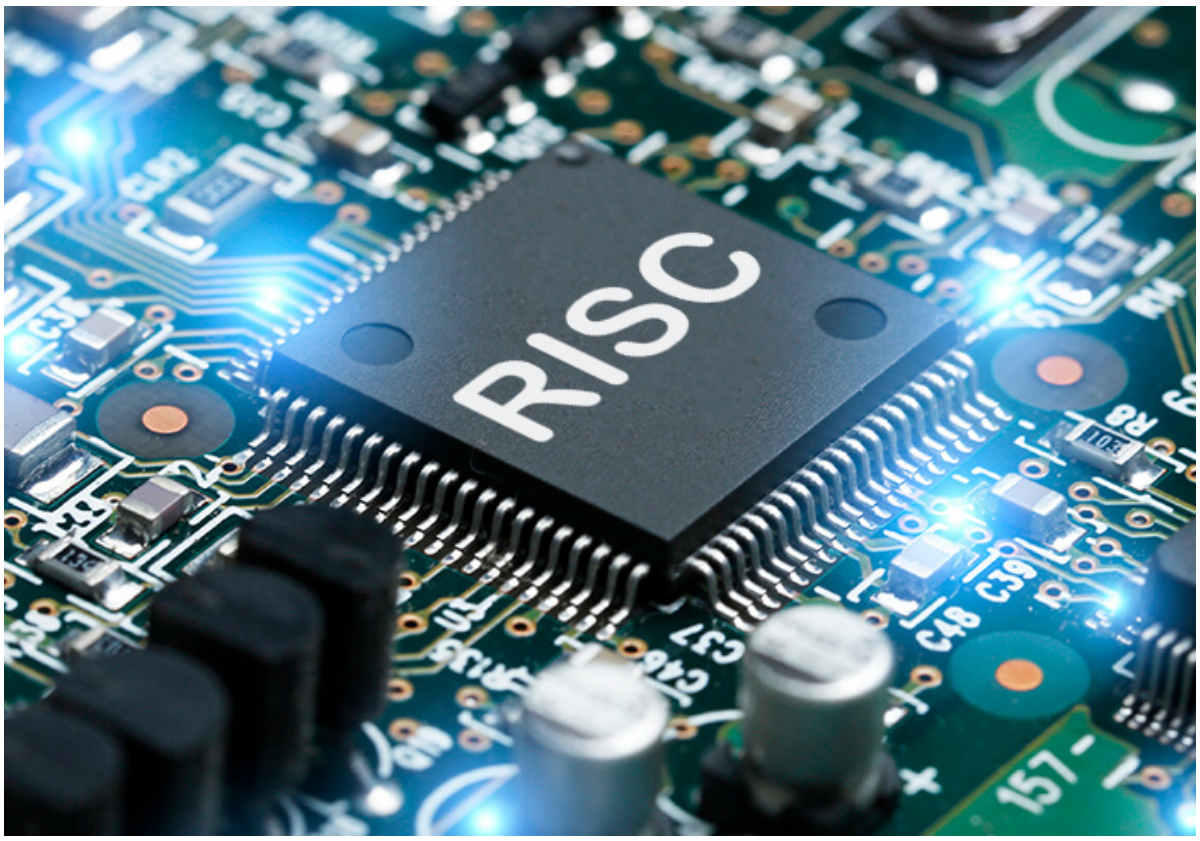
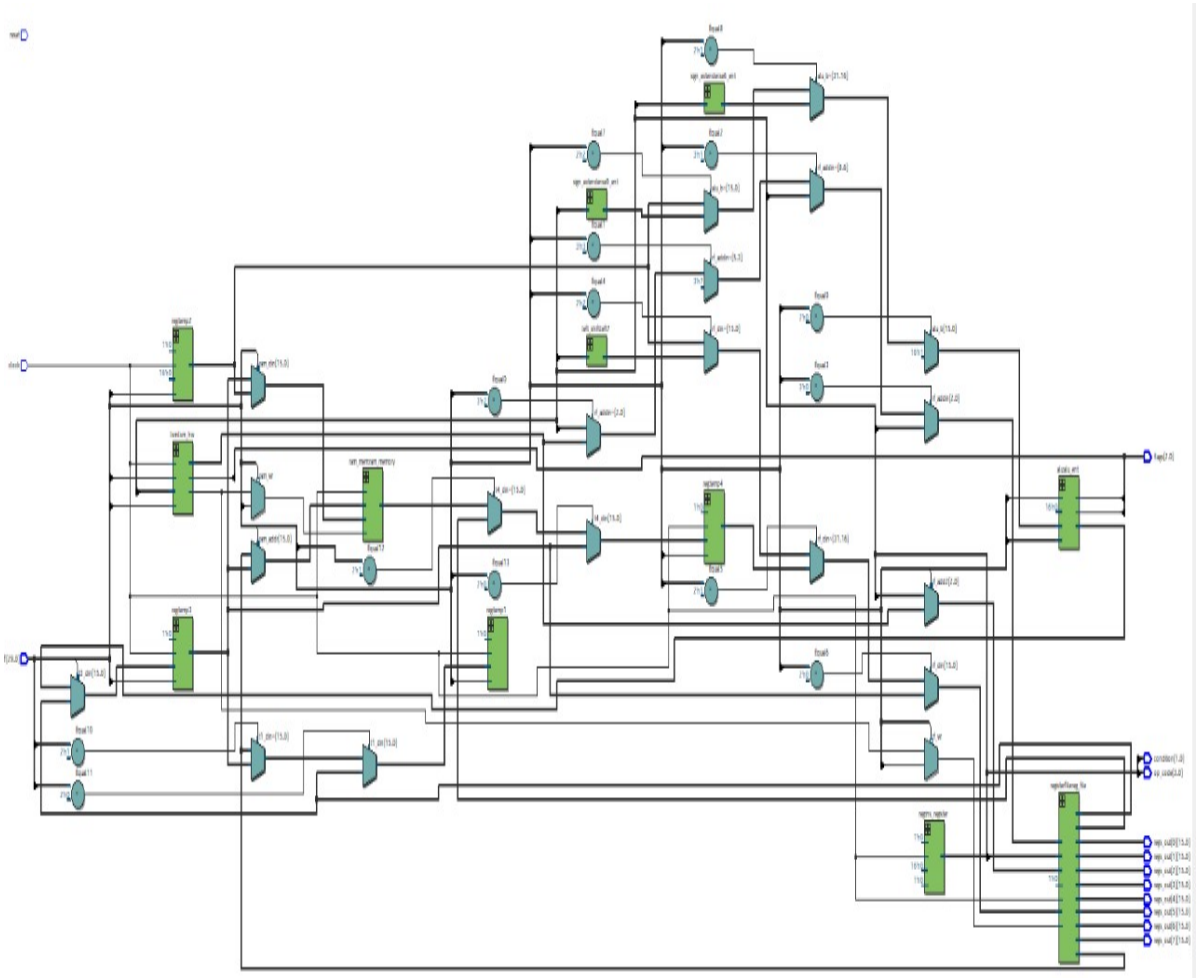
EE309 PROJECT-1 (COURSE PROJECT)

TEAM MEMBERS

AAYUSH M GOPAL- 200020004
NAVNEET- 200070048
TANMAY DOKANIA- 200070083
VANSH KAPOOR - 200100164



STATES



S_0: '111' → RF.A_1
RF.D_1 → T_1, Mem.A
Mem.D → IR

S_1: T_1 → Alu.A
t_1 → Alu.B
Alu.C → T_3

S_2: '111' → RF.A_3
T_3 → RF.D_3

S_3: IR_{11-9} → RF.A_1
RF.D_1 → T_1
IR_{8-6} → RF.A_2
RF.D_2 → T_2
T_3 → T_4

S_4: T_1 → Alu.A
T_2 → Alu.B
AluC → T_3

$S_{4'}$: t_4 → RF.D_3
'111' → RF.A_3

S_{43}: $T_4 \rightarrow T_3$

S_{44}: T_3 → RF.D_3
IQ_{11-9} → RF.A_3

S_5: T_3 → RF.D_3
$IR_{(5-3)}$ → RF.A_3

S_6: T_1 → Alu.A
IR_{5-0} → SE → Alu.B
AluC → T_3

S_8: T_3 → RF.D_3
$IR_{(8-6)}$ → RF.A_3

S_9: T_3 → Mem.A
T_2 → Mem.D

S_{11}: '111' → RF.A_3
$IR_{(8-6)}$ → RF.D_3
T_2

S_{13}: T_1 → Alu.A
$IR_{(8-0)}$ → SE → Alu.B
Alu.C → T_3

S_U: I_{0-8} → LS
LS → RF.D_3
IR_{9-11} → RF.A_3

S20 : $IR_{11-9} \longrightarrow RF.A_1$

$RF.D_1 \longrightarrow T_3$

$1 \longrightarrow HW\_reset$

$IR_{7-0} \longrightarrow HW\_read$

S21 : $T_3 \rightarrow Mem.A$, T_1

$Mem.D \rightarrow T_4$

S22 : $T_1 \rightarrow Alu.A$

$+1 \longrightarrow Alu.B$

$ALU.C \rightarrow T_3$

$T_4 \longrightarrow RF.D_3$ } If HW_write == 1

$HW\_add \rightarrow RF.A_3$

S23 : $HW\_add\ (out) \Rightarrow RF.A_2$

$RF.D_2 \longrightarrow T_4$

S24 : $T_4 \longrightarrow Mem.D$

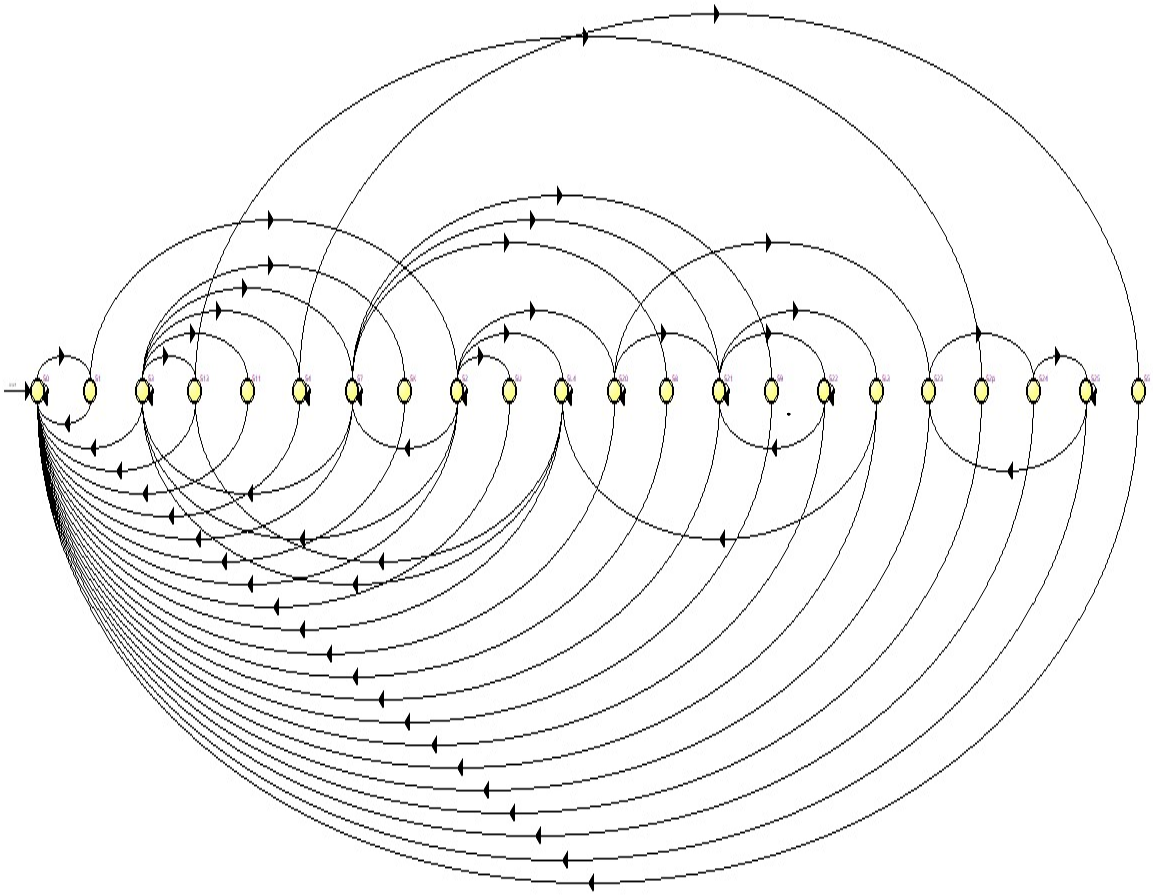
$T_3 \rightarrow Mem.A$, T_1

S25 : $T_1 \longrightarrow AluA$

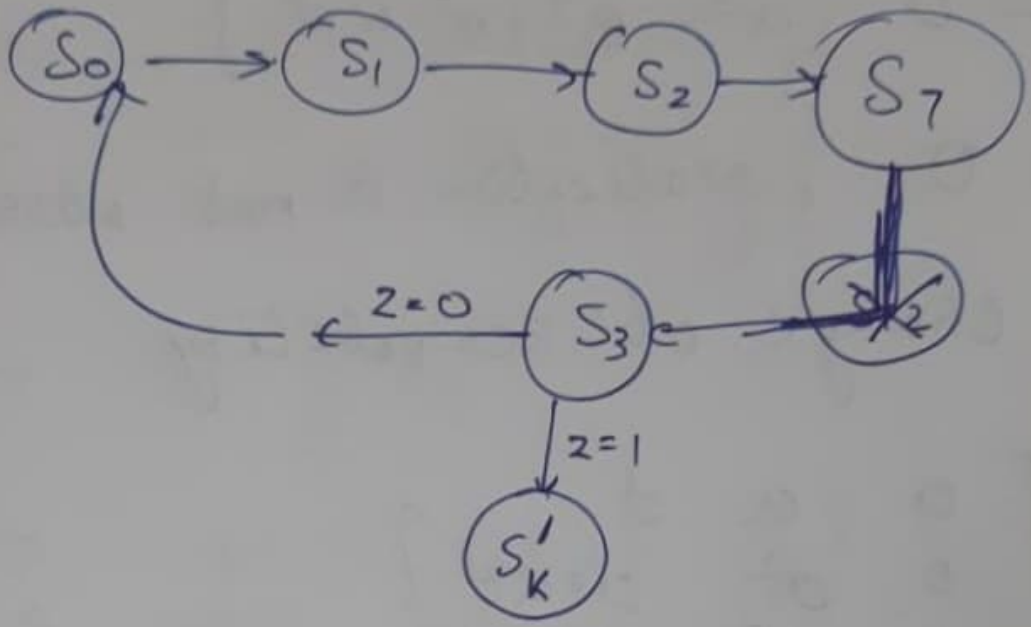
$+1 \longrightarrow AluB$

$Aluc \longrightarrow T_3$

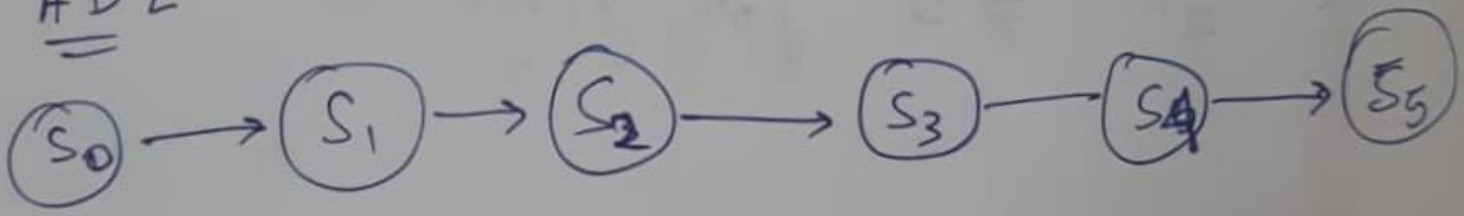
STATE TRANSITION DIAGRAM



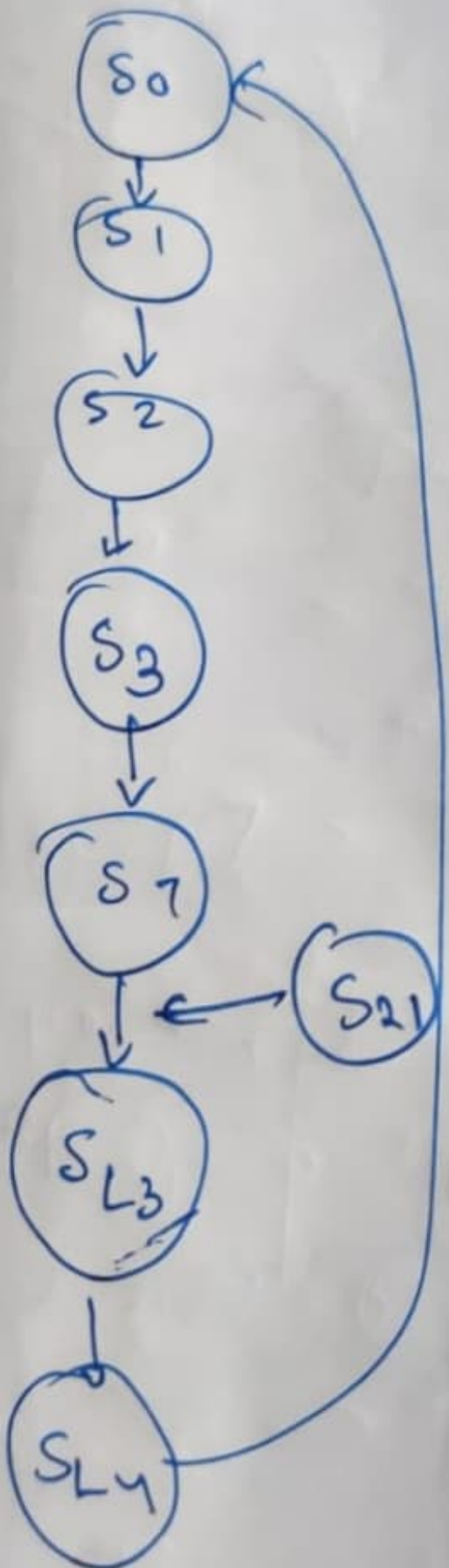
BEO



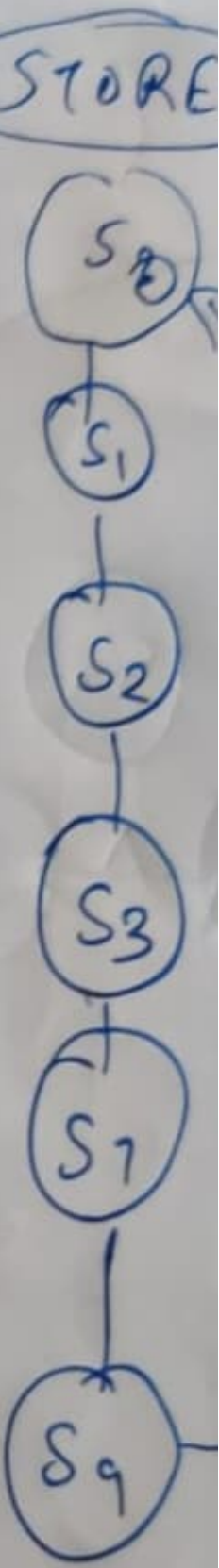
ADL



LOAD



STORE



ADC/ADZ/NDC/NDZ

S0
↓
S1
↓
(=0 or Z=0) S2 ——(C=1 or Z=1)——→
↓
S3
↓
S4
↓
S5 → S0

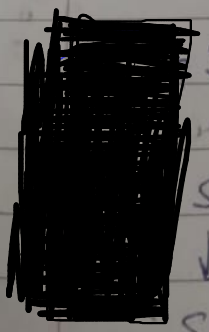
ADI

S0
↓
S1
↓
S2
↓
S3
↓
S7
↓
S8 →
S0

ADL

S0
↓
S1
↓
S2
↓
S3
↓
S4
↓
S11
↓
S8 →

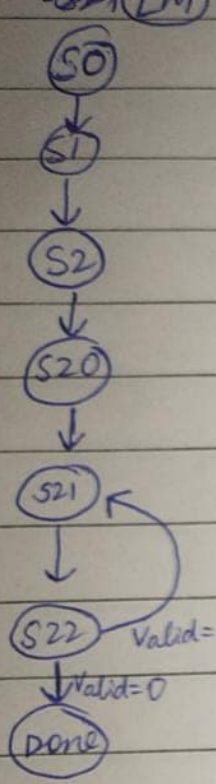
JAL



S0
↓
S1
↓
S2
↓
SL4
↓
S13
↓
S2' →

JRI

S0
↓
S1
↓
S2
↓
S3
↓
S13
↓
S2' →

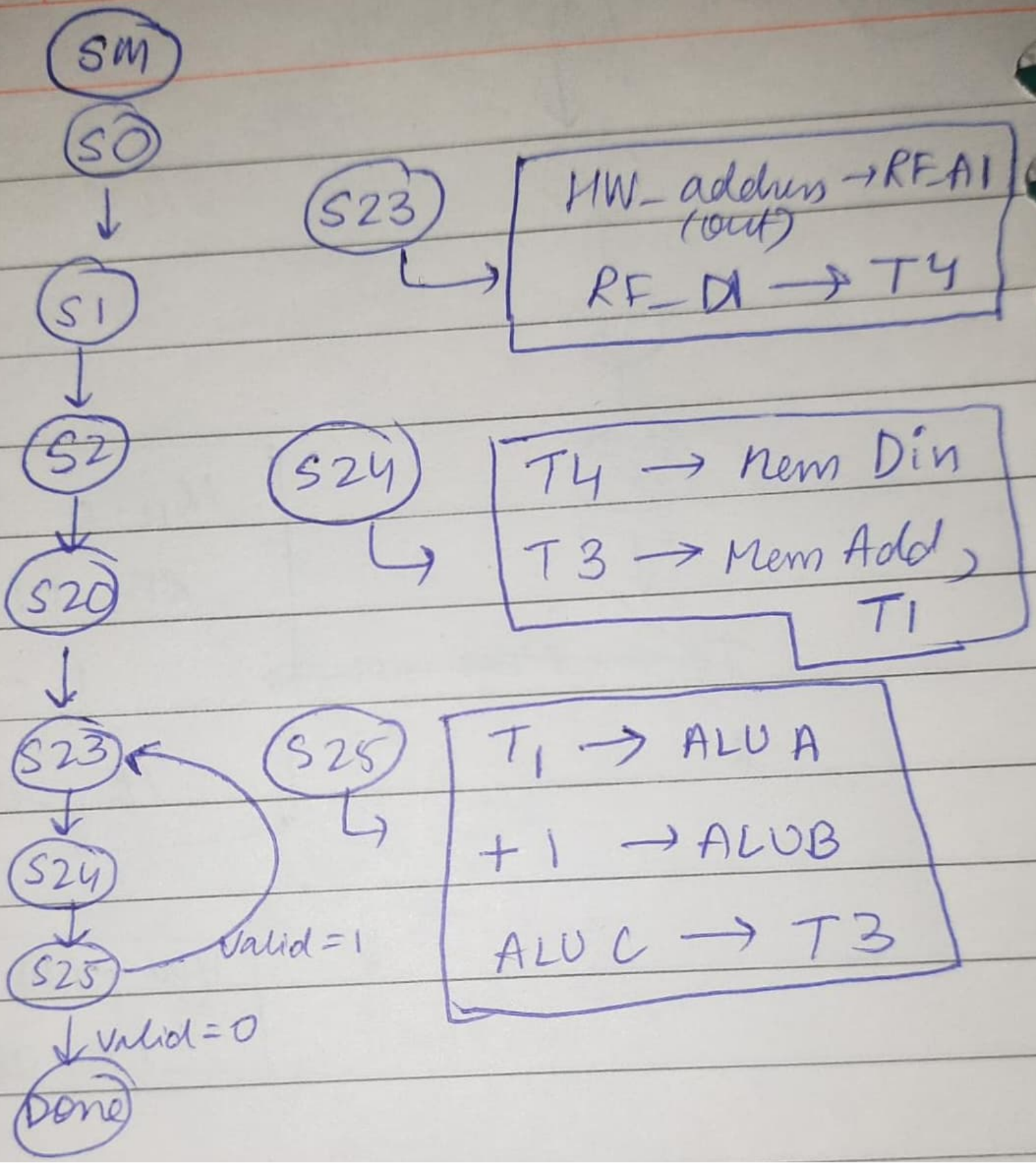


S20 →
IR_{11-9} → RF_A1
RF_D1 → T3
1 → HW_reset (control)
IR_{7-0} → HW_read

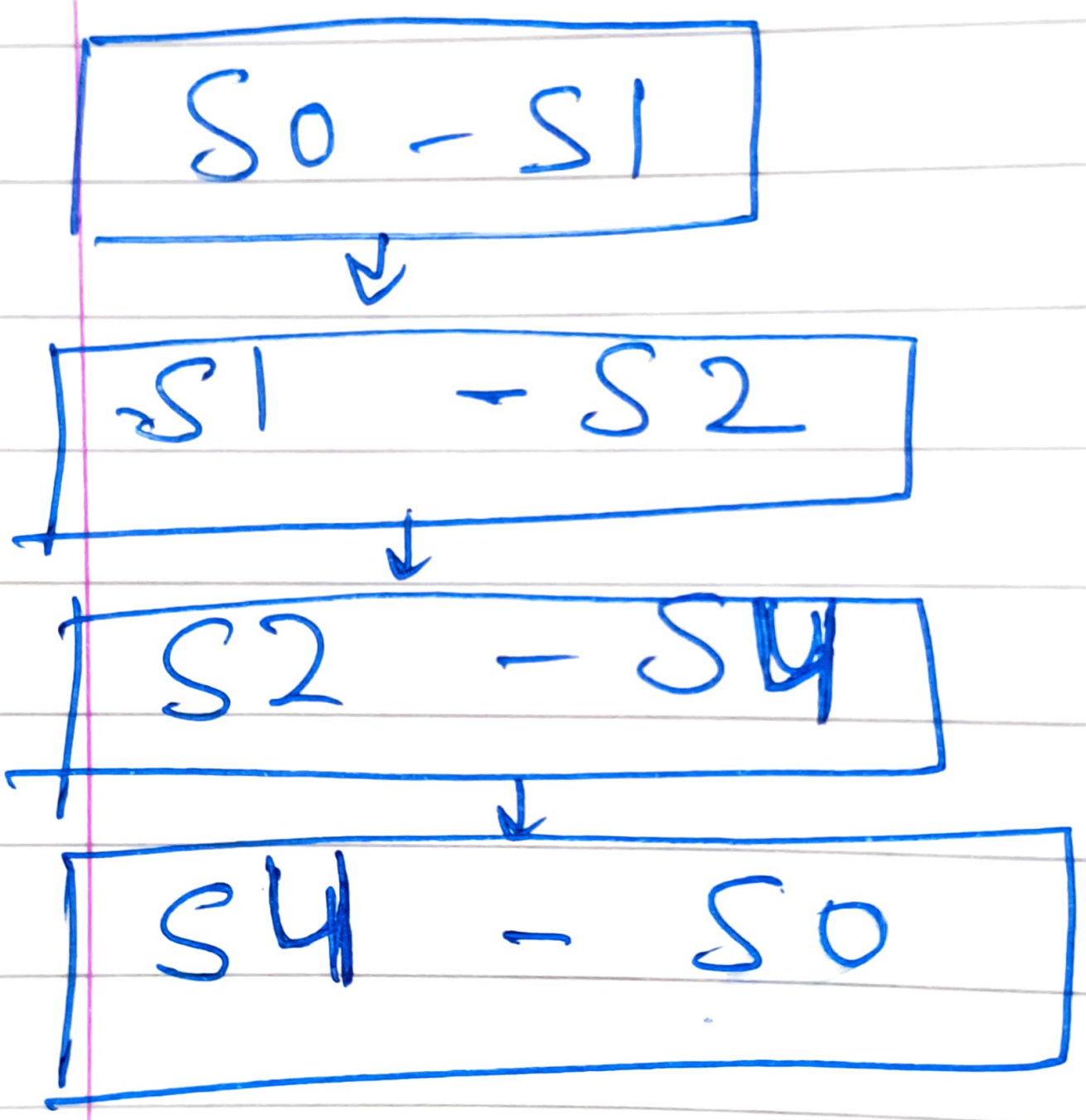
S21 →
T3 → MEM-ADDRESS, T1
Memout → Temp (T4)

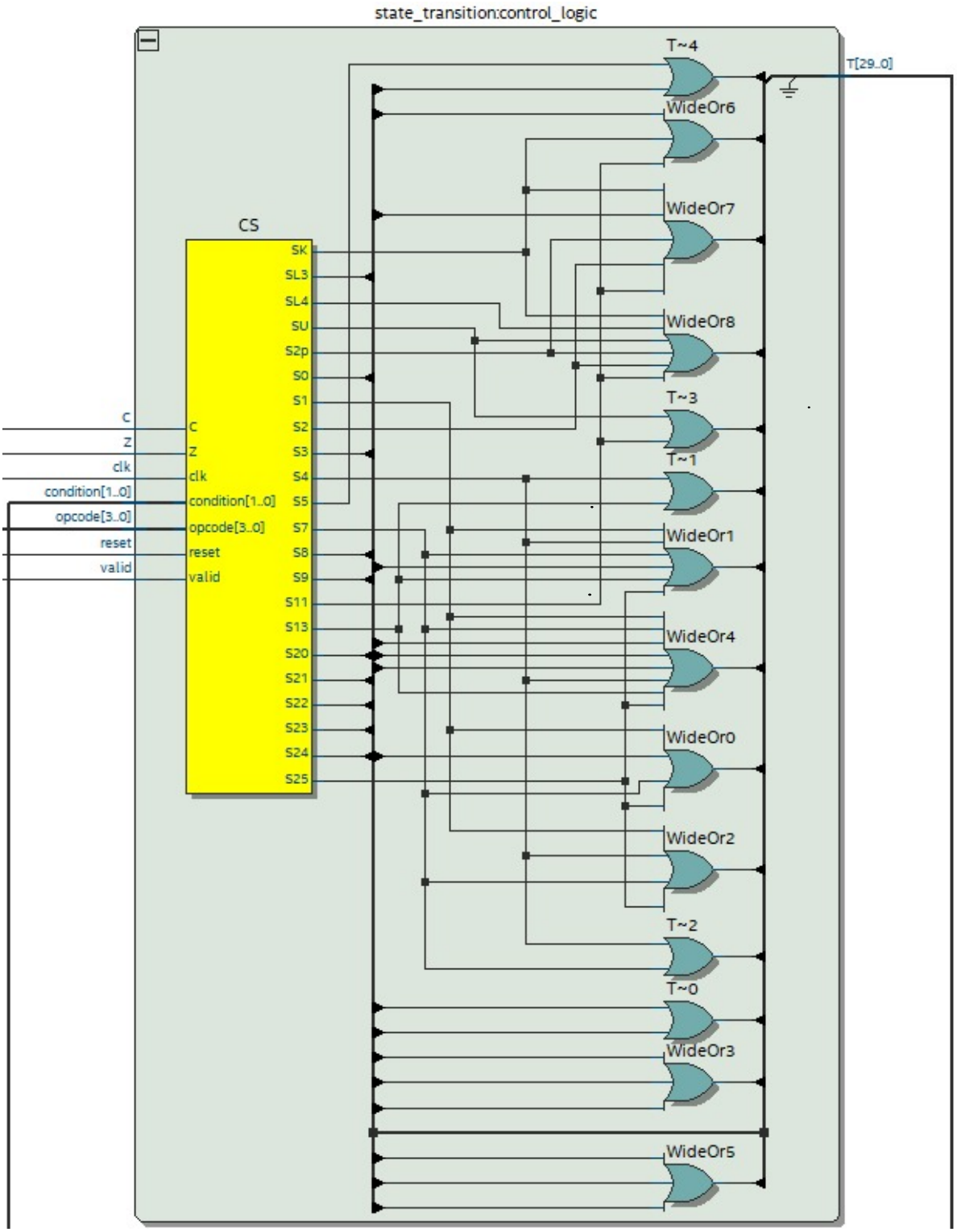
S22 →
T_1 → ALU-A
+1 → ALU-B
ALUC → T3
T4 → RF_D3
HW_address (out) → RF_A3
(HW_write == 1)

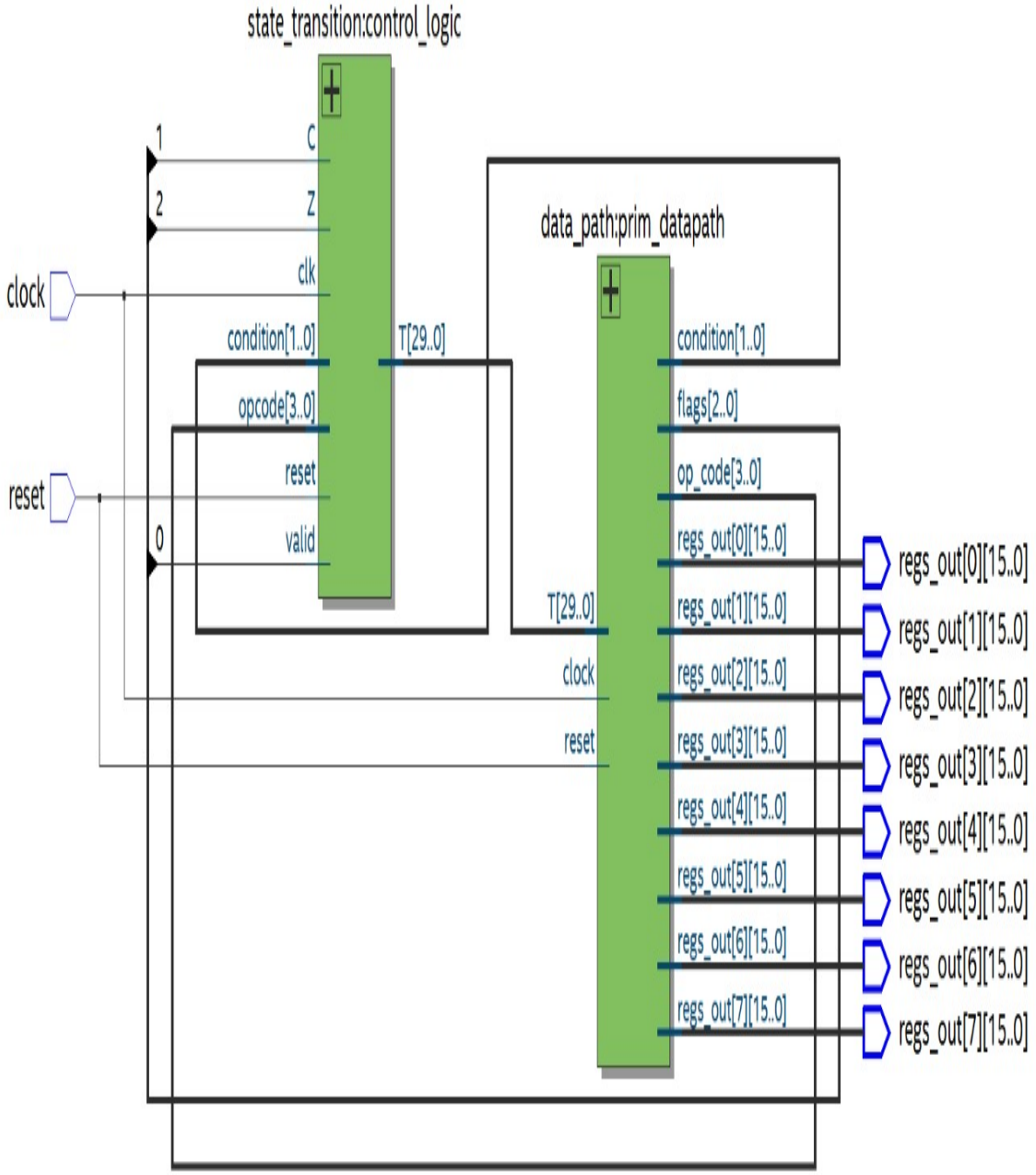
LM


SM
S0
S1
S2
S20
S23
S24
S25
Valid=1
Valid=0
Done
S23
HW_address (out) → RF_A1
RF_D1 → T4
S24
T4 → Mem Din
T3 → Mem Add
T1
S25
T1 → ALU A
+1 → ALUB
ALU C → T3

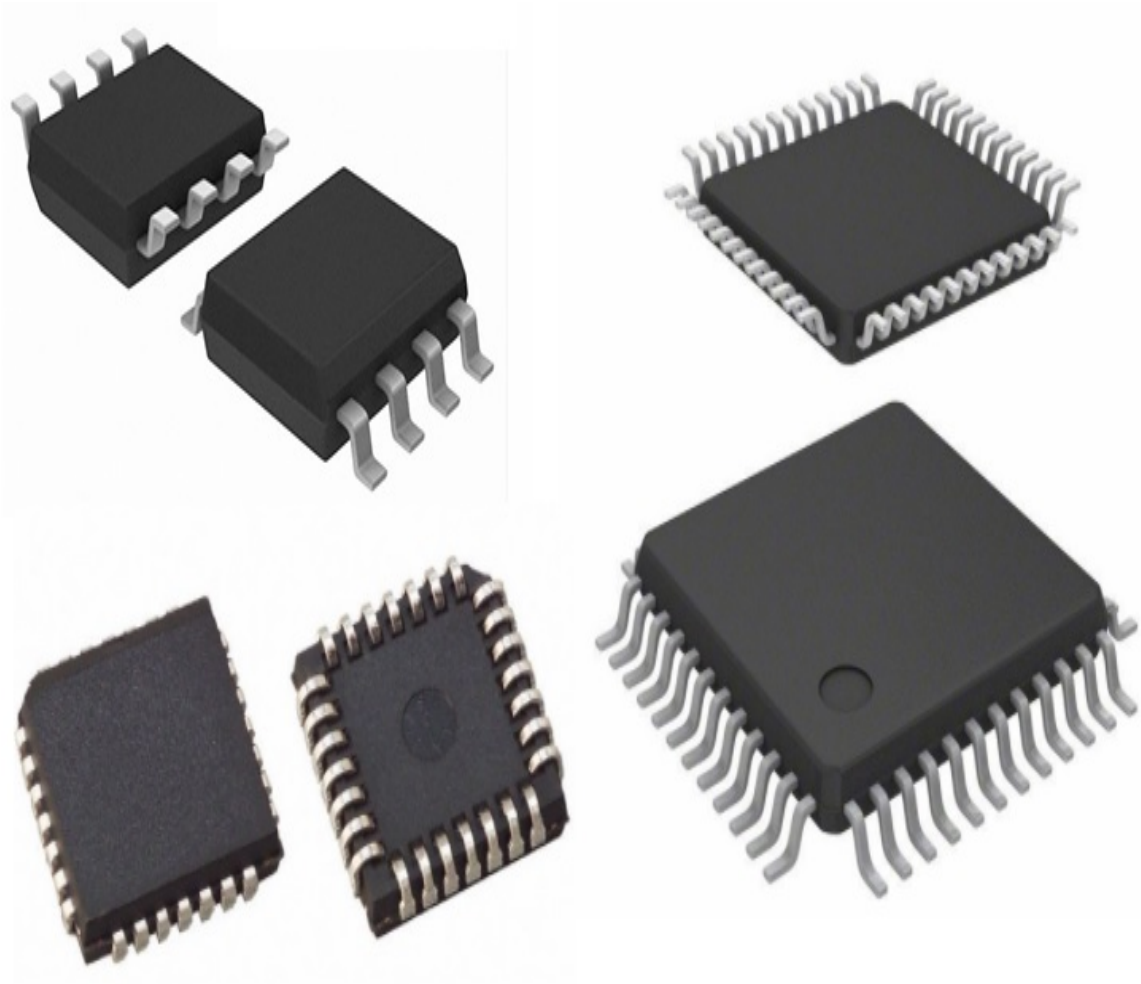
LHI




state_transition:control_logic
CS
C
Z
clk
condition[1..0]
opcode[3..0]
reset
valid
SK
SL3
SL4
SU
S2p
S0
S1
S2
S3
S4
S5
S7
S8
S9
S11
S13
S20
S21
S22
S23
S24
S25
T~4
T[29..0]
WideOr6
WideOr7
WideOr8
T~3
T~1
WideOr1
WideOr4
WideOr0
WideOr2
T~2
T~0
WideOr3
WideOr5


state_transition:control_logic
1
2
0
C
Z
clk
condition[1..0]
opcode[3..0]
reset
valid
T[29..0]
clock
reset
data_path:prim_datapath
condition[1..0]
flags[2..0]
op_code[3..0]
regs_out[0][15..0]
regs_out[1][15..0]
regs_out[2][15..0]
regs_out[3][15..0]
regs_out[4][15..0]
regs_out[5][15..0]
regs_out[6][15..0]
regs_out[7][15..0]
T[29..0]
clock
reset
regs_out[0][15..0]
regs_out[1][15..0]
regs_out[2][15..0]
regs_out[3][15..0]
regs_out[4][15..0]
regs_out[5][15..0]
regs_out[6][15..0]
regs_out[7][15..0]

COMPONENTS



REGISTER FILE

```
entity registerFile is
   generic(
         dataSize: integer := 16;
         numRegs: integer := 8
   );
   port(
         addr_out1, addr_out2, addr_in: in std_logic_vector(integer(ceil(log2(real(numRegs))))-1 downto 0);
         data_out1, data_out2, reg7_out : out std_logic_vector(dataSize-1 downto 0);
         data_in : in std_logic_vector(dataSize-1 downto 0);
         clock, wr_enable, clear: in std_logic;
         regbank_out : out regBank
   );
end entity;
```

ALU

```
entity alu is
       generic(
              operand_width : integer:= 16;
              sel_line: integer:= 2
       );
       port (
              opr1: in std_logic_vector(operand_width-1 downto 0);
              opr2: in std_logic_vector(operand_width-1 downto 0);
              dest: out std_logic_vector(operand_width-1 downto 0);
              sel: in std_logic_vector(sel_line-1 downto 0);
              enable: in std_logic;
              C, Z: out std_logic
       );
end alu;
```

RAM_MEMORY

```
  ENTITY ram_mem IS
         PORT
     (
      clock: IN   std_logic;
      ram_data_in:  IN   std_logic_vector (15 DOWNTO 0);
      ram_address:  IN   std_logic_vector(15 downto 0);
      ram_write_enable:    IN   std_logic;
      ram_data_out:     OUT  std_logic_vector (15 DOWNTO 0);
          reset : in std_logic
          );

  END ram_mem;
```

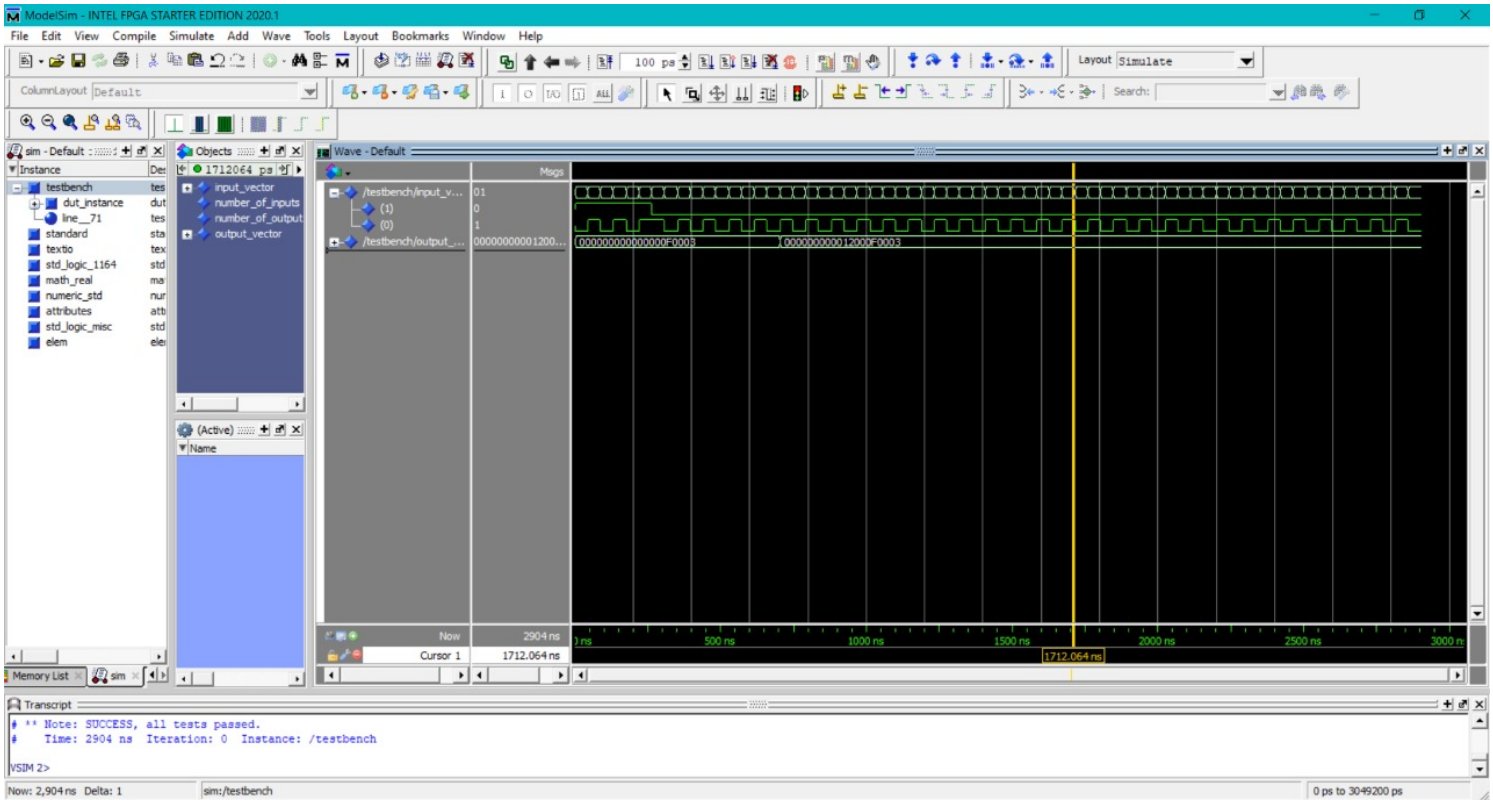
LEFT SHIFTER

```
entity left_shift is

	generic(
		input_length: integer := 9;	-- 9 bit input taken from immediate-9 field of the instruction
		output_length: integer := 16;	-- 16 bit output to be stored in the register
		shift_length: integer :=7);	-- 7 bit left shifter

	port(
		inp: in std_logic_vector(input_length-1 downto 0);
		outp: out std_logic_vector(output_length-1 downto 0)
	  );

end entity;
```

LOAD STORE MULTIPLE

```
  entity lsm is
    port (
       inc, reset, clock : in std_logic;
       insReg : in std_logic_vector(7 downto 0);
       valid, wr : out std_logic;
       addr : out std_logic_vector(2 downto 0)
    );
  end lsm;
```

SIMULATION



The least significant 16 bits have r0, then from 31 to 16 is r1 and 47 to 32 is r2. And we have instruction as ADD r2, r0, r1

The result of addition : 000F + 0003 = 0012

We have tested instructions by making different quartus projects for each. Screenshot of addition instruction running is attached and the corresponding quartus project is also attached. The program is coded in RAM and the regfile is initialized to said values to attain the given calculation.