Mini TPU: Tensor Processing Unit (TPU) Structure Implementation and Floor Planning

Chao-Jia Liu
Graduate Student,
Electrical and Computer Engineering
Klesse College of Engineering and Intergrated Design
University of Texas at San Antonio
chao-jia.liu@utsa.edu

***Abstract*—The Tensor Processing Unit (TPU), a powerful ASIC for machine learning and matrix multiplication, delivers high throughput and energy efficiency, setting new performance benchmarks in neural network computations. It has been utilized in Google's datacenters since its development. To better understand how data values and control signals propagate within the TPU structure, this project aims to replicate the TPU's microarchitecture, as detailed in the paper "In-Datacenter Performance Analysis of a Tensor Processing Unit," by implementing it in Verilog. The architectural design includes components such as the Matrix Multiplication Unit (MMU), accumulators, FIFO units, sequencer, memory models, and, most importantly, the control unit. A customized CISC instruction set will be encoded to realize the TPU's functionality. Additionally, floor planning and analysis using Cadence Innovus will be incorporated to explore potential future fabrication. To simplify the design and meet time constraints, components will be scaled down (e.g., the MMU from 256x256 to 4x4), as larger components primarily involve duplicating structures. This study will provide insights into TPU implementation and offer practical experience in computer architecture design.**

Keywords: Tensor Processing Unit (TPU), Matrix Multiplication Unit and Accumulators, FIFO units, memory models, control unit design, and microarchitecture using Verilog code.

I. Introduction

With the rapid advancement of deep learning and neural networks, significant research has been conducted to expand this field from various perspectives, encompassing both software and hardware aspects. From a software standpoint, researchers have focused on developing innovative algorithms to create more complex, precise, and efficient neural network architectures. These efforts have enabled the design of faster, deeper, and more robust networks capable of handling increasingly diverse and challenging tasks. On the hardware side, companies have dedicated substantial resources to developing specialized architectures that enhance the performance of deep learning computations, pushing the boundaries of speed, efficiency, and scalability.

Deep learning computations can often be simplified by treating them as matrix multiplication operations. The output of these operations is subsequently processed through activation functions such as ReLU, Sigmoid, or SoftMax, which introduce non-linear transformations while reducing precision. These transformations generate new datasets that can be recursively computed to approximate a function capable of representing the entire dataset. The Tensor Processing Unit (TPU), developed by Google in 2015 and now in its fifth iteration, is a specialized hardware architecture designed to tackle matrix multiplication tasks with exceptional throughput and efficiency. TPUs have become critical components in the deployment of large-scale neural networks, particularly in Google's datacenters. While Google's publications provide an overview of the TPU's block diagrams and functionality, details regarding its implementation and the control signals employed remain largely unexplored.

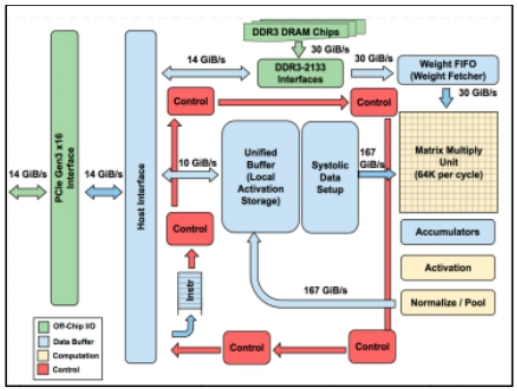


Figure 1. TPU Block Diagram.

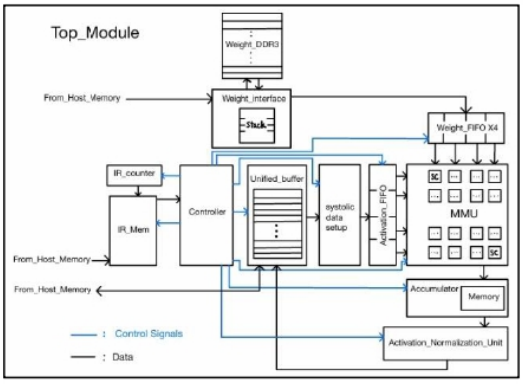


Figure 2. Block Diagram for this project. The blue lines represent the control signals, while the black lines indicate data movement.

To address this gap, this project aims to implement a scaled-down version of the TPU architecture to better understand how data values and control signals propagate within the system. The implementation is based on the descriptions provided in the paper "In-Datacenter Performance Analysis of a Tensor Processing Unit" and focuses on Verilog-based coding for each component. The second section of this report presents the Verilog code for each component described in the TPU paper, scaled to a 4x4 implementation, with detailed explanations of the functionality of each signal. The third section elaborates on the various states within the control unit and the usage of control signals in these states. The fourth section provides the simulation results of the implementation, while the fifth section presents the floorplanning results generated using Cadence Innovus, including timing reports and checks for geometry and connectivity. The sixth section discusses potential improvements and future work, concluding with a summary of findings in the final section. While this project does not claim to replicate the exact design of the TPU, it provides valuable insights into its architecture, practical experience in hardware implementation, and a foundation for further research and development in this area.

II. Compontent Implementation

Every component plays an important role in this project and has been implemented using Verilog. The following sections introduce and review all the components, including their functionalities and I/O signal usage. The components are organized and explained in separate sections: Instruction Register-Related Units, Weight-Related Units, Computation-Related Units, and the Control Unit.

A. Instruction Register-Related Units

1. IR Memory: The IR Memory stores instructions loaded from the Host Memory using the input signal *W_data* and waits for the control unit to load each instruction via the output signal *R_data*. The input signal *addr* from the IR counter determines which instruction is loaded into the control unit. The signals *IR_rd* and *IR_wr* serve as the read enable and write enable signals, respectively, allowing instructions to flow through the IR Memory. The memory size is set to 32 for floorplanning purposes, as increasing it to 64 would cause parallel effect problems.

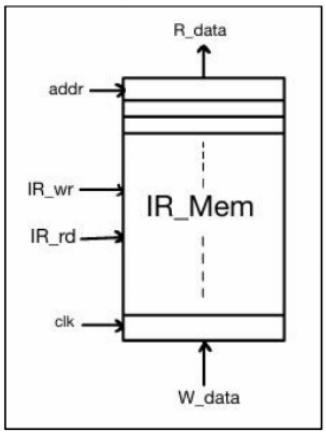


Figure 3. IR Memory.

2. IR counter: The IR counter determines which instruction is loaded into the control unit for decoding. If the *IR_clr* signal from the control unit is set to 1, the address signal *IR_addr* sent to the IR Memory will be 0. Otherwise, it increments by 1, and the IR Memory loads the next instruction.

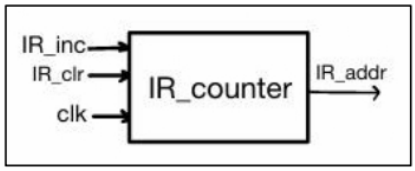


Figure 4. IR Counter.

B. Weight-Related Units

1. Weight DDR3: Similar to the structure of the IR Memory, the Weight DDR3 stores weights sent from the Host Memory through the Weight Interface using the input signal *W_data* and waits for further computation. The output signal *R_data* is connected to the Weight Interface, which propagates the weights to the Weight FIFO. The input signal *addr*, sent by the control unit, determines which weight in the memory location should be loaded. The input signals *rd* and *wr* serve as the read enable and write enable signals, respectively. In this project, the weights are pre-initialized in the Weight DDR3 for simulation instead of being loaded from the Host Memory, as this simplifies the memory structure and data movement.

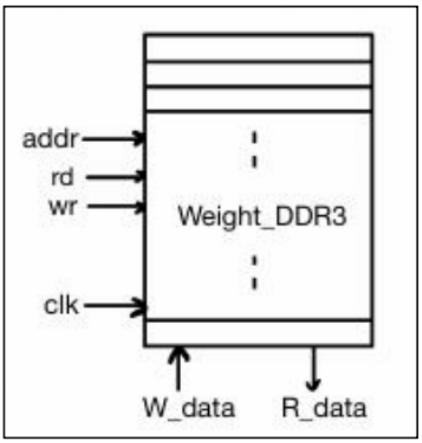


Figure 5. Weight DDR3.

2. Weight Interface: The Weight Interface is located between the Host Memory, Weight DDR3, and Weight FIFOs. In this project, since the weights are pre-initialized in the Weight DDR3, there is no port for loading weights from the Host Memory. When the input signal *push* is triggered, the weights are loaded from the Weight DDR3 using the 8-bit port *weight_in* and temporarily stored in the stack within the Weight Interface. The signal *push_time* determines how many times the weights will be pushed into the stack, while the count signal determines the specific location in the stack where each weight is stored. After all the weights have been pushed into the stack, the *pop* signal is sent to the Weight Interface. The weights stored in the stack are then sequentially popped to the related output ports, which are connected to four Weight FIFOs. Once all the weights in the stack are popped out, the *pop_complete* signal is set to 1 and sent to the control unit, indicating that all weights have been propagated to the Weight FIFOs.

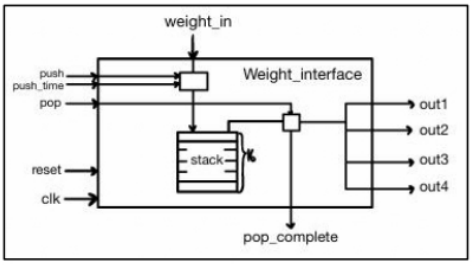


Figure 6. Weight Interface.

3. Weight FIFO: The Weight FIFO is a structure that propagates the input to the output in a first-in-first-out order. When *wr_en* is triggered, *buf_in* sends the weights into the FIFO memory. When *rd_en* is triggered, the weights are sent to *buf_out*, which then forwards them to the Matrix Multiplication Unit. The signals *fifo_counter*, *buf_empty*, and *buf_full* help track the FIFO's storage status.

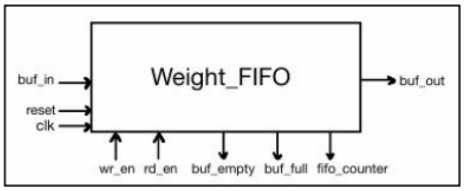


Figure 7. Weight FIFO.

C. Computation-Related Units

Before MMU:

1. Unified Buffer: The Unified Buffer stores activations waiting for matrix multiplication. There are two sources for activations: the Host Memory and the Activation Normalization Unit. In this project, since the activations are pre-initialized in the Unified Buffer, the only source sending data to the Unified Buffer is the Activation Normalization Unit. When rd = 1 and wr = 0, new activations that have completed computation are stored back into the Unified Buffer sequentially at specific locations. When rd = 0 and wr = 1, all values stored in the memory are sent to 16 different output ports, preparing to be sent to the Systolic Data Setup Unit for reordering.

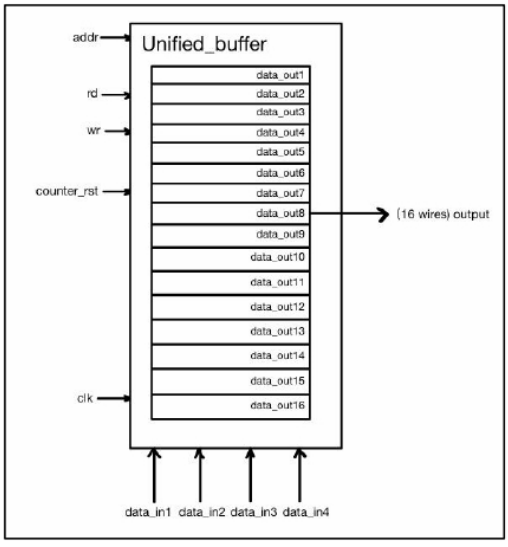


Figure 8. Unified Buffer.

2. Systolic Data Setup Unit: The Systolic Data Setup Unit receives activation values from the Unified Buffer and reorders them into four output ports in a specific sequence. The reordering ensures that the activations are sent to the MMU at the correct timing to perform multiplication with the appropriate output values. The specific order is demonstrated in the simulation results presented in the third paragraph.

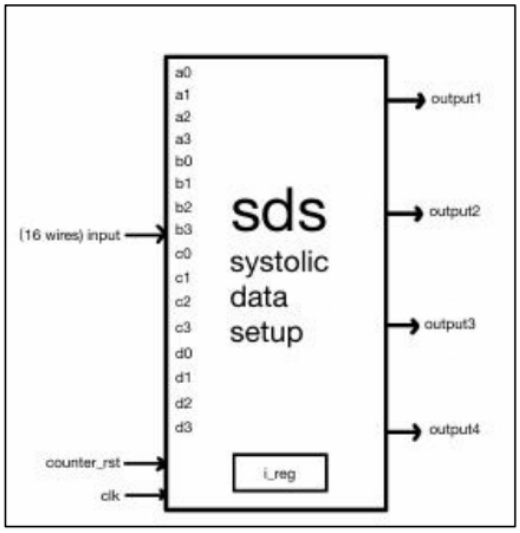


Figure 9. Systolic Data Setup.

3. Activation FIFO: The Activation FIFO has the same structure as the Weight FIFO and propagates input to output in a first-in-first-out order. When wr_en is triggered, buf_in sends activations into the FIFO memory. When rd_en is triggered, the activations are sent to buf_out, which then forwards them to the Matrix Multiplication Unit. The signals fifo_counter, buf_empty, and buf_full help track the FIFO's storage status.

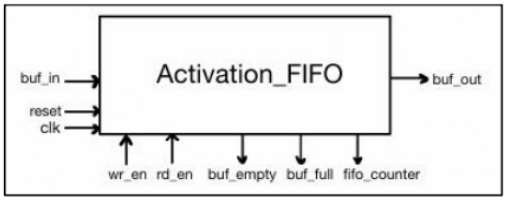


Figure 10. Activation FIFO.

MMU:

1. Row Detector: The Row Detector in the MMU is a unit that helps each systolic cell determine its row location. It facilitates weight loading by ensuring that weights are stored in the weight register at the appropriate time. The input signal *rowsignal* is sent from the control unit, and the output signal *weight_passtime* is connected to the systolic cells in each row.
2. Systolic Cell: The Systolic Cell in the MMU serves as a processing element for matrix multiplication applications.

   When the input signal *weight_pass* is set to 1, the Systolic Cell switches to weight loading mode, loading all the weights from the weight input ports at the top and storing them in the weight register at specific times. While the weights are being loaded, the output signal weightloading is set to 1; otherwise, it is set to 0 to indicate that the weight loading for this cell is complete.

After all the weights are loaded into the weight register, the input signals row_en and col_en will set to 1 to allow the computation processing in that cell, else it will just pass the previous sum values to the next cell.

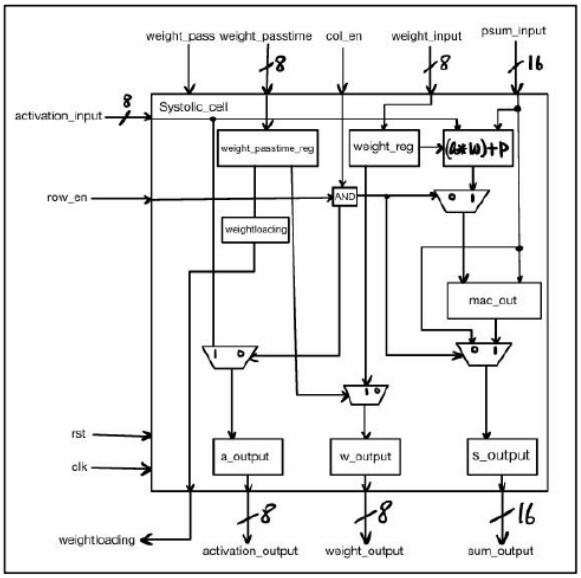


Figure 11. Systolic Cell.

3. Matrix Multiplication Unit: There are 4 Row Detectors and 16 Systolic Cells in the MMU. When the weight loading is complete, the signal weightload_complete is set to 1 to notify the control unit that the weight loading has finished and it can switch to the next state. The 4 output ports are connected to the Accumulator to store the results.

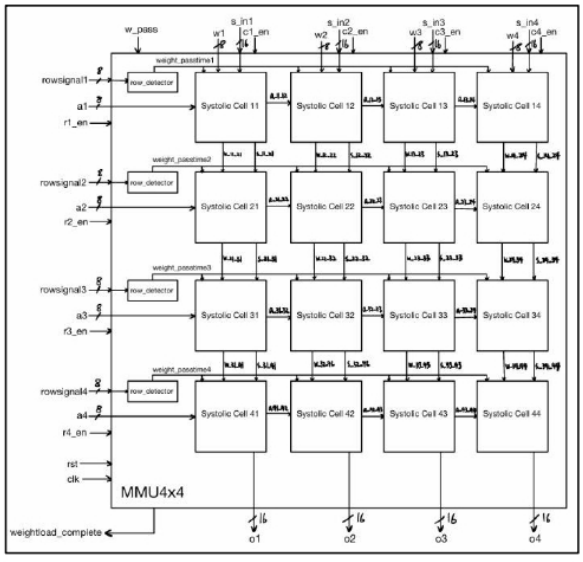


Figure 12. Matrix Multiplication Unit.

After MMU:

1. Accumulator: The current functionality of the Accumulator is to store the results computed by the MMU. The read_index and cycle_counter ensure that the memory stores inputs only at specific times and discards any invalid or outdated values. Once all the relevant values have been stored in the Accumulator's memory, the output signal *Accumulator_finish_storing* is set to 1, notifying the control unit that it can switch to the next state. In future work, the Accumulator will also need to accumulate exponential sum values for activation functions such as Sigmoid or SoftMax.

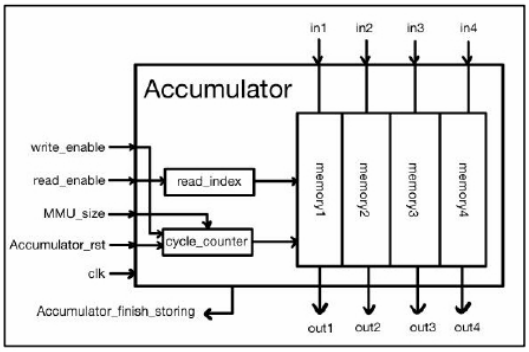


Figure 13. Accumulator.

2. Accumulator: The Activation Normalization Unit receives the *func_select* signal from the control unit. Currently, it supports only the ReLU activation function. Other functions, such as Sigmoid or SoftMax, are planned for future work due to time constraints. The result values are shifted right by 8 bits before being sent to the output ports and subsequently forwarded back to the Unified Buffer.

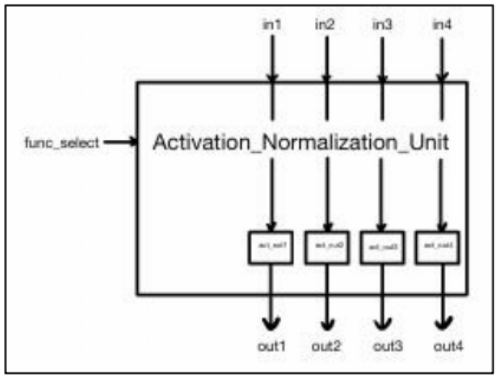


Figure 14. Activation Normalization Unit.

D. Control Unit

The control unit fetches instructions from the DRAM chips through the Host Interface (Control Unit for TPU), decodes them, and manages the operation of each component by coordinating control signals. In this project, the control unit will include a main control module responsible for sending signals and ensuring proper operation of the entire TPU structure, including the Unified Buffer, systolic sequencer, Matrix Multiplication Unit, accumulator, and activation unit. Sub-control units may also be needed to support the functionality of specific components.

The TPU uses a customized CISC ISA format that is not publicly available. To realize its functionality, this project will also attempt to break down all the necessary functions within this structure and encode an ISA format that includes the destination of the Unified Buffer, the size of the matrix, the training time (epochs), and other relevant parameters. The following instructions, along with additional ones, will be implemented in the ISA:

1. Read_Host_Memory: Reads data from the CPU host memory into the Unified Buffer (UB).
2. Read_Weights: Reads weights from Weight Memory into the Weight FIFO as input to the Matrix Unit.
3. Activate: Performs the nonlinear function of the artificial neuron, supporting ReLU, Sigmoid, etc. Its inputs are the Accumulators, and its output is the Unified Buffer. It can also perform the pooling operations required for convolutions using dedicated hardware on the die.
4. Write_Host_Memory: Writes data from the Unified Buffer into the CPU host memory.

In this project, since the Host Memory involves similar memory structures and data movement, Read_Host_Memory and Write_Host_Memory functionalities are deferred for future work. The Controller block diagram is shown below with the associated control signals.

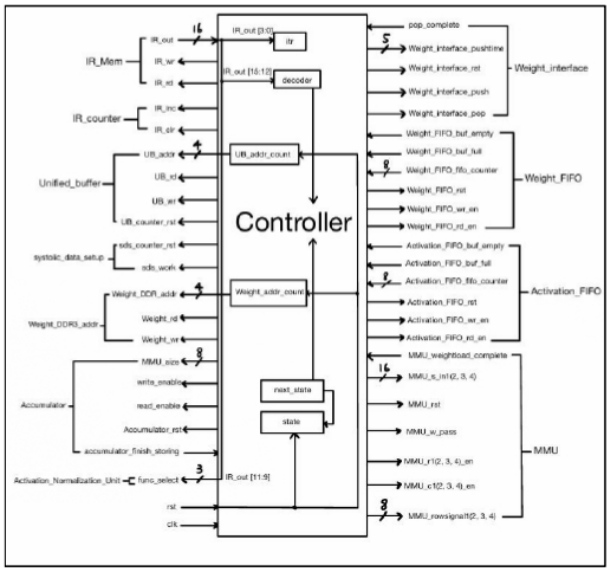


Figure 15. Controller.

III. Controller State Diagram

To better understand each state in the Controller, the complete state diagram is provided, along with the related control signals for each state.

The overall state diagram is shown below. If the opcode (*IR_out[15:12]*) is equal to 0, the Controller will begin the Read_Host_Memory process. If the opcode is 1, the Controller will initiate the Read_Weight process. For an opcode of 2, the Controller will start the Computation process, while an opcode of 3 triggers the Activate process. If the opcode is 4, the Write_Host_Memory process will be executed. For all other opcodes, the Controller transitions to the default stage, which further leads to the S_stay state.

In the initial state, all reset signals and the clear signal are set to 1 to ensure no dirty bits remain in any component. In the fetch state, *IR_rd* is set to 1 to read the instruction. If the opcode is 0, the Read_Host_Memory process will be executed, and then *IR_inc* will be set to 1 to increment the IR memory address location.

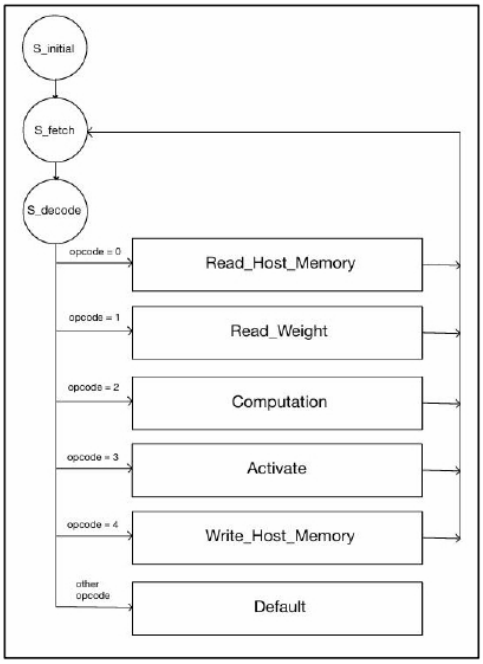


Figure 16. Overall State Diagram.

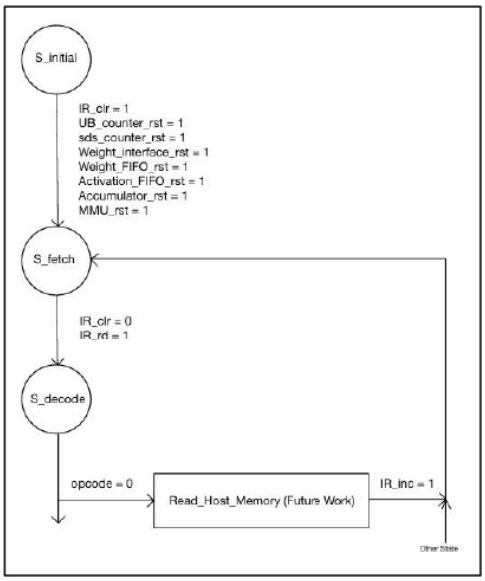


Figure 17. Initial State and Fetch State.

If the opcode is 1, the Read_Weight process will begin. First, the pre-initialized weights located in the Weight DDR3 are loaded into the Weight Interface and temporarily stored in the stack. Once all the weights are stored in the stack within the Weight Interface, the Controller sends the *Weight_interface_pop* signal to instruct the Weight Interface to pop all the weights into the Weight FIFO.

Next, the weights are loaded from the Weight FIFOs into the 16 Systolic Cells located in the Matrix Multiplication Unit (MMU). The *Weight_FIFO_rd_en* signal is set to 1 to enable reading from the FIFOs, and *MMU_w_pass* is set to 1 to allow weight passing and loading into the weight registers in each Systolic Cell. After all the weights are stored in the weight registers, the *MMU_weightload_complete* signal is set to 1, notifying the Controller to load the next instruction.

Finally, the *IR_inc* signal is set to 1 to ensure the IR Memory address increments to the next instruction.

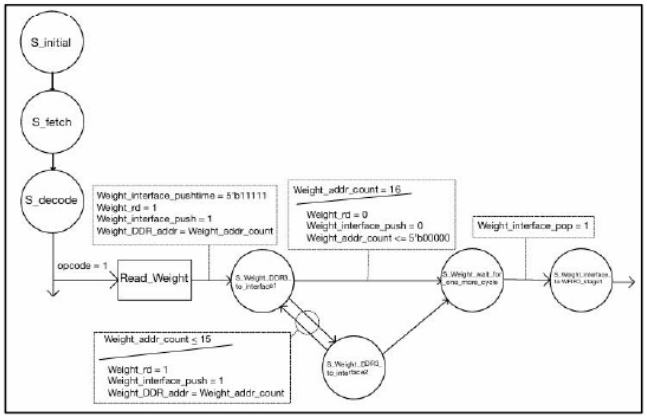


Figure 18. Read_Weight State Diagram Part 1 (from Weight DDR3 to Weight FIFOs).

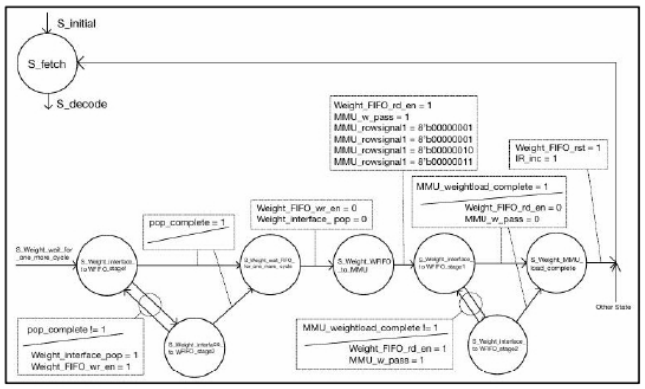


Figure 19. Read_Weight State Diagram Part 2 (from Weight FIFOs to the MMU).

For the Computation state, which occurs when the opcode is 2, the process begins with the pre-initialized activation values located in the Unified Buffer. The Controller sends *UB_rd* = 1 and *UB_wr* = 0 to the Unified Buffer, enabling it to read the values stored in memory. The control signal *sds_work* is set to 1, allowing the Systolic Data Setup Unit to reorder the values. Additionally, *Activation_FIFO_wr_en* is set to 1, enabling the values to be written into the Activation FIFOs, which then forward them to the MMU for computation.

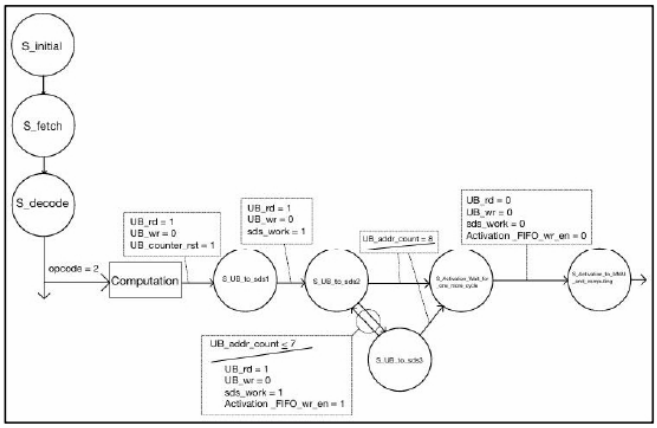


Figure 20. Computation State Diagram Part 1 (from Unified Buffer to Activation FIFOs).

After all activation values have been stored in the Activation FIFOs, the Controller sends the *MMU_row_enable* and *MMU_column_enable* signals to initiate the computation process in the MMU. The *write_enable* signal sent to the Accumulator is also set to 1, allowing the results to be stored in the Accumulator's memory.

During the storing process, the Accumulator generates the *accumulator_finish_storing* signal, which is sent back to the Controller. If this signal is 0, the computation and storing process will continue. Once the *accumulator_finish_storing* signal is set to 1, the Controller sets all enable signals to 0, indicating that the computation is complete and the result values have been successfully stored in the Accumulator.

The *IR_inc* signal is then set to 1 to load the next instruction, which will also determine the activation function to be applied to the stored values.

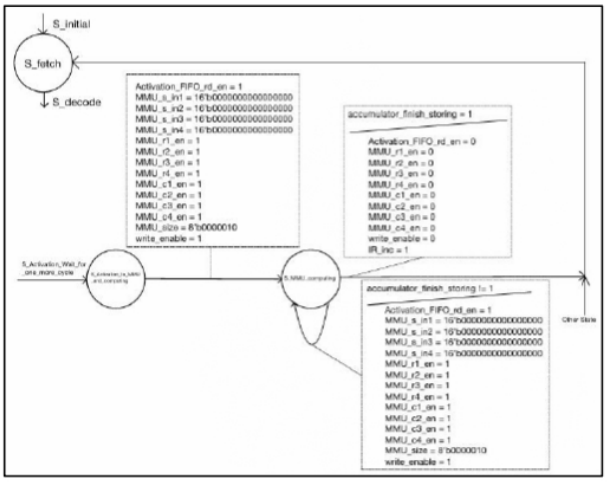


Figure 21. Computation State Diagram Part 2 (from Activation FIFOs to the Accumulator).

After all the result values have been stored in the Accumulator, the Controller loads the next instruction to determine which activation function should be applied to those values. In this project, only the ReLU function is implemented due to time constraints. The Controller then sets *UB_rd* = 0 and *UB_wr* = 1 to instruct the Unified Buffer to perform the write operation. Further details will be discussed in the Future Work and Improvement section.

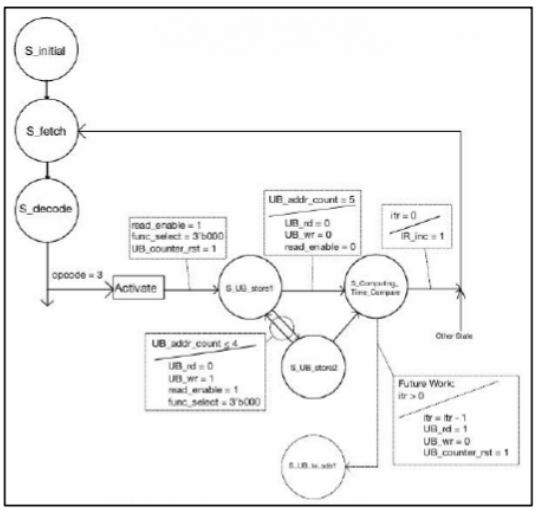


Figure 22. Activate State Diagram.

The remaining state, Write_Host_Memory, will be deferred to future work for the same reasons as Read_Host_Memory. A default state is also implemented to ensure that any invalid opcode transitions to this state, after which the Controller will move to the stay state and wait for a new instruction.

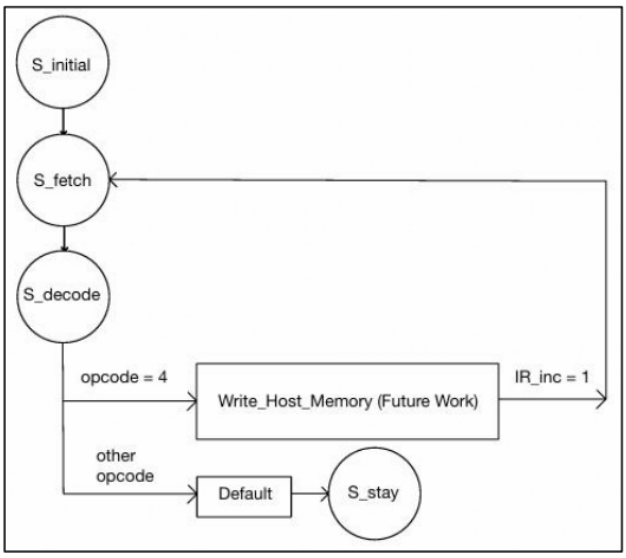


Figure 23. Remaining State.

IV. Simulation Result

The simulation results are shown below. Two pre-initialized datasets are stored in the Weight DDR3 and Unified Buffer. Both datasets contain values from 1 to 16, stored in row-major order in each memory structure. Value 1 is stored at memory[0], value 2 at memory[1], continuing sequentially until value 16 is stored at memory[15].

At time = 20, the Controller fetches the first instruction and decodes it as 0, corresponding to the Read_Host_Memory state. Since further functionality remains part of future work, the Controller simply increments the IR address to fetch the next instruction.

The next instruction is decoded as 1, representing the Read_Weight state. From time = 90 to time = 420, weights are loaded from the Weight DDR3 to the Weight Interface. From time = 430 to time = 530, the weights are propagated from the Weight Interface to the Weight FIFOs. Starting at time = 550, the weights are loaded into the MMU and finish loading at time = 700. By checking the values stored in each weight register within the MMU, it can be confirmed that the values match the pre-initialized weights in the Weight DDR3. After the weights are loaded into the MMU, the Controller fetches the next instruction, as shown after the weight register values are presented.

From time = 710 to time = 940, the activation values stored in the Unified Buffer are loaded into the Systolic Data Setup Unit and subsequently into four Activation FIFOs. By examining the values stored in the Activation FIFOs, the data order is validated. The rightmost values are the first stored in the FIFOs, while the leftmost values are the last.

```
# run 2000ns
State: S_initial
State: S_initial
State: S_initial
time = 20, clk = 0, IR_addr = 00, IR_out: 0000
State: S_fetch
time = 30, clk = 1, IR_addr = 00, IR_out: 0000
time = 40, clk = 0, IR_addr = 00, IR_out: 0000
opcode: 0000
State: Read_Host_Memory (future work)
time = 50, clk = 1, IR_addr = 00, IR_out: 0000
time = 60, clk = 0, IR_addr = 00, IR_out: 0000
State: S_fetch
State: S_fetch
time = 70, clk = 1, IR_addr = 01, IR_out: 1000
time = 80, clk = 0, IR_addr = 01, IR_out: 1000
opcode: 0001
State: Read_Weight
time = 90, clk = 1, IR_addr = 01, IR_out: 1000
time = 100, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 110, clk = 1, IR_addr = 01, IR_out: 1000
time = 120, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 130, clk = 1, IR_addr = 01, IR_out: 1000
time = 140, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 150, clk = 1, IR_addr = 01, IR_out: 1000
time = 160, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 170, clk = 1, IR_addr = 01, IR_out: 1000
time = 180, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 190, clk = 1, IR_addr = 01, IR_out: 1000
time = 200, clk = 0, IR_addr = 01, IR_out: 1000
```

```
State: Weight_load_to_WInterface. . .
time = 210, clk = 1, IR_addr = 01, IR_out: 1000
time = 220, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 230, clk = 1, IR_addr = 01, IR_out: 1000
time = 240, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 250, clk = 1, IR_addr = 01, IR_out: 1000
time = 260, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 270, clk = 1, IR_addr = 01, IR_out: 1000
time = 280, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 290, clk = 1, IR_addr = 01, IR_out: 1000
time = 300, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 310, clk = 1, IR_addr = 01, IR_out: 1000
time = 320, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 330, clk = 1, IR_addr = 01, IR_out: 1000
time = 340, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 350, clk = 1, IR_addr = 01, IR_out: 1000
time = 360, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 370, clk = 1, IR_addr = 01, IR_out: 1000
time = 380, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 390, clk = 1, IR_addr = 01, IR_out: 1000
time = 400, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WInterface. . .
time = 410, clk = 1, IR_addr = 01, IR_out: 1000
time = 420, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight completely loaded into WInterface ! ! !
time = 430, clk = 1, IR_addr = 01, IR_out: 1000
time = 440, clk = 0, IR_addr = 01, IR_out: 1000
```

```
State: Weight_load_to_WFIFO. . .
time = 450, clk = 1, IR_addr = 01, IR_out: 1000
time = 460, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WFIFO. . .
time = 470, clk = 1, IR_addr = 01, IR_out: 1000
time = 480, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WFIFO. . .
time = 490, clk = 1, IR_addr = 01, IR_out: 1000
time = 500, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WFIFO. . .
time = 510, clk = 1, IR_addr = 01, IR_out: 1000
time = 520, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_WFIFO. . .
time = 530, clk = 1, IR_addr = 01, IR_out: 1000
time = 540, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight completely loaded into WFIFO
time = 550, clk = 1, IR_addr = 01, IR_out: 1000
time = 560, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight_load_to_MMU
time = 570, clk = 1, IR_addr = 01, IR_out: 1000
time = 580, clk = 0, IR_addr = 01, IR_out: 1000
State: MMU row detect s1. . .
State: MMU row detect s1. . .
time = 590, clk = 1, IR_addr = 01, IR_out: 1000
time = 600, clk = 0, IR_addr = 01, IR_out: 1000
State: MMU row detect s2. . .
time = 610, clk = 1, IR_addr = 01, IR_out: 1000
time = 620, clk = 0, IR_addr = 01, IR_out: 1000
State: MMU row detect s1. . .
time = 630, clk = 1, IR_addr = 01, IR_out: 1000
time = 640, clk = 0, IR_addr = 01, IR_out: 1000
State: MMU row detect s2. . .
time = 650, clk = 1, IR_addr = 01, IR_out: 1000
time = 660, clk = 0, IR_addr = 01, IR_out: 1000
State: MMU row detect s1. . .
State: MMU row detect s1. . .
```

```
time = 670, clk = 1, IR_addr = 01, IR_out: 1000
time = 680, clk = 0, IR_addr = 01, IR_out: 1000
State: Weight complete loading into MMU
time = 690, clk = 1, IR_addr = 01, IR_out: 1000
time = 700, clk = 0, IR_addr = 01, IR_out: 1000
Value at MMU PE11 W_reg:   1
Value at MMU PE12 W_reg:   2
Value at MMU PE13 W_reg:   3
Value at MMU PE14 W_reg:   4
Value at MMU PE21 W_reg:   5
Value at MMU PE22 W_reg:   6
Value at MMU PE23 W_reg:   7
Value at MMU PE24 W_reg:   8
Value at MMU PE31 W_reg:   9
Value at MMU PE32 W_reg:   10
Value at MMU PE33 W_reg:   11
Value at MMU PE34 W_reg:   12
Value at MMU PE41 W_reg:   13
Value at MMU PE42 W_reg:   14
Value at MMU PE43 W_reg:   15
Value at MMU PE44 W_reg:   16
State: S_fetch
State: S_fetch
State: S_fetch
time = 710, clk = 1, IR_addr = 02, IR_out: 2001
time = 720, clk = 0, IR_addr = 02, IR_out: 2001
opcode: 0010
State: Computation
time = 730, clk = 1, IR_addr = 02, IR_out: 2001
time = 740, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s1. . .
time = 750, clk = 1, IR_addr = 02, IR_out: 2001
time = 760, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s2. . .
time = 770, clk = 1, IR_addr = 02, IR_out: 2001
time = 780, clk = 0, IR_addr = 02, IR_out: 2001
```

```
State: Activation_load_to_AFIFO s3. . .
time = 790, clk = 1, IR_addr = 02, IR_out: 2001
time = 800, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s2. . .
time = 810, clk = 1, IR_addr = 02, IR_out: 2001
time = 820, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s3. . .
time = 830, clk = 1, IR_addr = 02, IR_out: 2001
time = 840, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s2. . .
time = 850, clk = 1, IR_addr = 02, IR_out: 2001
time = 860, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s3. . .
time = 870, clk = 1, IR_addr = 02, IR_out: 2001
time = 880, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation_load_to_AFIFO s2. . .
time = 890, clk = 1, IR_addr = 02, IR_out: 2001
time = 900, clk = 0, IR_addr = 02, IR_out: 2001
time = 910, clk = 1, IR_addr = 02, IR_out: 2001
time = 920, clk = 0, IR_addr = 02, IR_out: 2001
State: Activation completely loaded into AFIFO
time = 930, clk = 1, IR_addr = 02, IR_out: 2001
time = 940, clk = 0, IR_addr = 02, IR_out: 2001
Value at AFIF01:   x   0   0   0  13   9   5   1
Value at AFIF02:   x   0   0  14  10   6   2   x
Value at AFIF03:   x   0  15  11   7   3   0   x
Value at AFIF04:   x  16  12   8   4   0   0   x
State: Activation start loading into MMU
time = 950, clk = 1, IR_addr = 02, IR_out: 2001
time = 960, clk = 0, IR_addr = 02, IR_out: 2001
State: MMU Computing. . .
time = 970, clk = 1, IR_addr = 02, IR_out: 2001
time = 980, clk = 0, IR_addr = 02, IR_out: 2001
time = 990, clk = 1, IR_addr = 02, IR_out: 2001
time = 1000, clk = 0, IR_addr = 02, IR_out: 2001
time = 1010, clk = 1, IR_addr = 02, IR_out: 2001
```

```
time = 1020, clk = 0, IR_addr = 02, IR_out: 200.
time = 1030, clk = 1, IR_addr = 02, IR_out: 200.
time = 1040, clk = 0, IR_addr = 02, IR_out: 200.
time = 1050, clk = 1, IR_addr = 02, IR_out: 200.
time = 1060, clk = 0, IR_addr = 02, IR_out: 200.
time = 1070, clk = 1, IR_addr = 02, IR_out: 200.
time = 1080, clk = 0, IR_addr = 02, IR_out: 200.
time = 1090, clk = 1, IR_addr = 02, IR_out: 200.
time = 1100, clk = 0, IR_addr = 02, IR_out: 200.
time = 1110, clk = 1, IR_addr = 02, IR_out: 200.
time = 1120, clk = 0, IR_addr = 02, IR_out: 200.
time = 1130, clk = 1, IR_addr = 02, IR_out: 200.
time = 1140, clk = 0, IR_addr = 02, IR_out: 200.
time = 1150, clk = 1, IR_addr = 02, IR_out: 200.
time = 1160, clk = 0, IR_addr = 02, IR_out: 200.
time = 1170, clk = 1, IR_addr = 02, IR_out: 200.
time = 1180, clk = 0, IR_addr = 02, IR_out: 200.
State: Values finishing loading into Accumulator!!!
time = 1190, clk = 1, IR_addr = 02, IR_out: 200.
Value at Accumulator (in Matrix Form):   90   .00   110   120
                                        202   228   254   280
                                        314   356   398   440
                                        426   484   542   600
time = 1200, clk = 0, IR_addr = 02, IR_out: 200.
State: S_fetch
State: S_fetch
time = 1210, clk = 1, IR_addr = 03, IR_out: 3000
time = 1220, clk = 0, IR_addr = 03, IR_out: 3000
opcode: 0011
State: Activate
time = 1230, clk = 1, IR_addr = 03, IR_out: 3000
time = 1240, clk = 0, IR_addr = 03, IR_out: 3000
State: UB storing s1. . .
time = 1250, clk = 1, IR_addr = 03, IR_out: 3000
time = 1260, clk = 0, IR_addr = 03, IR_out: 3000
State: UB storing s2. . .
```

From time = 970 to time = 1190, the values stored in the Activation FIFOs are sent to the MMU for computation and then loaded into the Accumulator after computation is complete. By verifying the values stored in the Accumulator, the results are confirmed to be correct, proving that the functionality of all components up to this point is working correctly.

Next, the Controller fetches the next instruction to determine which activation function should be applied to the values, and the normalized result is then sent back to the Unified Buffer. In this project, only the ReLU function is applied to the values. After the activation function is applied, the values are normalized to reduce their bit size and precision from 16-bit to 8-bit, and they are then sent back to the Unified Buffer.

By examining the new values stored in the Unified Buffer, the first seven values are equal to 0. This occurs because any value less than 256 becomes 0 after a right-shift operation by 8 bits. The next seven values are equal to 1, as values between 256 and 511 become 1 after the same right-shift operation. The remaining two values are both equal to 2, which is the correct result for values greater than 512 after performing a right-shift by 8 bits.

```
time = 1270, clk = 1, IR_addr = 03, IR_out: 3000
time = 1280, clk = 0, IR_addr = 03, IR_out: 3000
State: UB storing s1. . .
time = 1290, clk = 1, IR_addr = 03, IR_out: 3000
time = 1300, clk = 0, IR_addr = 03, IR_out: 3000
State: UB storing s2. . .
time = 1310, clk = 1, IR_addr = 03, IR_out: 3000
time = 1320, clk = 0, IR_addr = 03, IR_out: 3000
State: UB storing s1. . .
State: Values finishing storing back to Unified Buffer
time = 1330, clk = 1, IR_addr = 03, IR_out: 3000
time = 1340, clk = 0, IR_addr = 03, IR_out: 3000
Value at UB Mem[0]:   0
Value at UB Mem[1]:   0
Value at UB Mem[2]:   0
Value at UB Mem[3]:   0
Value at UB Mem[4]:   0
Value at UB Mem[5]:   0
Value at UB Mem[6]:   0
Value at UB Mem[7]:   1
Value at UB Mem[8]:   1
Value at UB Mem[9]:   1
Value at UB Mem[10]:   1
Value at UB Mem[11]:   1
Value at UB Mem[12]:   1
Value at UB Mem[13]:   1
Value at UB Mem[14]:   2
Value at UB Mem[15]:   2
State: Finish all the computation
time = 1350, clk = 1, IR_addr = 03, IR_out: 3000
time = 1360, clk = 0, IR_addr = 03, IR_out: 3000
State: S_fetch
State: S_fetch
time = 1370, clk = 1, IR_addr = 04, IR_out: 4000
time = 1380, clk = 0, IR_addr = 04, IR_out: 4000
opcode: 0100
```

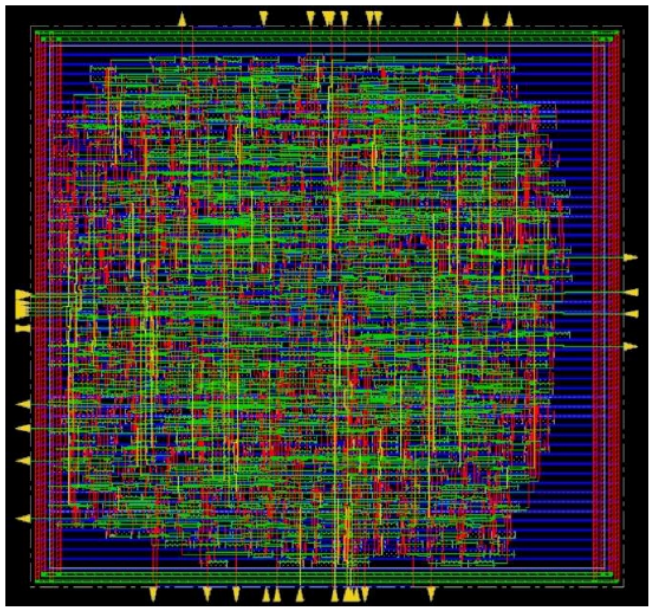
```
State: Write_Host_Memory (future work)
time = 1390, clk = 1, IR_addr = 04, IR_out: 4000
time = 1400, clk = 0, IR_addr = 04, IR_out: 4000
State: S_fetch
State: S_fetch
time = 1410, clk = 1, IR_addr = 05, IR_out: xxxx
time = 1420, clk = 0, IR_addr = 05, IR_out: xxxx
opcode: xxxx
Invalid opcode
time = 1430, clk = 1, IR_addr = 05, IR_out: xxxx
time = 1440, clk = 0, IR_addr = 05, IR_out: xxxx
State: S_stay
time = 1450, clk = 1, IR_addr = 05, IR_out: xxxx
time = 1460, clk = 0, IR_addr = 05, IR_out: xxxx
time = 1470, clk = 1, IR_addr = 05, IR_out: xxxx
$finish called at time : 1480 ns : File "C:/Xilinx/project/CA_project_16x16TPU/TPU4X4/TPU4X4.srcs/sim_1/new/TPU4X4_tb.v" Line 92
INFO: [USF-XSim-96] XSim completed. Design snapshot 'TPU4X4_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 2000ns
launch_simulation: Time (s): cpu = 00:00:00 ; elapsed = 00:00:08 . Memory (MB): peak = 1540.988 ; gain = 42.793
```

At time = 1370, after all previous operations are completed, the Controller fetches the next instruction and decodes it as Write_Host_Memory, which is designated as future work. For now, the Controller simply fetches the next instruction. The final instruction is invalid and causes the Controller to remain in the stay state.

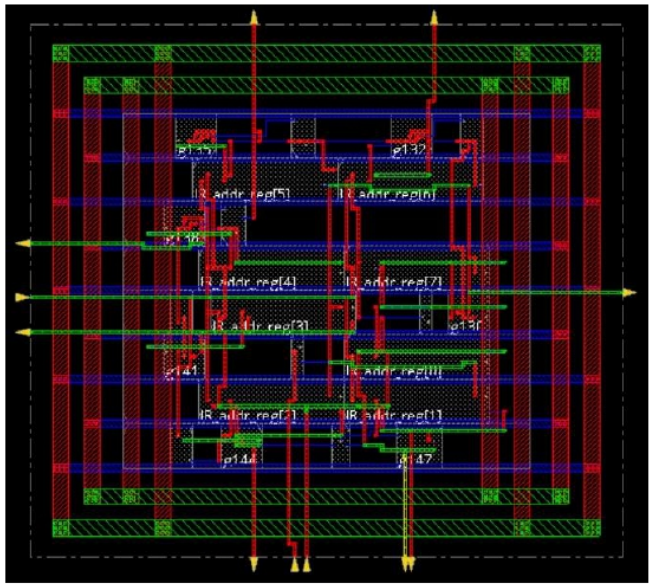
V. Floorplanning

To support potential future fabrication, floor plans for each component are created using Cadence Innovus. The floorplanning results are shown below, with the timing report and verification results for geometry and connectivity provided in the Appendix.

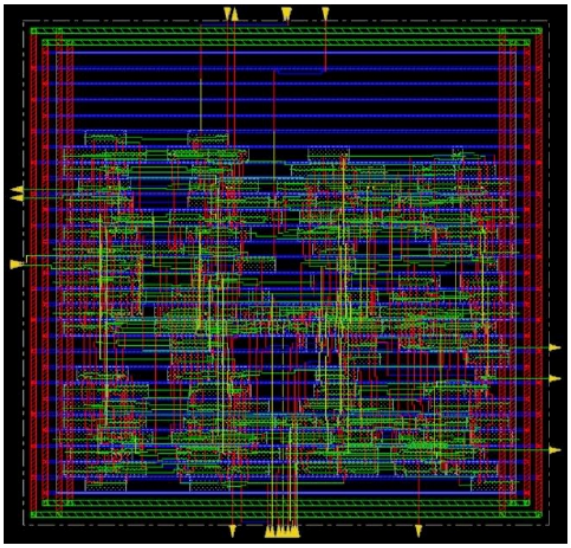
IR Memory:



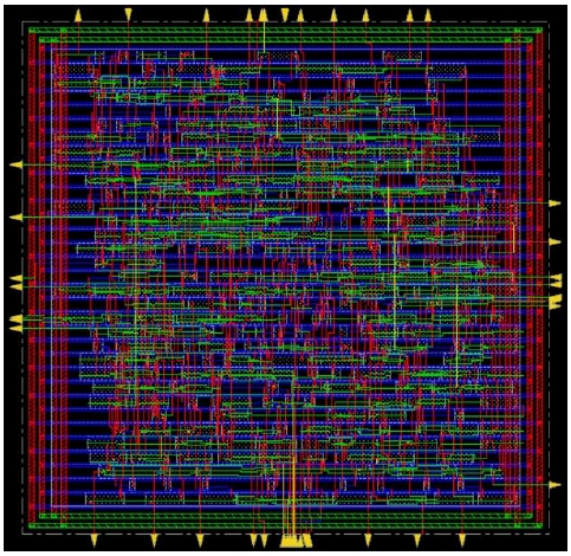
IR Counter:



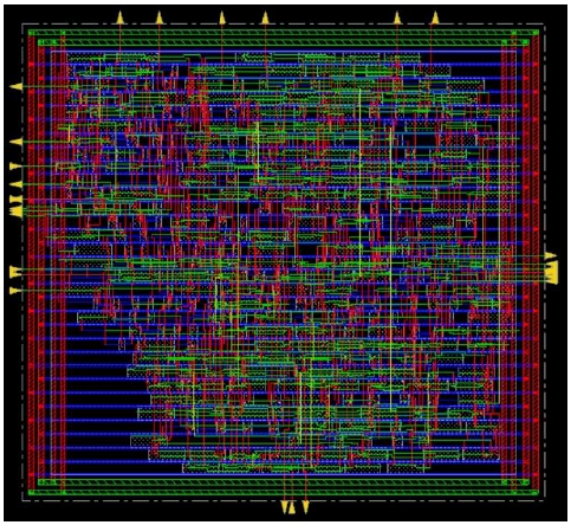
Weight DDR3:



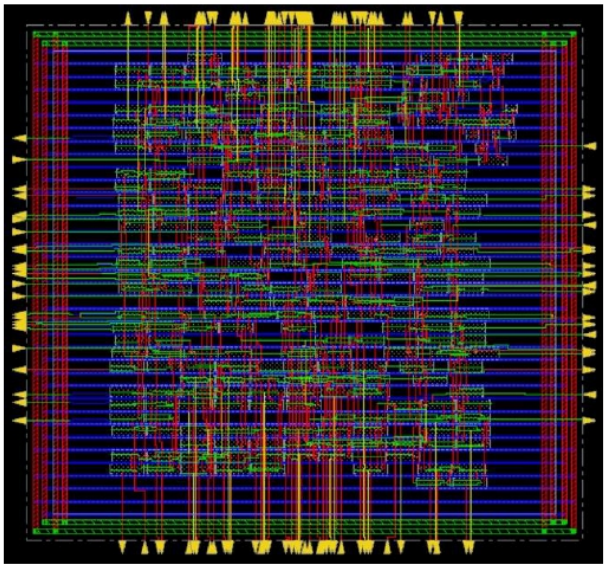
Weight Interface:



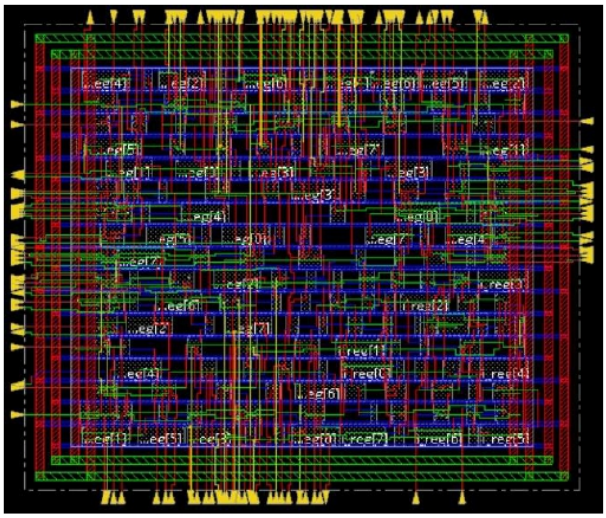
Weight FIFO:



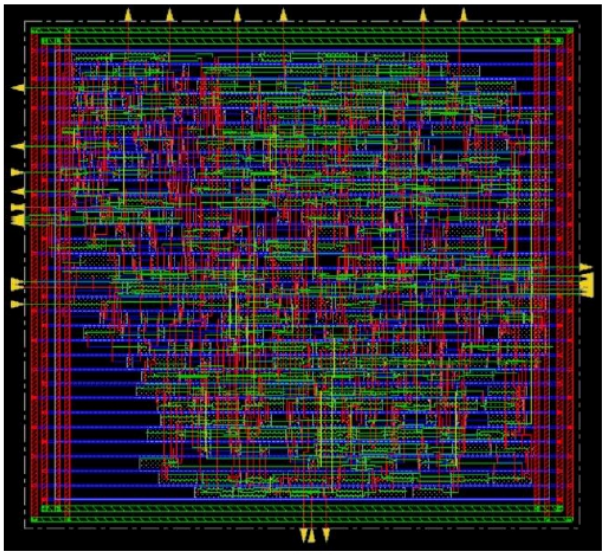
Unified Buffer:



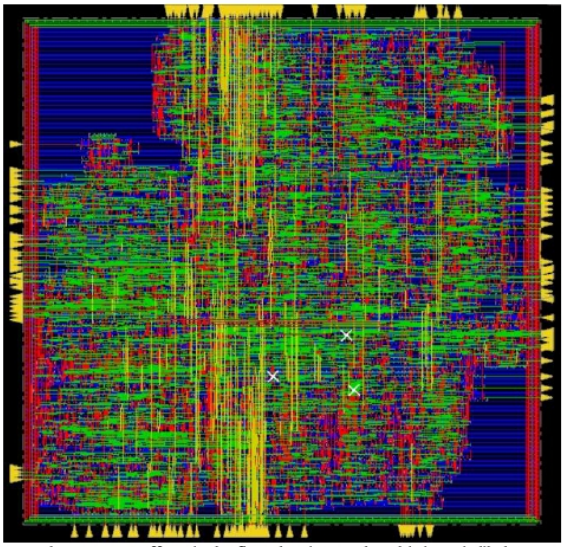
Systolic Data Setup:



Activation FIFO:

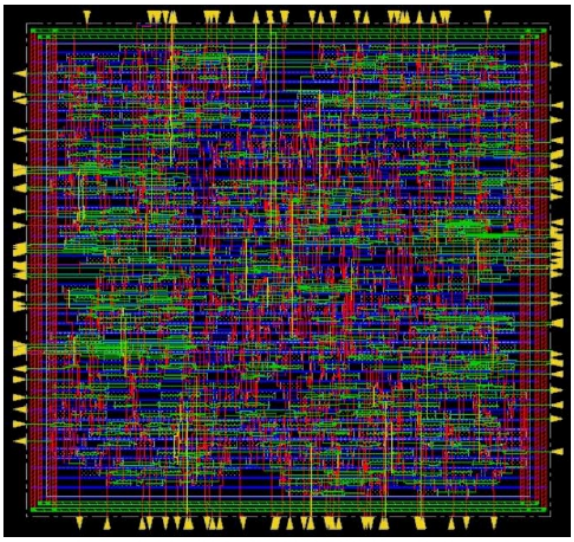


Matrix Multiplication Unit:

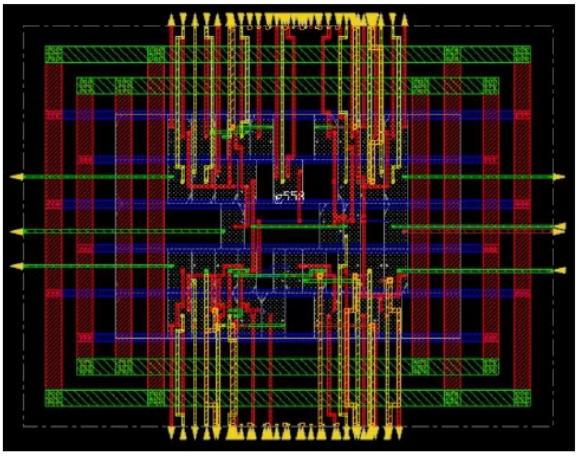


* There are three antenna effects in the floorplanning results. This issue is likely caused by a scaling problem and could potentially be resolved by adjusting the scaling parameters in the floorplanning process.

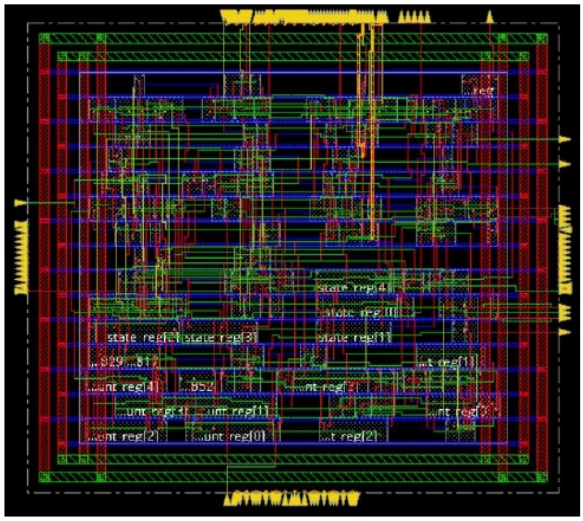
Accumulator:



Activation Normalization Unit:

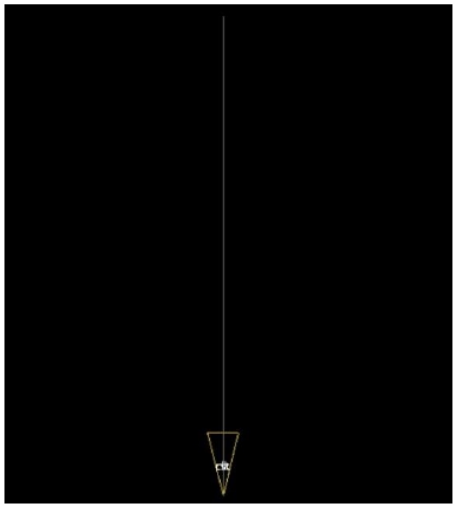


Control Unit:



Top module:

The floorplanning result for the top module appears as a line because the top module in the Verilog code only has two inputs: the clock signal and the reset signal. As a result, the software currently builds ports for these two signals but does not implement the internal structure of the module. In future work and real-world applications, additional ports for weights, activation values, and instructions will be added. These ports will connect the TPU structure to the Host Memory, potentially allowing Innovus to build the complete structure.



VI. Future Work and Improvement

Some works in this project can be improved and potential future work will be discussed in this section.

1. Optimizing State Design and Introducing Pipelining:

The first issue to address is the inefficiency in the state design. Since this is the first attempt at implementing such a large module, the control signals are somewhat disorganized, and many states could be removed to make the state diagram more concise and efficient.

Another potential improvement is the introduction of pipelining. Currently, the Controller waits for the weights to be fully stored in the MMU before fetching the next instruction,

which then triggers the loading of activation values from the Unified Buffer into the Activation FIFOs. By implementing pipelining, weights and activation values could propagate simultaneously, provided they do not access the same unit at the same time, thereby avoiding resource conflicts.

2. *Implementation of Host Memory for Enhanced Functionality:*

Although the Host Memory is a duplicated memory structure, it still needs to be implemented in the future to support additional states and their corresponding control signals, making the project more complete.

With the implementation of the Host Memory, the top module can incorporate more I/O ports, which may enable Innovus to generate a complete floorplanning result for the top module.

3. *Enhancing Iterative Computation and Resolving Pre-Initialization Conflicts:*

A major issue encountered during simulation result verification is that the computation can only run for a single iteration. An attempt was made in this project to determine how many iterations the MMU should compute for the same set of values. However, after the first set of results from the MMU is stored in the Unified Buffer and verified, the next set of values loaded into the Activation FIFOs remains the same as the pre-initialized values in the Unified Buffer. This issue persists even when the Controller fetches a new instruction for computation.

A potential reason for this problem could be the pre-initialized sets of values. To address this, a master-slave Unified Buffer architecture could be implemented. The pre-initialized values would be stored in the master Unified Buffer, while the slave Unified Buffer would receive a duplicate set from the master UB and serve as the primary component connected to the Systolic Data Setup and Activation Normalization Unit. This setup would allow verification of whether the pre-initialized values cause issues when attempting to reload the slave Unified Buffer multiple times.

Similarly, implementing the Host Memory could also help resolve this problem, as the pre-initialized values could be stored in the Host Memory instead of the Unified Buffer. This would separate the initialization process from the computation, potentially eliminating conflicts caused by reloading values.

4. *Addressing Antenna Effects and Optimizing Floorplanning Parameters:*

Addressing violations in the MMU structure floorplanning is another area that should be a focus for future improvement. Figure 24 illustrates the settings that can be modified. By adjusting parameters such as core utilization and core margins, Innovus can generate different floorplanning results. A common issue arises when the floorplan is too small to accommodate all the wires for a specific module, leading to parallel effects or antenna effects.

Due to limited training in Innovus, understanding how to effectively optimize these parameters remains a challenge and requires further study. However, ensuring that all floorplanning results have zero violations should be a primary goal to enhance the completeness and quality of this project or any future work.

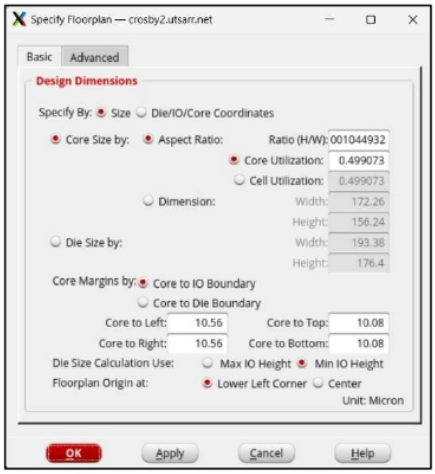


Figure 24. Floorplan Settings Page in Innovus.

VII. Conclusion

This project successfully implemented a scaled-down version of the Tensor Processing Unit (TPU) to gain insights into its microarchitecture and data flow. Key components, including the Matrix Multiplication Unit (MMU), Unified Buffer, Weight Interface, and Control Unit, were implemented using Verilog, and their functionality was validated through detailed simulation. The implementation demonstrated the ability to manage data propagation, control signals, and computation processes effectively. The floorplanning results, generated using Cadence Innovus, provided an initial understanding of how the TPU's physical structure might be organized for future fabrication, although some violations, such as antenna effects, remain to be resolved. This project serves as a significant step toward understanding TPU architecture and offers valuable experience in hardware design and system integration.

Despite these achievements, several areas for improvement have been identified. The current state diagram could be optimized to reduce redundancy and improve efficiency, while introducing pipelining could allow weights and activation values to propagate simultaneously, enhancing computation throughput. The implementation of Host Memory and a master-slave Unified Buffer architecture could address issues with single-iteration computations and data reloading conflicts. Additionally, further exploration of floorplanning parameters, such as core utilization and margins, could resolve existing violations and improve the quality of the floorplanning results.

In conclusion, this project lays a strong foundation for future projects and development in custom hardware architectures for machine learning. The experience of implementing a Mini TPU from scratch, including the module code, testbench, and control unit, provides valuable insights into the structure of a TPU. More importantly, it serves as a starting point for learning how to design a complete hardware architecture, addressing potential challenges, and identifying areas for improvement.

REFERENCES

[1] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Yoon, D. H. (2017, June). In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture (pp. 1-12).

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) (pp. 265-283).

[3] L. Bottou et al., "Comparison of classifier methods: a case study in handwritten digit recognition," Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5), Jerusalem, Israel, 1994, pp. 77-82 vol.2, doi: 10.1109/ICPR.1994.576879.

[4] Ross, J., Jouppi, N., Phelps, A., Young, C., Norrie, T., Thorson, G.,Luu, D., 2015. Neural Network Processor, Patent Application No. 62/164,931.

[5] Ross, J., Phelps, A., 2015. Computing Convolutions Using a Neural Network Processor, Patent Application No. 62/164,902.

[6] Ross, J., 2015. Prefetching Weights for a Neural Network Processor, Patent Application No. 62/164,981.

[7] Ross, J., Thorson, G., 2015. Rotating Data for Neural Network Computations, Patent Application No. 62/164,908.

[8] Thorson, G., Clark, C., Luu, D., 2015. Vector Computation Unit in a Neural Network Processor, Patent Application No. 62/165,022.

[9] Young, C., 2015. Batch Processing in a Neural Network Processor, Patent Application No. 62/165,020

APPENDIX

Verilog code:

IR Memory:

```
module IR_Mem(R_data, W_data, addr, clk, IR_rd, IR_wr);
    parameter address_size = 8;
    parameter word_size = 16;
    parameter memory_size = 32;
    output [word_size-1:0] R_data;
    input [word_size-1:0] W_data;
    input [address_size-1:0] addr;
    input clk, IR_rd, IR_wr;
    reg [word_size-1:0] memory [memory_size-1:0];

always @(posedge clk) begin
    if (IR_wr) begin
        memory[addr] <= W_data;
        end
    end

assign R_data = memory[addr];

endmodule
```

Weight DDR3:

```
module Weight_DDR3(R_data, W_data, addr, clk, rd, wr);
    parameter address_size = 4;
    parameter word_size = 8;
    parameter memory_size = 16;
    output [word_size-1:0] R_data;
    input [word_size-1:0] W_data;
    input [address_size:0] addr;
    input clk, rd, wr;
    reg [word_size-1:0] memory [memory_size-1:0];

always @(posedge clk) begin
    if (wr) begin
        memory[addr] <= W_data;
        end
    end

assign R_data = memory[addr];

endmodule
```

IR Counter:

```
module IR_counter(IR_addr, IR_inc, IR_clr, clk);
    parameter word_size = 8;
    output reg [word_size-1:0] IR_addr;
    input  IR_inc, IR_clr, clk;

always@(posedge clk or posedge IR_clr) begin
    if(IR_clr==1) IR_addr <= 8'b0;
        else if (IR_inc == 1) IR_addr <= IR_addr + 1;
    end
endmodule
```

Weight FIFO:

```
module Weight_FIFO(clk, rst, buf_in, buf_out, wr_en, rd_en, buf_empty, buf_full, fifo_counter);
    input clk, rst, wr_en, rd_en;
    input [7:0] buf_in;
    output [7:0] buf_out;
    output buf_empty, buf_full;
    output [7:0] fifo_counter;
    reg [7:0] buf_out;
    reg buf_empty, buf_full;
    reg [7:0] fifo_counter;
    reg [3:0] rd_ptr, wr_ptr;
    reg [7:0] buf_mem[63:0];

always @(fifo_counter) begin
    buf_empty = (fifo_counter==0);
    buf_full = (fifo_counter==64);
    end

always @(posedge clk or posedge rst) begin
    if(rst)
        fifo_counter <= 0;
        else if ( (!buf_full && wr_en) && (!buf_empty && rd_en) )
            fifo_counter <= fifo_counter;
        else if ( !buf_full && wr_en )
            fifo_counter <= fifo_counter + 1;
        else if ( !buf_empty && rd_en )
            fifo_counter <= fifo_counter - 1;
        else
        fifo_counter <= fifo_counter;
    end

always @(posedge clk or posedge rst) begin
    if(rst)
        buf_out <= 0;
            else begin
                if(rd_en && !buf_empty)
                    buf_out <= buf_mem[rd_ptr];
                    else
                    buf_out <= buf_out;
                    end
                end

always @(posedge clk) begin
    if(wr_en && !buf_full)
        buf_mem[wr_ptr] <= buf_in;
        else
            buf_mem[wr_ptr] <= buf_mem[wr_ptr];
            end

always @(posedge clk or posedge rst) begin
    if(rst) begin
        wr_ptr <= 0;
        rd_ptr <= 0;
        end else begin
            if(!buf_full && wr_en)
                wr_ptr <= wr_ptr + 1;
                else
                wr_ptr <= wr_ptr;
            if(!buf_empty && rd_en)
                rd_ptr <= rd_ptr + 1;
                else
                rd_ptr <= rd_ptr;
        end
    end
endmodule
```

Weight Interface:

```
module Weight_interface(clk, reset, weight_in, push_time, push, pop, pop_complete, out1, out2,
out3, out4);
    parameter STACK_SIZE = 16;
    reg [7:0] stack [STACK_SIZE-1:0];
    reg [4:0] count;
    input clk;
    input reset;
    input push, pop;
    input [7:0] weight_in;
    input [4:0] push_time;
    output reg pop_complete;
    output reg [7:0] out1;
    output reg [7:0] out2;
    output reg [7:0] out3;
    output reg [7:0] out4;

    reg [4:0] pushtime;
    reg [4:0] pop_count;

always @(posedge clk or posedge reset) begin
    if (reset) begin
        pushtime <= 5'b0;
        end else begin
            pushtime <= push_time;
            end
        end

always @(posedge clk or posedge reset) begin
    if (reset) begin
        count <= 0;
        end else if (push) begin
            stack[count] <= weight_in;
            count <= count + 1;
            end
        end

always @(posedge clk or posedge reset) begin
    if (reset) begin
        pop_count <= 0;
        out1 <= 8'b0;
        out2 <= 8'b0;
        out3 <= 8'b0;
        out4 <= 8'b0;
        pop_complete <= 1'b0;
        end else if (pop) begin
            case (pop_count)
                0: begin
                out4 <= stack[15];
                out3 <= stack[14];
                out2 <= stack[13];
                out1 <= stack[12];
                pop_complete <= 1'b0;
                end
                1: begin
                out4 <= stack[11];
                out3 <= stack[10];
                out2 <= stack[9];
                out1 <= stack[8];
                pop_complete <= 1'b0;
                end
                2: begin
                out4 <= stack[7];
                out3 <= stack[6];
                out2 <= stack[5];
                out1 <= stack[4];
                pop_complete <= 1'b0;
                end
                3: begin
                out4 <= stack[3];
                out3 <= stack[2];
                out2 <= stack[1];
                out1 <= stack[0];
                pop_complete <= 1'b0;
                end
                4: begin
                pop_complete <= 1'b1;
                end
                default: begin
                out1 <= 8'bx;
                out2 <= 8'bx;
                out3 <= 8'bx;
                out4 <= 8'bx;
                end
                endcase
                pop_count <= pop_count + 1;
                end
        end
endmodule
```

Unified Buffer:

```
module unified_buffer(addr, clk, rd, wr, counter_rst, data_in1, data_in2, data_in3,
data_in4, data_out1, data_out2, data_out3, data_out4, data_out5, data_out6, data_out7,
data_out8, data_out9, data_out10, data_out11, data_out12, data_out13, data_out14,
data_out15, data_out16);
    parameter BIT_WIDTH = 8;
    parameter address_size = 4;
    parameter word_size = 8;
    parameter memory_size = 16;
    input [word_size-1:0] data_in1, data_in2, data_in3, data_in4;
    output reg [word_size-1:0] data_out1, data_out2, data_out3, data_out4, data_out5,
    data_out6, data_out7, data_out8, data_out9, data_out10, data_out11, data_out12,
    data_out13, data_out14, data_out15, data_out16;
    input [word_size-1:0] W_data;
    input [address_size-1:0] addr;
    input counter_rst;
    input clk;
    input rd, wr;

    reg [word_size-1:0] memory [memory_size-1:0];
    reg [BIT_WIDTH-1:0] i;

always @(posedge clk) begin
    if (counter_rst) begin
        i <= 0;
        end else begin
            i <= i + 1;
        end
    end

always @(posedge clk) begin
    if (wr && !rd) begin
        case (i)
        3'd0: begin
        memory[0] <= data_in1;
        memory[1] <= data_in2;
        memory[2] <= data_in3;
        memory[3] <= data_in4;
        end
        3'd1: begin
        memory[4] <= data_in1;
        memory[5] <= data_in2;
        memory[6] <= data_in3;
        memory[7] <= data_in4;
        end
        3'd2: begin
        memory[8] <= data_in1;
        memory[9] <= data_in2;
        memory[10] <= data_in3;
        memory[11] <= data_in4;
        end
        3'd3: begin
        memory[12] <= data_in1;
        memory[13] <= data_in2;
        memory[14] <= data_in3;
        memory[15] <= data_in4;
        end
        default: ;
        endcase
    end
    if (rd && !wr) begin
        data_out1 <= memory[0];
        data_out2 <= memory[1];
        data_out3 <= memory[2];
        data_out4 <= memory[3];
        data_out5 <= memory[4];
        data_out6 <= memory[5];
        data_out7 <= memory[6];
        data_out8 <= memory[7];
        data_out9 <= memory[8];
        data_out10 <= memory[9];
        data_out11 <= memory[10];
        data_out12 <= memory[11];
        data_out13 <= memory[12];
        data_out14 <= memory[13];
        data_out15 <= memory[14];
        data_out16 <= memory[15];
        end
    end
endmodule
```

Systolic Data Setup:

```
module sds4x4(clk, counter_rst, sds_work, a0, a1, a2, a3, b0, b1, b2, b3, c0, c1, c2, c3,
d0, d1, d2, d3, output1, output2, output3, output4);
    parameter BIT_WIDTH = 8;
    input clk;
    input counter_rst;
    input sds_work;
    input [BIT_WIDTH-1:0] a0, a1, a2, a3, b0, b1, b2, b3, c0, c1, c2, c3, d0, d1, d2, d3;
    output reg [BIT_WIDTH-1:0] output1, output2, output3, output4;

    reg [BIT_WIDTH-1:0] i;

always @(posedge clk) begin
    if (counter_rst) begin
        i <= 0;
        end else if (sds_work) begin
            i <= i + 1;
            end
        end

always @(posedge clk) begin
    if (sds_work) begin
        case (i)
            3'd0: begin
            output1 <= a0;
            end
            3'd1: begin
            output1 <= b0;
            output2 <= a1;
            output3 <= 8'b0;
            output4 <= 8'b0;
            end
            3'd2: begin
            output1 <= c0;
            output2 <= b1;
            output3 <= a2;
            output4 <= 8'b0;
            end
            3'd3: begin
            output1 <= d0;
            output2 <= c1;
            output3 <= b2;
            output4 <= a3;
            end
            3'd4: begin
            output1 <= 8'b0;
            output2 <= d1;
            output3 <= c2;
            output4 <= b3;
            end
            3'd5: begin
            output1 <= 8'b0;
            output2 <= 8'b0;
            output3 <= d2;
            output4 <= c3;
            end
            3'd6: begin
            output1 <= 8'b0;
            output2 <= 8'b0;
            output3 <= 8'b0;
            output4 <= d3;
            end
            default: begin
            output1 <= 8'b0;
            output2 <= 8'b0;
            output3 <= 8'b0;
            output4 <= 8'b0;
            end
        endcase
    end
end
endmodule
```

Activation FIFO:

```
module Activation_FIFO (clk, rst, buf_in, buf_out, wr_en, rd_en, buf_empty,
buf_full, fifo_counter);
    input clk, rst, wr_en, rd_en;
    input [7:0] buf_in;
    output [7:0] buf_out;
    output buf_empty, buf_full;
    output [7:0] fifo_counter;
    reg [7:0] buf_out;
    reg buf_empty, buf_full;
    reg [7:0] fifo_counter;
    reg [3:0] rd_ptr, wr_ptr;
    reg [7:0] buf_mem[63:0];

always @(fifo_counter) begin
    buf_empty = (fifo_counter==0);
    buf_full = (fifo_counter==64);
    end

always @(posedge clk or posedge rst) begin
    if(rst)
        fifo_counter <= 0;
        else if ( (!buf_full && wr_en) && (!buf_empty && rd_en) )
            fifo_counter <= fifo_counter;
        else if ( !buf_full && wr_en )
            fifo_counter <= fifo_counter + 1;
        else if ( !buf_empty && rd_en )
            fifo_counter <= fifo_counter - 1;
        else
        fifo_counter <= fifo_counter;
    end

always @(posedge clk or posedge rst) begin
    if(rst)
        buf_out <= 0;
            else begin
                if(rd_en && !buf_empty)
                    buf_out <= buf_mem[rd_ptr];
                    else
                    buf_out <= buf_out;
                    end
                end

always @(posedge clk) begin
    if(wr_en && !buf_full)
        buf_mem[wr_ptr] <= buf_in;
        else
            buf_mem[wr_ptr] <= buf_mem[wr_ptr];
            end

always @(posedge clk or posedge rst) begin
    if(rst) begin
        wr_ptr <= 0;
        rd_ptr <= 0;
        end else begin
            if(!buf_full && wr_en)
                wr_ptr <= wr_ptr + 1;
                else
                wr_ptr <= wr_ptr;
            if(!buf_empty && rd_en)
                rd_ptr <= rd_ptr + 1;
                else
                rd_ptr <= rd_ptr;
        end
    end
endmodule
```

Row Detector:

```
module row_detector(rowsignal, weight_passtime);
    parameter BIT_WIDTH = 8;
    input [BIT_WIDTH-1:0] rowsignal;
    output reg [BIT_WIDTH-1:0] weight_passtime;

always @(*) begin
    weight_passtime = 8'b00000011 - rowsignal;
    end

endmodule
```

Systolic Cell:

```
module systolic_cell(clk, rst, row_en, col_en, weight_pass, weight_passtime, weightloading, activation_input, weight_input, psum_input, activation_output, sum_output, weight_output);
    parameter BIT_WIDTH = 8;
    input [2*BIT_WIDTH-1:0] psum_input;
    input [BIT_WIDTH-1:0] activation_input;
    input [BIT_WIDTH-1:0] weight_input;
    input [BIT_WIDTH-1:0] weight_passtime;
    input rst;
    input clk;
    input row_en;
    input col_en;
    input weight_pass;
    output reg weightloading;
    output reg [BIT_WIDTH-1:0] activation_output;
    output reg [2*BIT_WIDTH-1:0] sum_output;
    output reg [BIT_WIDTH-1:0] weight_output;

    reg [BIT_WIDTH-1:0] weight_passtime_reg;
    reg [BIT_WIDTH-1:0] weight_reg;
    reg [2*BIT_WIDTH-1:0] mac_out;

always @(*) begin
    mac_out = psum_input;
    weight_passtime_reg = weight_passtime;
        if (row_en && col_en) begin
            mac_out = (activation_input * weight_reg) + psum_input;
            end
        end

always @(posedge clk or posedge rst) begin
    if (rst) begin
        activation_output <= 1'b0;
        sum_output <= 1'b0;
        weight_reg <= 1'b0;
        weight_output <= 1'b0;
        end else if (weight_pass) begin
            if (weight_passtime_reg > 0) begin
            weightloading <= 1'b1;
            weight_output <= weight_input;
            weight_passtime_reg <= weight_passtime_reg - 1;
            end else begin
                weightloading <= 1'b0;
                weight_reg <= weight_input;
                weight_output <= weight_input;
                end
            end else begin
                if (row_en && col_en) begin
                    activation_output <= activation_input;
                    sum_output <= mac_out;
                end else begin
                    sum_output <= psum_input;
                end
            end
        end
endmodule
```

Matrix Multiplication Unit:

```
module MMU4x4(a1, a2, a3, a4, w1, w2, w3, w4, s_in1, s_in2, s_in3, s_in4, o1, o2, o3, o4, rst, clk, w_pass, weightload_complete, r1_en, r2_en, r3_en, r4_en, c1_en, c2_en, c3_en, c4_en, rowsignal1, rowsignal2, rowsignal3, rowsignal4);
    parameter BIT_WIDTH = 8;
    input [BIT_WIDTH-1:0] a1, a2, a3, a4;
    input [BIT_WIDTH-1:0] w1, w2, w3, w4;
    input [2*BIT_WIDTH-1:0] s_in1, s_in2, s_in3, s_in4;
    input r1_en, r2_en, r3_en, r4_en, c1_en, c2_en, c3_en, c4_en;
    input [BIT_WIDTH-1:0] rowsignal1, rowsignal2, rowsignal3, rowsignal4;
    input clk, rst, w_pass;
    output wire [2*BIT_WIDTH-1:0] o1, o2, o3, o4;
    output reg weightload_complete;

    wire [2*BIT_WIDTH-1:0] s_11_21, s_21_31, s_31_41, s_12_22, s_22_32, s_32_42, s_13_23, s_23_33, s_33_43, s_14_24, s_24_34, s_34_44;
    wire [BIT_WIDTH-1:0] w_11_21, w_21_31, w_31_41, w_12_22, w_22_32, w_32_42, w_13_23, w_23_33, w_33_43, w_14_24, w_24_34, w_34_44;
    wire [BIT_WIDTH-1:0] a_11_12, a_12_13, a_13_14, a_21_22, a_22_23, a_23_24, a_31_32, a_32_33, a_33_34, a_41_42, a_42_43, a_43_44;
    wire weightloading1, weightloading2, weightloading3, weightloading4;
    wire [BIT_WIDTH-1:0] weight_passtime1, weight_passtime2, weight_passtime3, weight_passtime4;

always @(*) begin
    if (weightloading2 || weightloading3 || weightloading4) begin
        weightload_complete <= 1'b0;
        end else begin
            weightload_complete <= 1'b1;
            end
        end
//--------------------------------ROW1--------------------------------//
row_detector u1(.rowsignal(rowsignal1), .weight_passtime(weight_passtime1));
//--------------------------------ROW2--------------------------------//
row_detector u2(.rowsignal(rowsignal2), .weight_passtime(weight_passtime2));
//--------------------------------ROW3--------------------------------//
row_detector u3(.rowsignal(rowsignal3), .weight_passtime(weight_passtime3));
//--------------------------------ROW4--------------------------------//
row_detector u4(.rowsignal(rowsignal4), .weight_passtime(weight_passtime4));
//--------------------------------COL 1--------------------------------//
systolic_cell
pe11(.clk(clk), .rst(rst), .row_en(r1_en), .col_en(c1_en), .weight_pass(w_pass), .weight_passtime(weight_passtime1), .weightloading(weightloading1), .activation_input(a1), .weight_input(w1), .psum_input(s_in1), .activation_output(a_11_12), .sum_output(s_11_21), .weight_output(w_11_21));
systolic_cell
pe21(.clk(clk), .rst(rst), .row_en(r2_en), .col_en(c1_en), .weight_pass(w_pass), .weight_passtime(weight_passtime2), .weightloading(weightloading2), .activation_input(a2), .weight_input(w_11_21), .psum_input(s_11_21), .activation_output(a_21_22), .sum_output(s_21_31), .weight_output(w_21_31));
systolic_cell
pe31(.clk(clk), .rst(rst), .row_en(r3_en), .col_en(c1_en), .weight_pass(w_pass), .weight_passtime(weight_passtime3), .weightloading(weightloading3), .activation_input(a3), .weight_input(w_21_31), .psum_input(s_21_31), .activation_output(a_31_32), .sum_output(s_31_41), .weight_output(w_31_41));
systolic_cell
pe41(.clk(clk), .rst(rst), .row_en(r4_en), .col_en(c1_en), .weight_pass(w_pass), .weight_passtime(weight_passtime4), .weightloading(weightloading4), .activation_input(a4), .weight_input(w_31_41), .psum_input(s_31_41), .activation_output(a_41_42), .sum_output(o1), .weight_output());
//--------------------------------COL 2--------------------------------//
systolic_cell
pe12(.clk(clk), .rst(rst), .row_en(r1_en), .col_en(c2_en), .weight_pass(w_pass), .weight_passtime(weight_passtime1), .weightloading(), .activation_input(a_11_12), .weight_input(w2), .psum_input(s_in2), .activation_output(a_12_13), .sum_output(s_12_22), .weight_output(w_12_22));
systolic_cell
pe22(.clk(clk), .rst(rst), .row_en(r2_en), .col_en(c2_en), .weight_pass(w_pass), .weight_passtime(weight_passtime2), .weightloading(), .activation_input(a_21_22), .weight_input(w_12_22), .psum_input(s_12_22), .activation_output(a_22_23), .sum_output(s_22_32), .weight_output(w_22_32));
systolic_cell
pe32(.clk(clk), .rst(rst), .row_en(r3_en), .col_en(c2_en), .weight_pass(w_pass), .weight_passtime(weight_passtime3), .weightloading(), .activation_input(a_31_32), .weight_input(w_22_32), .psum_input(s_22_32), .activation_output(a_32_33), .sum_output(s_32_42), .weight_output(w_32_42));
systolic_cell
pe42(.clk(clk), .rst(rst), .row_en(r4_en), .col_en(c2_en), .weight_pass(w_pass), .weight_passtime(weight_passtime4), .weightloading(), .activation_input(a_41_42), .weight_input(w_32_42), .psum_input(s_32_42), .activation_output(a_42_43), .sum_output(o2), .weight_output());
//--------------------------------COL 3--------------------------------//
systolic_cell
pe13(.clk(clk), .rst(rst), .row_en(r1_en), .col_en(c3_en), .weight_pass(w_pass), .weight_passtime(weight_passtime1), .weightloading(), .activation_input(a_12_13), .weight_input(w3), .psum_input(s_in3), .activation_output(a_13_14), .sum_output(s_13_23), .weight_output(w_13_23));
systolic_cell
pe23(.clk(clk), .rst(rst), .row_en(r2_en), .col_en(c3_en), .weight_pass(w_pass), .weight_passtime(weight_passtime2), .weightloading(), .activation_input(a_22_23), .weight_input(w_13_23), .psum_input(s_13_23), .activation_output(a_23_24), .sum_output(s_23_33), .weight_output(w_23_33));
systolic_cell
pe33(.clk(clk), .rst(rst), .row_en(r3_en), .col_en(c3_en), .weight_pass(w_pass), .weight_passtime(weight_passtime3), .weightloading(), .activation_input(a_32_33), .weight_input(w_23_33), .psum_input(s_23_33), .activation_output(a_33_34), .sum_output(s_33_43), .weight_output(w_33_43));
systolic_cell
pe43(.clk(clk), .rst(rst), .row_en(r4_en), .col_en(c3_en), .weight_pass(w_pass), .weight_passtime(weight_passtime4), .weightloading(), .activation_input(a_42_43), .weight_input(w_33_43), .psum_input(s_33_43), .activation_output(a_43_44), .sum_output(o3), .weight_output());
//--------------------------------COL 4--------------------------------//
systolic_cell
pe14(.clk(clk), .rst(rst), .row_en(r1_en), .col_en(c4_en), .weight_pass(w_pass), .weight_passtime(weight_passtime1), .weightloading(), .activation_input(a_13_14), .weight_input(w4), .psum_input(s_in4), .activation_output(), .sum_output(s_14_24), .weight_output(w_14_24));
systolic_cell
pe24(.clk(clk), .rst(rst), .row_en(r2_en), .col_en(c4_en), .weight_pass(w_pass), .weight_passtime(weight_passtime2), .weightloading(), .activation_input(a_23_24), .weight_input(w_14_24), .psum_input(s_14_24), .activation_output(), .sum_output(s_24_34), .weight_output(w_24_34));
systolic_cell
pe34(.clk(clk), .rst(rst), .row_en(r3_en), .col_en(c4_en), .weight_pass(w_pass), .weight_passtime(weight_passtime3), .weightloading(), .activation_input(a_33_34), .weight_input(w_24_34), .psum_input(s_24_34), .activation_output(), .sum_output(s_34_44), .weight_output(w_34_44));
systolic_cell
pe44(.clk(clk), .rst(rst), .row_en(r4_en), .col_en(c4_en), .weight_pass(w_pass), .weight_passtime(weight_passtime4), .weightloading(), .activation_input(a_43_44), .weight_input(w_34_44), .psum_input(s_34_44), .activation_output(), .sum_output(o4), .weight_output());

endmodule
```

Accumulator:

```
module Accumulator4x4(MMU_size, write_enable, read_enable, in1, in2, in3, in4,
out1, out2, out3, out4, accumulator_finish_storing, clk, rst);
    parameter BIT_WIDTH = 8;
    parameter word_size = 16;
    parameter memory_size = 4;
    input [BIT_WIDTH-1:0] MMU_size;
    input [2*BIT_WIDTH-1:0] in1, in2, in3, in4;
    input clk, rst;
    input write_enable, read_enable;
    output reg [2*BIT_WIDTH-1:0] out1, out2, out3, out4;
    output reg accumulator_finish_storing;

    reg [word_size-1:0] memory1 [memory_size-1:0];
    reg [word_size-1:0] memory2 [memory_size-1:0];
    reg [word_size-1:0] memory3 [memory_size-1:0];
    reg [word_size-1:0] memory4 [memory_size-1:0];

    reg [7:0] cycle_count1, cycle_count2, cycle_count3,
    cycle_count4;
    reg [1:0] read_index;

always @(posedge clk) begin
    if (rst) begin
        cycle_count1 <= 0;
        cycle_count2 <= 0;
        cycle_count3 <= 0;
        cycle_count4 <= 0;
        accumulator_finish_storing <= 1'b0;
        read_index <= 0;
        end
        if (write_enable) begin
            if (cycle_count1 < MMU_size + 5) cycle_count1 <= cycle_count1 + 1;
            if (cycle_count2 < MMU_size + 6) cycle_count2 <= cycle_count2 + 1;
            if (cycle_count3 < MMU_size + 7) cycle_count3 <= cycle_count3 + 1;
            if (cycle_count4 < MMU_size + 8) cycle_count4 <= cycle_count4 + 1;

            if (cycle_count1 >= 5 && cycle_count1 < (5 + MMU_size))
            memory1[cycle_count1 - 5] <= in1;
            if (cycle_count2 >= 6 && cycle_count2 < (6 + MMU_size))
            memory2[cycle_count2 - 6] <= in2;
            if (cycle_count3 >= 7 && cycle_count3 < (7 + MMU_size))
            memory3[cycle_count3 - 7] <= in3;
            if (cycle_count4 >= 8 && cycle_count4 < (8 + MMU_size))
            memory4[cycle_count4 - 8] <= in4;

            if ((cycle_count1 >= MMU_size + 4) && (cycle_count2 >= MMU_size + 5)
            && (cycle_count3 >= MMU_size + 6) && (cycle_count4 >= MMU_size +
            7)) begin
                accumulator_finish_storing <= 1'b1;
                end
            end else begin
                accumulator_finish_storing <= 1'b0;
                end
            end

always @(posedge clk) begin
    if (read_enable) begin
        if (read_index < memory_size) begin
            out1 <= memory1[read_index];
            out2 <= memory2[read_index];
            out3 <= memory3[read_index];
            out4 <= memory4[read_index];
            read_index <= read_index + 1;
            end else begin
                out1 <= 0;
                out2 <= 0;
                out3 <= 0;
                out4 <= 0;
                end
            end
        end

endmodule
```

Activation Normalization Unit:

```
module Activation_Normalization_Unit4x4(func_select, in1, in2, in3, in4,
out1, out2, out3, out4);
    parameter BIT_WIDTH = 8;
    input [2:0] func_select;
    input [15:0] in1, in2, in3, in4;
    output reg [7:0] out1, out2, out3, out4;

    reg [15:0] act_out1, act_out2, act_out3, act_out4;

always @(*) begin
    case (func_select)
        2'b01: begin
        // Placeholder for sigmoid logic (FUTURE WORK)
        act_out1 = in1;
        act_out2 = in2;
        act_out3 = in3;
        act_out4 = in4;
        end
        2'b10: begin
        // Placeholder for SoftMAX logic (FUTURE WORK)
        act_out1 = in1;
        act_out2 = in2;
        act_out3 = in3;
        act_out4 = in4;
        end
        2'b11: begin
        // Placeholder for other activation logic (FUTURE WORK)
        act_out1 = in1;
        act_out2 = in2;
        act_out3 = in3;
        act_out4 = in4;
        end
        default: begin // ReLU logic
        act_out1 = (in1[15] == 0) ? in1 : 16'b0;
        act_out2 = (in2[15] == 0) ? in2 : 16'b0;
        act_out3 = (in3[15] == 0) ? in3 : 16'b0;
        act_out4 = (in4[15] == 0) ? in4 : 16'b0;
        end
    endcase
end

always @(*) begin
    out1 = act_out1 >> 8;
    out2 = act_out2 >> 8;
    out3 = act_out3 >> 8;
    out4 = act_out4 >> 8;
    end

endmodule
```

Controller:

```
module Controller(IR_out, IR_wr, IR_rd, // for IR_Mem
IR_inc, IR_clr, // for IR_counter
UB_addr, UB_rd, UB_wr, UB_counter_rst, // for Unified Buffer
sds_counter_rst, sds_work, // for systolic data setup
Weight_DDR_addr, Weight_rd, Weight_wr, // for Weight DDR3 memory
Weight_interface_pushtime, Weight_interface_rst, Weight_interface_push, Weight_interface_pop, pop_complete, // for Weight interface
Weight_FIFO_rst,Weight_FIFO_wr_en, Weight_FIFO_rd_en, Weight_FIFO_buf_empty, Weight_FIFO_buf_full, Weight_FIFO_fifo_counter, // for Weight FIFO
Activation_FIFO_rst,    Activation_FIFO_wr_en,    Activation_FIFO_rd_en    Activation_FIFO_buf_empty,    Activation_FIFO_buf_full,
Activation_FIFO_fifo_counter, // for Activation FIFO
MMU_size, write_enable, read_enable, Accumulator_rst, accumulator_finish_storing, // for Accumulator
func_select, // for Activation_Normalization_Unit
MMU_rst, MMU_w_pass, MMU_weightload_complete, MMU_s_in1, MMU_s_in2, MMU_s_in3, MMU_s_in4, // MMU_weightload_complete is an 'input' to detect whether the weights are finishing loading into MMU
MMU_r1_en, MMU_r2_en, MMU_r3_en, MMU_r4_en, MMU_c1_en, MMU_c2_en, MMU_c3_en, MMU_c4_en, MMU_rowsignal1, MMU_rowsignal2, MMU_rowsignal3, MMU_rowsignal4, // for MMU
clk, rst);
    parameter Instruction_size = 16;
    parameter address_size = 4;
    parameter MMUsize = 8;
    parameter S_initial = 0, S_fetch = 1; // fetch instructions for IR (in TPU)
    parameter S_decode = 2; // decode the instruction from IR
    // Weight load //
    parameter S_Weight_DDR3_to_interface1 = 3, S_Weight_DDR3_to_interface2 = 4, S_Weight_wait_stack_for_one_more_cycle = 5; parameter S_Weight_interface_to_WFIFO_stage1 = 6, S_Weight_interface_to_WFIFO_stage2 = 7, S_Weight_interface_to_WFIFO = 8,
    S_Weight_wait_FIFO_for_one_more_cycle = 9; parameter S_Weight_WFIFO_to_MMU = 10, S_Weight_MMU_rowdetect_stage1 = 11, S_Weight_MMU_rowdetect_stage2 = 12, S_Weight_MMU_load_complete = 13;
    parameter  S_UB_to_sfu1 =  14,  S_UB_to_sds2 =  15,  S_UB_to_sds3 =  16,  S_Activation_Wait_for_one_more_cycle =  17, S_Activation_AFIFO_to_MMU_and_computing = 18;
    parameter S_MMU_computing = 19;
    parameter S_UB_store1 = 20, S_UB_store2 = 21;
    parameter S_Computing_Time_Compase = 22;
    parameter S_stay = 23;
    // opcode //
    parameter Read_Host_Memory = 0; //----------------   future work   ----------------//
    parameter Read_Weight = 1; // weight_load (to MMU)
    parameter Computation = 2; //input from UB to A_FIFO to MMU then store in Accumulator (Called MatrixMultiply/Convolve in TPU paper)
    parameter Activate = 3, // values from accumulator through Activation-Normalization Unit and back to UB (Called Activate in TPU paper)
    parameter Write_Host_Memory = 4; //----------------   future work   ----------------//

    // for IR_Mem
    input [Instruction_size-1:0]IR_out; output reg IR_wr, IR_rd;
    // for IR_counter
    output reg IR_inc, IR_clr;
    // for Unified Buffer
    output reg [address_size-1:0] UB_addr; // for 16 address locations
    output reg UB_rd, UB_wr, UB_counter_rst;
    // for systolic data setup
    output reg sds_counter_rst;
    output reg sds_work;
    // for Weight DDR3 memory
    output reg [address_size:0] Weight_DDR_addr;
    output reg Weight_rd, Weight_wr;
    // for Weight interface
    output reg [4:0] Weight_interface_pushtime; // for stack inside Weight interface
    output reg Weight_interface_rst;
    output reg Weight_interface_push, Weight_interface_pop;
    input pop_complete;
    // for Weight FIFO
    output reg Weight_FIFO_rst, Weight_FIFO_wr_en, Weight_FIFO_rd_en;
    input Weight_FIFO_buf_empty, Weight_FIFO_buf_full;
    input [2*address_size-1:0] Weight_FIFO_fifo_counter;
    // for Activation FIFO
    output reg Activation_FIFO_rst, Activation_FIFO_wr_en, Activation_FIFO_rd_en;
    input Activation_FIFO_buf_empty, Activation_FIFO_buf_full;
    input [2*address_size-1:0] Activation_FIFO_fifo_counter;
    // for Accumulator
    output reg [MMUsize-1:0] MMU_size;
    output reg write_enable, read_enable, Accumulator_rst;
    input accumulator_finish_storing;
    // for Activation_Normalization_Unit
    output reg [2:0] func_select; // for 4 different activation functions (fix in 00 for ReLU now and more functions are future work)
    // for MMU
    output reg MMU_rst, MMU_w_pass;
    input MMU_weightload_complete;
    output reg [15:0] MMU_s_in1, MMU_s_in2, MMU_s_in3, MMU_s_in4; // fix at 0
    output reg MMU_r1_en, MMU_r2_en, MMU_r3_en, MMU_r4_en, MMU_c1_en, MMU_c2_en, MMU_c3_en, MMU_c4_en;
    output reg [7:0] MMU_rowsignal1, MMU_rowsignal2, MMU_rowsignal3, MMU_rowsignal4;
    // general signal
    input clk, rst;

    reg [4:0] state, next_state;
    wire [Instruction_size-1:0]IR_out;
    wire [3:0] opcode = IR_out[15:12];
    wire [2:0] function_select = IR_out[11:9];
    reg [3:0] itr;
    reg [4:0] Weight_addr_count;
    reg [4:0] UB_addr_count;

// Sequential logic to manage Weight_addr_count for each clock cycle in S_Weight_load_loop
always @(posedge clk or posedge rst) begin
    if (rst) begin
        Weight_addr_count <= 5'b00000;
    end else if ((next_state == S_Weight_DDR3_to_interface1 || next_state == S_Weight_DDR3_to_interface2) && Weight_addr_count < 16) begin
        Weight_addr_count <= Weight_addr_count + 5'b00001;
        end
    end

// Sequential logic to manage UB_addr_count for each clock cycle in S_UB_loop (rd and wr)
always @(posedge clk or posedge rst) begin
    if (rst) begin
        UB_addr_count <= 5'b00000;
        end else if (next_state == S_Activation_Wait_for_one_more_cycle || next_state == S_UB_to_sds1) begin
            UB_addr_count <= 5'b00000;
        end else if ((next_state==S_UB_to_sds2 | next_state ==S_UB_to_sds3 | next_state == S_UB_store1 | next_state == S_UB_store2) && UB_addr_count < 8) begin
            UB_addr_count <= UB_addr_count + 5'b00001;
            end
        end

always@(IR_out) begin // maybe for computation more than once, 2 times or even a loop
    if (IR_out[15:12] == 2) begin
        itr <= IR_out[3:0]; // if opcode = 2 (computation, if itr >= 0, then loop back to computation state again)
        end else begin
        itr <= itr;
        end
    end

always@(posedge clk or posedge rst)begin: State_transitions
    if(rst == 1) state <= S_initial; else state <= next_state;
    end
```

```
always@(state or opcode or UB_addr or Weight_DDR_addr or Weight_interface_pushtime or MMU_size or accumulator_finish_storing or func_select or MMU_weightload_complete or pop_complete or itr)begin
    IR_wr = 0; IR_rd = 0; IR_inc = 0; IR_clr = 0; UB_addr = 4'b0000;
    UB_rd =0; UB_wr = 0; UB_counter_rst = 0; sds_counter_rst = 0;
    Weight_DDR_addr = 5'b0000; Weight_rd = 0; Weight_wr = 0; Weight_interface_pushtime = 5'b00000; Weight_interface_push = 0;
    Weight_interface_pop = 0; Weight_interface_rst = 0; Weight_FIFO_rst = 0; Weight_FIFO_wr_en = 0; Weight_FIFO_rd_en = 0;
    Activation_FIFO_rst = 0; Activation_FIFO_wr_en = 0;Activation_FIFO_rd_en = 0; MMU_size = 8'b00000000; write_enable= 0; read_enable=0;
    Accumulator_rst = 0; func_select = 3'b000; MMU_rst = 0; MMU_w_pass = 0;
    MMU_s_in1 = 16'b0000000000000000; MMU_s_in2 = 16'b0000000000000000; MMU_s_in3 = 16'b0000000000000000;
    MMU_s_in4 = 16'b0000000000000000; MMU_r1_en = 0; MMU_r2_en = 0; MMU_r3_en = 0; MMU_r4_en = 0; MMU_c1_en = 0;
    MMU_c2_en = 0; MMU_c3_en = 0; MMU_c4_en = 0; MMU_rowsignal1 = 8'b00000000; MMU_rowsignal2 = 8'b00000000;
    MMU_rowsignal3 = 8'b00000000; MMU_rowsignal4 = 8'b00000000;
    next_state = state;

    case (state)
        S_initial   :begin
            next_state = S_fetch;
            IR_clr = 1;
            UB_counter_rst = 1;
            sds_counter_rst = 1;
            Weight_interface_rst = 1;
            Weight_FIFO_rst = 1;
            Activation_FIFO_rst = 1;
            Accumulator_rst = 1;
            MMU_rst = 1;
            $display("State: S_initial");
        end
        S_fetch     :begin
            next_state = S_decode;
            IR_clr = 0;
            IR_rd = 1;
            $display("State: S_fetch");
        end
        S_decode    :begin
            $display("opcode: %b", opcode);
            case(opcode)
                // opcode = 0
                Read_Host_Memory: begin //----------------   future work   ----------------//
                next_state = S_fetch; // fetch next instruction
                IR_inc = 1;
                $display("State: Read_Host_Memory (future work)"); //----------------   future work   ----------------//
            end //----------------   future work   ----------------//
                // opcode = 1
                Read_Weight: begin
                Weight_interface_pushtime = 5'b11111; // prepare for the weight input from DDR3
                // data read out from Weight DDR
                Weight_rd = 1;
                Weight_interface_push = 1;
                Weight_DDR_addr = Weight_addr_count;
                next_state = S_Weight_DDR3_to_interface1; // Move to the loop state to begin loading weights
                $display("State: Read_Weight");
            end
                // opcode = 2
                Computation: begin
                //--------Use 1st value stored in UB_mem[0] for example--------//
                // UB data1 from mem[0] to UB output_reg
                UB_rd = 1;
                UB_wr = 0;
                UB_counter_rst = 1; // reset UB_counter i (in UB)
                //--------Should not allow these two signal, or it will load 8'bx into sds_reg and AFIFO--------//
                //                sds_work = 1;
                //                Activation_FIFO_rd_en = 1;
                next_state = S_UB_to_sds1;
                $display("State: Computation");
            end
                //opcode = 3
                Activate: begin
                read_enable = 1; // Accumulator read_enable = 1
                func_select = 3'b000;
                UB_counter_rst = 1; // reset UB_counter i (in UB)
                next_state = S_UB_store1;
                $display("State: Activate");
            end
                // opcode = 4
                Write_Host_Memory: begin //----------------   future work   ----------------//
                next_state = S_fetch; // fetch next instruction
                IR_inc = 1;
                $display("State: Write_Host_Memory (future work)"); //----------------   future work   ----------------//
            end //----------------   future work   ----------------//
            default:
                begin next_state = S_stay;
                $display("Invalid opcode");
            end
            endcase
        end
        S_Weight_DDR3_to_interface1: begin
            if(Weight_addr_count <= 15)begin
                $display("State: Weight_load_to_WInterface. . .");
                // data read out from Weight DDR
                Weight_rd = 1;
                // Weight interface push
                Weight_interface_push = 1;
                Weight_DDR_addr = Weight_addr_count;
                next_state = S_Weight_DDR3_to_interface2; // Remain in the loop until count reaches 16
            end else begin
                // All weights have been loaded into the FIFO
                $display("State: Weight_load_to_WInterface. . .");
                Weight_rd = 0;
                Weight_interface_push = 0;
                Weight_addr_count <= 5'b00000;
                next_state = S_Weight_wait_stack_for_one_more_cycle; // Proceed to the next state
            end
        end
        S_Weight_DDR3_to_interface2: begin
            if(Weight_addr_count <= 15)begin
                $display("State: Weight_load_to_WInterface. . .");
                // data read out from Weight DDR
                Weight_rd = 1;
                // Weight interface push
                Weight_interface_push = 1;
                Weight_DDR_addr = Weight_addr_count;
                next_state = S_Weight_DDR3_to_interface1; // Remain in the loop until count reaches 16
            end else begin
                // All weights have been loaded into the FIFO
                $display("State: Weight_load_to_WInterface. . .");
                Weight_rd = 0;
                Weight_interface_push = 0;
                next_state = S_Weight_wait_stack_for_one_more_cycle; // Proceed to the next state
            end
        end
        S_Weight_wait_stack_for_one_more_cycle: begin
            $display("State: Weight completely loaded into WInterface ! ! !");
            Weight_interface_pop = 1; // have to pressed this pop signal one state earlier, or, all FIFOs[0] will load output = 0.
            next_state = S_Weight_interface_to_WFIFO_stage1;
            end
```

```
S_Weight_interface_to_WFIFO_stage1: begin
    if(pop_complete != 1) begin
        Weight_interface_pop = 1;
        Weight_FIFO_wr_en = 1;
        $display("State: Weight_load_to_WFIFO...");
        next_state = S_Weight_interface_to_WFIFO_stage2;
    end else begin
        next_state = S_Weight_wait_FIFO_for_one_more_cycle;
    end
end
S_Weight_interface_to_WFIFO_stage2: begin
    if(pop_complete != 1) begin
        Weight_interface_pop = 1;
        Weight_FIFO_wr_en = 1;
        $display("State: Weight_load_to_WFIFO...");
        next_state = S_Weight_interface_to_WFIFO_stage1;
    end else begin
        next_state = S_Weight_wait_FIFO_for_one_more_cycle;
    end
end
S_Weight_wait_FIFO_for_one_more_cycle: begin
    $display("State: Weight completely loaded into WFIFO");
    Weight_FIFO_wr_en = 0;
    Weight_interface_pop = 0;
    next_state = S_Weight_WFIFO_to_MMU;
end
S_Weight_WFIFO_to_MMU: begin
    $display("State: Weight_load_to_MMU");
    Weight_FIFO_rd_en = 1;
    MMU_w_pass = 1;
    MMU_rowsignal1 = 8'b00000001; // should be 1 for correct counting
    MMU_rowsignal2 = 8'b00000001;
    MMU_rowsignal3 = 8'b00000010;
    MMU_rowsignal4 = 8'b00000011;
    next_state = S_Weight_MMU_rowdetect_stage1;
end
S_Weight_MMU_rowdetect_stage1: begin
    $display("State: MMU row detect s1...");
    Weight_FIFO_rd_en = 1;
    MMU_w_pass = 1;
        if (MMU_weightload_complete == 1) begin
            next_state = S_Weight_MMU_load_complete; // Fetch next instruction when weights are fully loaded
            Weight_FIFO_rd_en = 0;
            MMU_w_pass = 0;
        end else begin
            next_state = S_Weight_MMU_rowdetect_stage2; // Continue weight loading
        end
    end
S_Weight_MMU_rowdetect_stage2: begin
    $display("State: MMU row detect s2...");
    Weight_FIFO_rd_en = 1;
    MMU_w_pass = 1;
        if (MMU_weightload_complete == 1) begin
            next_state = S_Weight_MMU_load_complete; // Fetch next instruction when weights are fully loaded
            Weight_FIFO_rd_en = 0;
            MMU_w_pass = 0;
        end else begin
            next_state = S_Weight_MMU_rowdetect_stage1; // Continue weight loading
        end
    end
S_Weight_MMU_load_complete: begin
    $display("State: Weight complete loading into MMU");
    next_state = S_fetch;
    IR_inc = 1;
    Weight_FIFO_rst = 1; // clear Weight FIFO in case of contain useless weight values
end
S_UB_to_sds1: begin
    $display("State: Activation_load_to_AFIFO s1...");
    //---------Use 1st value stored in UB_mem[0] for example---------//
    // Other data be sent to output_reg's'
    UB_rd = 1;
    UB_wr = 0;
    //---------Should allow sds_work = 1 for Systolic Data Setup---------//
    // UB data1 from UB output_reg to sds output1(reg)
    sds_work = 1;
    //---------However, should not allow Activation_FIFO_wr_en = 1 or it will load 8'dx into AFIFO---------//
    //                      Activation_FIFO_wr_en = 1;
    next_state = S_UB_to_sds2;
end
S_UB_to_sds2: begin
    if (UB_addr_count <= 7) begin
        $display("State: Activation_load_to_AFIFO s2...");
        //---------Use 1st value stored in UB_mem[0] for example---------//
        // Other data be sent to output_reg's'
        UB_rd = 1;
        UB_wr = 0;
        // Other data be sent to sds_output_reg's'
        sds_work = 1;
        // UB data1 from sds output1(reg) to AFIFO1[0]
        Activation_FIFO_wr_en = 1;
        next_state = S_UB_to_sds3; // need this extra state for signal detection
    end else begin
        next_state = S_Activation_Wait_for_one_more_cycle;
    end
end
S_UB_to_sds3: begin
    if (UB_addr_count <= 7) begin
        $display("State: Activation_load_to_AFIFO s3...");
        UB_rd = 1;
        UB_wr = 0;
        sds_work = 1;
        Activation_FIFO_wr_en = 1;
        next_state = S_UB_to_sds2;
    end else begin
        next_state = S_Activation_Wait_for_one_more_cycle;
    end
end
S_Activation_Wait_for_one_more_cycle: begin
    $display("State: Activation completely loaded into AFIFO");
    UB_rd = 0;
    UB_wr = 0;
    sds_work = 0;
    Activation_FIFO_wr_en = 0;
    next_state = S_Activation_AFIFO_to_MMU_and_computing;
end
```

```
S_Activation_AFIFO_to_MMU_and_computing: begin
    $display("State: Activation start loading into MMU");
    Activation_FIFO_rd_en = 1;
    MMU_s_in1 = 16'b0000000000000000;
    MMU_s_in2 = 16'b0000000000000000;
    MMU_s_in3 = 16'b0000000000000000;
    MMU_s_in4 = 16'b0000000000000000;
    MMU_r1_en = 1; MMU_r2_en = 1; MMU_r3_en = 1; MMU_r4_en = 1;
    MMU_c1_en = 1; MMU_c2_en = 1; MMU_c3_en = 1; MMU_c4_en = 1;
    MMU_size = 8'b00000100;
    write_enable = 1; // Accumulator write_enable = 1
    next_state = S_MMU_computing;
end
S_MMU_computing: begin
    if (accumulator_finish_storing != 1) begin
        $display("State: MMU Computing...");
        Activation_FIFO_rd_en = 1;
        MMU_s_in1 = 16'b0000000000000000;
        MMU_s_in2 = 16'b0000000000000000;
        MMU_s_in3 = 16'b0000000000000000;
        MMU_s_in4 = 16'b0000000000000000;
        MMU_r1_en = 1; MMU_r2_en = 1; MMU_r3_en = 1; MMU_r4_en = 1;
        MMU_c1_en = 1; MMU_c2_en = 1; MMU_c3_en = 1; MMU_c4_en = 1;
        MMU_size = 8'b00000100;
        write_enable = 1; // Accumulator write_enable = 1
        next_state = S_MMU_computing;
    end else begin
//-Finish Computing and values are stored in Accumulator and waiting for Activation Function instruction-//
        $display("State: Values finishing loading into Accumulator!!!");
        Activation_FIFO_rd_en = 0;
        MMU_r1_en = 0; MMU_r2_en = 0; MMU_r3_en = 0; MMU_r4_en = 0;
        MMU_c1_en = 0; MMU_c2_en = 0; MMU_c3_en = 0; MMU_c4_en = 0;
        write_enable = 0; // Accumulatpor write_enable = 0
        next_state = S_fetch; // fetch for Activate (ReLU, Sigmoid, SoftMax...)
        IR_inc = 1;
    end
end
S_UB_store1: begin
    $display("State: UB storing s1...");
    if (UB_addr_count <= 4) begin
        UB_rd = 0;
        UB_wr = 1;
        read_enable = 1; // Accumulator read_enable = 1
        func_select = 3'b000;
        next_state = S_UB_store2;
    end else begin
        $display("State: Values finishing storing back to Unified Buffer");
        UB_rd = 0;
        UB_wr = 0;
        read_enable = 0; // Accumulator read_enable = 0
        next_state = S_Computing_Time_Compare;
    end
end
S_UB_store2: begin
    $display("State: UB storing s2...");
    if (UB_addr_count <= 4) begin
        UB_rd = 0;
        UB_wr = 1;
        read_enable = 1; // Accumulator read_enable = 1
        func_select = 3'b000;
        next_state = S_UB_store1;
    end else begin
        $display("State: Values finishing storing back to Unified Buffer");
        UB_rd = 0;
        UB_wr = 0;
        read_enable = 0; // Accumulator read_enable = 0
        next_state = S_Computing_Time_Compare;
    end
end
S_Computing_Time_Compare: begin
    //---------future work---------//
    //                  $display("State: Checking iteration...");
    //                  if (itr > 0) begin
    //                  $display("State: Compute for the next itration!!!");
    //                  itr = itr - 1;
    //                  UB_rd = 1;
    //                  UB_wr = 0;
    //                  UB_counter_rst = 1; // reset UB_counter i (in UB)
    //                  next_state = S_UB_to_sds1;
    //                  end
    //                  else begin
    $display("State: Finish all the computation");
    next_state = S_fetch;
    IR_inc = 1;
    end
    //                  end
S_stay      :begin
    IR_rd = 1;
    $display("State: S_stay");
    end
default: begin next_state = S_initial;
    $display("State: Default, resetting to S_initial");
    end
endcase
end

endmodule
```

Top Module:

```
module TPU4x4(clk, rst);
    input clk, rst;

    parameter Instruction_size = 16;
    parameter address_size = 4;
    parameter MMUsize = 8;
    parameter Data_size = 8;

    wire [Instruction_size-1:0]IR_out;
    wire IR_wr, IR_rd;
    // for IR_counter
    wire [2*address_size-1:0] IR_addr;
    wire IR_inc, IR_clr;
    // for Unified Buffer
    wire [address_size-1:0] UB_addr; // for 16 address locations
    wire UB_rd, UB_wr, UB_counter_rst;
    wire [Data_size-1:0] UB_dataout1, UB_dataout2, UB_dataout3, UB_dataout4, UB_dataout5, UB_dataout6, UB_dataout7, UB_dataout8;
    wire [Data_size-1:0] UB_dataout9, UB_dataout10, UB_dataout11, UB_dataout12, UB_dataout13, UB_dataout14, UB_dataout15, UB_dataout16;
    // for systolic data setup
    wire sds_counter_rst, sds_work;
    wire [Data_size-1:0] Activation_sds_to_AFIFO1, Activation_sds_to_AFIFO2, Activation_sds_to_AFIFO3, Activation_sds_to_AFIFO4;
    // for Weight DDR3 memory
    wire [address_size:0] Weight_DDR_addr;
    wire Weight_rd, Weight_wr;
    wire [Data_size-1:0] Weight_DDR3_to_Interface;
    // for Weight interface
    wire [4:0] Weight_interface_pushtime; // for stack inside Weight interface
    wire Weight_interface_push, Weight_interface_pop;
    wire pop_complete;
    wire Weight_interface_rst;
    wire [Data_size-1:0] Weight_Interface_to_FIFO1, Weight_Interface_to_FIFO2, Weight_Interface_to_FIFO3, Weight_Interface_to_FIFO4;
    // for Weight FIFO
    wire Weight_FIFO_rst, Weight_FIFO_wr_en, Weight_FIFO_rd_en;
    wire Weight_FIFO_buf_empty, Weight_FIFO_buf_full;
    wire [2*address_size-1:0] Weight_FIFO_fifo_counter;
    wire [Data_size-1:0] Weight_FIFO_to_MMU1, Weight_FIFO_to_MMU2, Weight_FIFO_to_MMU3, Weight_FIFO_to_MMU4;
    // for Activation FIFO
    wire Activation_FIFO_rst, Activation_FIFO_wr_en, Activation_FIFO_rd_en;
    wire Activation_FIFO_buf_empty, Activation_FIFO_buf_full;
    wire [2*address_size-1:0] Activation_FIFO_fifo_counter;
    wire [Data_size-1:0] Activation_FIFO_to_MMU1, Activation_FIFO_to_MMU2, Activation_FIFO_to_MMU3, Activation_FIFO_to_MMU4;
    // for Accumulator
    wire [MMUsize-1:0] MMU_size;
    wire write_enable, read_enable, Accumulator_rst;
    wire accumulator_finish_storing;
    wire [15:0] Accumulator_to_AN_out1, Accumulator_to_AN_out2, Accumulator_to_AN_out3, Accumulator_to_AN_out4;
    // for Activation_Normalizaion_Unit
    wire [2:0] func_select; // for 4 different activation functions (fix in 00 for ReLU now and more functions are future work)
    wire [Data_size-1:0] AN_to_UB1, AN_to_UB2, AN_to_UB3, AN_to_UB4;
    // for MMU
    wire MMU_rst, MMU_w_pass;
    wire MMU_weightload_complete;
    wire [15:0] MMU_s_in1, MMU_s_in2, MMU_s_in3, MMU_s_in4; // fix at 0
    wire MMU_r1_en, MMU_r2_en, MMU_r3_en, MMU_r4_en, MMU_c1_en, MMU_c2_en, MMU_c3_en, MMU_c4_en;
    wire [7:0] MMU_rowsignal1, MMU_rowsignal2, MMU_rowsignal3, MMU_rowsignal4;
    wire [15:0] MMU_to_Accumulator_out1, MMU_to_Accumulator_out2, MMU_to_Accumulator_out3, MMU_to_Accumulator_out4;

    //----------------------------IR related----------------------------//
    IR_Mem IR_Mem1(.R_data(IR_out), .W_data(), .addr(IR_addr), .clk(clk), .IR_rd(IR_rd), .IR_wr());
    IR_counter IR_counter1(.IR_addr(IR_addr), .IR_inc(IR_inc), .IR_clr(IR_clr), .clk(clk));
    //----------------------------Weight related----------------------------//
    Weight_DDR3 WDDR3_1(.R_data(Weight_DDR3_to_Interface), .W_data(), .addr(Weight_DDR_addr), .clk(clk), .rd(Weight_rd), .wr());
    Weight_interface Winterface1(.clk(clk), .reset(Weight_interface_rst), .weight_in(Weight_DDR3_to_Interface), .push_time(Weight_interface_pushtime), .push(Weight_interface_push), .pop(Weight_interface_pop),.pop_complete(pop_complete),
    .out1(Weight_Interface_to_FIFO1), .out2(Weight_Interface_to_FIFO2), .out3(Weight_Interface_to_FIFO3), .out4(Weight_Interface_to_FIFO4));
    Weight_FIFO WFIFO1(.clk(clk), .rst(Weight_FIFO_rst), .buf_in(Weight_Interface_to_FIFO1), .buf_out(Weight_FIFO_to_MMU1), .wr_en(Weight_FIFO_wr_en), .rd_en(Weight_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Weight_FIFO WFIFO2(.clk(clk), .rst(Weight_FIFO_rst), .buf_in(Weight_Interface_to_FIFO2), .buf_out(Weight_FIFO_to_MMU2), .wr_en(Weight_FIFO_wr_en), .rd_en(Weight_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Weight_FIFO WFIFO3(.clk(clk), .rst(Weight_FIFO_rst), .buf_in(Weight_Interface_to_FIFO3), .buf_out(Weight_FIFO_to_MMU3), .wr_en(Weight_FIFO_wr_en), .rd_en(Weight_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Weight_FIFO WFIFO4(.clk(clk), .rst(Weight_FIFO_rst), .buf_in(Weight_Interface_to_FIFO4), .buf_out(Weight_FIFO_to_MMU4), .wr_en(Weight_FIFO_wr_en), .rd_en(Weight_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    //----------------------------MMU related----------------------------//
    MMU4x4 MMU1(.a1(Activation_FIFO_to_MMU1), .a2(Activation_FIFO_to_MMU2), .a3(Activation_FIFO_to_MMU3), .a4(Activation_FIFO_to_MMU4), .w1(Weight_FIFO_to_MMU1), .w2(Weight_FIFO_to_MMU2),
    .w3(Weight_FIFO_to_MMU3), .w4(Weight_FIFO_to_MMU4), .s_in1(MMU_s_in1), .s_in2(MMU_s_in2), .s_in3(MMU_s_in3), .s_in4(MMU_s_in4), .o1(MMU_to_Accumulator_out1), .o2(MMU_to_Accumulator_out2),
    .o3(MMU_to_Accumulator_out3), .o4(MMU_to_Accumulator_out4), .rst(MMU_rst), .clk(clk), .w_pass(MMU_w_pass), .weightload_complete(MMU_weightload_complete), .r1_en(MMU_r1_en), .r2_en(MMU_r2_en), .r3_en(MMU_r3_en),
    .r4_en(MMU_r4_en), .c1_en(MMU_c1_en), .c2_en(MMU_c2_en), .c3_en(MMU_c3_en), .c4_en(MMU_c4_en), .rowsignal1(MMU_rowsignal1), .rowsignal2(MMU_rowsignal2), .rowsignal3(MMU_rowsignal3), .rowsignal4(MMU_rowsignal4));
    //----------------------------Other Computation related----------------------------//
    //----------------------------before MMU----------------------------//
    unified_buffer UB1(.addr(UB_addr), .clk(clk), .rd(UB_rd), .wr(UB_wr), .counter_rst(UB_counter_rst), .data_in1(AN_to_UB1), .data_in2(AN_to_UB2), .data_in3(AN_to_UB3), .data_in4(AN_to_UB4), .data_out1(UB_dataout1),
    .data_out2(UB_dataout2), .data_out3(UB_dataout3), .data_out4(UB_dataout4), .data_out5(UB_dataout5), .data_out6(UB_dataout6), .data_out7(UB_dataout7), .data_out8(UB_dataout8), .data_out9(UB_dataout9), .data_out10(UB_dataout10),
    .data_out11(UB_dataout11), .data_out12(UB_dataout12), .data_out13(UB_dataout13), .data_out14(UB_dataout14), .data_out15(UB_dataout15), .data_out16(UB_dataout16));
    sds4x4 sds1(.clk(clk), .counter_rst(sds_counter_rst), .sds_work(sds_work), .a0(UB_dataout1), .a1(UB_dataout2), .a2(UB_dataout3), .a3(UB_dataout4), .b0(UB_dataout5), .b1(UB_dataout6), .b2(UB_dataout7), .b3(UB_dataout8), .c0(UB_dataout9),
    .c1(UB_dataout10), .c2(UB_dataout11), .c3(UB_dataout12), .d0(UB_dataout13), .d1(UB_dataout14), .d2(UB_dataout15), .d3(UB_dataout16), .output1(Activation_sds_to_AFIFO1), .output2(Activation_sds_to_AFIFO2),
    .output3(Activation_sds_to_AFIFO3), .output4(Activation_sds_to_AFIFO4));
    Activation_FIFO
    AFIFO1(.clk(clk), .rst(Activation_FIFO_rst), .buf_in(Activation_sds_to_AFIFO1), .buf_out(Activation_FIFO_to_MMU1), .wr_en(Activation_FIFO_wr_en), .rd_en(Activation_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Activation_FIFO
    AFIFO2(.clk(clk), .rst(Activation_FIFO_rst), .buf_in(Activation_sds_to_AFIFO2), .buf_out(Activation_FIFO_to_MMU2), .wr_en(Activation_FIFO_wr_en), .rd_en(Activation_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Activation_FIFO
    AFIFO3(.clk(clk), .rst(Activation_FIFO_rst), .buf_in(Activation_sds_to_AFIFO3), .buf_out(Activation_FIFO_to_MMU3), .wr_en(Activation_FIFO_wr_en), .rd_en(Activation_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    Activation_FIFO
    AFIFO4(.clk(clk), .rst(Activation_FIFO_rst), .buf_in(Activation_sds_to_AFIFO4), .buf_out(Activation_FIFO_to_MMU4), .wr_en(Activation_FIFO_wr_en), .rd_en(Activation_FIFO_rd_en), .buf_empty(), .buf_full(), .fifo_counter());
    //----------------------------After MMU----------------------------//
    Accumulator4x4
    Accumulator1(.MMU_size(MMU_size), .write_enable(write_enable), .read_enable(read_enable), .in1(MMU_to_Accumulator_out1), .in2(MMU_to_Accumulator_out2), .in3(MMU_to_Accumulator_out3), .in4(MMU_to_Accumulator_out4),
    .out1(Accumulator_to_AN_out1), .out2(Accumulator_to_AN_out2), .out3(Accumulator_to_AN_out3), .out4(Accumulator_to_AN_out4), .accumulator_finish_storing(accumulator_finish_storing), .clk(clk), .rst(Accumulator_rst));
    Activation_Normalization_Unit4x4
    AN1(.func_select(func_select), .in1(Accumulator_to_AN_out1), .in2(Accumulator_to_AN_out2), .in3(Accumulator_to_AN_out3), .in4(Accumulator_to_AN_out4), .out1(AN_to_UB1), .out2(AN_to_UB2), .out3(AN_to_UB3), .out4(AN_to_UB4));
    //----------------------------Controller related----------------------------//
    Controller controller1(.IR_out(IR_out), .IR_wr(), .IR_rd(IR_rd), // for IR_Mem
    .IR_inc(IR_inc), .IR_clr(IR_clr), // for IR_counter
    .UB_addr(UB_addr), .UB_rd(UB_rd), .UB_wr(UB_wr), .UB_counter_rst(UB_counter_rst), // for Unified Buffer
    .sds_counter_rst(sds_counter_rst), .sds_work(sds_work), // for systolic data setup
    .Weight_DDR_addr(Weight_DDR_addr), .Weight_rd(Weight_rd), .Weight_wr(), // for Weight DDR3 memory
    .Weight_interface_pushtime(Weight_interface_pushtime), .Weight_interface_rst(Weight_interface_rst), .Weight_interface_push(Weight_interface_push), .Weight_interface_pop(Weight_interface_pop), // for Weight interface
    .pop_complete(pop_complete), .Weight_FIFO_rst(Weight_FIFO_rst), .Weight_FIFO_wr_en(Weight_FIFO_wr_en), .Weight_FIFO_rd_en(Weight_FIFO_rd_en),
    .Weight_FIFO_buf_empty(), .Weight_FIFO_buf_full(), .Weight_FIFO_fifo_counter(), // for Weight FIFO
    .Activation_FIFO_rst(Activation_FIFO_rst), .Activation_FIFO_wr_en(Activation_FIFO_wr_en), .Activation_FIFO_rd_en(Activation_FIFO_rd_en),
    .Activation_FIFO_buf_empty(), .Activation_FIFO_buf_full(), .Activation_FIFO_fifo_counter(), // for Activation FIFO
    .MMU_size(MMU_size), .write_enable(write_enable), .read_enable(read_enable), .Accumulator_rst(Accumulator_rst), .accumulator_finish_storing(accumulator_finish_storing), // for Accumulator
    .func_select(func_select), // for Activation_Normalizaion_Unit
    .MMU_rst(MMU_rst), .MMU_w_pass(MMU_w_pass), .MMU_weightload_complete(MMU_weightload_complete), .MMU_s_in1(MMU_s_in1), .MMU_s_in2(MMU_s_in2), .MMU_s_in3(MMU_s_in3),
    .MMU_s_in4(MMU_s_in4), // MMU_weightload_complete is an input to detect whether the weights are finishing loading into MMU
    .MMU_r1_en(MMU_r1_en), .MMU_r2_en(MMU_r2_en), .MMU_r3_en(MMU_r3_en), .MMU_r4_en(MMU_r4_en), .MMU_c1_en(MMU_c1_en), .MMU_c2_en(MMU_c2_en), .MMU_c3_en(MMU_c3_en), .MMU_c4_en(MMU_c4_en),
    .MMU_rowsignal1(MMU_rowsignal1), .MMU_rowsignal2(MMU_rowsignal2), .MMU_rowsignal3(MMU_rowsignal3), .MMU_rowsignal4(MMU_rowsignal4), // for MMU
    .clk(clk), .rst(rst));

endmodule
```

IR Memory:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.928%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.68 sec
Total Real time: 4.0 sec
Total Memory Usage: 1187.367188 Mbytes
Reset AAE Options
```

```
innovus 1> *** Starting Verify Geometry (MEM: 1193.0) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 1.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.5  MEM: 89.5M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 16:02:41 2024

Design Name: IR_Mem
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (314.8200, 297.3600)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 16:02:41 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.1  MEM: 0.000M)
```

IR Counter:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.464%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.83 sec
Total Real time: 3.0 sec
Total Memory Usage: 1171.046875 Mbytes
Reset AAE Options
```

```
innovus 1> *** Starting Verify Geometry (MEM: 1178.8) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify_geometry (CPU: 0:00:00.1  MEM: 63.5M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 15:57:04 2024

Design Name: IR_counter
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (67.3200, 60.4800)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 15:57:04 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Weight DDR3:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.841%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.97 sec
Total Real time: 3.0 sec
Total Memory Usage: 1175.78125 Mbytes
Reset AAE Options
```

```
innovus 1> *** Starting Verify Geometry (MEM: 1181.4) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.2  MEM: 64.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 21:07:15 2024

Design Name: Weight_DDR3
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (168.9600, 161.2800)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 21:07:15 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

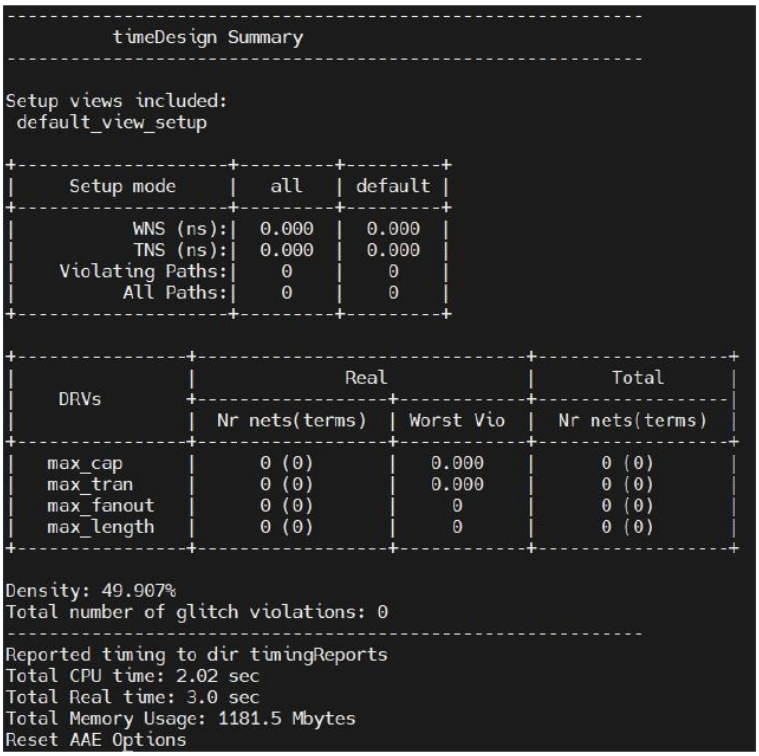
Weight Interface:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.836%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.0 sec
Total Real time: 3.0 sec
Total Memory Usage: 1184.148438 Mbytes
Reset AAE Options
```

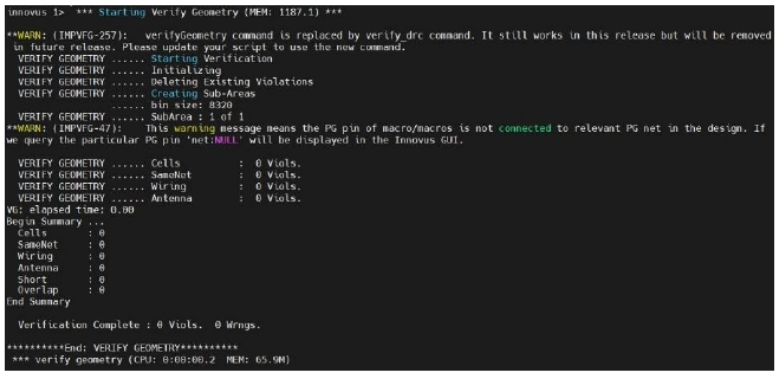
```
innovus 1> *** Starting Verify Geometry (MEM: 1189.8) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

*********End: VERIFY GEOMETRY*********
 *** verify geometry (CPU: 0:00:00.2  MEM: 67.5M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 21:21:30 2024

Design Name: Weight_interface
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (191.4000, 186.4800)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 21:21:30 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Weight FIFO:



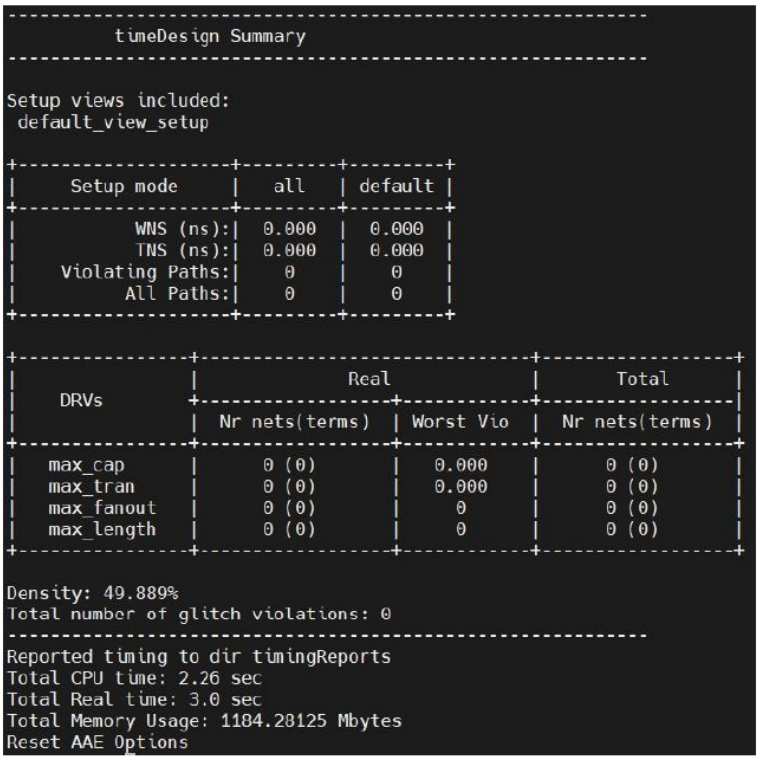
```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.907%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.02 sec
Total Real time: 3.0 sec
Total Memory Usage: 1181.5 Mbytes
Reset AAE Options
```



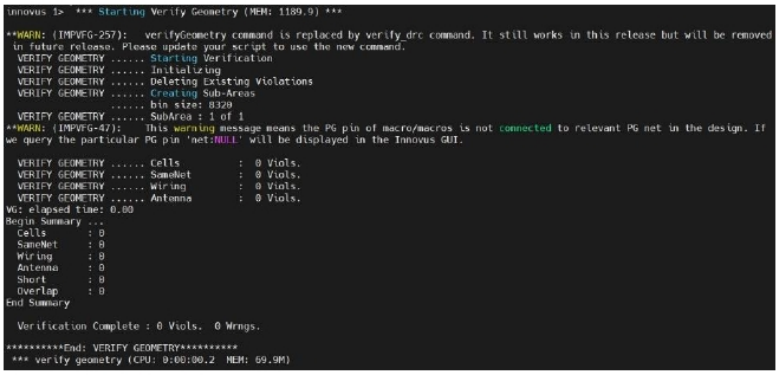
```
innovus 1> *** Starting Verify Geometry (MEM: 1187.1) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY*********
 *** verify geometry (CPU: 0:00:00.2  MEM: 65.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 21:14:14 2024

Design Name: Weight_FIFO
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (193.3800, 176.4000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 21:14:14 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

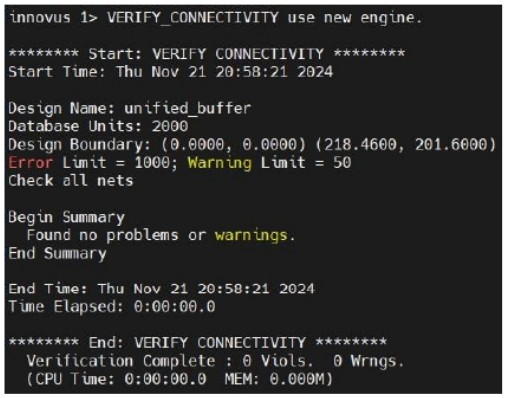
Unified Buffer:



```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.889%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.26 sec
Total Real time: 3.0 sec
Total Memory Usage: 1184.28125 Mbytes
Reset AAE Options
```



```
innovus 1> *** Starting Verify Geometry (MEM: 1189.9) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

*********End: VERIFY GEOMETRY*********
 *** verify geometry (CPU: 0:00:00.2  MEM: 69.9M)
```



```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 20:58:21 2024

Design Name: unified_buffer
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (218.4600, 201.6000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 20:58:21 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Systolic Data Setup:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.871%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.61 sec
Total Real time: 3.0 sec
Total Memory Usage: 1179.90625 Mbytes
Reset AAE Options
```

```
innovus 1> *** Starting Verify Geometry (MEM: 1186.5) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.1  MEM: 61.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 20:36:27 2024

Design Name: sds4x4
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (126.0600, 105.8400)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 20:36:27 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

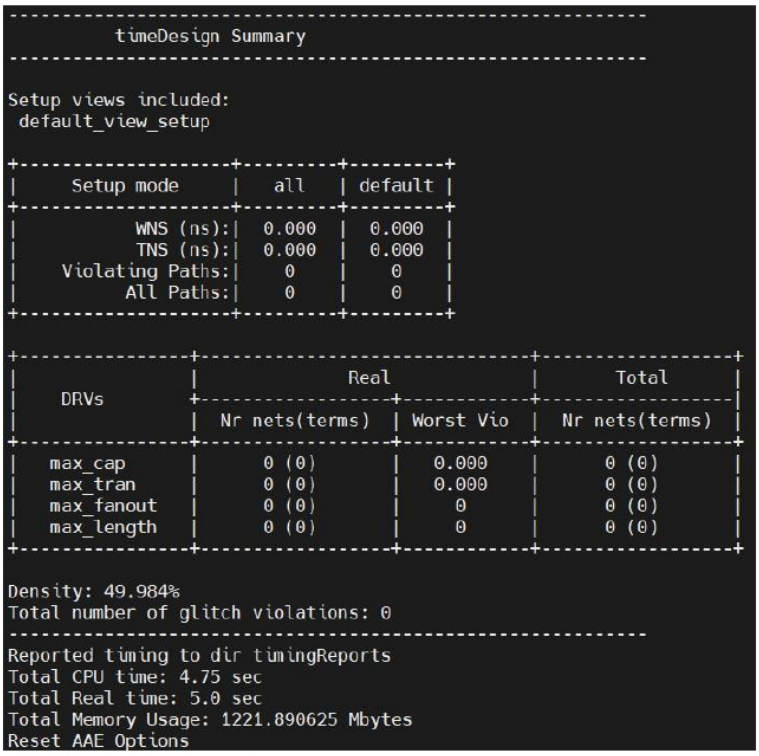
Activation FIFO:

```
------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.907%
Total number of glitch violations: 0
------------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.06 sec
Total Real time: 3.0 sec
Total Memory Usage: 1181.835938 Mbytes
Reset AAE Options
```

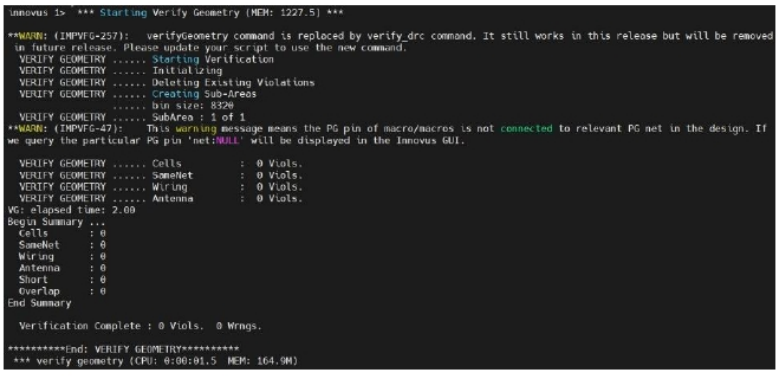
```
innovus 1>  *** Starting Verify Geometry (MEM: 1187.5) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin "net:NULL" will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY*********
 *** verify geometry (CPU: 0:00:00.2  MEM: 65.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 15:29:19 2024

Design Name: Activation_FIFO
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (193.3800, 176.4000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 15:29:19 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Matrix Multiplication Unit:



```
------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.984%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 4.75 sec
Total Real time: 5.0 sec
Total Memory Usage: 1221.890625 Mbytes
Reset AAE Options
```



```
innovus 1> *** Starting Verify Geometry (MEM: 1227.5) ***

**WARN: (IMPVFG-257):   verifyGeometry command is reploced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                        ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 2.00
Begin Summary ...
  Cells     : 0
  SameNet   : 0
  Wiring    : 0
  Antenna   : 0
  Short     : 0
  Overlap   : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:01.5  MEM: 164.0M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 16:10:28 2024

Design Name: MMU4x4
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (573.5400, 559.4400)
Error Limit = 1000; Warning Limit = 50
Check all nets
**** 16:10:28 **** Processed 5000 nets.

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 16:10:28 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.2  MEM: 0.000M)
```

Accumulator:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.875%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 2.39 sec
Total Real time: 4.0 sec
Total Memory Usage: 1192.117188 Mbytes
Reset AAE Options
```

```
innovus 1>  *** Starting Verify Geometry (MEM: 1197.7) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be removed
 in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY*********
 *** verify geometry (CPU: 0:00:00.3  MEM: 78.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 15:18:48 2024

Design Name: Accumulator4x4
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (273.9000, 257.0400)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 15:18:48 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Activation Normalization Unit:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.831%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.84 sec
Total Real time: 3.0 sec
Total Memory Usage: 1172.648438 Mbytes
Reset AAE Options
```

```
innovus 1> *** Starting Verify Geometry (MEM: 1170.3) ***

**WARN: (IMPVFG-257):   VerifyGeometry command is replaced by verify_drc command. It still works in this release but will be remove
d in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
 we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  0 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
VG: elapsed time: 0.00
Begin Summary ...
  Cells     : 0
  SameNet   : 0
  Wiring    : 0
  Antenna   : 0
  Short     : 0
  Overlap   : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

*********End: VERIFY GEOMETRY*********
*** verify geometry (CPU: 0:00:00.1  MEM: 61.9M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 15:41:21 2024

Design Name: Activation_Normalization_Unit4x4
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (60.0600, 45.3600)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 15:41:21 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Control Unit:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 default_view_setup

+--------------------+---------+---------+
|     Setup mode     |   all   | default |
+--------------------+---------+---------+
|           WNS (ns):|  0.000  |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |
|          All Paths:|    0    |    0    |
+--------------------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 49.899%
Total number of glitch violations: 0
------------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.89 sec
Total Real time: 3.0 sec
Total Memory Usage: 1176.03125 Mbytes
Reset AAE Options
```

```
innovus 1>  *** Starting Verify Geometry (MEM: 1179.6) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this release but will be remove
d in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea : 1 of 1
**WARN: (IMPVFG-47):    This warning message means the PG pin of macro/macros is not connected to relevant PG net in the design. If
 we query the particular PG pin 'net:NULL' will be displayed in the Innovus GUI.

  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
Wk: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.1  MEM: 63.5M)
```

```
innovus 1> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 21 15:51:26 2024

Design Name: Controller
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (108.2400, 95.7600)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 21 15:51:26 2024
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```