,

Assignment Six
Hardware Lab: CS224

Shelke Durgesh Balkrishna: 230101093

Kavya Kumar Agrawal: 230101053

Parth Sunil Aher: 230101072

Dhruv Pansuriya: 230101071

1st April 2025

1 Problem Statement

The given function is defined as:

```
module MaxMin(
input [3:0] wAddress, // address where dataIn to be written
input [4:0] dataIn, // data to be written at wAddress location
input dataValid, // will be on when we want to write input data to the memory
input clk, // clock signal
output reg outValid, // is 1 when processing is done and valid data is showing at output
output reg [3:0] outAddress, // address corresponding to dataOut
output reg [4:0] dataOut // output data corresponding to outAddress location
);

Memory operations and max/min finding algorithm
Read data into memory when dataValid is 1
Process data to find maximum and minimum values
Output results one by one with outValid set to 1
```

2 ASM Chart

The ASM diagram represents a sequential process for calculating and outputting the minimum and maximum values from a dataset. It begins with initialization, followed by input handling, memory referencing, and iterative comparisons to determine the minimum and maximum values. Conditional branching guides the flow between states, ensuring accurate updates to variables holding these values. Once the calculations are complete, the results are outputted, and the process terminates. The diagram effectively combines memory operations, decision-making, and output handling in a structured manner.

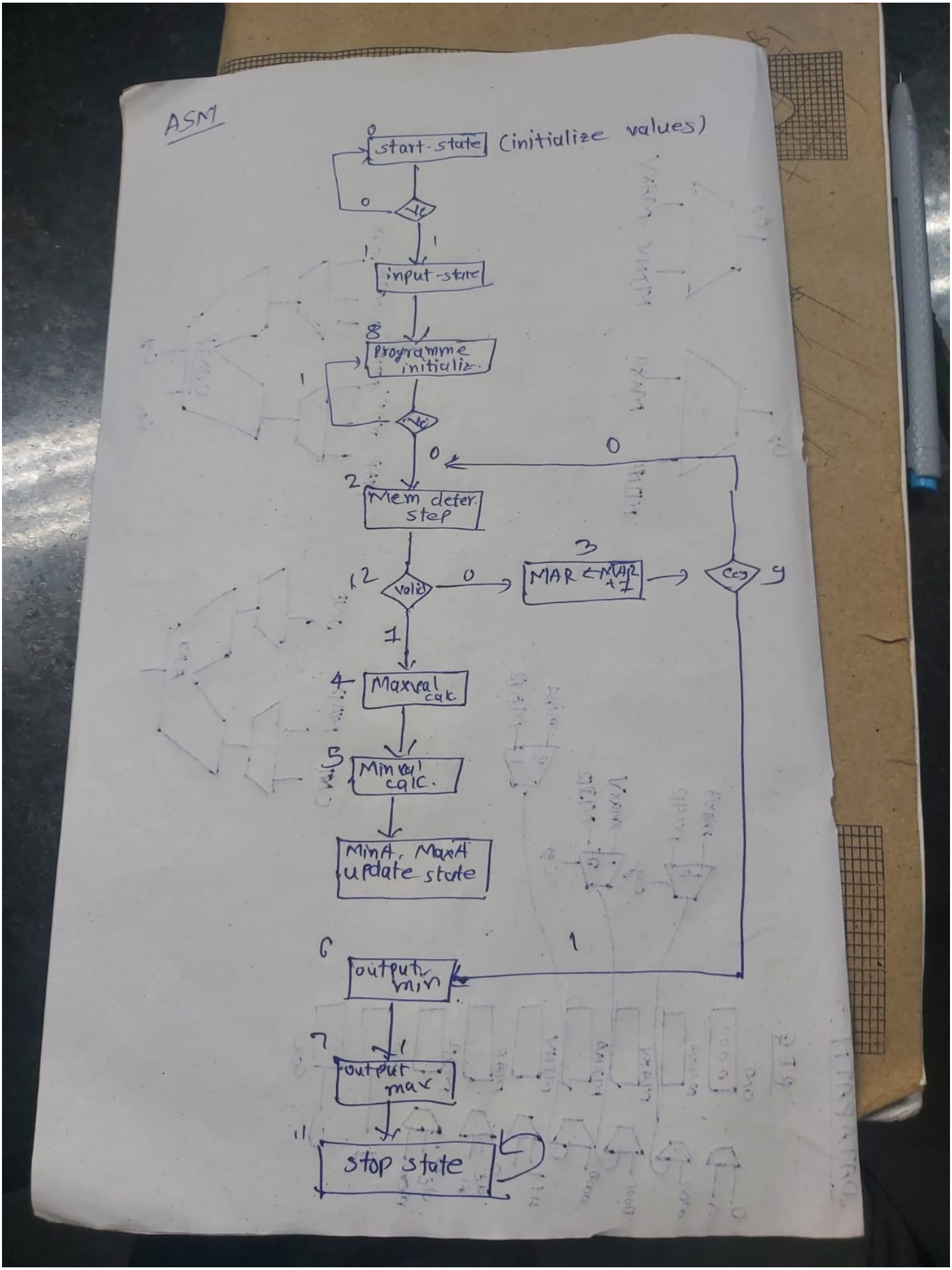

Figure 1: ASM Chart (Conceptual Representation)

3 Controller Section

3.1 FSM Chart

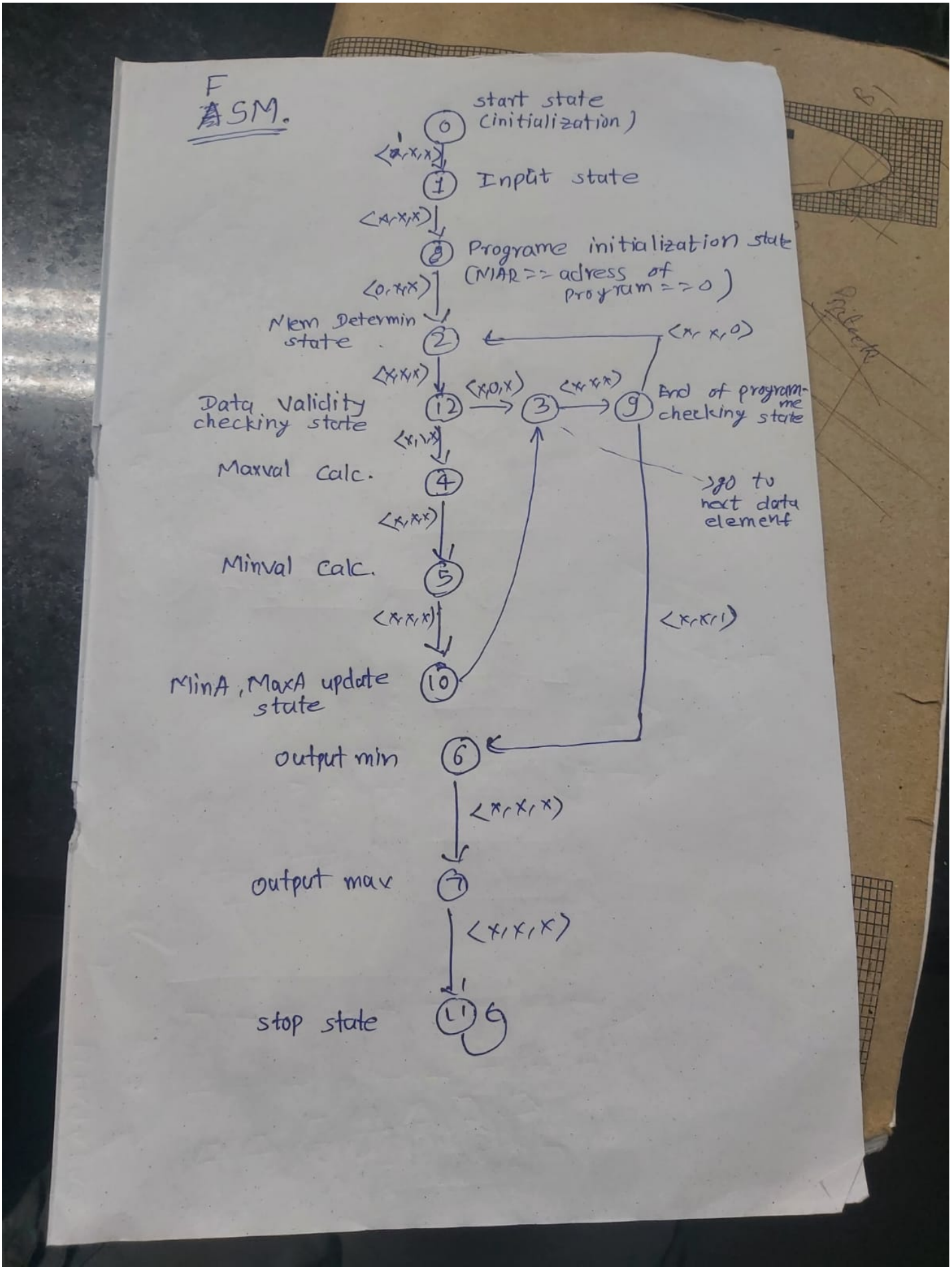

Figure 2: FSM Chart

4 Datapath Section

The datapath includes several registers: CNT (initialized to zero), MAXA and MAXV (maximum address and value), MINA and MINV (minimum address and value), MAR (Memory Address Register), MDR (Memory Data Register), and a constant register called ONE (value of 1). These registers are connected via multiplexers and control signals, which select inputs based on signals like CY (carry) and CE (control enable). The datapath features an adder (ADD) for arithmetic operations with inputs such as CNT and ONE, as well as a comparator (COMP) for comparing MAR, MAXV, and MINV. Outputs are represented by AO (address output) and DO (data output) sourced from the minimum and maximum registers.

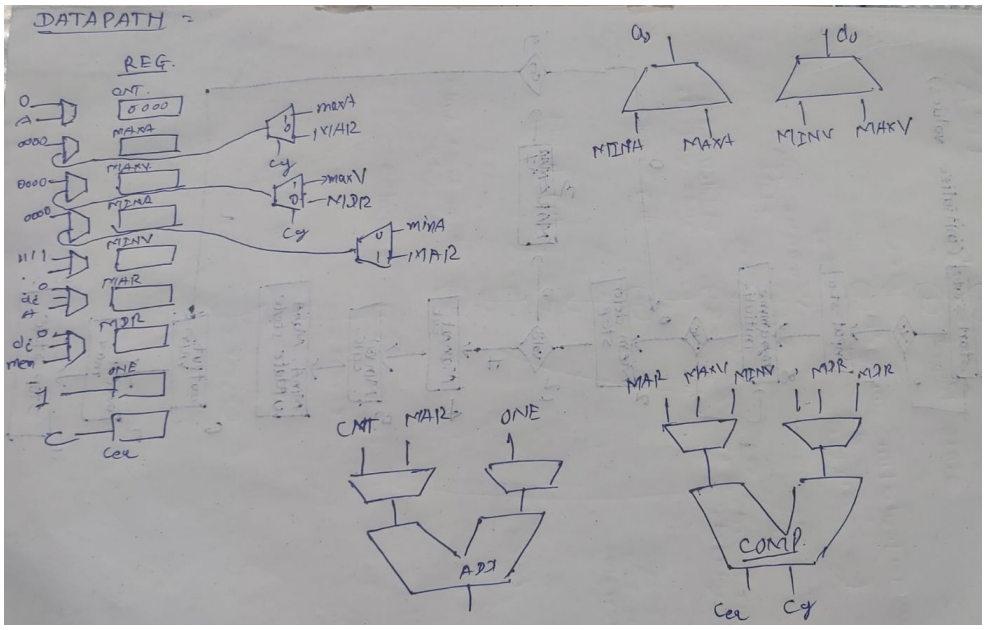

Figure 3: Datapath Diagram

5 Testbench

5.1 Simulation Results

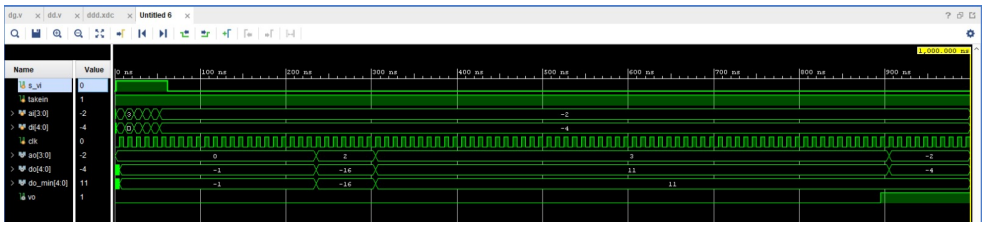

Figure 4: Simulation waveform in Vivado (Placeholder)

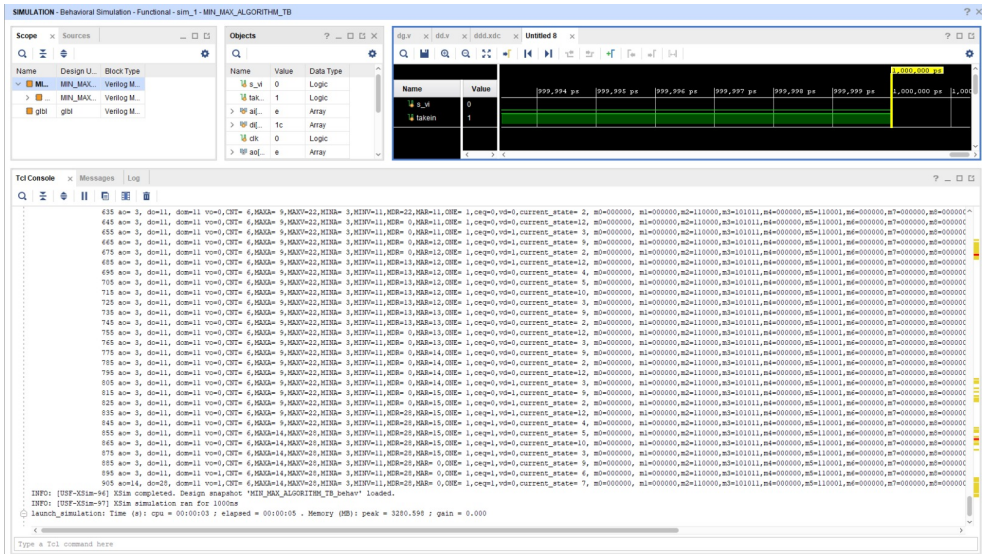

Figure 5: Simulation data in Vivado

6 I/O Pin Mapping

Table 1: I/O Pin Mapping

Signal Name	Port Name	FPGA Pin
s_vi	s_vi	V10
ai[3]	ai[3]	U11
ai[2]	ai[2]	U12
ai[1]	ai[1]	H6
ai[0]	ai[0]	T13
di[4]	di[4]	R16
di[3]	di[3]	U8
di[2]	di[2]	T8
di[1]	di[1]	R13
di[0]	di[0]	U18
ao[0]	ao[0]	H17
ao[1]	ao[1]	K15
ao[2]	ao[2]	J13
ao[3]	ao[3]	N14
takein	takein	T18
vo	vo	V11
do_min[4]	do_min[4]	V14
do_min[3]	do_min[3]	V15
do_min[2]	do_min[2]	T16
do_min[1]	do_min[1]	U14
do_min[0]	do_min[0]	T15
do[0]	do[0]	R18
do[1]	do[1]	V17
do[2]	do[2]	U17
do[3]	do[3]	U16
do[4]	do[4]	V16

7 Images

7.1 Block Diagrams

This image displays a schematic diagram of a digital system design in Vivado. It includes three main modules controller_module, memory_module, and datapath_module, interconnected with various signals. Inputs like ai[3:0], di[4:0], and control signals (takein, s_vi) are connected to the controller, while outputs such as ao[3:0], do[4:0], and vo are generated. The design uses multiplexers, registers, and memory elements to manage data flow and control logic.

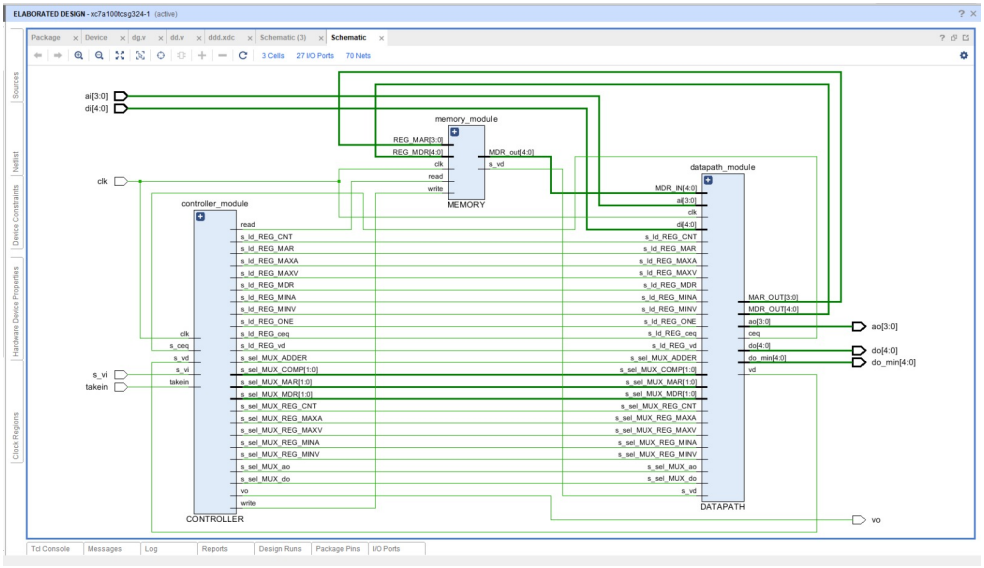

Figure 6: Block Diagram 1

This image represents a Vivado block design showcasing a digital system architecture. It includes three primary modules: controller_module, memory_module, and datapath_module, connected through a complex network of signals and nets. The design features 31 cells, 27 I/O ports, and 122 nets. Input signals such as ‘ai[3:0]‘, ‘di[4:0]‘, and control signals (‘takein‘, ‘s_vi‘) are routed into the modules, while outputs like ‘ao[3:0]‘, ‘do[4:0]‘, and ‘vo‘ are generated. The system uses buffers, multiplexers, and registers to manage data flow and control logic efficiently.

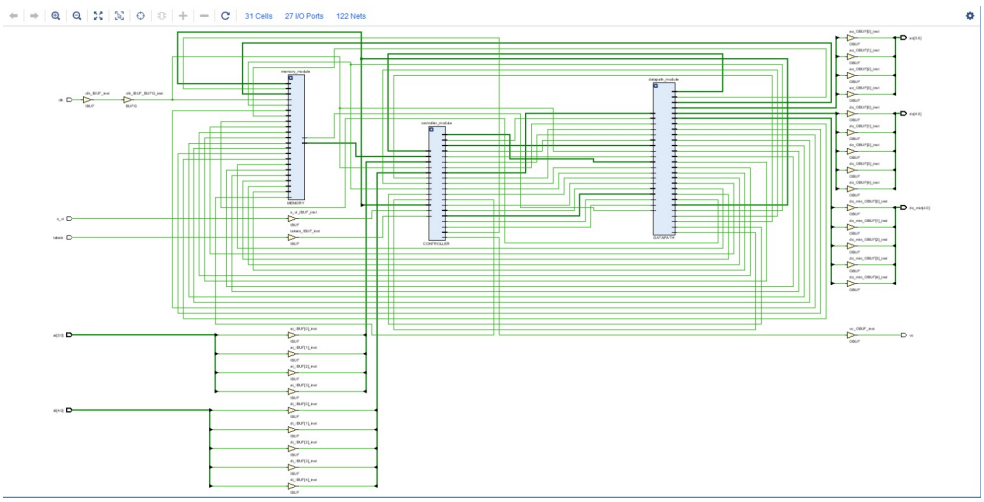

Figure 7: Block Diagram 2

8 Conclusion

In this assignment, we successfully implemented the specified iterative function in Verilog and deployed it on the Artix-7 XC7A100T-CSG324 FPGA using Vivado 2018. The design process included creating a structured ASM chart, an optimized FSM controller, and a well-defined datapath to efficiently execute the computations.

We analyzed hardware costs and performance trade-offs by evaluating different scheduling strategies. Our findings indicated that a balanced approach (K=4) provided the best trade-off between area and latency. The FSM state transitions and Karnaugh map simplifications contributed to reducing the complexity of the control logic, ensuring efficient execution.

The simulation results confirmed the correctness of our design, and the generated waveforms demonstrated that the system behaves as expected. We optimized hardware resource utilization to minimize unnecessary multipliers and adders, resulting in a cost-effective implementation.

Overall, this project offered valuable insights into digital design methodologies, finite state machine implementations, and datapath optimization, which are essential for developing efficient FPGA-based hardware solutions.