

# IIIT-B Chip Design Studio Weekly Report

Week : 1

Team Name/Project: Circuit Crafters

## Sparse Systolic Array for AI Acceleration and Matrix Computation Based Chip Design

### 1. Updates

**Current Progress:** During Week-1, we set up the AMD Vivado design environment and implemented the fundamental compute building block required for our proposed sparse systolic array architecture. A sparsity-aware Processing Element (PE) based on a Multiply–Accumulate (MAC) operation was designed using Verilog HDL. The PE supports conditional execution using a valid signal to enable or skip accumulation, representing zero-skipping behavior. Behavioral simulation was performed to verify correct functionality.

#### Challenges Faced:

- Initial learning curve in using AMD Vivado and understanding RTL schematic generation.
- Interpreting simulation waveforms and hexadecimal accumulator values during verification.
- Abstracting sparsity behavior without implementing full CSR/COO logic in the first week.

#### Next Steps:

- Extend the PE by adding automatic zero-detection logic.
- Replicate the PE to form a small 2×2 or 4×4 systolic array.
- Begin designing sparse dataflow control logic for PE activation.
- Perform synthesis to analyse resource utilization in Vivado.

### 2. GitHub Link: Provide the GitHub repository link for the project, if any:

**Repository:** (Not created yet)

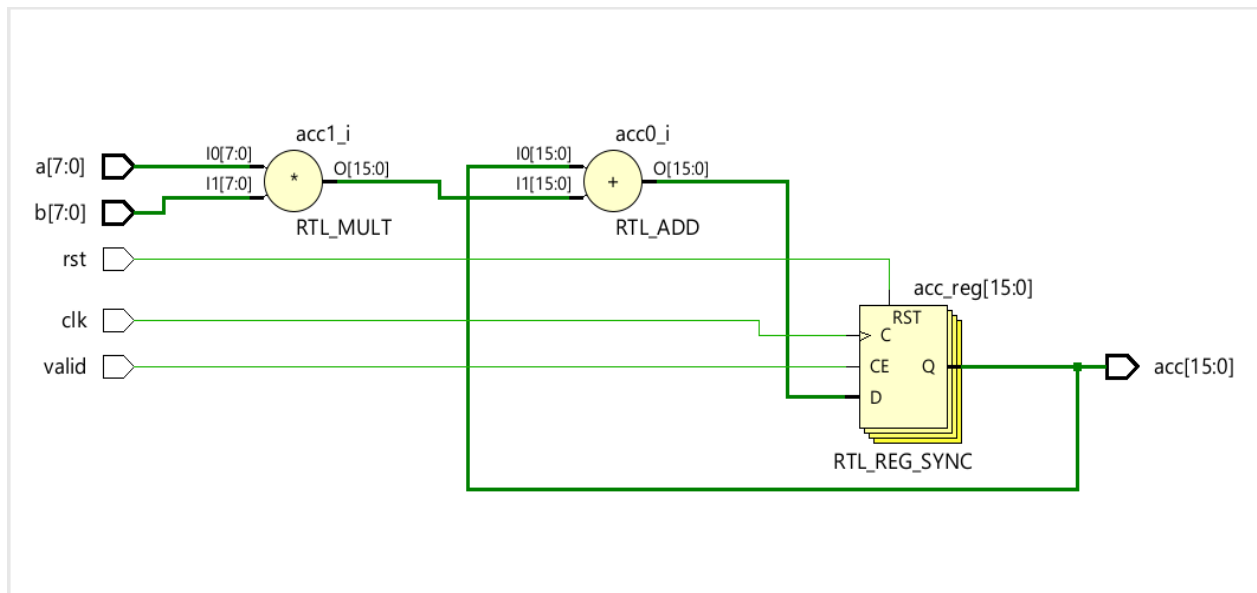
### 3. Project Idea

The objective of this project is to design a sparsity-aware systolic array architecture optimized for matrix multiplication in AI acceleration. Since modern AI workloads often use sparse matrices, conventional systolic arrays waste energy and computation on zero-valued operands. Our approach introduces zero-skipping mechanisms and compressed data representations to reduce redundant MAC operations, improve energy efficiency, and enhance throughput. The system targets AI accelerators, DSP applications, and low-power edge devices.

## 4. Schematic/Simulations

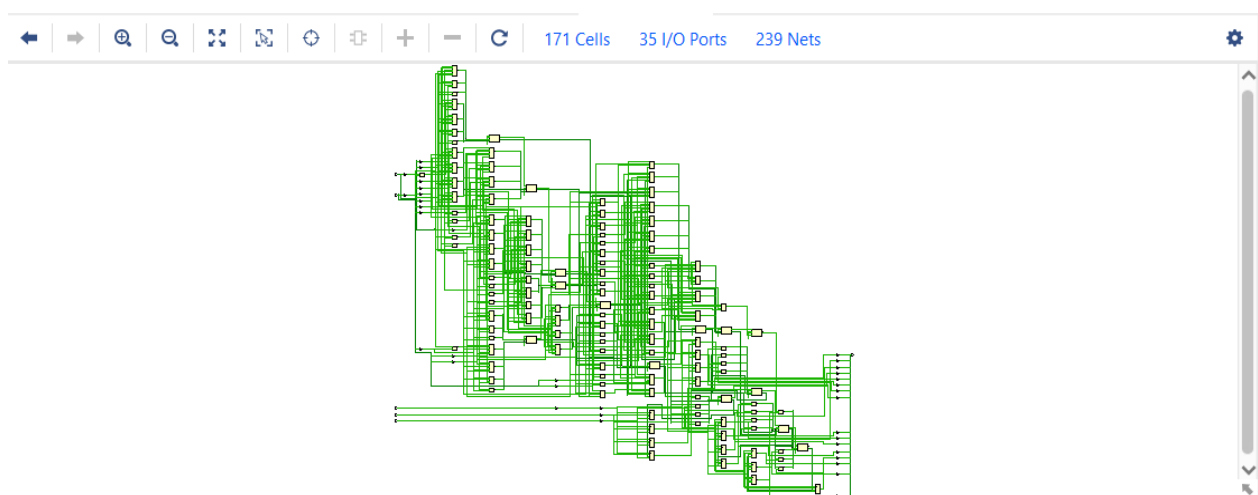
### • Schematics:

**Figure 1:** RTL Schematic of Sparsity-Aware Processing Element (PE)



**Figure 1** shows the RTL schematic of the sparsity-aware Processing Element (PE) implemented in AMD Vivado. The PE consists of an 8-bit multiplier, a 16-bit adder, and a clock-enabled accumulator register. The valid signal controls the accumulator update, enabling zero-skipping behavior.

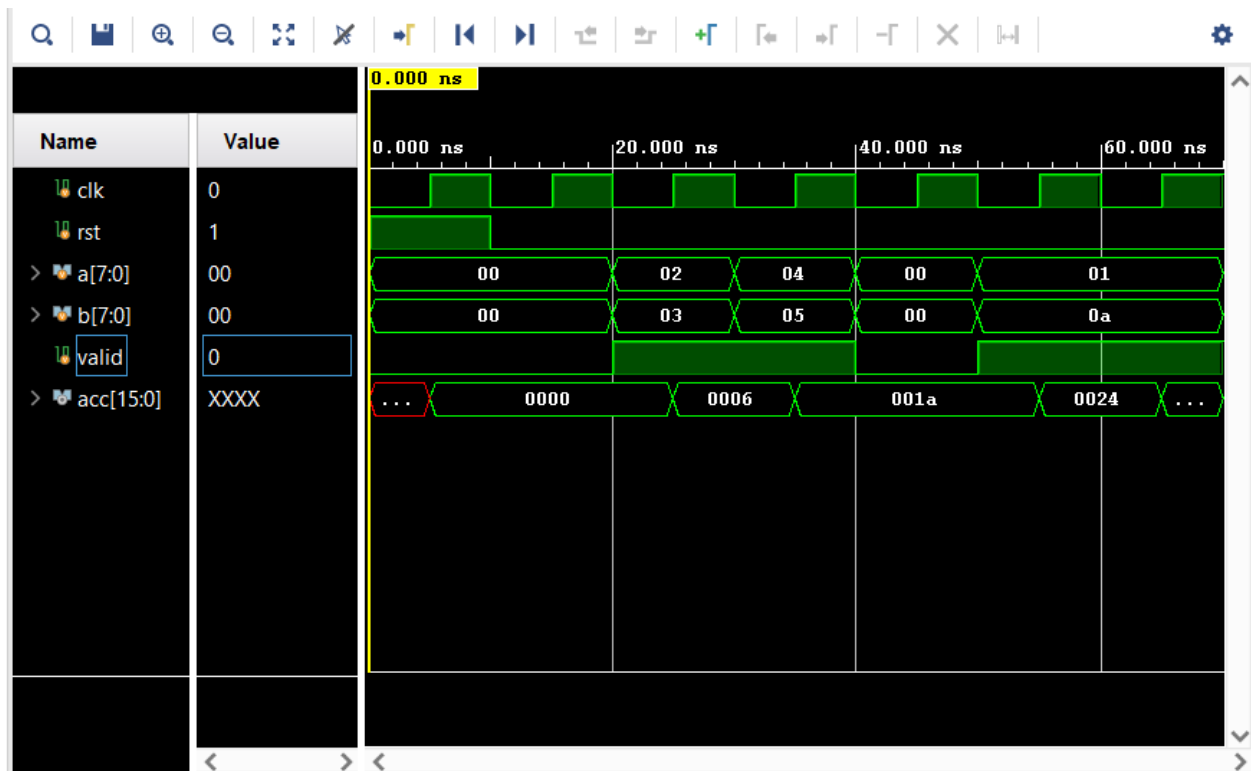
**Figure 2:** RTL Elaborated Netlist of the Processing Element (PE)



**Figure 2** shows the RTL elaborated netlist of the sparsity-aware Processing Element (PE) generated in AMD Vivado. The diagram represents the logical connectivity between the multiplier, adder, and accumulator register after RTL elaboration.

- **Simulation Results:**

Figure 3: Behavioral Simulation Waveform of the Processing Element (PE)



**Figure 3** shows the behavioral simulation waveform of the sparsity-aware Processing Element. When the valid signal is asserted, the PE performs a multiply–accumulate operation, updating the accumulator output. When valid is de-asserted, the accumulator retains its previous value, demonstrating zero-skipping behavior. The output values match the expected results for the applied input vectors.

**Example:**

When **a=02** and **b=03** with **valid=1**, the accumulator updates to **0x0006**, confirming correct MAC functionality.

## 5. Analysis

- **Key Findings:**

- The MAC-based Processing Element functions correctly and produces expected accumulated results.
- Clock-enable based control effectively skips unnecessary accumulation, modelling sparsity-aware behaviour.
- The PE forms a solid foundational block for building larger systolic arrays.

- **Insights or Learnings:**

- A Processing Element is the fundamental unit of matrix multiplication hardware.
- Zero-skipping can be efficiently implemented using control signals without modifying the core data path.
- Vivado RTL schematics and waveforms provide valuable insight into hardware behaviour.

- **Improvements or Modifications Needed:**

- We need to replace externally driven valid signal with automatic zero-detection logic.
- We must integrate sparse matrix indexing and scheduling logic in later stages.
- We should scale the design to a multi-PE systolic array for full matrix multiplication.

### TEAM DETAILS:

**Team Name:** Circuit Crafters

1. Jeswin S - [sec23ec089@sairamtap.edu.in](mailto:sec23ec089@sairamtap.edu.in) ([sec23ec089@iiitb.net](mailto:sec23ec089@iiitb.net))
2. Moneswaran P - [sec23ec225@sairamtap.edu.in](mailto:sec23ec225@sairamtap.edu.in) ([sec23ec225@iiitb.net](mailto:sec23ec225@iiitb.net))
3. Mokshith M - [sec23ec192@sairamtap.edu.in](mailto:sec23ec192@sairamtap.edu.in) ([sec23ec192@iiitb.net](mailto:sec23ec192@iiitb.net))

**College Name:** Sri Sai Ram Engineering College, Chennai.