Hardware Lab (CS 224)
Assignment 5

Group No. 18

Team Members:

Daksh Agarwal (230101123)
Aditya Shukla (230101115)
Kartik Maheshwari (230101116)
Ryan Aggarwal (230101117)

March 18, 2025

1 Finite State Machine (FSM)

The heart of the system is the FSM, which orchestrates the sequence of operations based on inputs and internal states. Here's how it operates:

1.1 State Transition

The FSM transitions between different states, each representing a specific stage of the computation process. Transitions are triggered by clock pulses and depend on various input conditions.

1.2 Operation Control

In each state, the FSM controls which operations are enabled or disabled. It activates or deactivates the enable signals for arithmetic operations, data storage, and comparison based on the current state and input conditions.

1.3 Data Flow Management

The FSM manages the flow of data between different modules. It selects the appropriate inputs for arithmetic operations, decides which data to store in registers, and determines when to perform comparisons.

2 Arithmetic and Comparison Modules

The FSM interacts with several modules to perform arithmetic operations (addition, subtraction, multiplication) and comparisons:

2.1 Adder-Subtractor Module

Performs addition or subtraction of two operands based on a control signal. It handles both addition and subtraction operations efficiently by exploiting the value of "invert".

2.2 Multiplier Module

Computes the product of two operands using simple multiplication logic. It efficiently calculates the multiplication result.

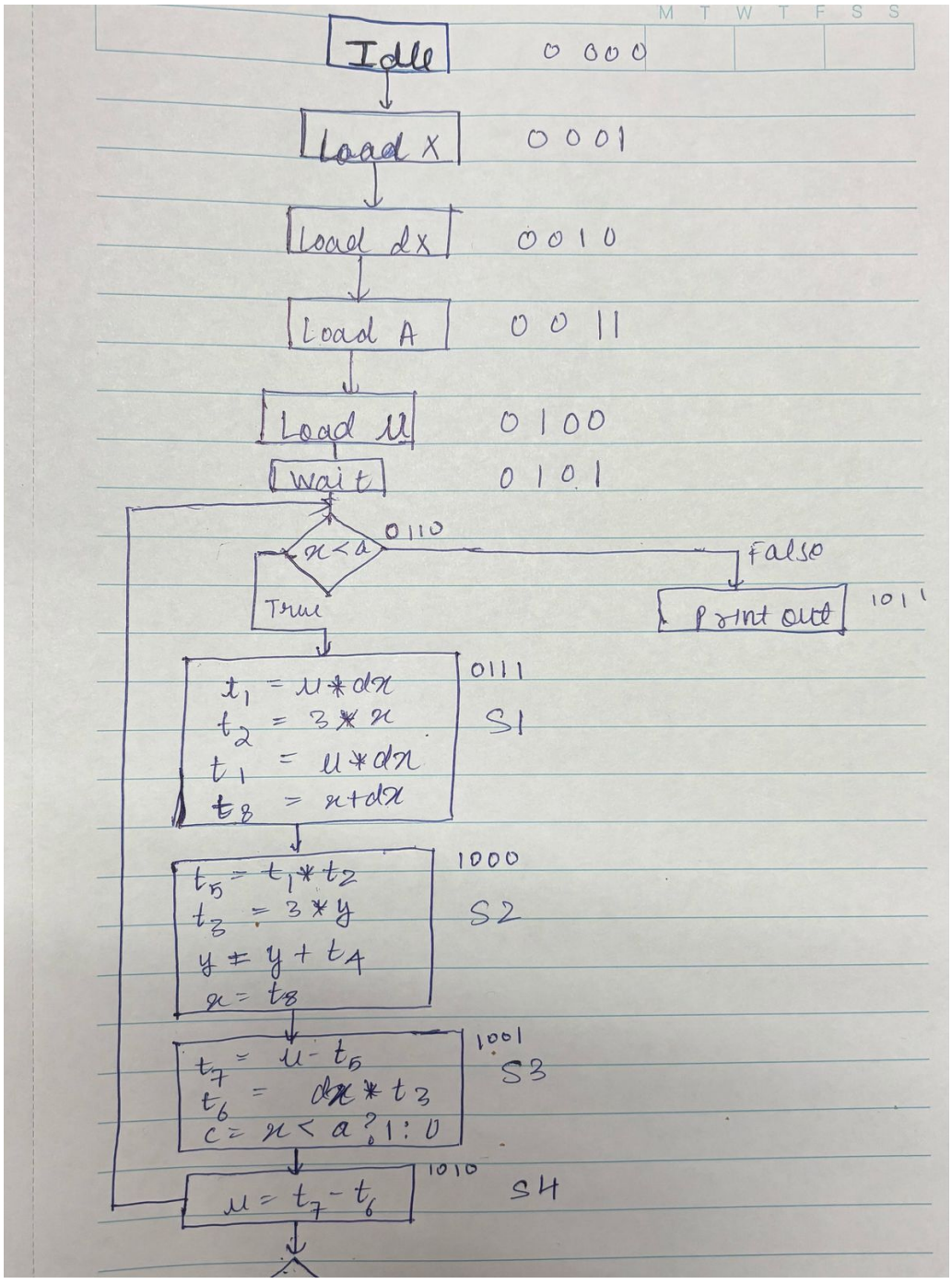
Idle
0000
Load X
0001
Load dx
0010
Load A
0011
Load u
0100
wait
0101
x<a
0110
False
Print out
1011
True
t1 = u*dx
t2 = 3*x
t1 = u*dx
t8 = x+dx
0111
S1
t5 = t1*t2
t3 = 3*y
y = y + t4
x = t8
1000
S2
t7 = u - t5
t6 = dx * t3
c = x < a ? 1 : 0
1001
S3
u = t7 - t6
1010
S4

Figure 1: FSM

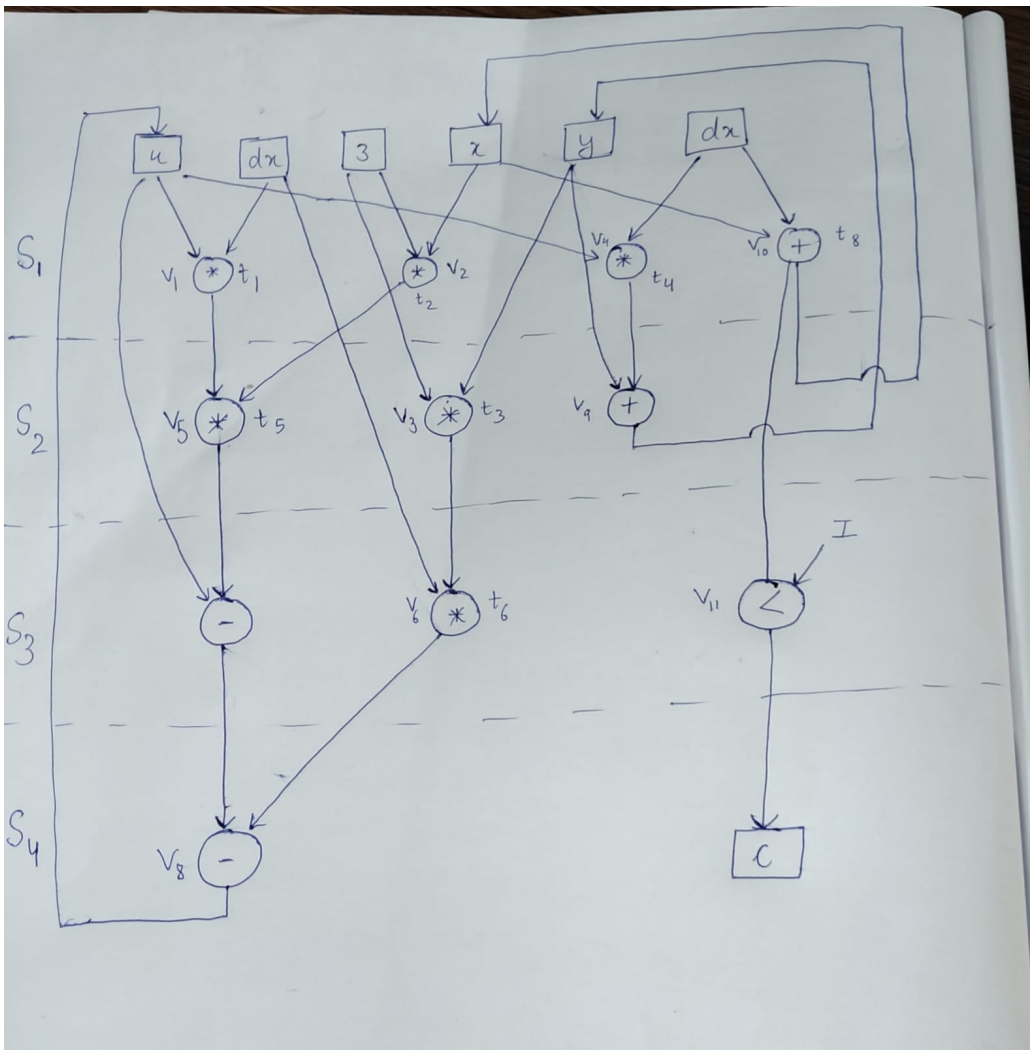

Figure 2: FU Allocation and Binding

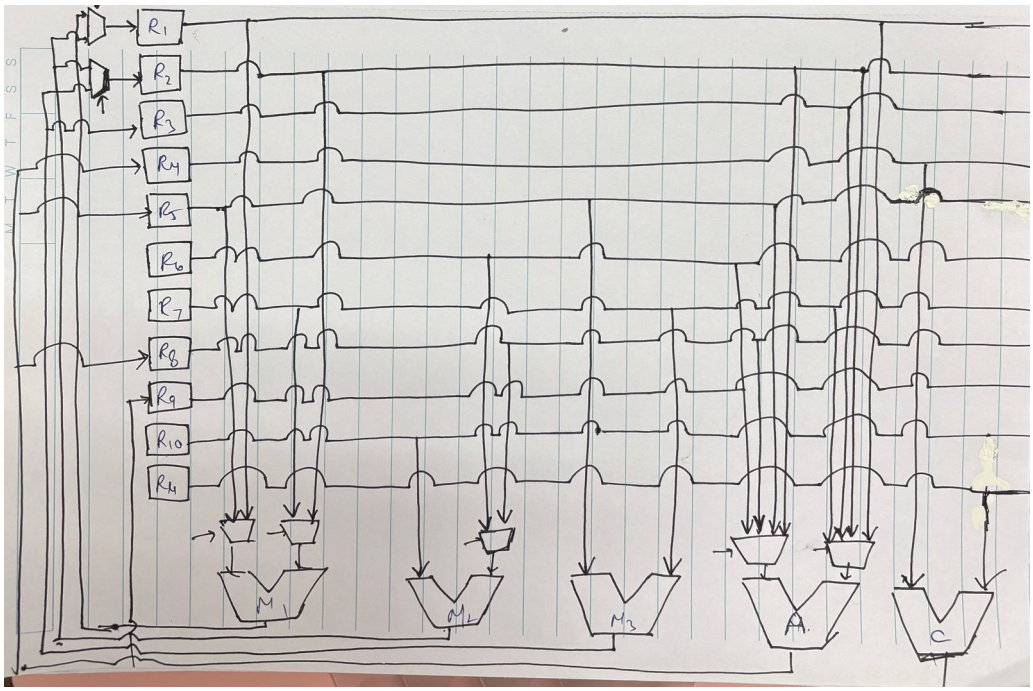

Figure 3: Data Path Generation

2.3 Comparison Module

Determines whether one operand is less than the other. It outputs a signal indicating the result of the comparison.

3 Multiplexers and Registers

Multiplexers (MUX) and registers play a crucial role in data selection and storage:

3.1 MUX Modules

Select one of multiple input data based on control signals. They facilitate data routing and selection within the system.

3.2 Register Modules

Store data temporarily and allow for selective loading based on control signals. They enable the system to hold intermediate results and input data for further processing.

4 Overall Operation

Here's how the system operates as a whole:

1. **Initialization:** The system starts in an initial state where all operations are disabled, awaiting input signals.
2. **State Transition:** Upon receiving input signals (e.g., "ready" signal), the FSM transitions to the next state, enabling specific operations and determining the flow of data.
3. **Arithmetic and Comparison:** In each state, the enabled operations (such as addition, subtraction, multiplication, and comparison) are performed based on the input data and control signals.
4. **Data Storage:** Intermediate and final results are stored in registers for further processing or output.
5. **State Advancement:** After completing the current state's operations, the FSM transitions to the next state based on the current state and input conditions.

6. **Completion and Output:** The system continues through the state transitions until the computation is complete. Final results are outputted when the computation is finished.