CS201 – Computer Organization

Project — A simple Processor

Rupesh Yadav (Btech 3rd year)

1. Objective

We have to design and simulate a simple processor that contains a number of 8 bit registers, a multiplexer, an ALU unit, and a control unit (finite state machine).

- Data is input to this system via the 8 bit DIN input. This data can be loaded through the 8 bit wide multiplexer into the various registers, such as R0….R7 and A. The multiplexer also allows data to be transferred from one register to another.
- The multiplexer's output wires are called a bus in the figure because this term is often used for wiring that allows data to be transferred from one location in a system to another.
- Addition, Subtraction, And, Or, Xor and Complement operations is performed by using the multiplexer to first place one 8 bit number onto the bus wires and loading this number into register A.
- Once this is done, a second 8 bit number is placed onto the bus, the adder/subtractor unit performs the required operation, and the result is loaded into register G.
- The data in G can then be transferred to one of the other registers as required.
- The system can perform different operations in each clock cycle, as governed by the control unit. This unit determines when particular data is placed onto the bus wires and it controls which of the registers is to be loaded with this data.

Table 1 lists the instructions that the processor has to support for this exercise. The left column shows the name of an instruction and its operand. The meaning of the syntax RX [RY] is that the contents of register RY are loaded into register RX. The mv (move) instruction allows data to be copied from one register to another. For the mvi (move immediate) instruction the expression RX.

Table 1: Instructions performed in the processor

Operation	Function
mv Rx, Ry	Rx [Ry]
mvi Rx, #D	Rx D
add Rx, Ry	Rx [Rx]+[Ry]
sub Rx, Ry	Rx [Rx]-[Ry]
and Rx, Ry	Rx [Rx]Λ[Ry]
or Rx, Ry	Rx [Rx]V[Ry]
xor Rx, Ry	Rx [Rx] [Ry]
not Rx	Rx ¬[Rx]

Each instruction can be encoded and stored in the IR register using the 9-bit format IIIXXXYYY, where III represents the instruction, XXX gives the RX register, and YYY gives the RY register.

2. Architecture and Control Signals

Based on the required specification, the system can be represented by the following architecture.

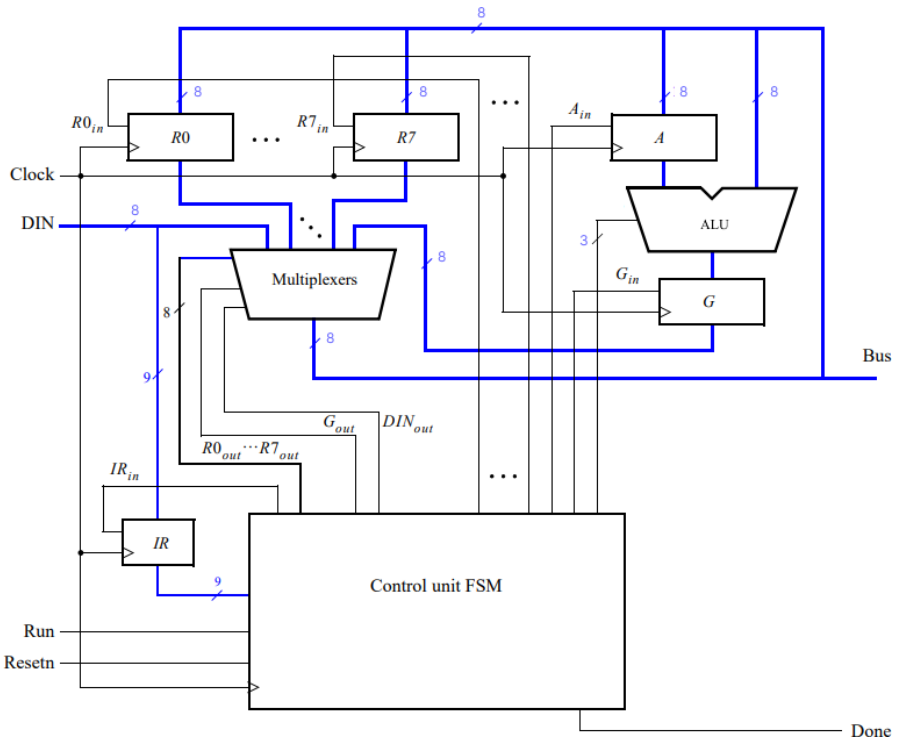


Fig. Architecture of the processor

Control signals asserted in each instruction/time step.

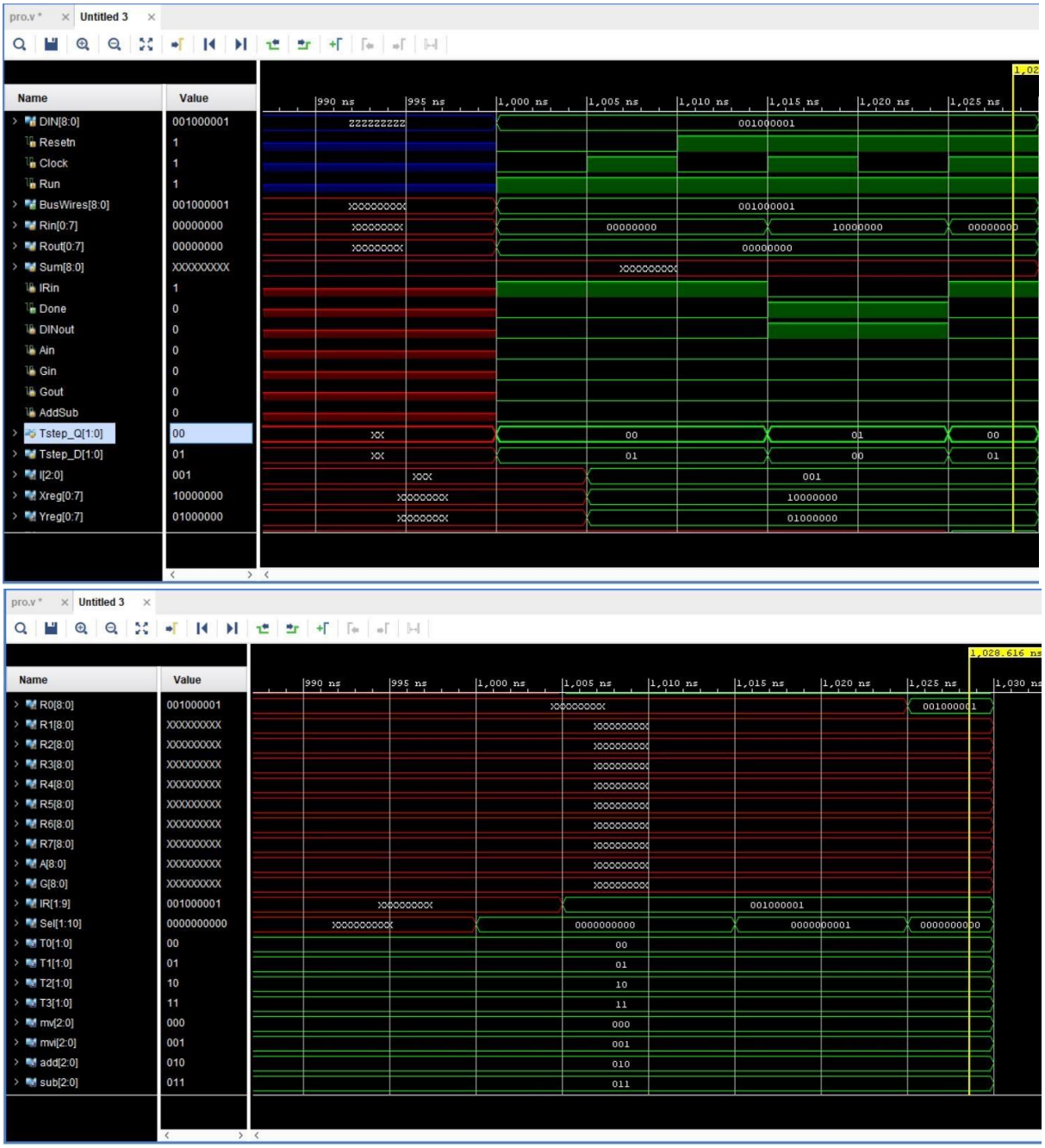
	T_1	T_2	T_3
(mv): I_0	RY_{out}, RX_{in}, Done		
(mvi): I_1	DIN_{out}, RX_{in}, Done		
(add): I_2	RX_{out}, A_{in}	RY_{out}, Gin, ALU	G_{out}, RX_{in}, Done
(SUB): I_3	RX_{out}, A_{in}	RY_{out}, Gin, ALU	G_{out}, RX_{in}, Done
(and): I_4	RX_{out}, A_{in}	RY_{out}, Gin, ALU	G_{out}, RX_{in}, Done
(OR): I_5	RX_{out}, A_{in}	RY_{out}, Gin, ALU	G_{out}, RX_{in}, Done
(XOR): I_6	RX_{out}, A_{in}	RY_{out}, Gin, ALU	G_{out}, RX_{in}, Done
(neg): I_7	RX_{out}, Gin,	RX_{in}, Done	

3. Output

1. IR = 000001111

IR = 000 001 111
000 = Move
001 = R1
111 = R7
Hence it moves the data of R7 into R1. The same can be verified in the output waveform.

4. IR = 001000001



IR = 001 000 001

001 = Move immediate

000 = R0

001 = R1

Din = 001000001

Hence it moves the Din to the register R0. The Ry is ignored in this case. The same can be verified in the output waveform.

5. IR = 010001010

IR = 010 001 010

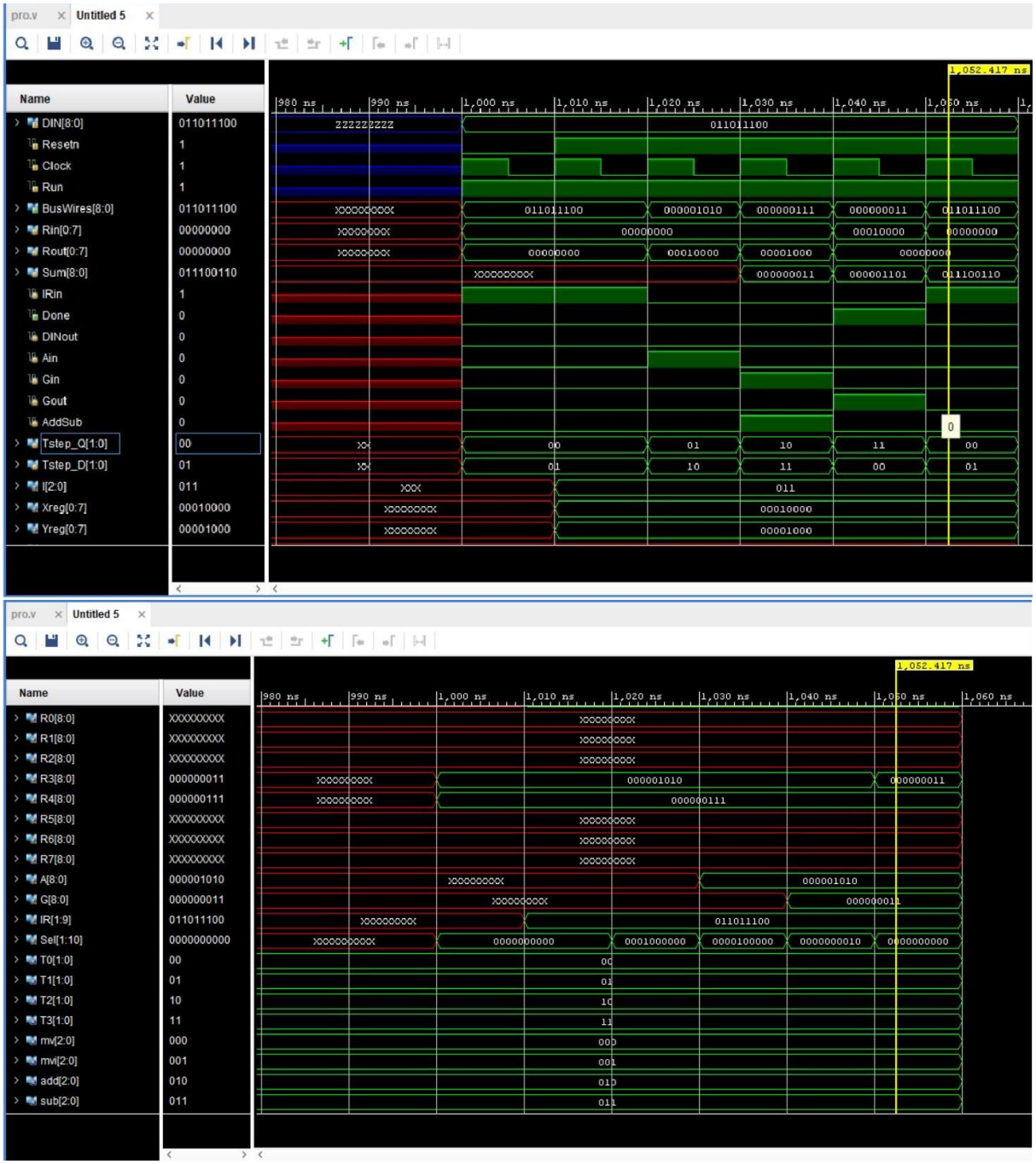
010 = Sum

001 = R1

010 = R2

Hence it should add the data of the registers R1 and R2 and store the sum in R1. The same can be verified in the output waveform.

6. IR = 011011100



IR = 011 011 100

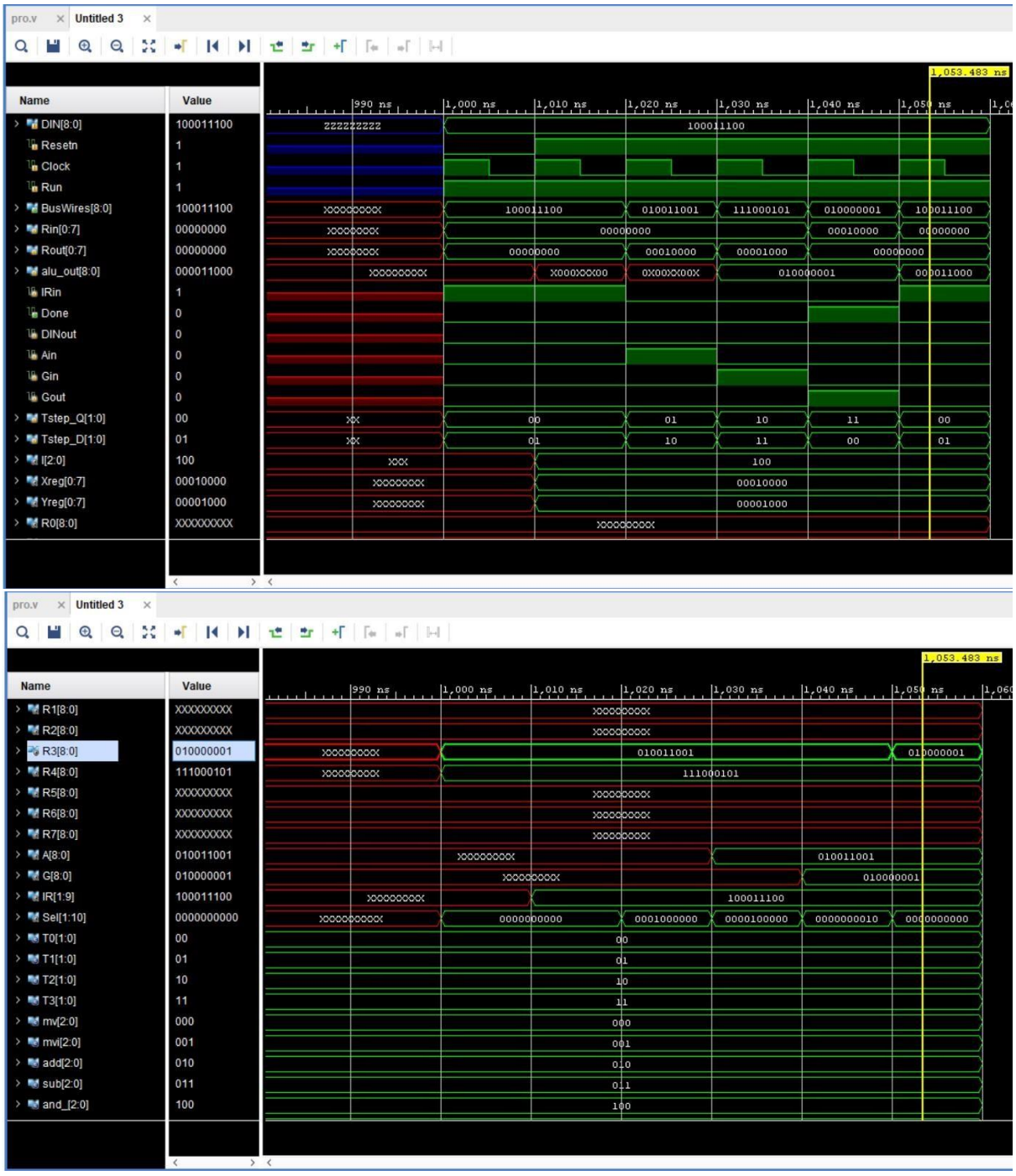
011 = Subtract

011 = R3

100 = R4

Hence it should subtract R4 from R3 and store in the register R3. The same can be verified in the output waveform.

7. IR = 100011100



IR = 100 011 100

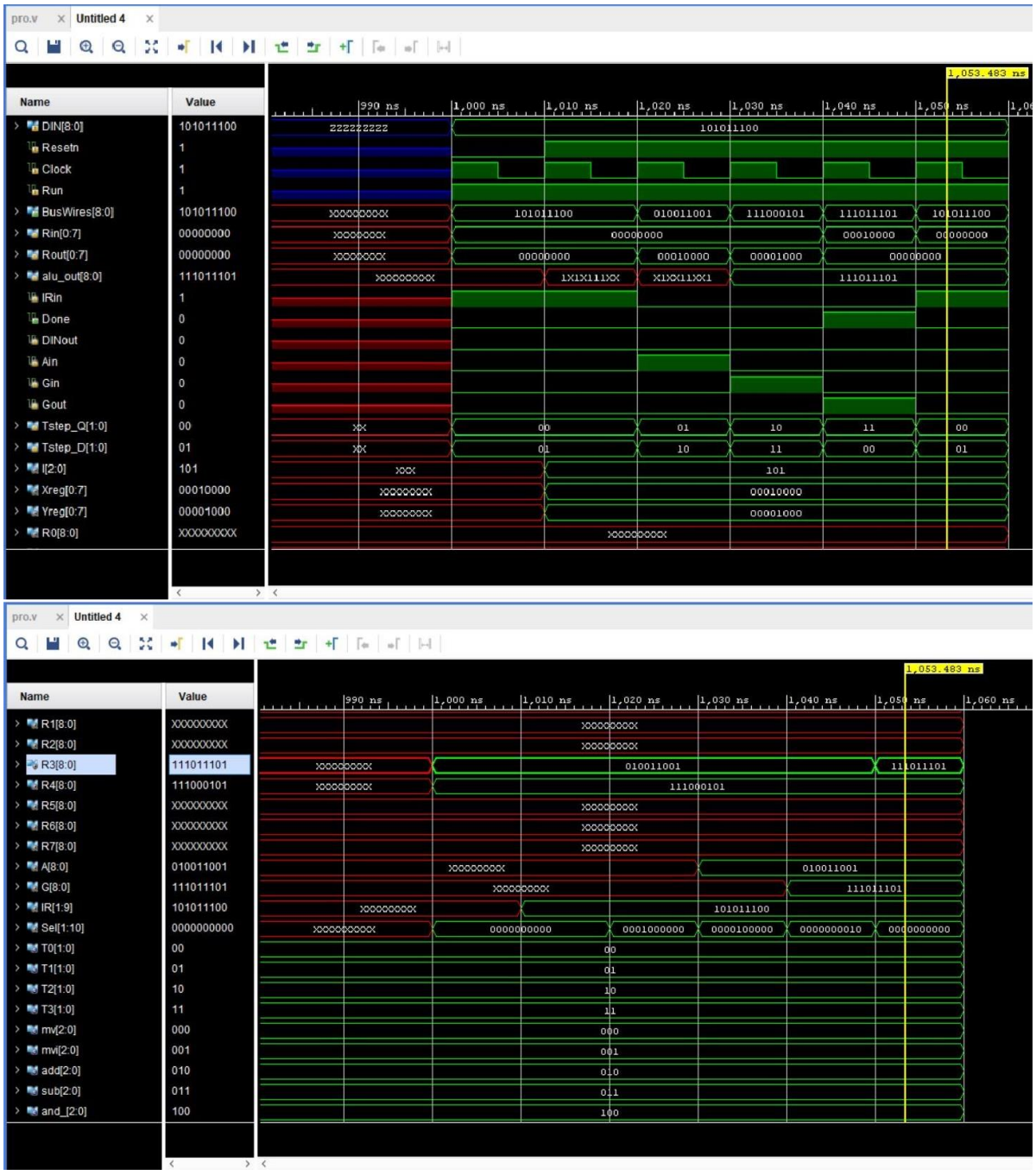
100 = And

011 = R3

100 = R4

Hence it should store the output of AND operation of data in R3 and R4 in R3. The same can be verified in the output waveform.

8. IR = 101011100



IR = 101 011 100

101 = OR

011 = R3

100 = R4

Hence it should store the output of OR operation of data in R3 and R4 in R3. The same can be verified in the output waveform.

9. IR =110011100

IR = 110 011 100

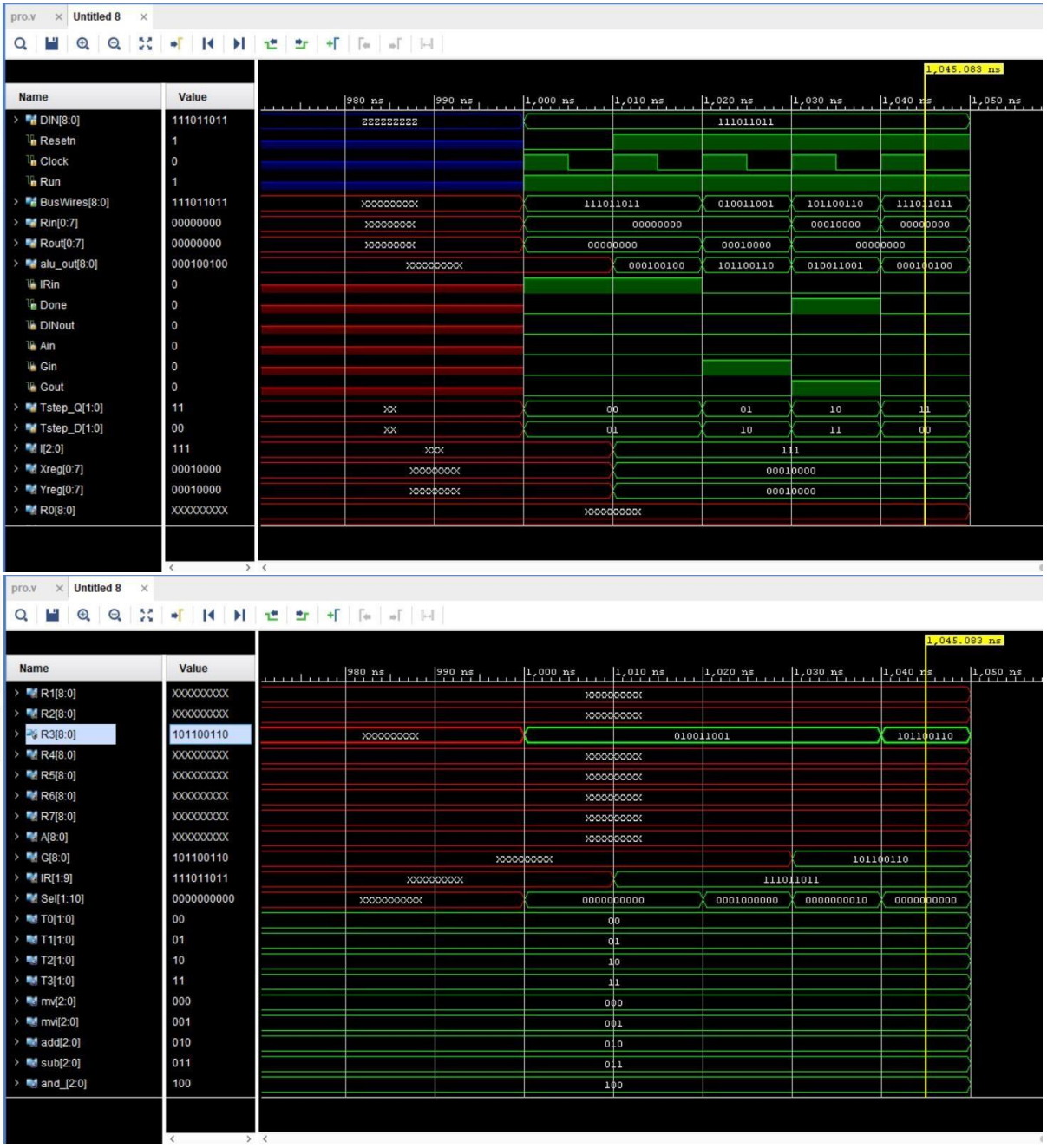
110 = XOR

011 = R3

100 = R4

Hence it should store the output of XOR operation of data in R3 and R4 in R3. The same can be verified in the output waveform.

10. IR = 111011011



IR = 111 011 011

111 = Negation

011 = R3

011 = R3

Hence it should store the complement of data in R3 in R3. The Ry register is ignore in case of negation.

4. Conclusion

- In this lab exercise, we got the understanding to design a simple processor which can perform eight operations viz, move, move immediate, addition, subtraction, AND, OR, XOR, and complement.
- We used an ALU to perform various arithmetic and logical operations.
- A control unit was designed to control various functionalities of ALU, multiplexers, registers to data transfer and operations.
- We understood the application of OPCODE. The IR register stored a 9 bit value. The three initial bits gave information about the operation to be performed, the middle three bits addressed the data of Rx and the last three bits addressed the data of Ry. Subsequently, ALU performed various operations based on the instructions.