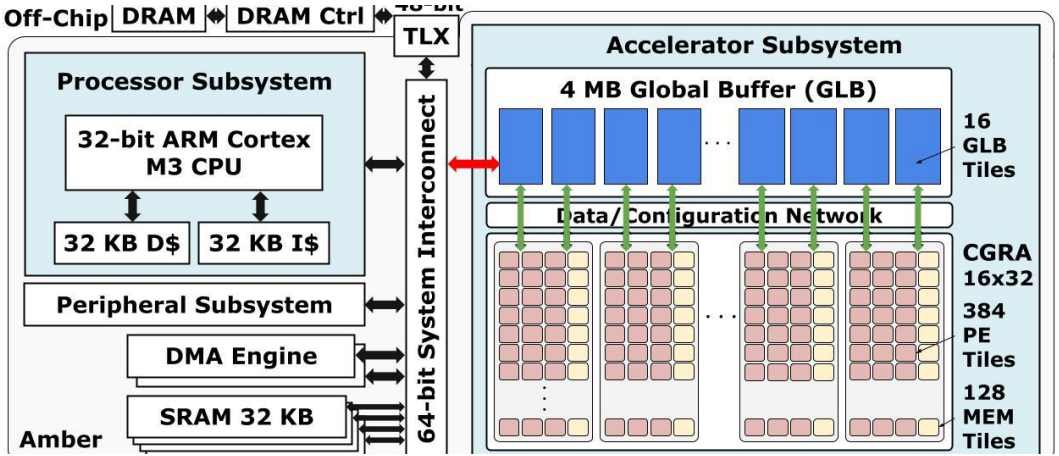


Fig. Amber architecture block diagram

1. GLB communicates with processor through AXI interface (red)
2. There are 16 GLB Tiles (blue)
3. Each GLB Tile communicates with CGRA (green)

Amber - Power Breakdown by Blocks

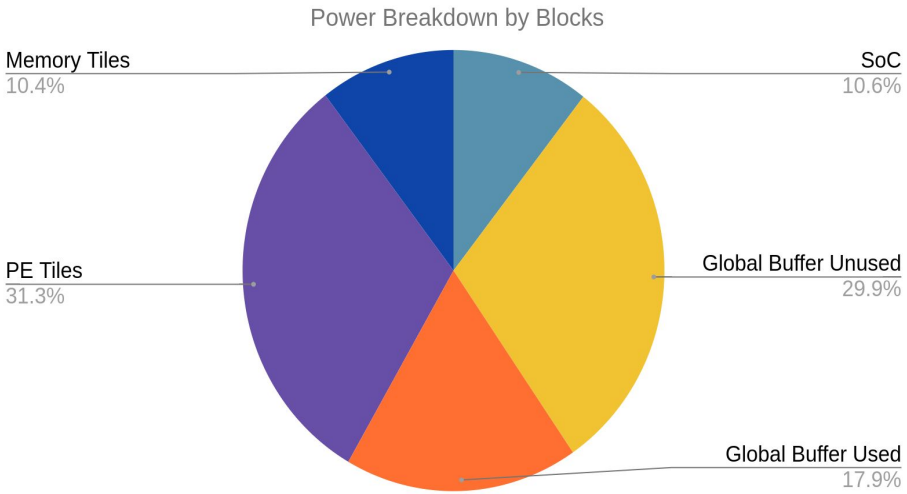


Fig. Power breakdown from Harris+Demosaic simulation at 1GHz.
(No SDF. Averaged mode.)

1. GLB consumes 47.8% of total power
   a. Used: 17.9%
   b. Unused: 29.9%
2. Problem
   a. High power consumption of unused GLB tiles (due to limited clock-gating control)

=> Why?

GLB Tile Architecture

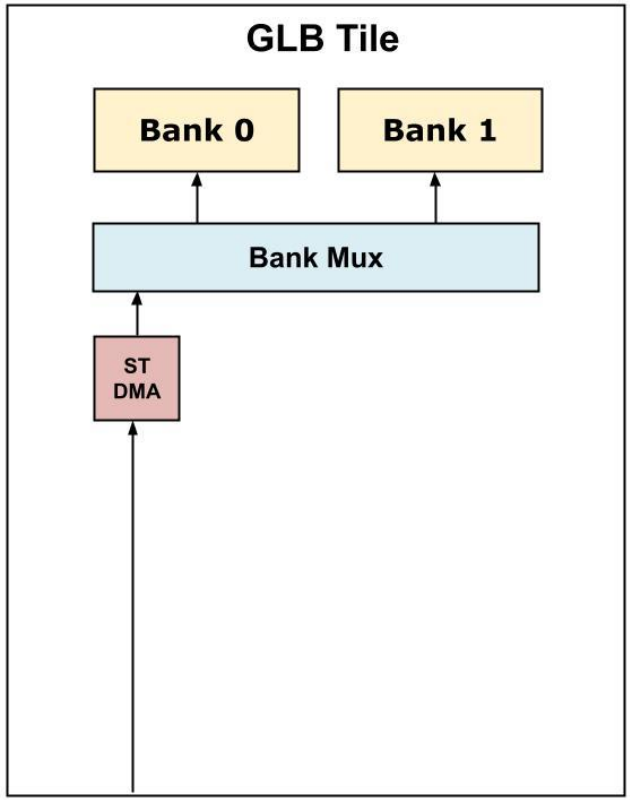


Fig. Glb-tile microarchitecture

1. **Two banks, Bank Mux, ST DMA**

GLB Tile Architecture

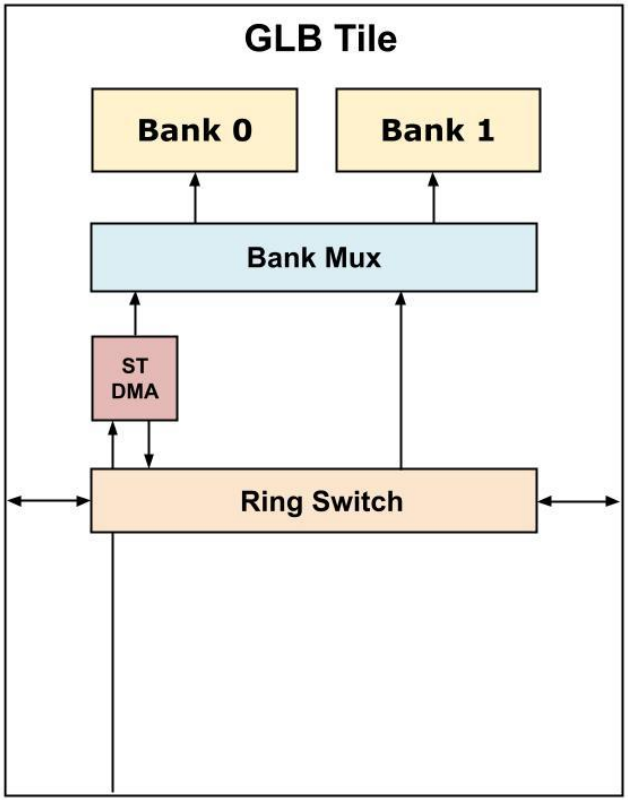


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. **+ Ring switch**

GLB Tile Architecture

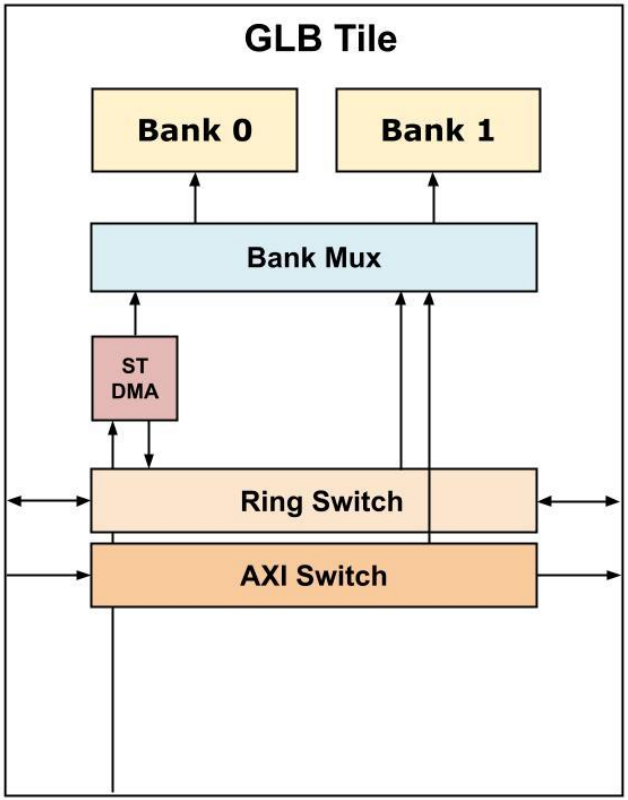


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. **+ AXI switch**

GLB Tile Architecture

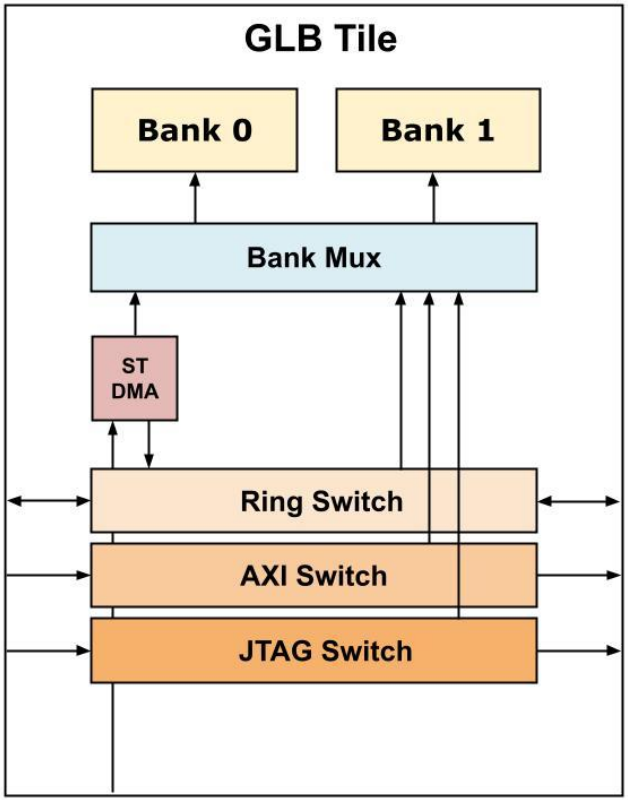


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. + AXI switch
4. **+ JTAG Switch**

GLB Tile Architecture

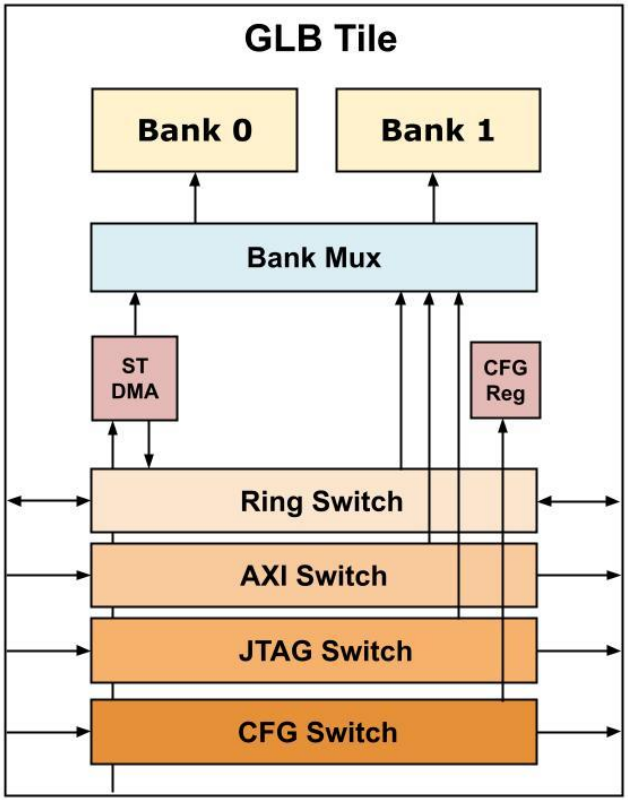


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. + AXI switch
4. + JTAG Switch
5. **+ CFG Switch, CFG Registers**

GLB Tile Architecture

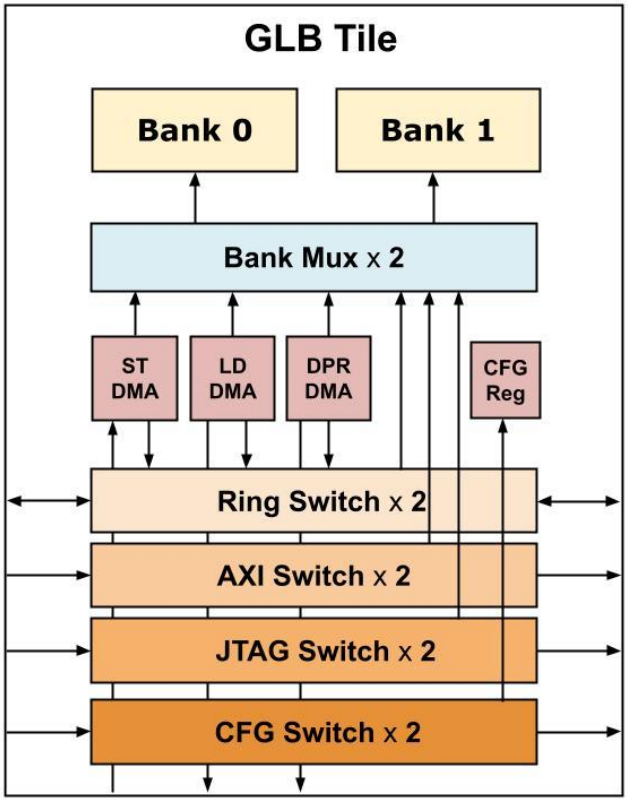


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. + AXI switch
4. + JTAG Switch
5. + CFG Switch, CFG Registers
6. **(1-5) x 2 (Read Channel & LD DMA)**

GLB Tile Architecture

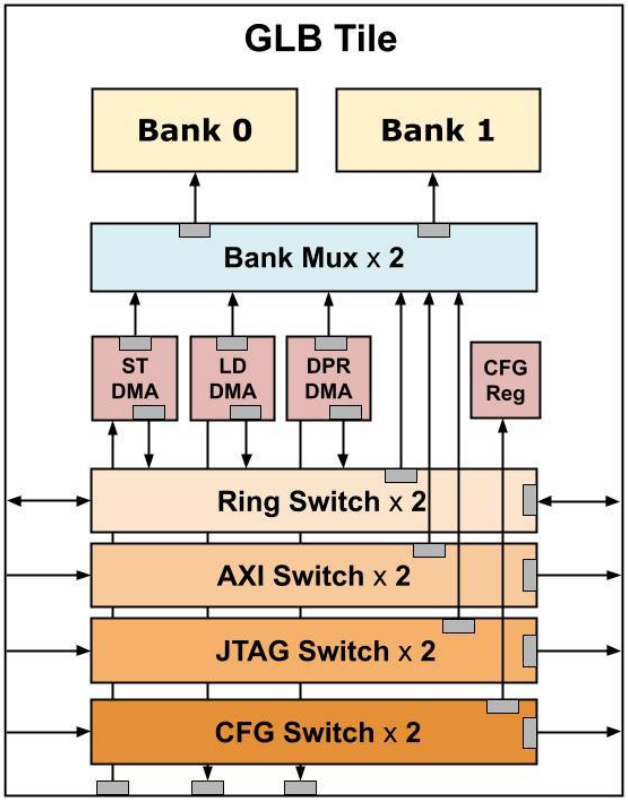


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. + AXI switch
4. + JTAG Switch
5. + CFG Switch, CFG Registers
6. (1-5) x 2 (Read Channel & LD DMA)
7. **Pipeline Registers**

GLB Tile Architecture

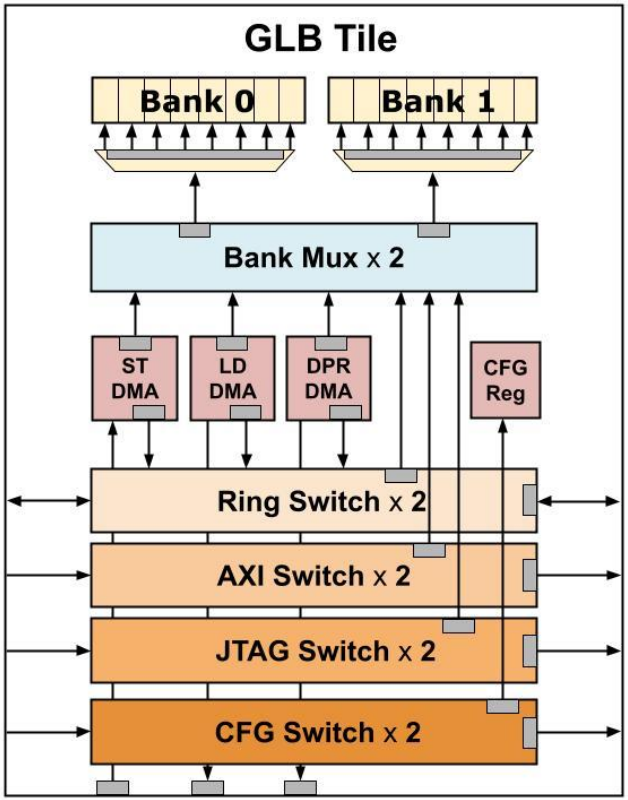


Fig. Glb-tile microarchitecture

1. Two banks, Bank Mux, ST DMA
2. + Ring switch
3. + AXI switch
4. + JTAG Switch
5. + CFG Switch, CFG Registers
6. (1-5) x 2 (Read Channel & LD DMA)
7. Pipeline Registers
8. **SRAM macro Demux**

Amber - GLB Tile Clock Gating

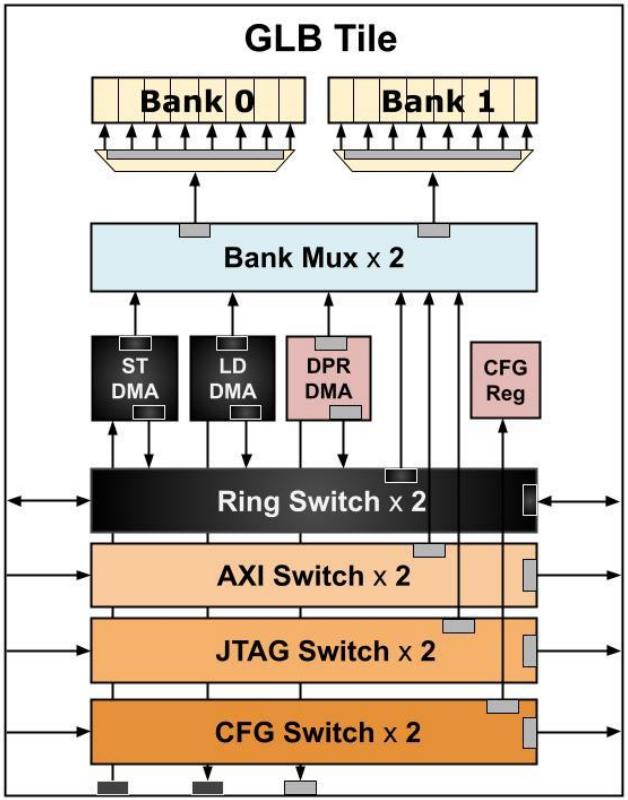


Fig. Glb-tile clock-gated modules

Amber - GLB Tile Clock Gating

1. One ***stall*** signal clock gates ST DMA, LD DMA, and Ring Switches
2. Only stall (clock-gate) blocks that are used to communicate with CGRA (breakpoint)

Problems with Amber clock-gating / power

1. Many blocks are not clock-gated
2. One stall signal clock-gates DMAs + Ring switch
3. Too many pipeline registers are used

Onyx Goal - GLB Power Improvement (How?)

1. One stall signal clock-gates DMAs + Ring switch
    a. ***Fine-grain clock-gating*** to control each block
2. Too many pipeline registers are used
    a. ***Optimize*** pipeline registers and microarchitecture
3. Many blocks are not clock-gated
    a. ***Dynamic clock-gating*** when a block is not used

Why not power gating?

1. Too many blocks
2. For an unused GLB Tile, only 0.65% of the total power is leakage power (Amber)
3. Physical design difficulty

Experiment Setup - Workload

- Double Buffering
  - Processor/CGRA writes/reads **32KB block** to/from GLB Tile 1 and GLB Tile 2
  - Widely used workload in many applications to hide memory latency
  - Good benchmark to show benefits of fine-grain clock gating

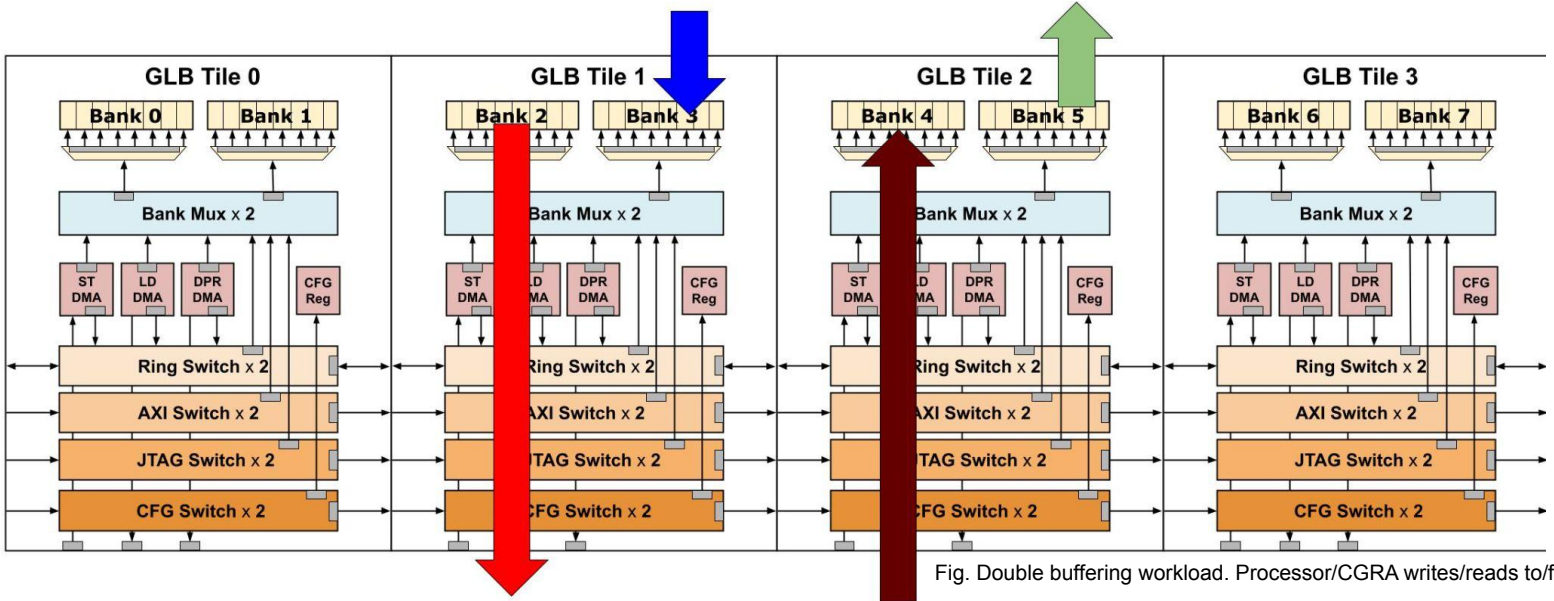


Fig. Double buffering workload. Processor/CGRA writes/reads to/from bank2 - bank5

Experiment Setup - Workload

- Double Buffering
  - Tile 0 - AXI Switch x 2
  - Tile 1 - AXI Switch x 2, LD DMA, Bank Mux, Bank x 2
  - Tile 2 - AXI Switch, ST DMA, Bank Mux, Bank x 2
  - Tile 3 - None

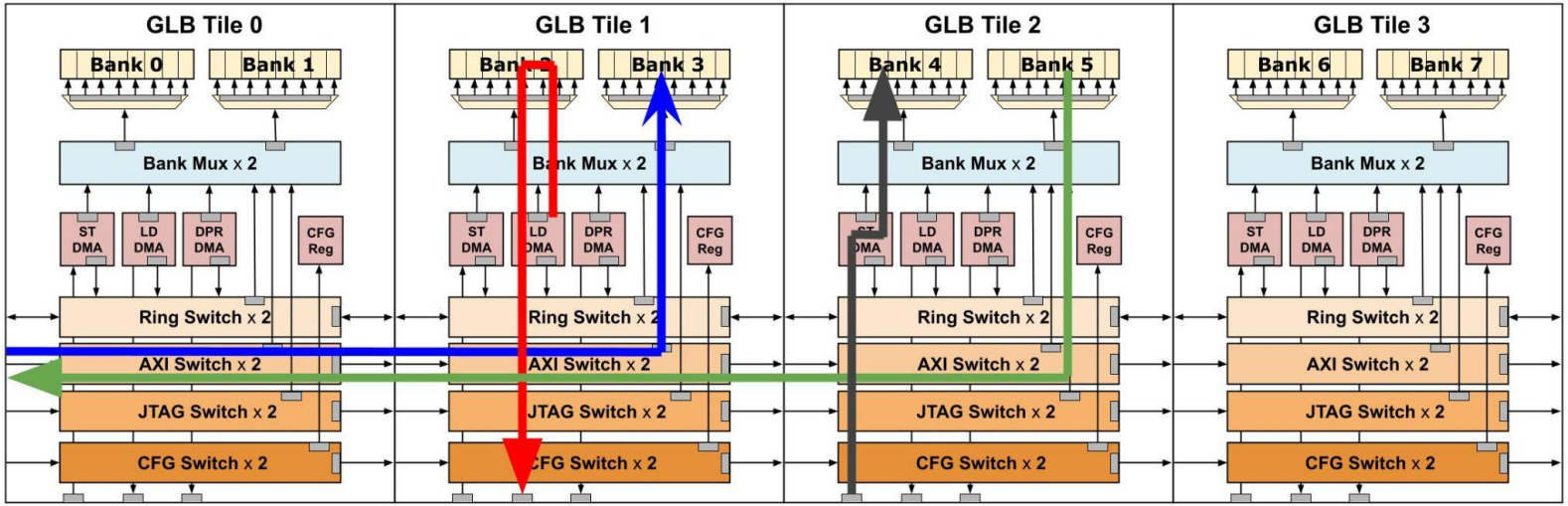


Fig. Each arrow shows how transactions actually flow in GLB

Experiment Setup - Methodology

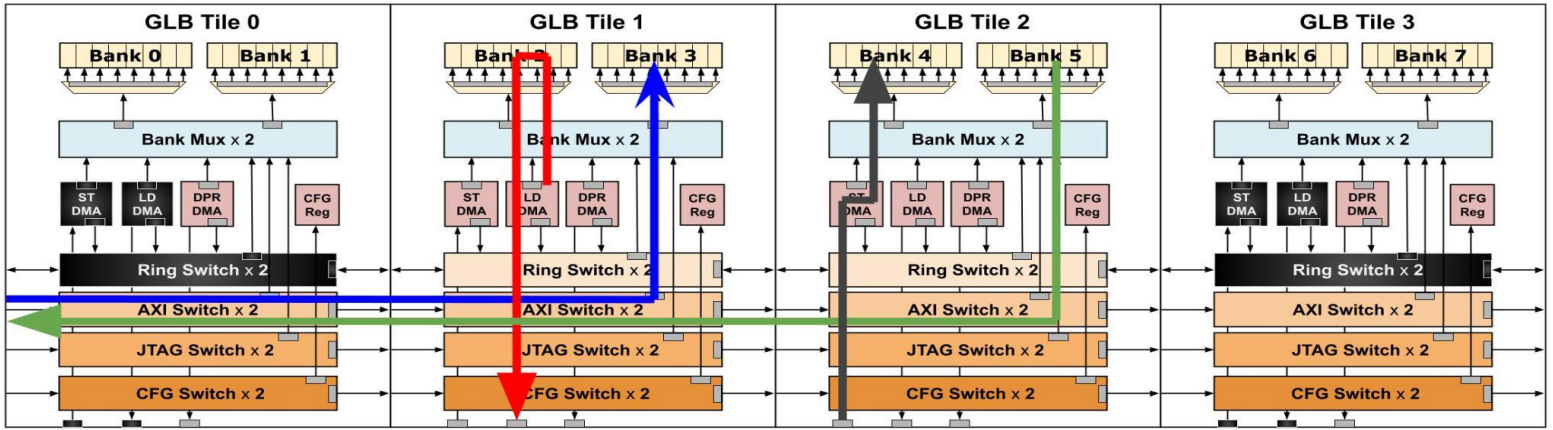
Technology	TSMC 16nm
PVT	TT / 0.8V / 25C
Frequency	700MHz
Tool	PrimeTime - Q-2019.12
Power analysis mode	Time-based
SDF	SDF annotated
Top design	global_buffer
Test	test3_axi_wr_3_axi_rd_5_cgra_ld_2_cgra_st_4

Table. GLB power experiment setup

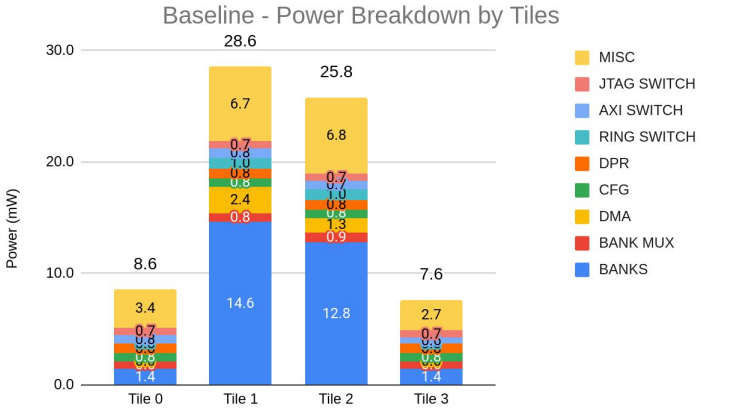
Baseline - Amber version

- One ***stall*** signal clock gates ST DMA, LD DMA, and Ring Switches
- Workload flow block diagram shows tile 0 and tile 3 are stalled
  - For each improvement, we will show which modules are clock-gated in the workload flow block diagram

Baseline Workload Flow

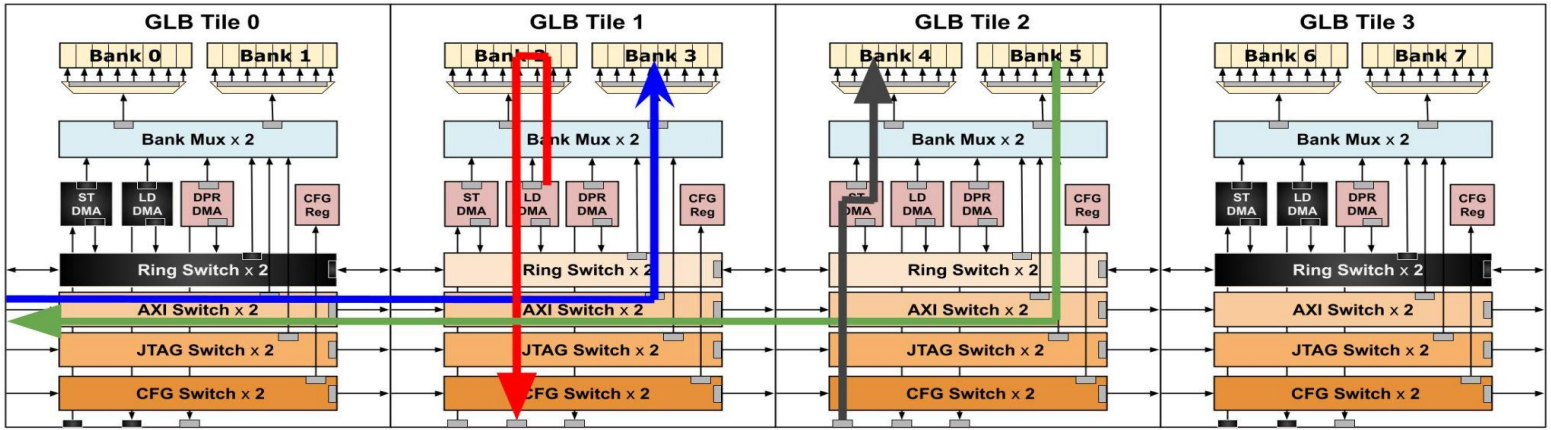


Fine-Grain Clock Gating vs Baseline - Power by Tiles / Modules

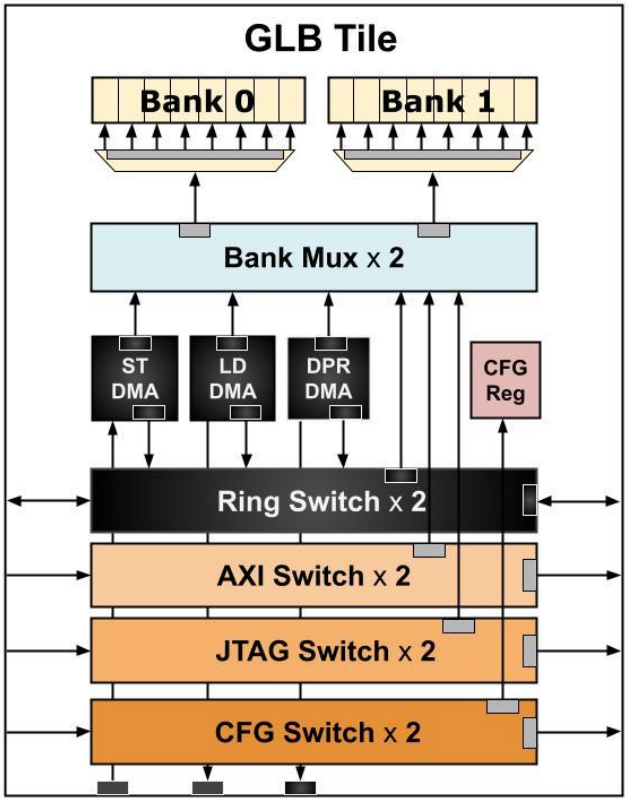


- Used Tiles: 25.8mW~28.6mW
- Unused Tiles: 7.6mW~8.6mW
  - Unused tile power is ~30% of used tile power

Baseline Workload Flow



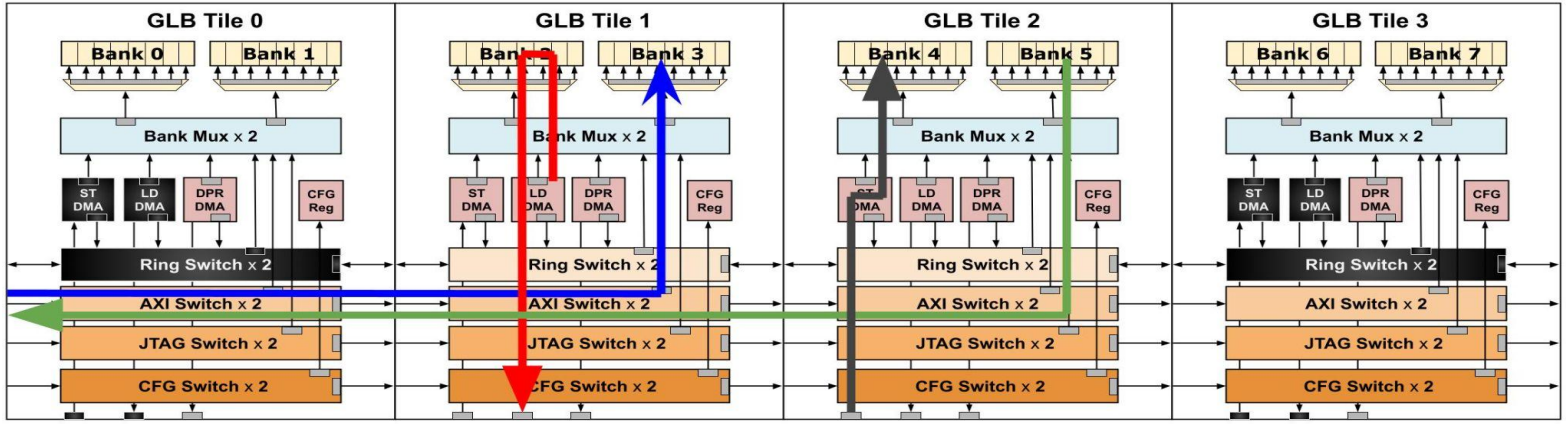
Ver.1 - DMAs / Ring Switch Fine-Grain Clock Gating (FG-CG)



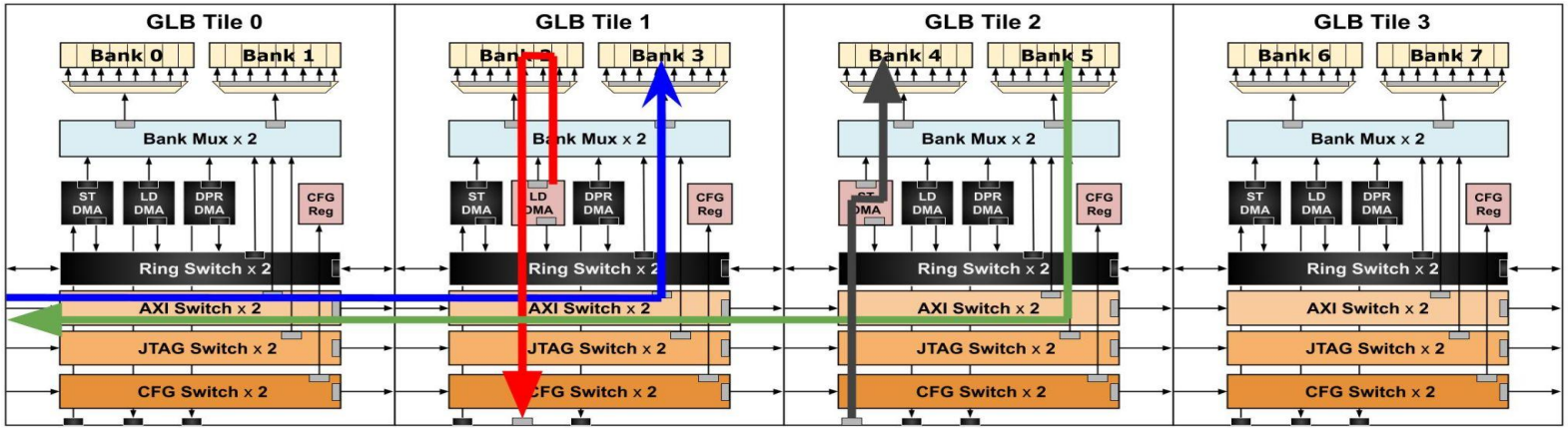
- Remove an external stall signal
- Use configuration registers to do fine-grain clock gating each module
  - DMAs have configuration registers to turn them on, start streaming, and send an interrupt
  - Ring Switch has a configuration register to chain to left/right tile

Fine-Grain Clock Gating vs Baseline - Block Diagram

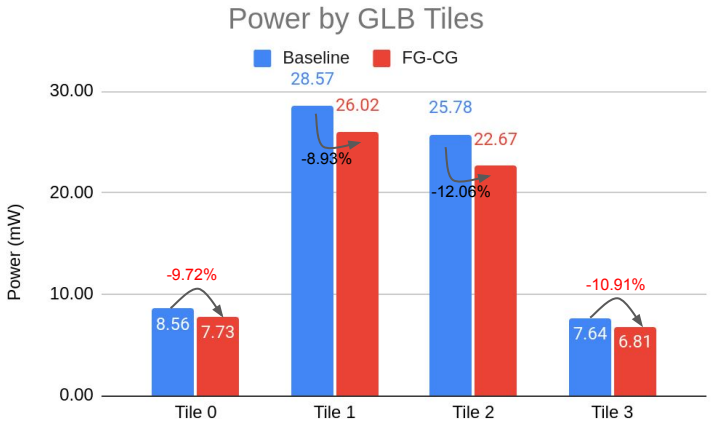
Baseline Workload Flow



Fine-grain Clock Gating Workload Flow

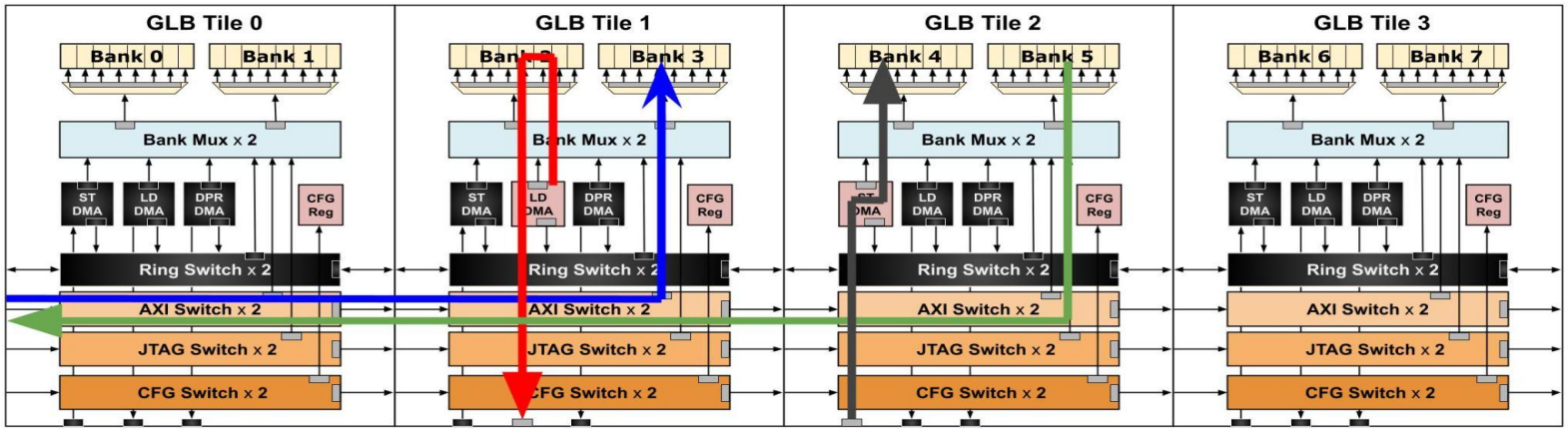


Fine-Grain Clock Gating vs Baseline - Power by Tiles

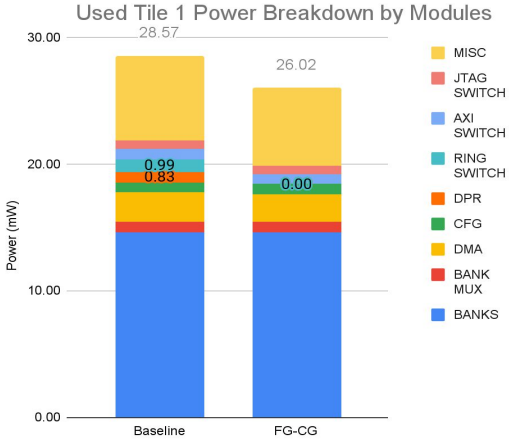


- 1~3mW power reduction for each tile
- 8~12% power reduction for each tile
- More power reduction for Used Tile 1 / Tile 2 with fine-grain clock gating
  - Clock-gate unused modules even when tile is used

Fine-grain Clock Gating Workload Flow

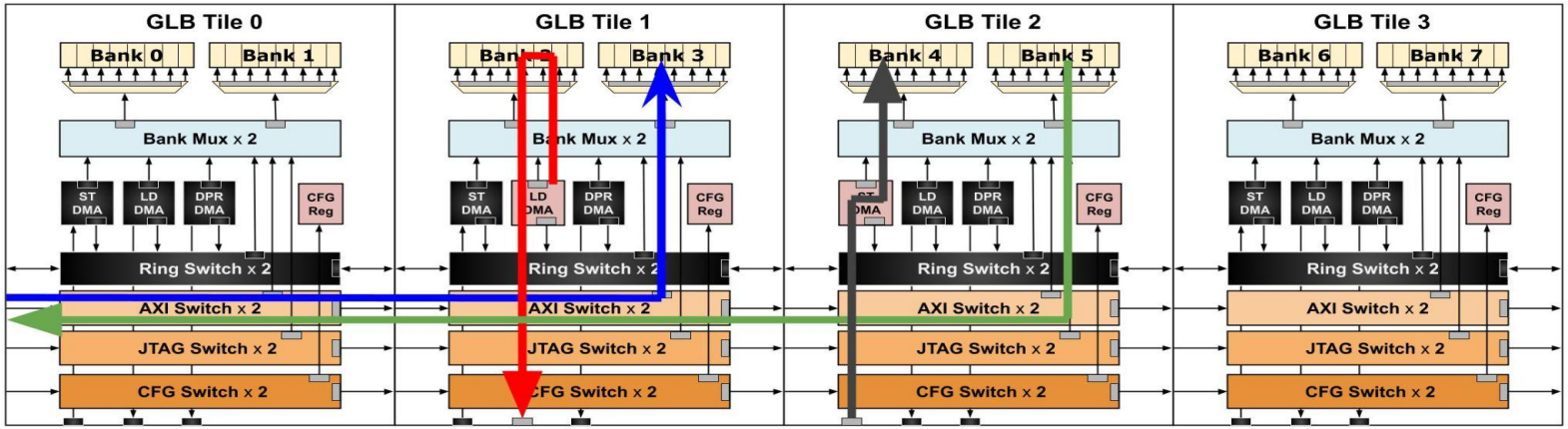


Fine-Grain Clock Gating vs Baseline - Power by Modules

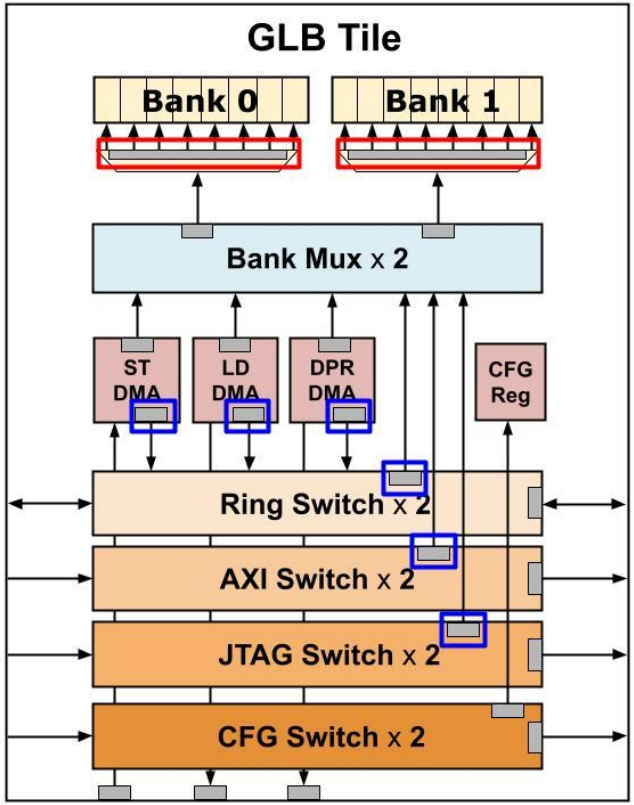


- Tile 1 - Ring Switch / ST DMA / DPR DMA are clock gated (vs Baseline)
  - Total: 28.57mW -> 26.02mW (-8.93%)
  - Ring Switch: 0.99mW -> 0.00mW
  - DPR: 0.83mW -> 0.00mW

Fine-grain Clock Gating Workload Flow

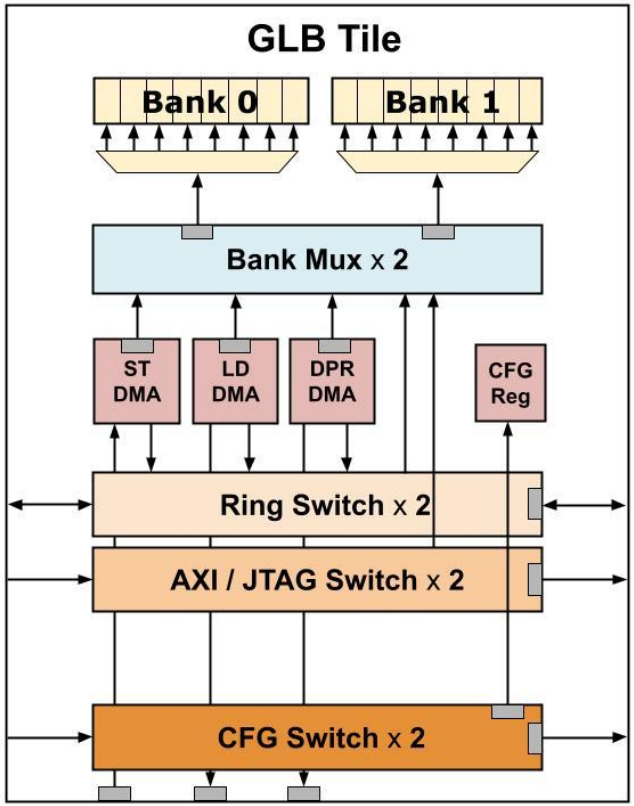


Ver.2.1 - Pipeline Register Optimization



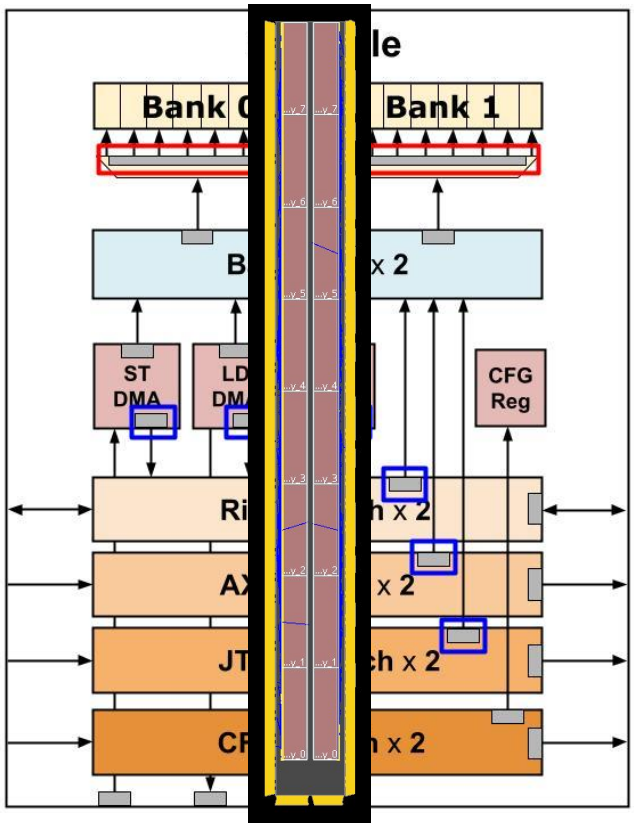
- Add a parameter to choose if pipeline registers are used or not at each location
  - Run P&R and GLS with different pipeline parameters
- Best case is to remove pipeline registers at
  - SRAM macro demux (red)
  - DMA->Switch / Switch->Bank Mux (blue)

Ver.2.2 - AXI Switch / JTAG Switch Merge



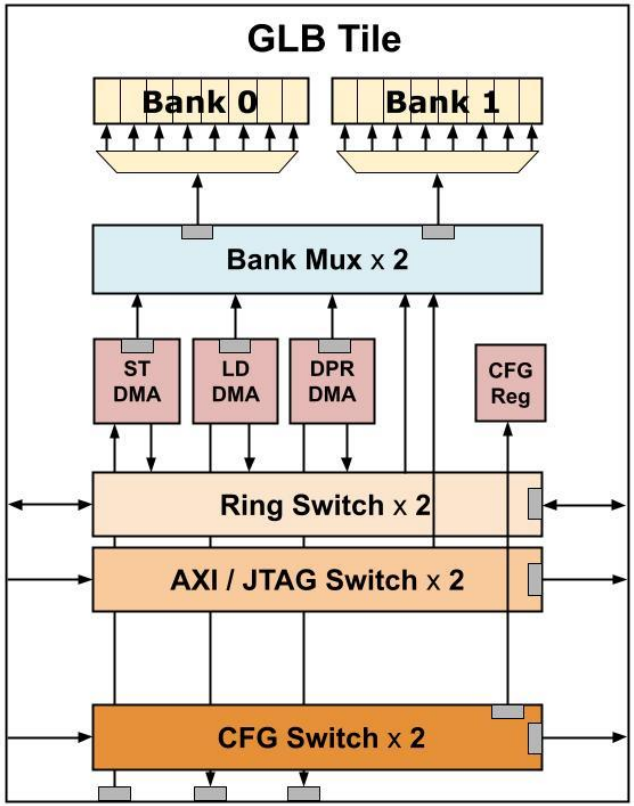
- Add a parameter to choose if pipeline registers are used or not at each location
  - Run P&R and GLS with different pipeline parameters
- Best case is to remove pipeline registers at
  - SRAM macro demux (red)
  - DMA->Switch / Switch->Bank Mux (blue)
- Merge AXI Switch and JTAG Switch
  - AXI and JTAG are not used at the same time

Ver.2.1 - Pipeline Register Optimization



- Add a parameter to choose if pipeline registers are used or not at each location
  - Run P&R and GLS with different pipeline parameters
- Best case is to remove pipeline registers at
  - SRAM macro demux (red)
  - DMA->Switch / Switch->Bank Mux (blue)
- Setup time violation at *Macro2All* and *All2Macro* paths
  - There are violations regardless of target frequency or whether there are pipeline registers or not
  - Related to a tall & skinny floorplan of SRAM macros

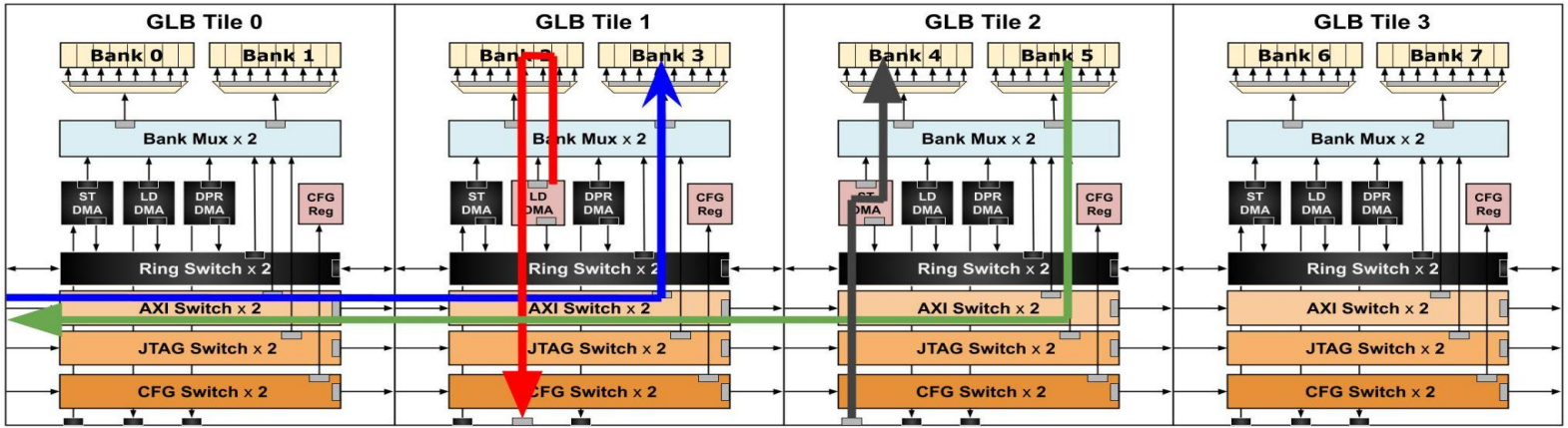
Ver.2.2 - AXI Switch / JTAG Switch Merge



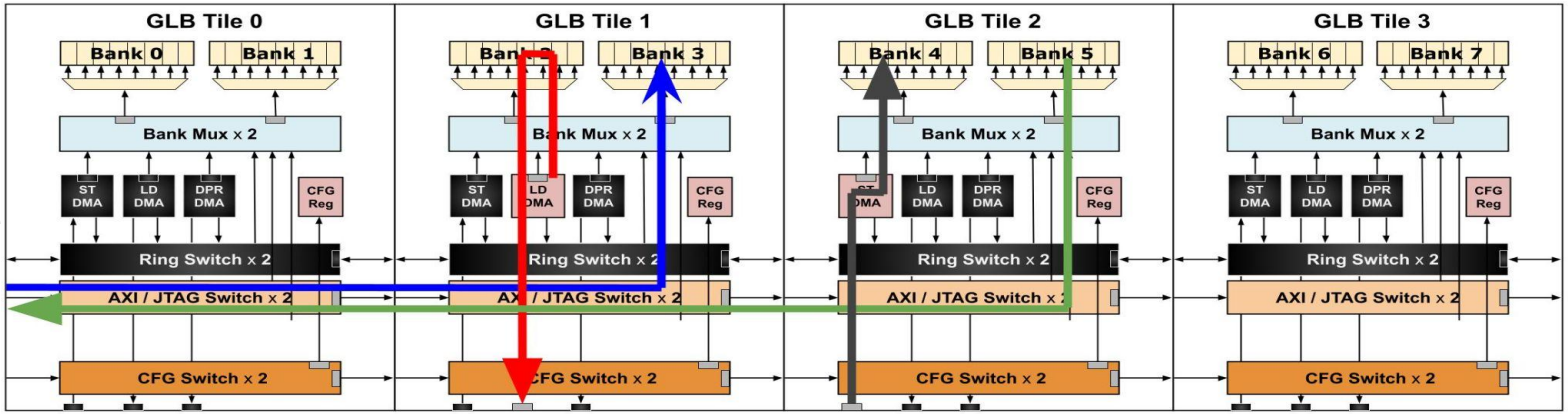
- Merge AXI Switch and JTAG Switch
  - AXI and JTAG are not used at the same time
- Need extra controller at the top
  - AXI: 64-bit word
  - JTAG: 32-bit word
  - Power reduction is multiplied by the number of GLB Tiles

PL Optimization / Merge vs Fine-Grain Clock Gating - Block Diagram

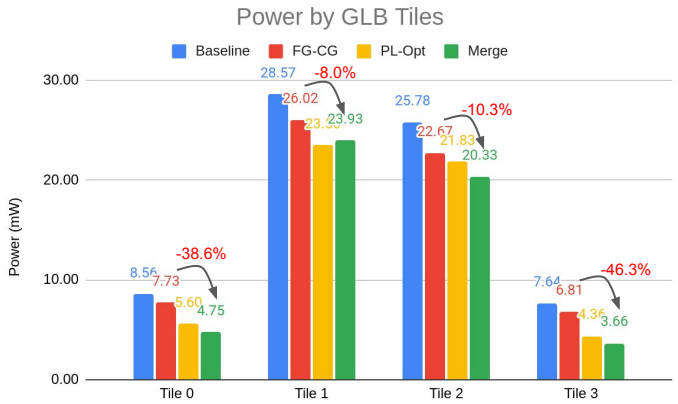
Fine-Grain Clock Gating Workload Flow



PL Opt. AXI/JTAG Merge Workload Flow

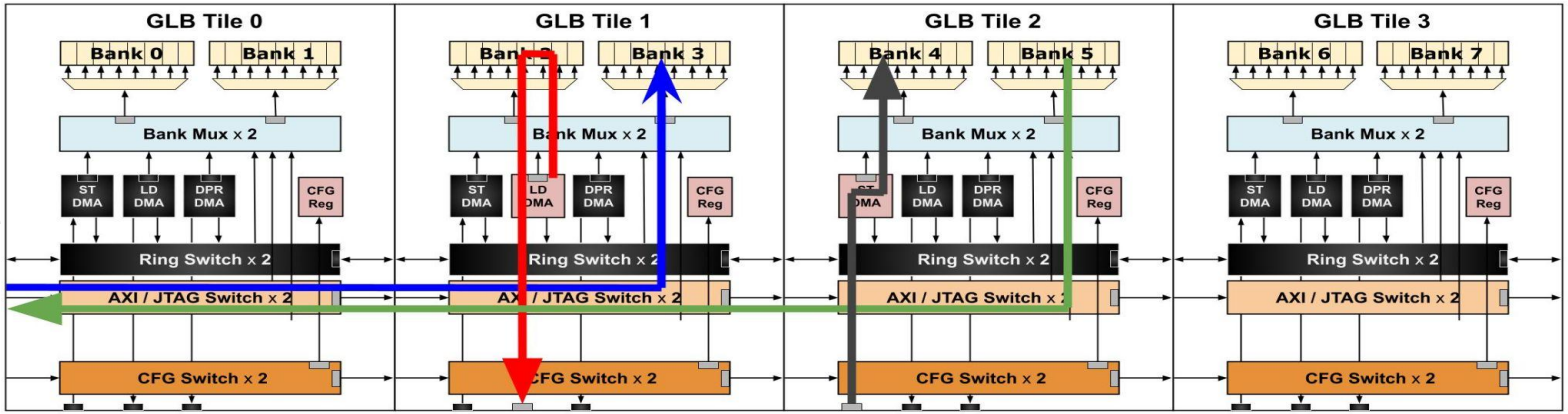


PL Optimization / Merge vs Fine-Grain Clock Gating - Power by Tiles

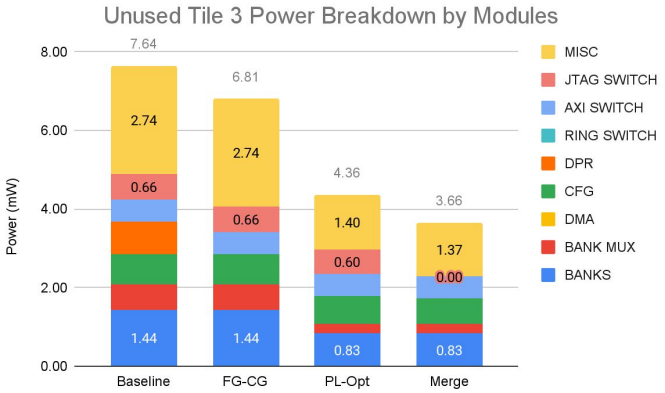


- 2~3mW power reduction for each tile
- 8~46% power reduction for each tile
- Similar power reduction for all tiles

PL Opt.
AXI/JTAG Merge
Workload
Flow

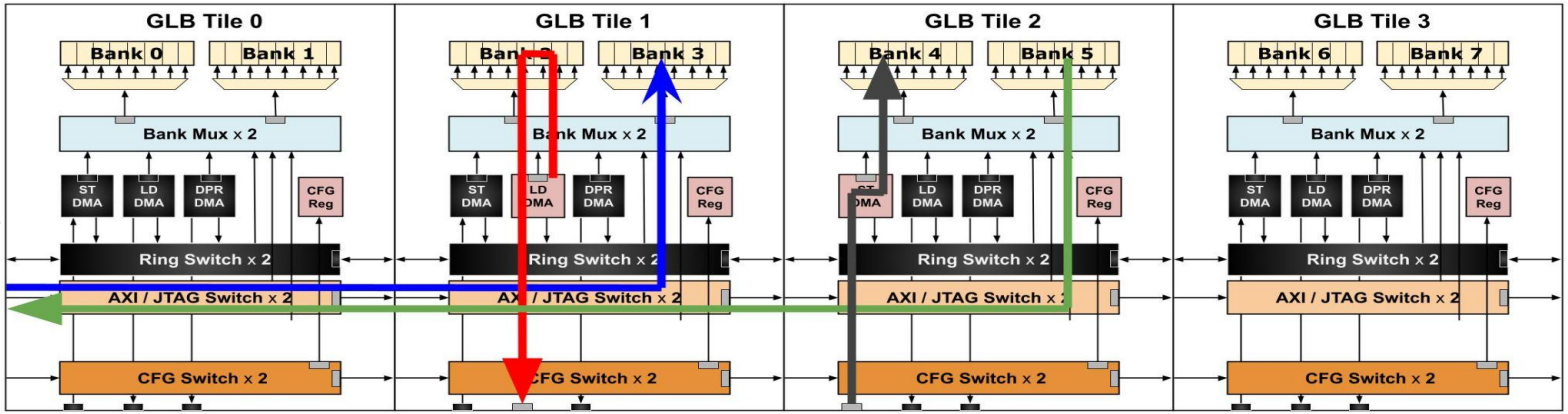


PL Optimization / Merge vs Fine-Grain Clock Gating - Power by Modules

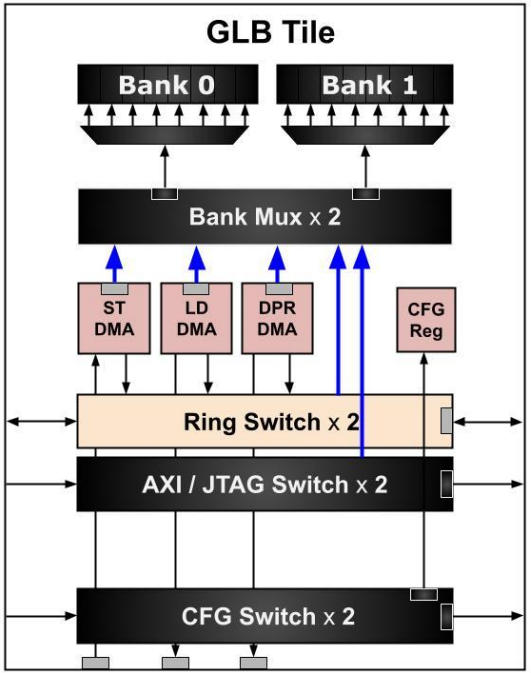


- Bank (blue): 1.44mW -> 0.83mW
  - No registers at SRAM demux
- JTAG (pink): 0.66mW -> 0mW
  - Merged into AXI
- Misc. (yellow): 6.81mW -> 3.66mW
  - Less registers == Less buffers

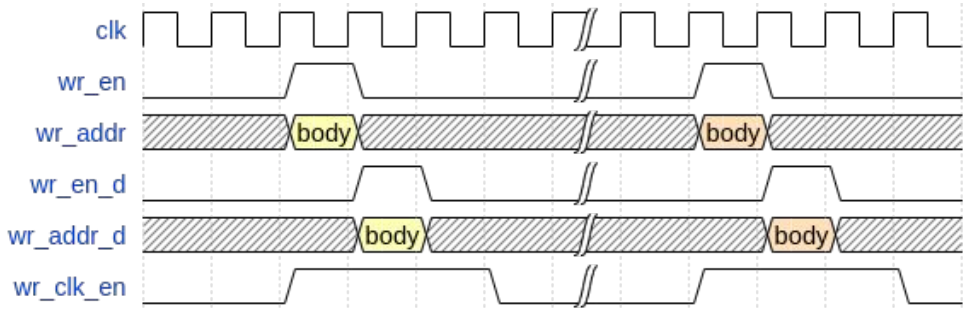
PL Opt.
AXI/JTAG Merge
Workload
Flow



Ver.3 - Dynamic Clock Gating

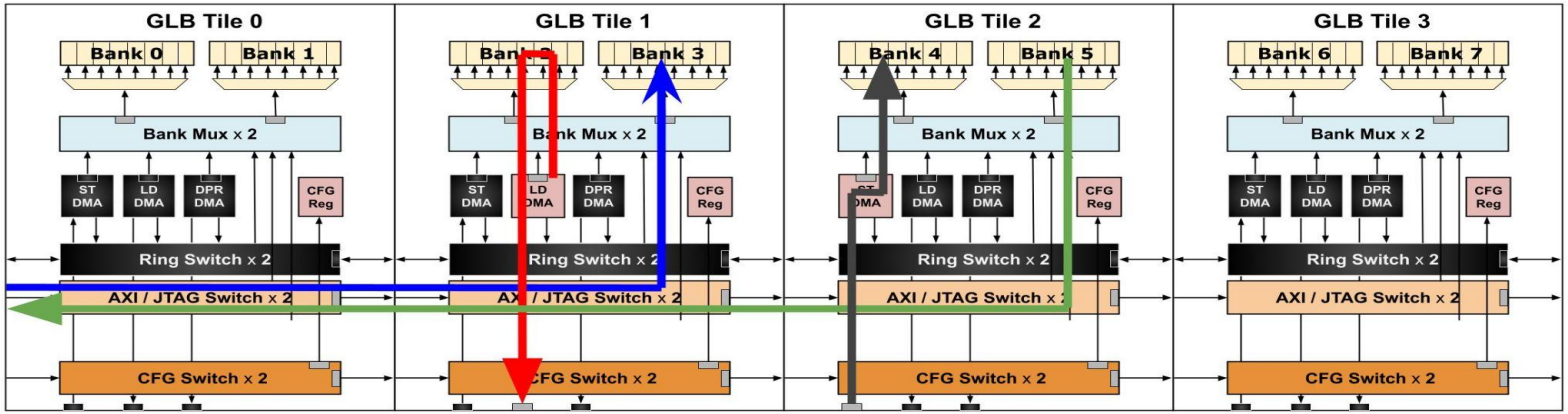


- AXI/JTAG/CFG Switch
  - Use ***wr_en*** signal to generate ***clk_en*** signal
  - Use ***wr_en*** and ***wr_addr*** signals to decide where to route signals
    - If ***wr_addr*** matches bank#, routes to bank
    - Otherwise, routes to the next GLB Tile
- DMAs that send packets to banks
  - Use ***wr_en*** signal to generate ***clk_en*** signal
  - All ***clk_en*** signals going to banks (blue) are ***ORed*** to turn clocks on for banks

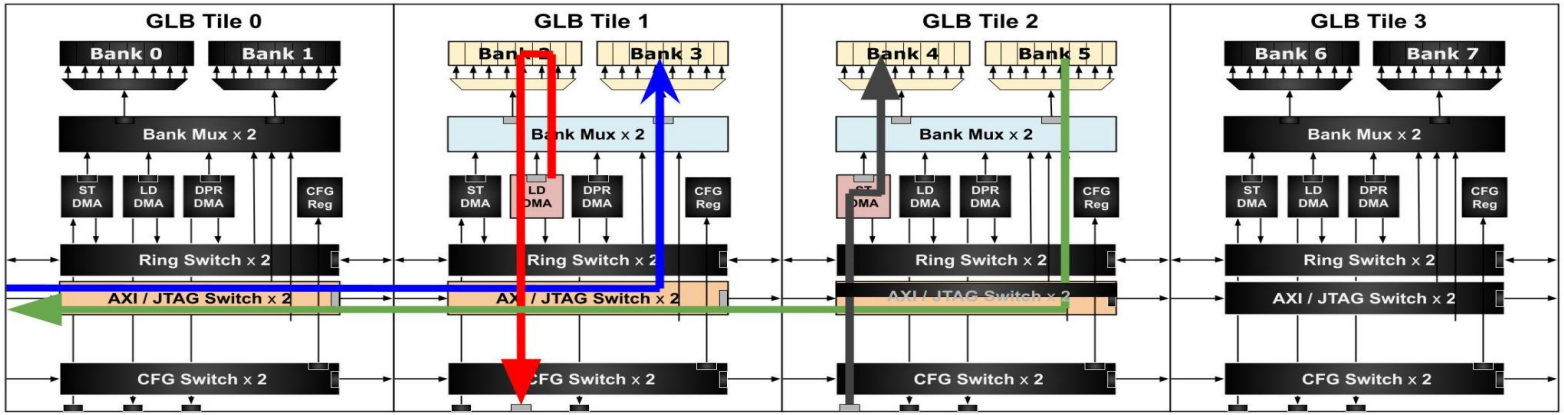


Dynamic Clock Gating vs PL Opt. + Merge - Block Diagram

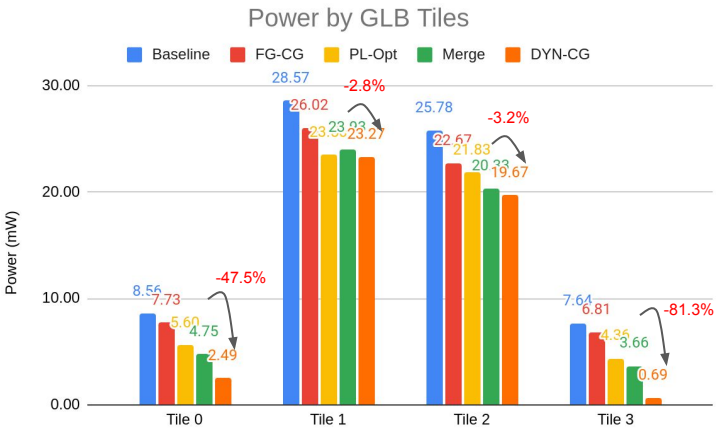
PL Opt.
AXI/JTAG Merge
Workload
Flow



Dynamic
Clock Gating
Workload
Flow

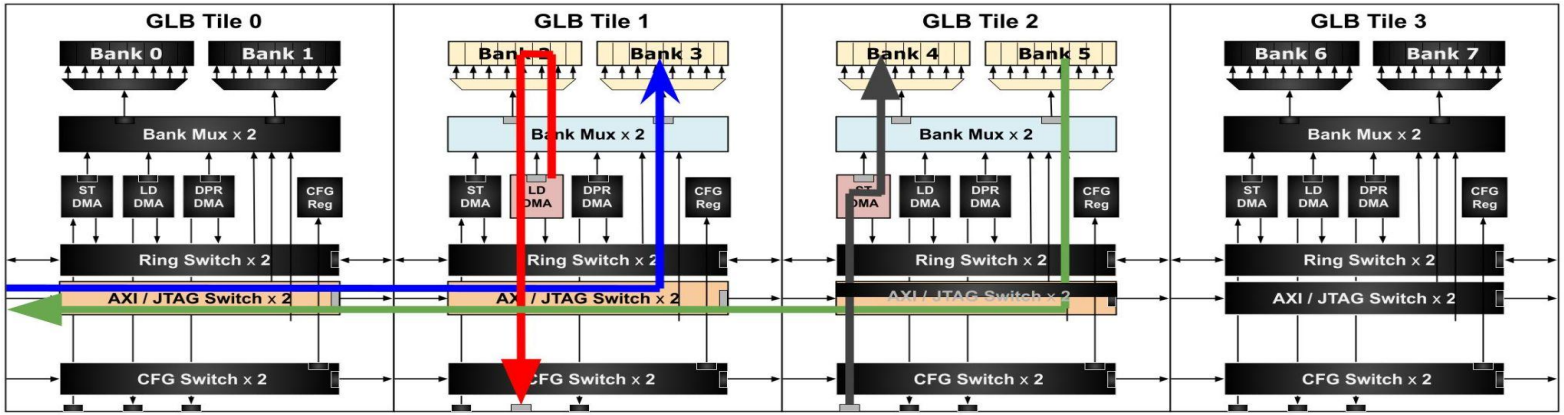


Dynamic Clock Gating vs PL Opt. + Merge - Power by Tiles

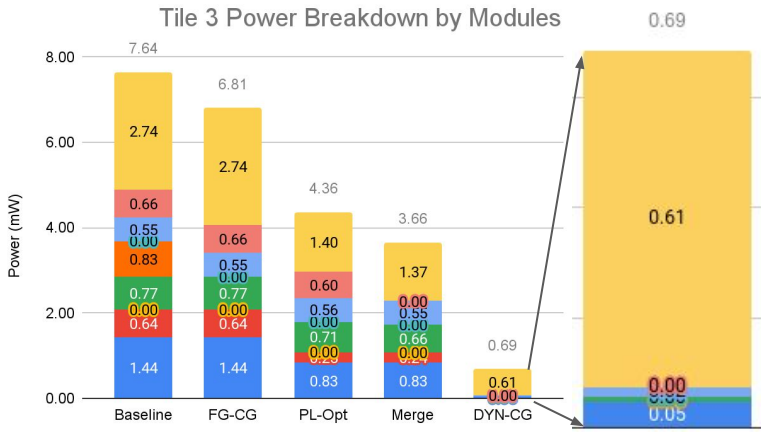


- 1~3mW power reduction for each tile
- 2~81% power reduction for each tile
- More power reduction for Unused Tile 0 / Tile 3 with dynamic clock gating
  - Banks are clock-gated
  - All switches are clock-gated

Dynamic Clock Gating Workload Flow

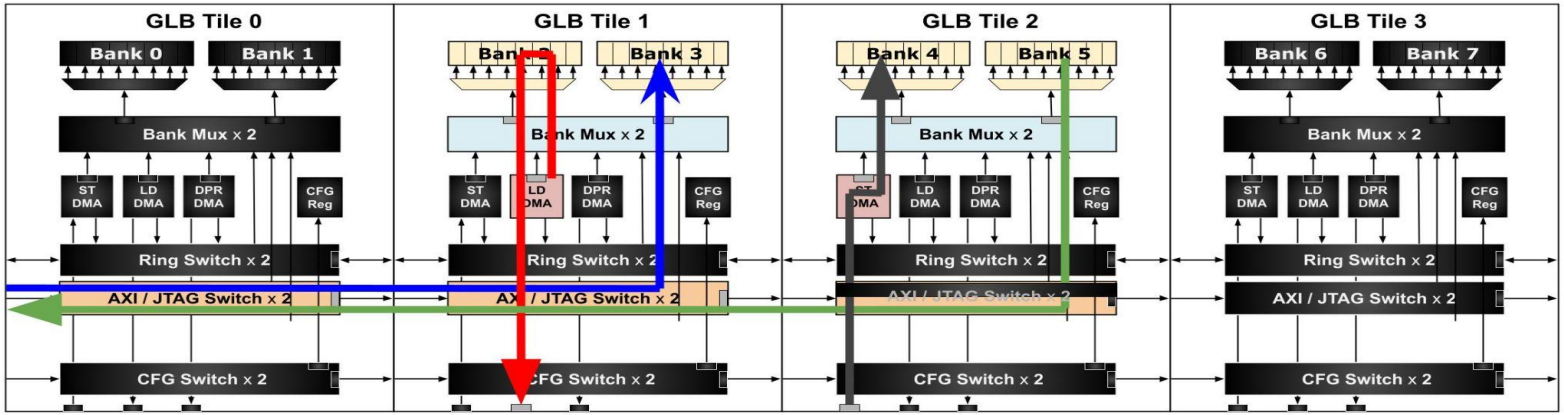


Dynamic Clock Gating vs PL Opt. + Merge - Power by Tiles



- Tile 3 (Unused) : 3.66mW -> 0.69mW
  - All modules are clock-gated
  - SRAM leakage + always_on dynamic clock-gating + Miscellaneous
  - Miscellaneous
    - Clock tree buffers
    - Timing buffers

Dynamic Clock Gating Workload Flow



Ver.4 - Use Clock Gating Integrated Cell (CGIC)

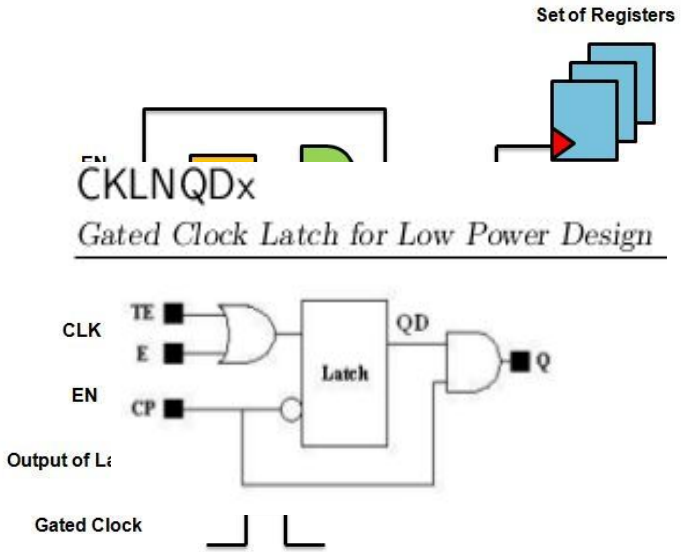


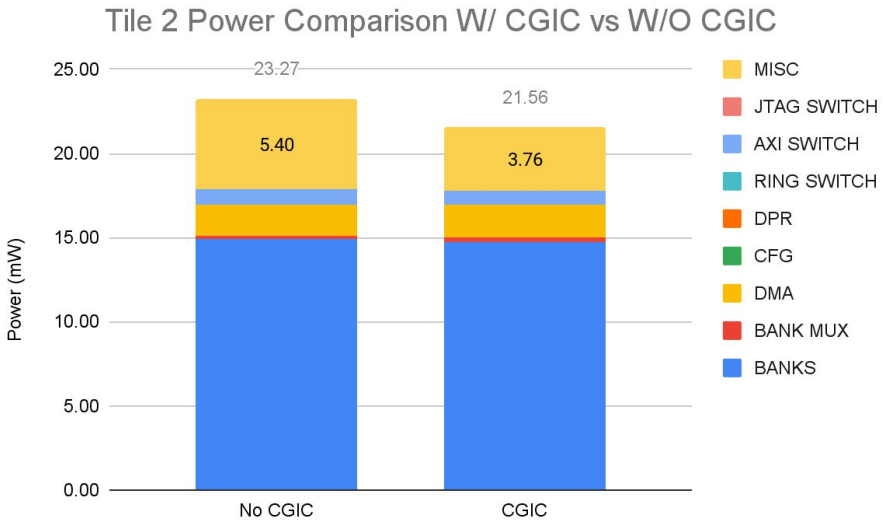
Fig. Clock gating (from www.vlsi-soc.blogspot.com)

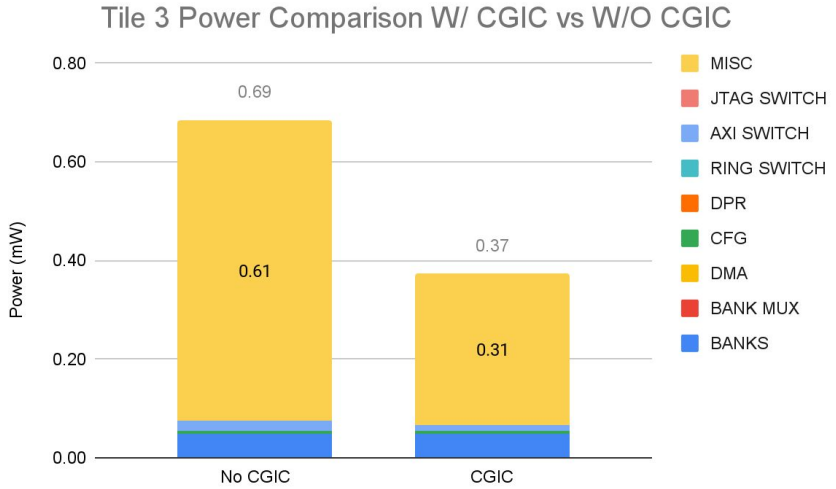
- Clock Gating = Latch + AND gate
  - Enable signal is latched to prevent glitches
- SystemVerilog Code example:

```
module clk_gate CKLNQDx (
    input logic clk,
    input logic enable,
    output logic gclk);
logic en_latch;
always_latch begin
    if (!clk) begin
        en_latch = enable;
    end
end
assign gclk = clk & en_latch;
endmodule
```

- Set module name same as CGIC from standard cell library to force EDA tools to use it

Power Breakdown of Active/Inactive Tiles - Ver.4 CGIC

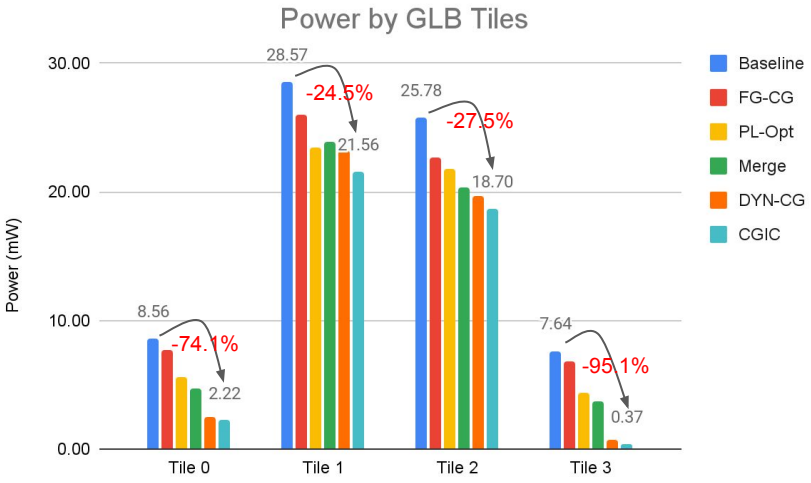




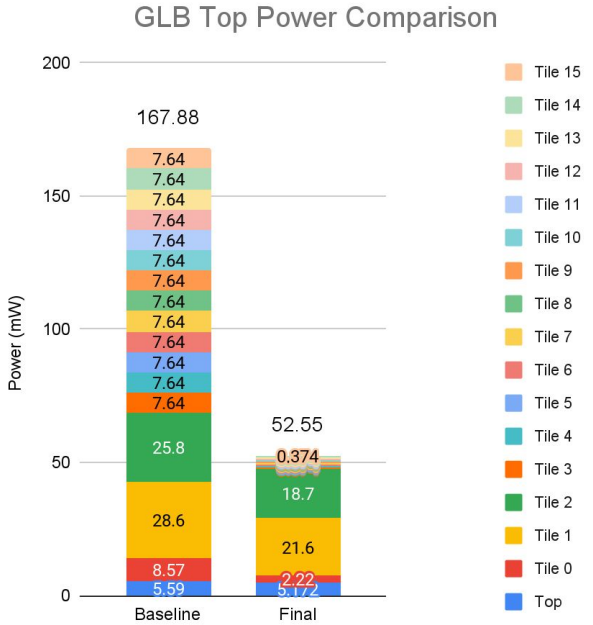
- Tile 2 (Active) Total power: 23.27mW -> 21.56mW
- Miscellaneous power: 5.40mW -> 3.76mW (-30.4%)

- Tile 3 (Inactive) Total power: 0.69mW -> 0.37mW
- Miscellaneous power: 0.61mW -> 0.37mW (-39.3%)

Overall Result



- Tile 0: 8.56mW -> 2.22mW (-74.1%)
- Tile 1: 28.57mW -> 21.56mW (-24.5%)
- Tile 2: 25.78mW -> 18.70mW (-27.5%)
- Tile 3: 7.64mW -> 0.37mW (-95.1%)



- GLB Power: 167.88mW -> 52.55mW (-68.7%)

Summary

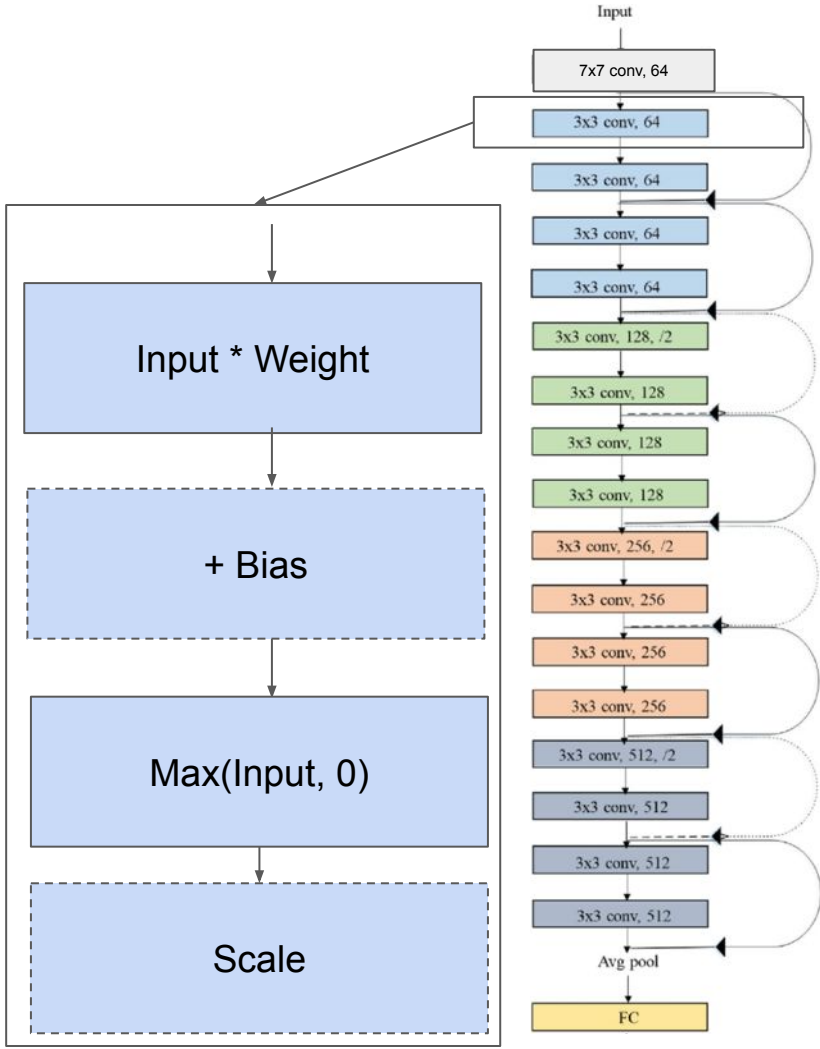
1. Optimized the global buffer power consumption by
    a. Fine-Grain clock-gating
    b. Pipeline optimization and module merging
    c. Dynamic clock-gating
    d. Manual CGIC insertion
2. Reduced the overall power by 68.7% for the benchmark (167.88mW -> 52.55mW)
    a. Active Tile: 20~30% Reduction
    b. Inactive Tile: 95% Reduction
3. Further optimization
    a. SRAM macro floorplan exploration

End to End Resnet

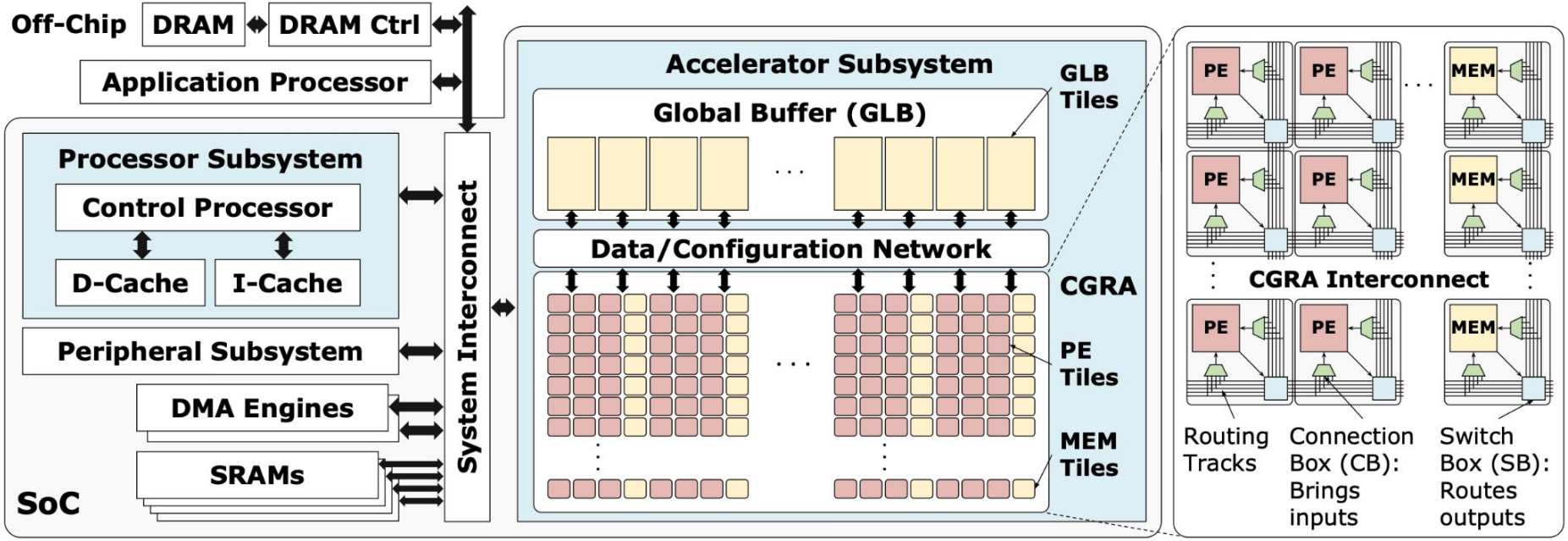
Kalhan Koul

Resnet 18

- Currently we have been accelerating one convolutional layer at a time
  - Single parameterized application in H2H
- Relu is performed at the end of each block
- Currently our application does not implement biases and scaling, but they should be added similar to skip connections (shown later)
- Goal: Look at accelerating the whole application and determine hardware and compiler changes for Onyx

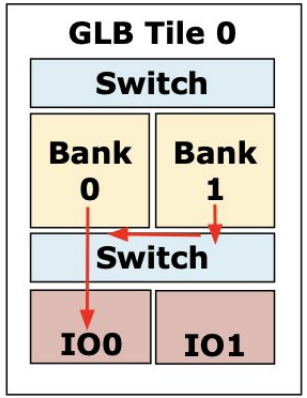


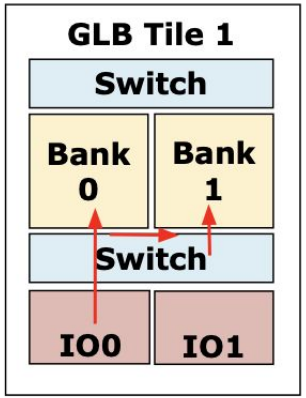
SoC Review

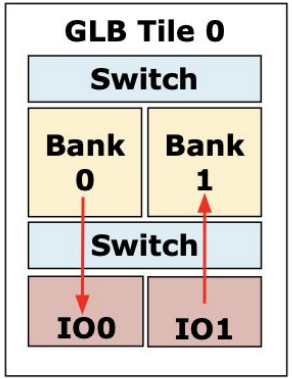

Off-Chip
DRAM
DRAM Ctrl
Application Processor
Processor Subsystem
Control Processor
D-Cache
I-Cache
Peripheral Subsystem
DMA Engines
SRAMs
SoC
System Interconnect
Accelerator Subsystem
Global Buffer (GLB)
GLB Tiles
Data/Configuration Network
CGRA
PE Tiles
MEM Tiles
PE
MEM
CGRA Interconnect
Routing Tracks
Connection Box (CB): Brings inputs
Switch Box (SB): Routes outputs

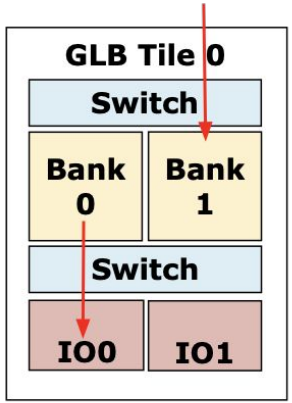
Global Buffer Review

- Pixel = 16 bits
- DMA from SoC to GLB is 2.37 pixels/cycle
  - Due to AXI3 issues Po-Han presented
- Each Bank is 125KB, 2 Banks per tile, 16 tiles
- Can use one whole tile as load and another whole tile as store
- Per tile each bank can both store or load during operation
  - Preferred - increased bandwidth to array







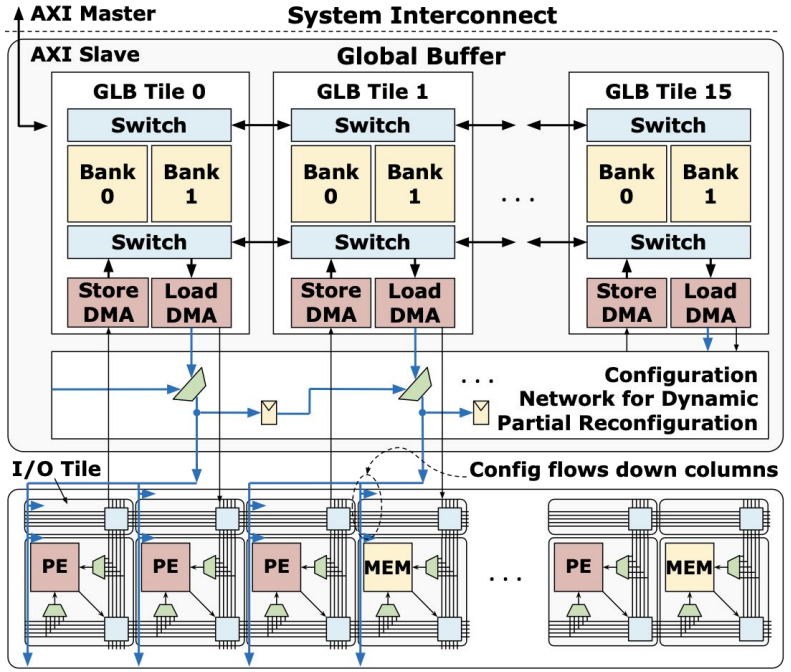


Assumptions for Running Resnet End to End on the SoC

- Using a schedule that copies all of the weights for a given layer to the GLB
  - Current working schedule
  - No tiling at DRAM <-> GLB level
  - Resnet input image is brought into the GLB once, and intermediate activations are reused from the GLB
- 128 MACs/cycle (256/384 PEs)
  - Using 256/384 PEs since its the current working schedule and routing issues start occuring
  - MACs/cycle will increase with Jack's PEs
- 100% compute occupancy on conv layers
  - Layers are not input, weight, or output limited from GLB to CGRA
  - Note: Current occupancy is 50% for the first layer and 90% for the rest of the layers
- No problems with routing applications given a specific IO placement
  - Noticed during VLSI push that we could not successfully route with inputs and outputs on the same side (ex. both left) of the CGRA

Constraints

- Bitstream (~25KB), Inputs, Weights and Outputs must fit in Global Buffer
- Placement of Inputs, Outputs, Identities (output of one layer is input of next)
  - Avoid shuffling data between layers
- Loading the Weights for Layer N+1 < Kernel Processing for Layer N
  - Maintain high compute occupancy
- Bitstream placed in leftmost tile
  - DPR configuration network flows from left to right
- Went layer by layer and found a mapping that works

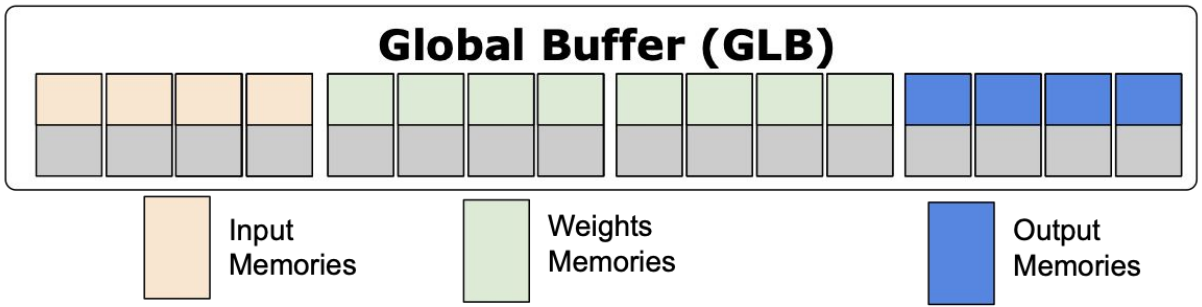


Early Resnet Layers need Input/Output Banking

- Inputs/Outputs are too large for a single GLB Bank (125KB)

Layer	input size	weights size	output size
L1.0 conv1	401KB	73KB	401KB
L1.0 conv2	401KB	73KB	401KB

- Bank data to fit Inputs/Outputs in the GLB



Last Resnet Layers need Weight Banking

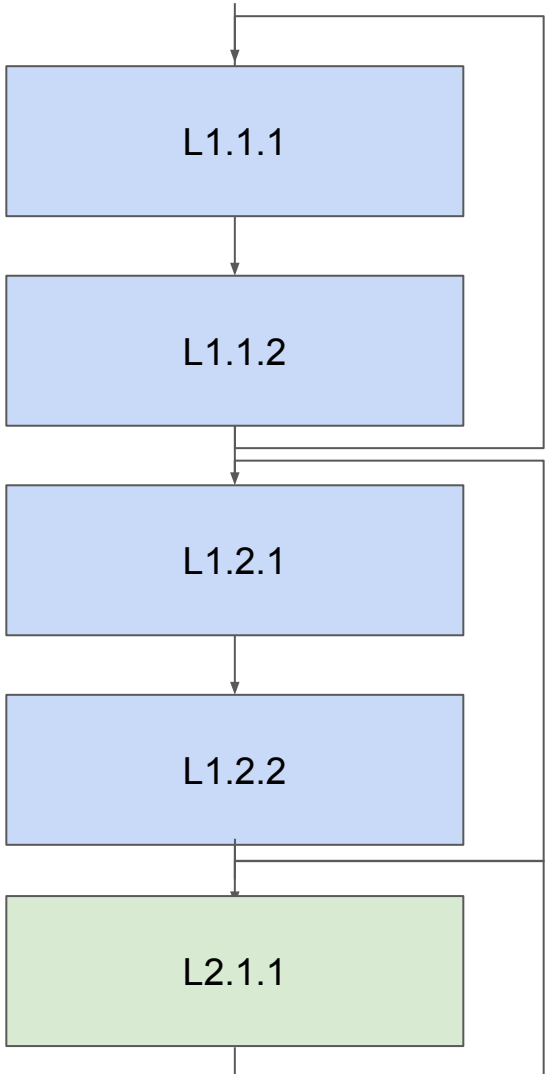
- Latter layers have more weights ~4MB, smaller inputs and outputs
  - Increase weight banking -> 12
  - Split output channels (n_oc=512) into quarters only mapping a quarter of the layer on the CGRA at a time (4MB/48 < 125KB)

Layer	input size	weight size	output size
L4.1 conv1	50KB	4MB	50KB
L4.1 conv2	50KB	4MB	50KB

Resnet Skip Connections

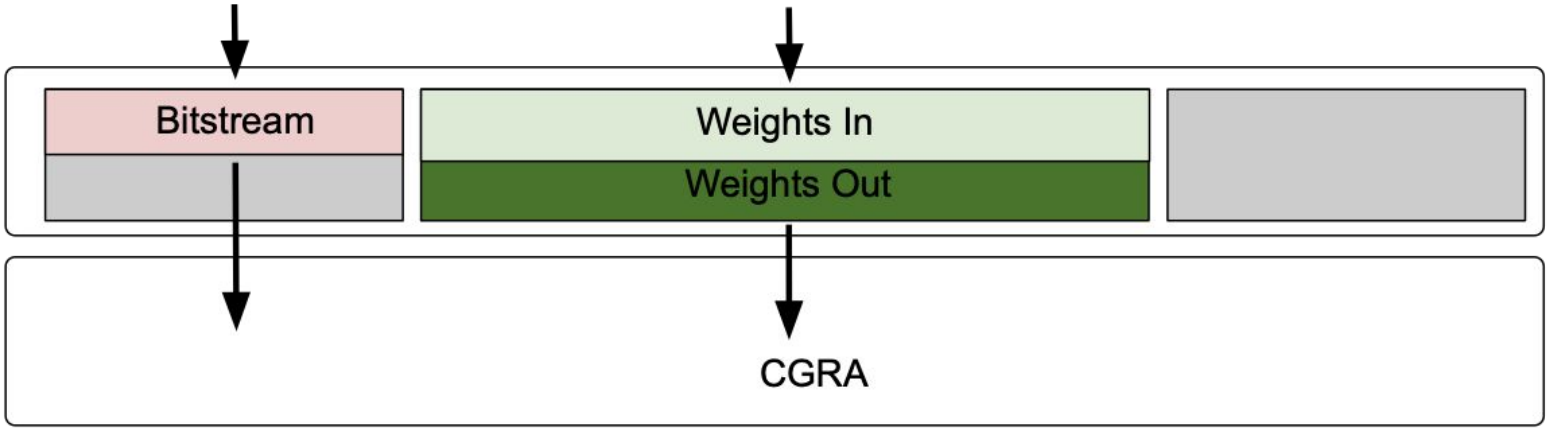
Notes

- Each of these layers has the same convolutional computation
- Input of one layer becomes output of the next
- We need to store identities in the global buffer between computation
- Also need to save space for bitstream!



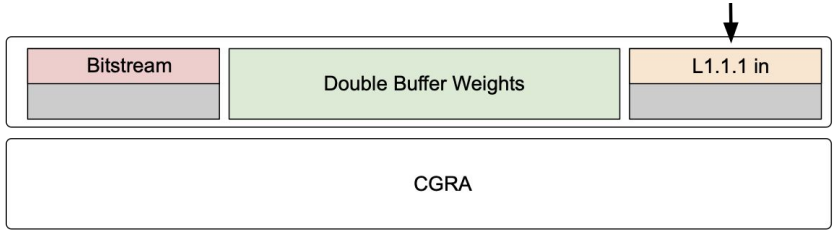
Bitstream and Weights

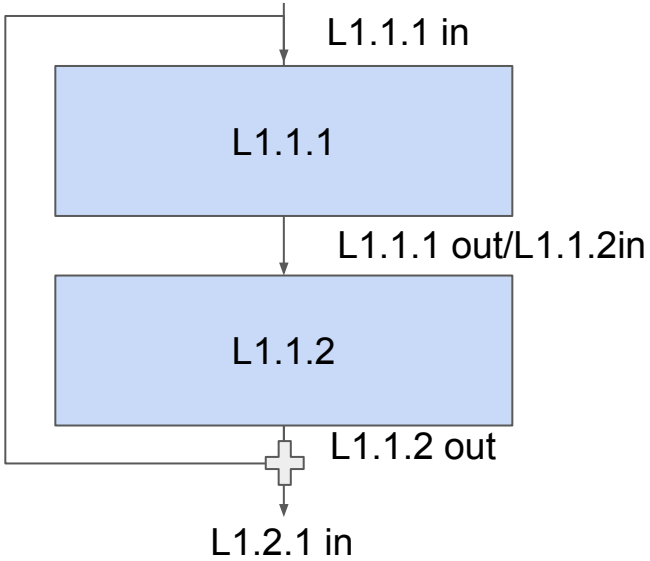
- Every layer has a bitstream and different weights
- Configure Layer
- Run Layer (wait for interrupt)
  - Replace Bitstream (we do not have space to double buffer – shown later)
  - Double Buffer weights



Flow on Amber

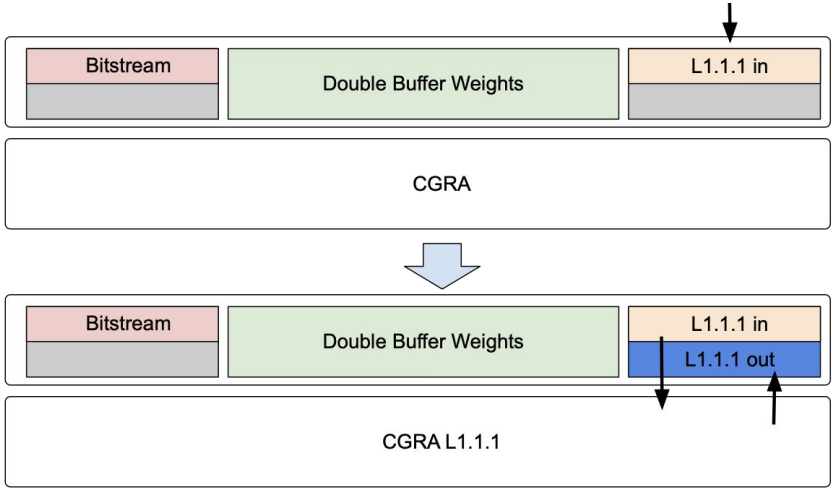
- Load in inputs

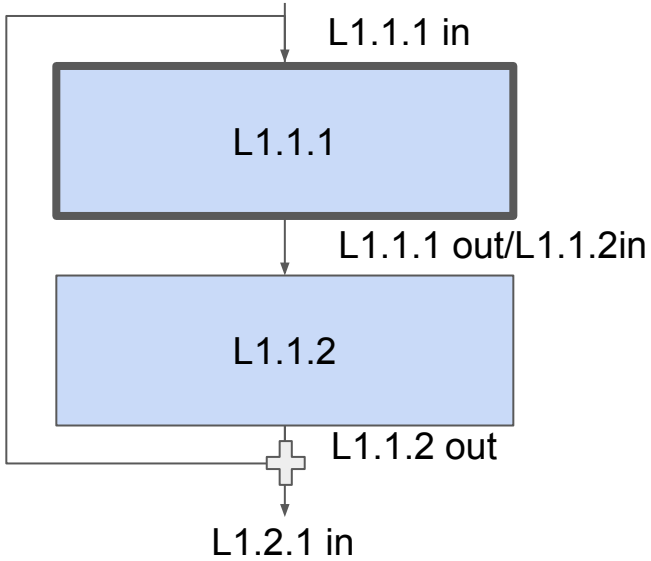




Flow on Amber

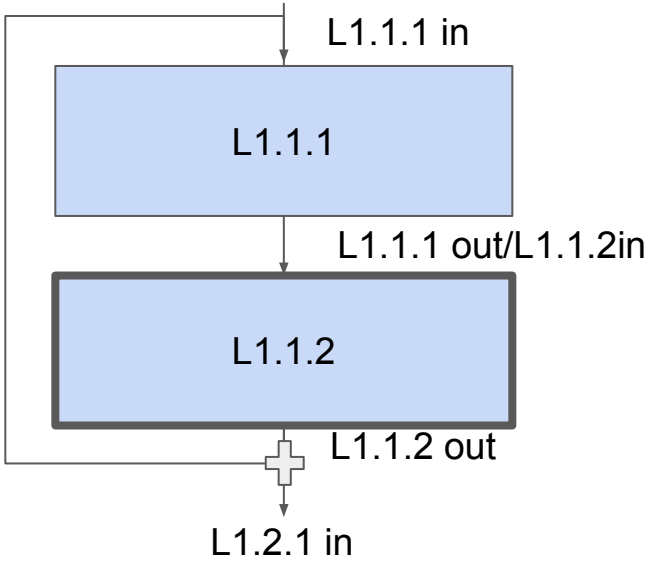
- Perform L1.1.1

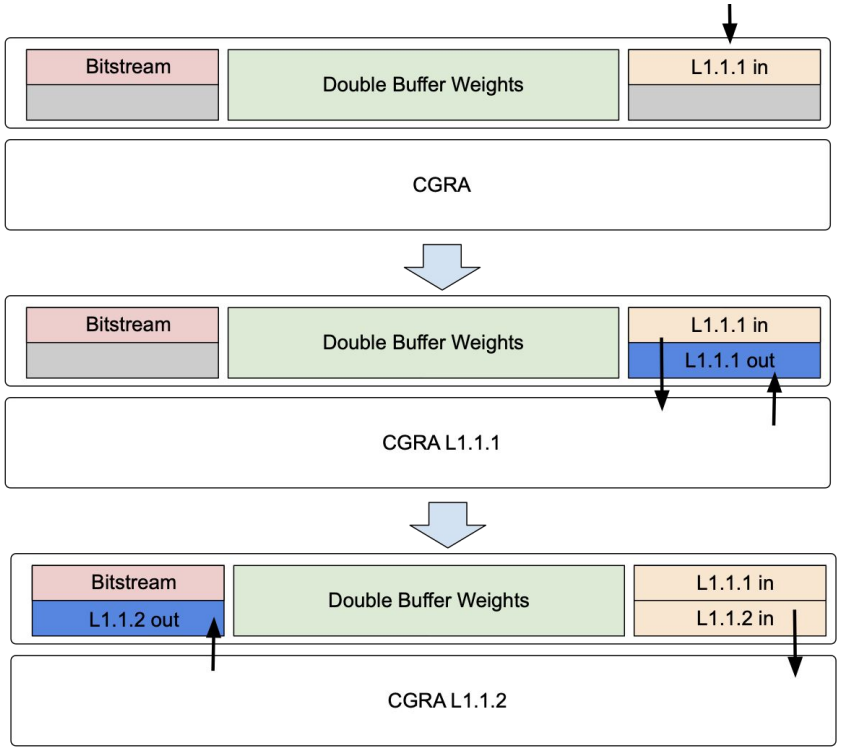




Flow on Amber

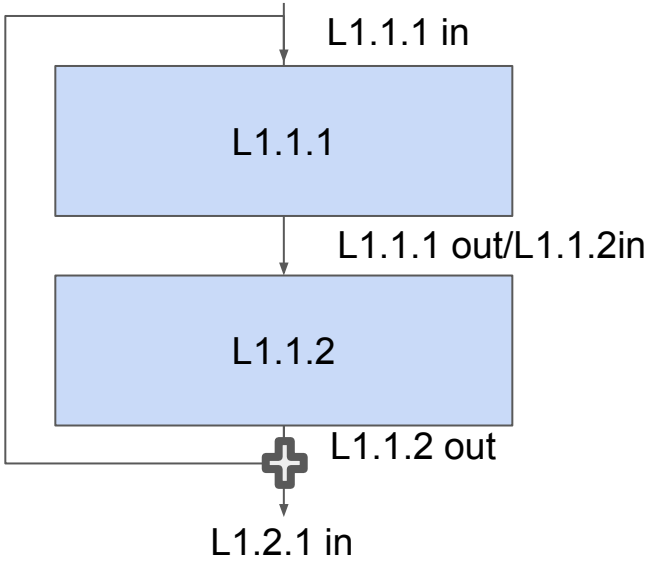
- Perform Layer L1.1.2

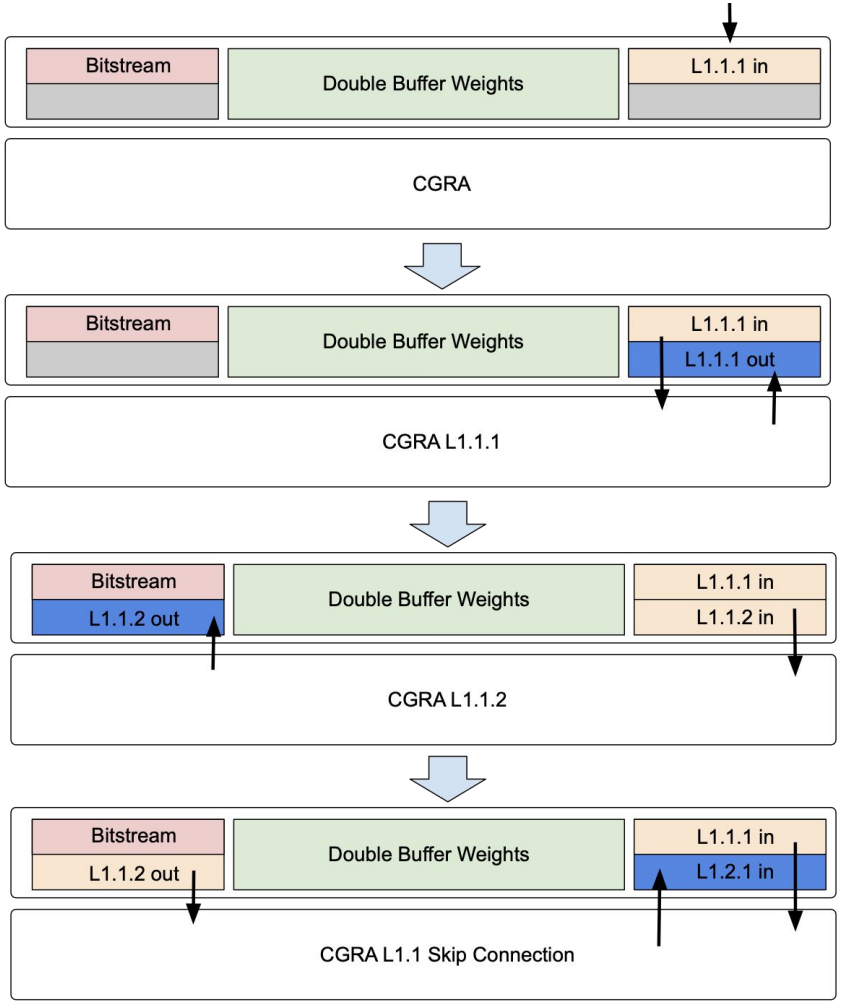




Flow on Amber

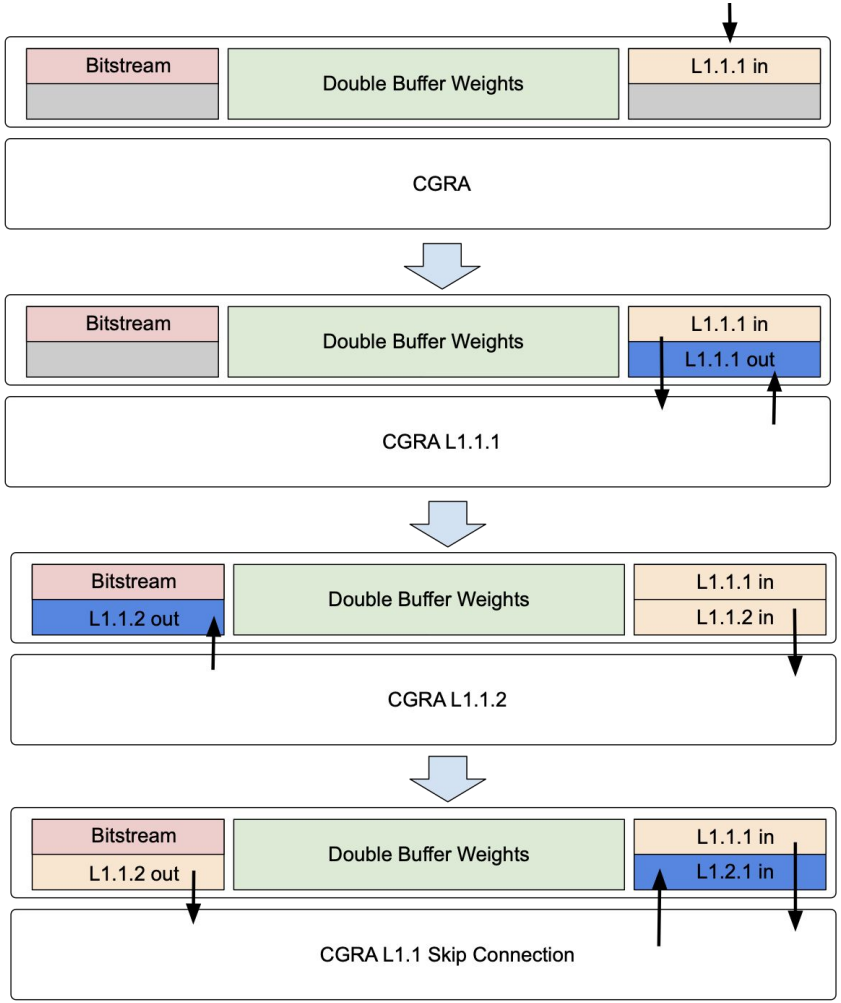
- Perform skip connection
- Back to starting point





Notes

- Why configure each layer?
  - Equivalent computation
  - Need to change IO to save skip values
  - IO are changing so bitstream changes
- Skip connection sum
  - Runtime = input size/glb unrolling
  - First layers - 11% of Conv layer
  - Last layers - 6% of Conv layer



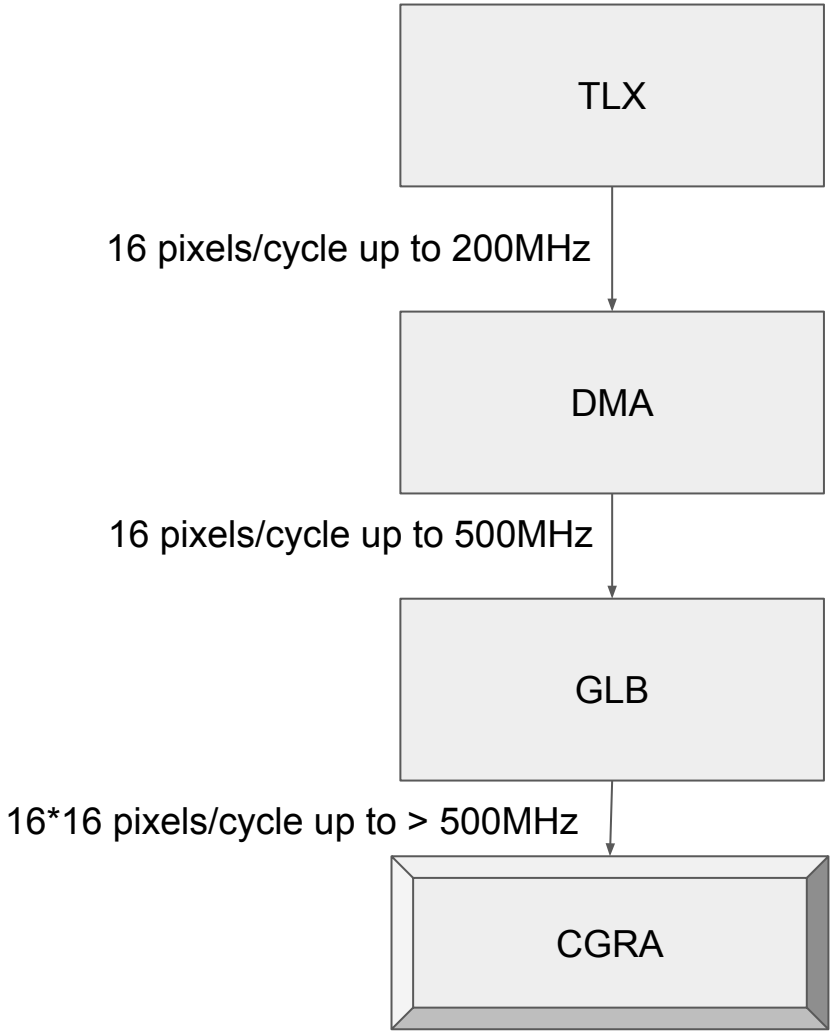
Loading Weights vs Compute

- Early layers have fewer weights, not a problem
- Latter layers we run into issues with bandwidth to GLB
- Gedeon designed an improved DMA engine (untested)
- However, we are still weight limited for latter layers
- Jack's PEs will allow for more MACs/cycle
- Add a second axi port to GLB?

		amber dma 2.37 pixels/cycle	onyx expected 4 pixels/cycle	amber 128 MACs
Layer	kernel size	weight loading time	weight loading time	ideal cycles
L1.1.1	73KB	31109	18432	903168
L4.2.2	4.7MB	1990967	1179648	903168

TLX

- TLX = 64bits x TLX_CLOCK_FREQ x TLX_DATA_REV_CHANNEL_UTILIZATION
- TLX = 64bits x 200MHz x 0.98 (12Gbit/s)
- TLX can feed one DMA engine up to 200MHz (FPGA Limited, Amber SDC-250MHz)
- Currently, at 200MHz latter layers will be weight limited
- Image Processing Applications (16 bit pixels)
  - Unroll=4 at 200MHz
  - Unroll=1 at 800MHz (low utilization)
- Options:
  - 8-bit Resnet (would require GLB changes, or shifting data inside CGRA)
  - FPGA with new board (correct, clock pin 250GHz)
  - More REV pins for TLX (~100 bumps left)
  - SERDES (1 GHz) - We don't have IP
- FPGAs have 400Gbit/s bandwidth



Next Steps - Bandwidth

- Determine what we would like to do for Chip IO Bandwidth
- Second AXI port to GLB
  - Allows us to use two DMA engines to move weights to two GLB tiles in the same cycle
- Rerun analysis with Jack’s PEs and increased compute utilization
- Test and refine schedules and mapping
  - Global Buffer banking increases input, weight, and output bandwidth
  - Can we write schedules that are compute bound with this mapping?

Next Steps - AHA Flow

- Test pool layers and FC layer
- Quantize Resnet for our hardware
- Need ability to pass in IO configuration per resnet layer
  - Currently the flow greedily lays out IO from left to right on the GLB
  - Ideally specified in the schedule, amount of unroll and IO location
- Express Multi-layer Application in Halide