DLX Processor Design

Written by Noam Yakar

This file presents a full design of a DLX processor. The program used for this matter is the XILINX ISE Design Suite. The design was tested on a XILINX FPGA.

During the planning stages, the Datapath and the Control were tested using testbenches. The simulation results are also present in this file.

Table of Contents

All Schematics and VHDL designs…… Page 2

Testbench and Simulation results…… Page 18

Schematics and VHDL designs:

MAC:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MAC is
    Port ( clk : in  STD_LOGIC;
           ACK_N : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           MW : in  STD_LOGIC;
           MR : in  STD_LOGIC;
           busy : out  STD_LOGIC;
           AS_N : out  STD_LOGIC;
           WR_N : out  STD_LOGIC;
           STATE : out  STD_LOGIC_VECTOR (1 downto 0);
           STOP_N : out  STD_LOGIC);
end MAC;

architecture Behavioral of MAC is
-- Setting 3 different states for the state machine
constant STATE0_WAIT4REQ : STD_LOGIC_VECTOR(1 downto 0) := "00";
constant STATE1_WAIT4ACK : STD_LOGIC_VECTOR(1 downto 0) := "01";
constant STATE2_NEXT : STD_LOGIC_VECTOR(1 downto 0) := "10";
--initial states is STATE0_WAIT4REQ
signal current_state : STD_LOGIC_VECTOR(1 downto 0) := STATE0_WAIT4REQ;
signal previous_state : STD_LOGIC_VECTOR(1 downto 0) := STATE0_WAIT4REQ;

begin
main: process(clk)
begin
   if((clk'event) and (clk='1')) then
      if(reset='1') then
         current_state <= STATE0_WAIT4REQ;
      else
         case current_state is
            --STATE0_WAIT4REQ
            when STATE0_WAIT4REQ =>
               if ((MR='1') or (MW='1')) then current_state <= STATE1_WAIT4ACK;
               else current_state <= STATE0_WAIT4REQ;
               end if;

            --STATE1_WAIT4ACK
            when STATE1_WAIT4ACK =>
               if(ACK_N='0') then current_state <= STATE2_NEXT;
               else current_state <= STATE1_WAIT4ACK;
               end if;

            --STATE2_NEXT
            when STATE2_NEXT =>
               current_state <= STATE0_WAIT4REQ;
```

```
           --else
           when others =>
              NULL;
        end case;
     end if;

     --update previous state
     previous_state <= current_state;
  end if;
end process main;

STATE <= current_state;
WR_N <= '1' when (MW='0') else '0';
AS_N <= '0' when (current_state = STATE1_WAIT4ACK) else '1';
busy <= '1' when ((ACK_N = '1') AND ((MR='1') or (MW='1'))) else '0';
STOP_N <= '1' WHEN (((current_state=STATE0_WAIT4REQ) or (current_state=STATE2_NEXT)) or (ACK_N='0') or (not(previous_state=STATE1_WAIT4ACK))) else '0';


end Behavioral;
```

DLX State Machine:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity DLX_state_machine_EX7 is
33      Port ( clk : in  STD_LOGIC;
34             reset : in  STD_LOGIC;
35             STEP_EN : in  STD_LOGIC;
36             busy : in  STD_LOGIC;
37             ACK_N : in  STD_LOGIC;
38             opcode : in  STD_LOGIC_VECTOR (5 downto 0);
39             FUNC_R : in  STD_LOGIC_VECTOR (5 downto 0);
40             AEQZ : in  STD_LOGIC;
41             IN_INIT : out  STD_LOGIC;
42             PC_CE : out  STD_LOGIC;
43             A_CE : out  STD_LOGIC;
44             B_CE : out  STD_LOGIC;
45             C_CE : out  STD_LOGIC;
46             MR : out  STD_LOGIC;
47             MW : out  STD_LOGIC;
48             IR_CE : out  STD_LOGIC;
49             S1_SEL : out  STD_LOGIC_VECTOR (1 downto 0);
50             S2_SEL : out  STD_LOGIC_VECTOR (1 downto 0);
51             ADD : out  STD_LOGIC;
52             TEST : out  STD_LOGIC;
53             I_TYPE : out  STD_LOGIC;
54             DINT_SEL : out  STD_LOGIC;
55             SHIFT_SIG : out  STD_LOGIC;
56             MAR_CE : out  STD_LOGIC;
57             MDR_CE : out  STD_LOGIC;
58             MDR_SEL : out  STD_LOGIC;
59             A_SEL : out  STD_LOGIC;
60             GPR_WE : out  STD_LOGIC;
61             J_LINK : out  STD_LOGIC;
62             state : out  STD_LOGIC_VECTOR (4 downto 0));
63  end DLX_state_machine_EX7;
64
65  architecture Behavioral of DLX_state_machine_EX7 is
```

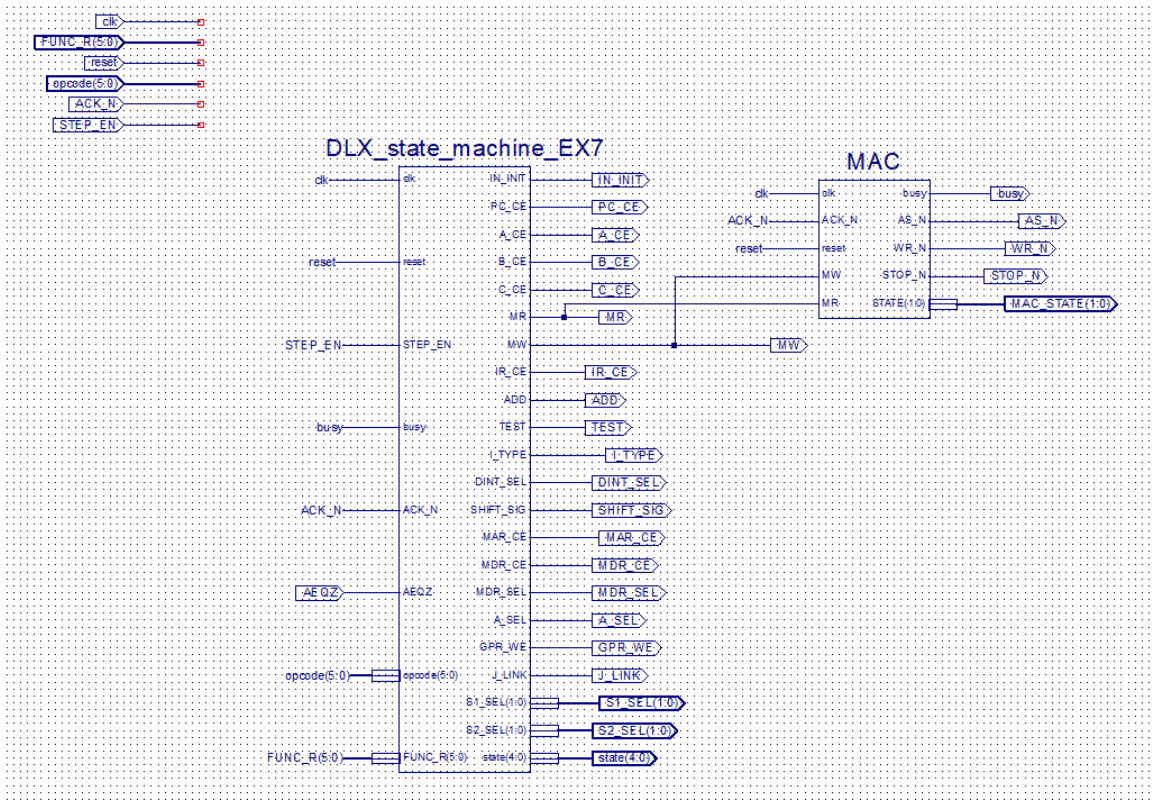
```
67  --states
68  constant INIT: STD_LOGIC_VECTOR (4 downto 0) := "00000";
69  constant FETCH: STD_LOGIC_VECTOR (4 downto 0) := "00001";
70  constant DECODE: STD_LOGIC_VECTOR (4 downto 0) := "00010";
71  constant HALT: STD_LOGIC_VECTOR (4 downto 0) := "00011";
72  constant ALU: STD_LOGIC_VECTOR (4 downto 0) := "00100";
73  constant SHIFT: STD_LOGIC_VECTOR (4 downto 0) := "00101";
74  constant ALUI: STD_LOGIC_VECTOR (4 downto 0) := "00110";
75  constant TESTI: STD_LOGIC_VECTOR (4 downto 0) := "00111";
76  constant ADDRESSCMP: STD_LOGIC_VECTOR (4 downto 0) := "01000";
77  constant JR: STD_LOGIC_VECTOR (4 downto 0) := "01001";
78  constant SAVEPC: STD_LOGIC_VECTOR (4 downto 0) := "01010";
79  constant BRANCH: STD_LOGIC_VECTOR (4 downto 0) := "01011";
80  constant WBR: STD_LOGIC_VECTOR (4 downto 0) := "01100";
81  constant WBI: STD_LOGIC_VECTOR (4 downto 0) := "01101";
82  constant LOAD: STD_LOGIC_VECTOR (4 downto 0) := "01110";
83  constant COPYGPR2MDR: STD_LOGIC_VECTOR (4 downto 0) := "01111";
84  constant JALR: STD_LOGIC_VECTOR (4 downto 0) := "10000";
85  constant BTAKEN: STD_LOGIC_VECTOR (4 downto 0) := "10001";
86  constant COPYMDR2C: STD_LOGIC_VECTOR (4 downto 0) := "10010";
87  constant STORE: STD_LOGIC_VECTOR (4 downto 0) := "10011";
88
89  --opcodes
90  constant opcode_NOP: STD_LOGIC_VECTOR (5 downto 0) := "110000";
91  constant opcode_RTYPE: STD_LOGIC_VECTOR (5 downto 0) := "000000";
92  constant opcode_ALUI_5to3: STD_LOGIC_VECTOR (2 downto 0) := "001";
93  constant opcode_LOAD: STD_LOGIC_VECTOR (5 downto 0) := "100011";
94  constant opcode_STORE: STD_LOGIC_VECTOR (5 downto 0) := "101011";
95  constant opcode_TESTI_5to3: STD_LOGIC_VECTOR (2 downto 0) := "011";
96  constant opcode_JUMP_5to3: STD_LOGIC_VECTOR (2 downto 0) := "010";
97  constant opcode_JR_2to0: STD_LOGIC_VECTOR (2 downto 0) := "110";
98  constant opcode_BRANCH_5to2: STD_LOGIC_VECTOR (3 downto 0) := "0001";
99  constant opcode_HALT: STD_LOGIC_VECTOR (5 downto 0) := "111111";
100
101 signal current_state: STD_LOGIC_VECTOR (4 downto 0) := INIT;
102 signal bt: STD_LOGIC;
103 begin
104
105 main: process(clk)
106 begin
107 if ((CLK'event) and (CLK='1')) then
108    if (reset='1') then
109    current_state<=INIT;
110    else
111    case current_state is
112       when INIT =>
113          if(STEP_EN='1') then current_state<=FETCH;
114          else current_state<=INIT;
115          end if;
116       when FETCH =>
117          if(ACK_N='0') then current_state<=DECODE;
118          else current_state <= FETCH;
119          end if;
120       when DECODE =>
```

```
121          if(opcode=opcode_NOP) then
122             if(STEP_EN='1') then current_state<=FETCH;
123             else current_state<=INIT;
124             end if;
125          elsif(opcode=opcode_RTYPE) then
126             if(FUNC_R(5)='1') then current_state<=ALU;
127             else current_state<=SHIFT;
128             end if;
129          elsif(opcode(5 downto 3)=opcode_ALUI_5to3) then current_state<=ALUI;
130          elsif(opcode(5 downto 3)=opcode_TESTI_5to3) then current_state<=TESTI;
131          elsif((opcode=opcode_LOAD) or (opcode=opcode_STORE)) then current_state<=ADDRESSCMP;
132          elsif(opcode(5 downto 3)=opcode_JUMP_5to3) then
133             if(opcode(2 downto 0)=opcode_JR_2to0) then current_state<= JR;
134             else current_state<=SAVEPC;
135             end if;
136          elsif(opcode(5 downto 2)=opcode_BRANCH_5to2) then current_state<=BRANCH;
137          elsif(opcode=opcode_HALT) then current_state<=HALT;
138          else if (STEP_EN='1') then current_state<=FETCH;
139             else current_state<=INIT;
140             end if;
141          end if;
142       when ALUI =>
143          current_state<=WBI;
144       when TESTI =>
145          current_state<=WBI;
146       when ADDRESSCMP =>
147          if(opcode=opcode_LOAD) then current_state<=LOAD;
148          else current_state<=COPYGPR2MDR;
149          end if;
150       when ALU =>
151          current_state<=WBR;
152       when SHIFT =>
153          current_state<=WBR;
154       when LOAD =>
155          if(busy='0') then current_state<=COPYMDR2C;
156          else current_state<=LOAD;
157          end if;
158       when COPYMDR2C =>
159          current_state<=WBI;
160       when COPYGPR2MDR =>
161          current_state<=STORE;
162       when STORE =>
163          if(busy='0') then
164             current_state<=INIT;
165          else current_state<=STORE;
166          end if;
167       when JR =>
168          if(STEP_EN='1') then current_state<= FETCH;
169          else current_state<=INIT;
170          end if;
171       when JALR =>
172          if(STEP_EN='1') then current_state<=FETCH;
173          else current_state<=INIT;
174          end if;
```

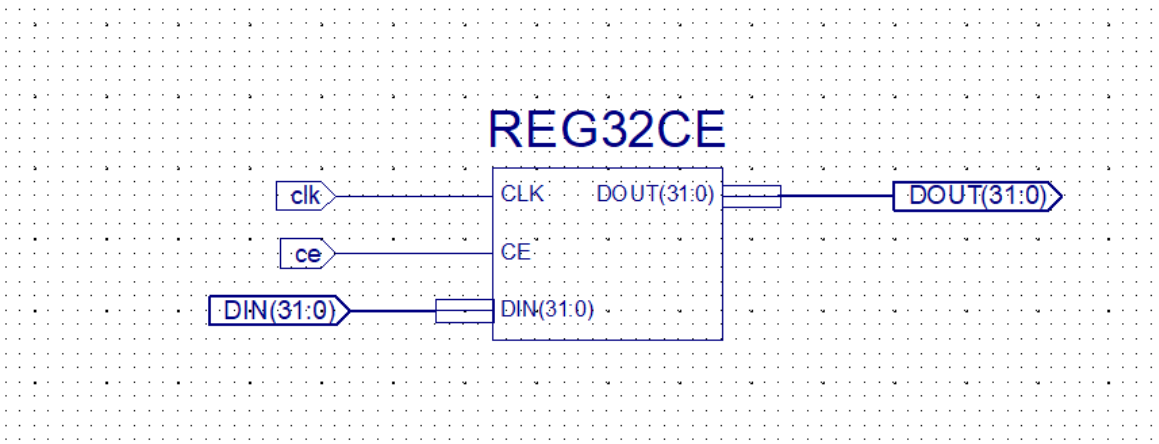
```
175       when BTAKEN =>
176          if(STEP_EN='1') then current_state<=FETCH;
177          else current_state<=INIT;
178          end if;
179       when WBI =>
180          if(STEP_EN='1') then current_state<=FETCH;
181          else current_state<=INIT;
182          end if;
183       when WBR =>
184          if(STEP_EN='1') then current_state<=FETCH;
185          else current_state<=INIT;
186          end if;
187       when SAVEPC =>
188          current_state<=JALR;
189       when BRANCH =>
190          if(bt='1') then current_state<=BTAKEN;
191          else
192             if(STEP_EN='1') then current_state<=FETCH;
193             else current_state<=INIT;
194             end if;
195          end if;
196       when HALT =>
197          current_state<=HALT;
198       when others => NULL;
199    end case;
200    end if;
201 end if;
202 end process main;
```

```
203
204 --control signals
205 bt <= '1' when (AEQZ xor opcode(0))='1' else '0';
206 IN_INIT <= '1' when ((current_state=INIT) or (current_state=HALT)) else '0';
207 PC_CE <= '1' when ((current_state=DECODE) or (current_state=JR) or (current_state=JALR) or (current_state=BTAKEN)) else '0';
208 A_CE <= '1' when (current_state=DECODE) else '0';
209 B_CE <= '1' when (current_state=DECODE) else '0';
210 C_CE <= '1' when ((current_state=ALU) or (current_state=TESTI) or (current_state=ALUI) or (current_state=SHIFT) or (current_state=COPYMDR2C) or (current_state=SAVEPC)) else '0';
211 MR <= '1' when ((current_state=FETCH) or (current_state=LOAD)) else '0';
212 MW <= '1' when (current_state=STORE) else '0';
213 IR_CE <= '1' when ((current_state=FETCH) and (ACK_N='0')) else '0';
214 S1_SEL(0) <= '1' when ((current_state=ALU) or (current_state=TESTI) or (current_state=ALUI) or (current_state=SHIFT) or (current_state=ADDRESSCMP) or (current_state=COPYMDR2C) or (current_state=JR) or (current_state=JALR)) else '0';
215 S1_SEL(1) <= '1' when ((current_state=COPYMDR2C) or (current_state=COPYGPR2MDR)) else '0';
216 S2_SEL(0) <= '1' when ((current_state=DECODE) or (current_state=TESTI) or (current_state=ALUI) or (current_state=ADDRESSCMP) or (current_state=BTAKEN)) else '0';
217 S2_SEL(1) <= '1' when ((current_state=DECODE) or (current_state=COPYMDR2C) or (current_state=COPYGPR2MDR) or (current_state=JR) or (current_state=SAVEPC) or (current_state=JALR)) else '0';
218 ADD <= '1' when ((current_state=DECODE) or (current_state=ALUI) or (current_state=ADDRESSCMP) or (current_state=BTAKEN) or (current_state=JR) or (current_state=SAVEPC) or (current_state=JALR)) else '0';
219 TEST <= '1' when (current_state=TESTI) else '0';
220 I_TYPE <= '1' when ((current_state=TESTI) or (current_state=ALUI) or (current_state=WBI)) else '0';
221 DINT_SEL <= '1' when ((current_state=SHIFT) or (current_state=COPYMDR2C) or (current_state=COPYGPR2MDR)) else '0';
222 SHIFT_SIG <= '1' when (current_state=SHIFT) else '0';
223 MAR_CE <= '1' when (current_state=ADDRESSCMP) else '0';
224 MDR_CE <= '1' when ((current_state=LOAD) or (current_state=COPYGPR2MDR)) else '0';
225 MDR_SEL <= '1' when (current_state=LOAD) else '0';
226 A_SEL <= '1' when ((current_state=STORE) or (current_state=LOAD)) else '0';
227 GPR_WE <= '1' when ((current_state=WBI) or (current_state=WBR) or (current_state=JALR)) else '0';
228 J_LINK <= '1' when (current_state=JALR) else '0';
229 state <= current_state;
230
231
232 end Behavioral;
233
```

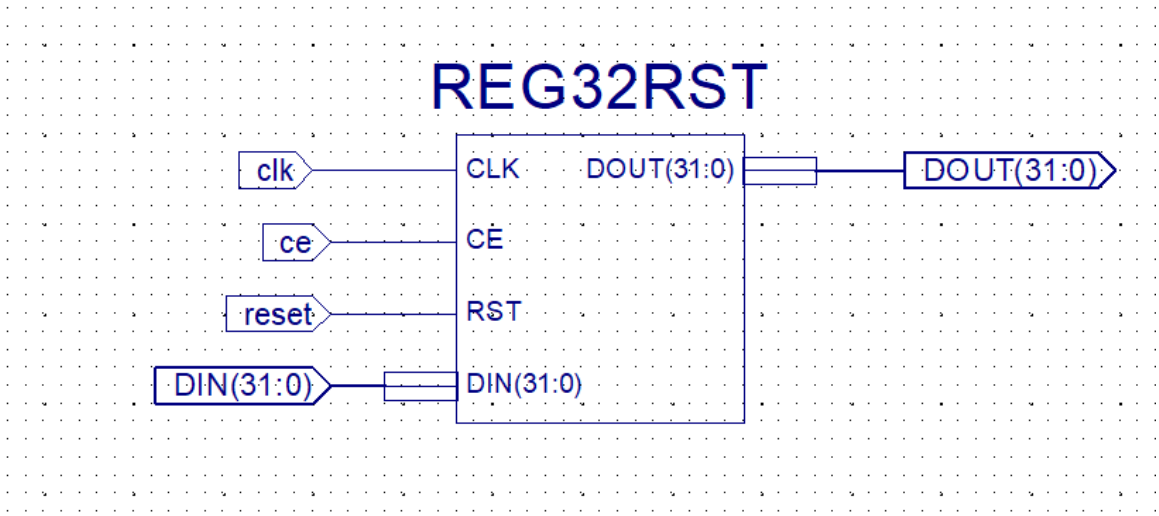
DLX Control:



REG A, B, C:



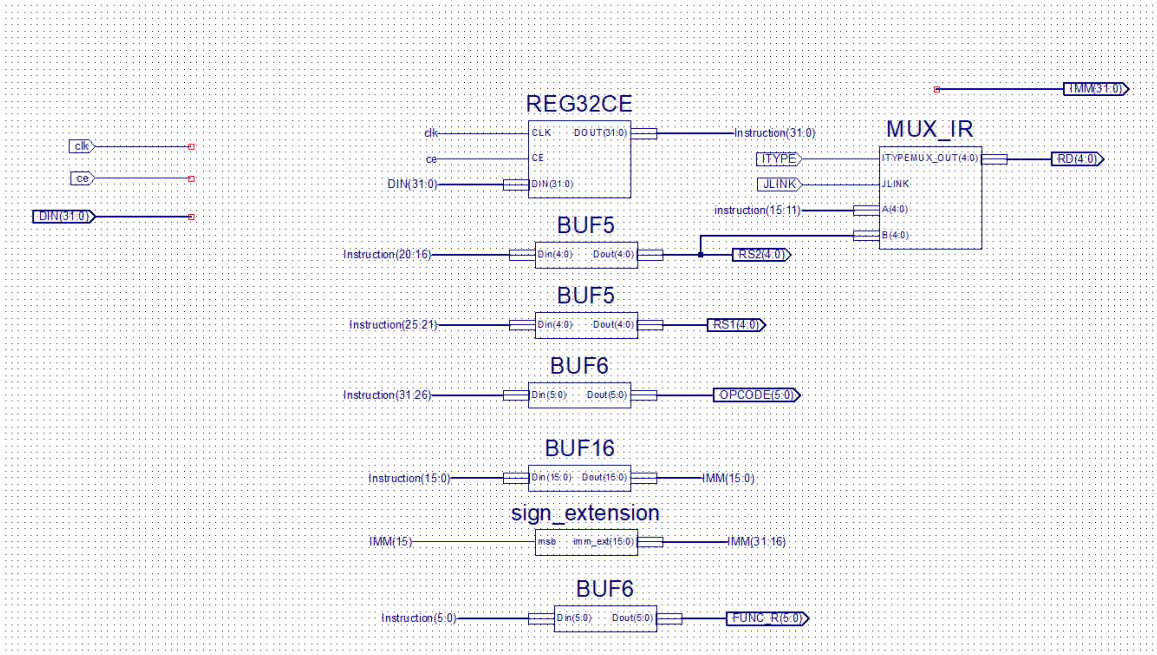
REG PC:



MUX IR:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity MUX_IR is
33      Port ( A : in  STD_LOGIC_VECTOR (4 downto 0);
34             B : in  STD_LOGIC_VECTOR (4 downto 0);
35             ITYPE : in  STD_LOGIC;
36             JLINK : in  STD_LOGIC;
37             MUX_OUT : out  STD_LOGIC_VECTOR (4 downto 0));
38  end MUX_IR;
39
40  architecture Behavioral of MUX_IR is
41
42  begin
43
44  MUX_OUT <= B when((ITYPE='1') and (JLINK='0')) else A when ((ITYPE='0') and (JLINK='0')) else "11111";
45
46  end Behavioral;
47
```

IR Environment:



MUX32BIT:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  --  Uncomment the following lines to use the declarations that are
7  --  provided for instantiating Xilinx primitive components.
8  --library UNISIM;
9  --use UNISIM.VComponents.all;
10
11 entity MUX32bit is
12     Port ( A : in std_logic_vector(31 downto 0);
13            B : in std_logic_vector(31 downto 0);
14            sel : in std_logic;
15            O : out std_logic_vector(31 downto 0));
16 end MUX32bit;
17
18 architecture Behavioral of MUX32bit is
19
20 begin
21
22 O <= A when (sel = '0') else B;
23
24 end Behavioral;
25
```

MUX4_32bit:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  --  Uncomment the following lines to use the declarations that are
7  --  provided for instantiating Xilinx primitive components.
8  --library UNISIM;
9  --use UNISIM.VComponents.all;
10
11 entity MUX4_32bit is
12     Port ( A0 : in std_logic_vector(31 downto 0);
13            A1 : in std_logic_vector(31 downto 0);
14            A2 : in std_logic_vector(31 downto 0);
15            A3 : in std_logic_vector(31 downto 0);
16            sel : in std_logic_vector(1 downto 0);
17            O : out std_logic_vector(31 downto 0));
18 end MUX4_32bit;
19
20 architecture Behavioral of MUX4_32bit is
21
22 begin
23
24 O <=  A0 when (sel = "00") else
25       A1 when (sel = "01") else
26       A2 when (sel = "10") else
27       A3;
28
29 end Behavioral;
```

Sign Extension:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity sign_extension is
33      Port ( msb : in  STD_LOGIC;
34             imm_ext : out  STD_LOGIC_VECTOR (15 downto 0));
35  end sign_extension;
36
37  architecture Behavioral of sign_extension is
38
39  begin
40
41  imm_ext <= (X"FFFF") when (msb='1') else (X"0000");
42
43  end Behavioral;
44
```

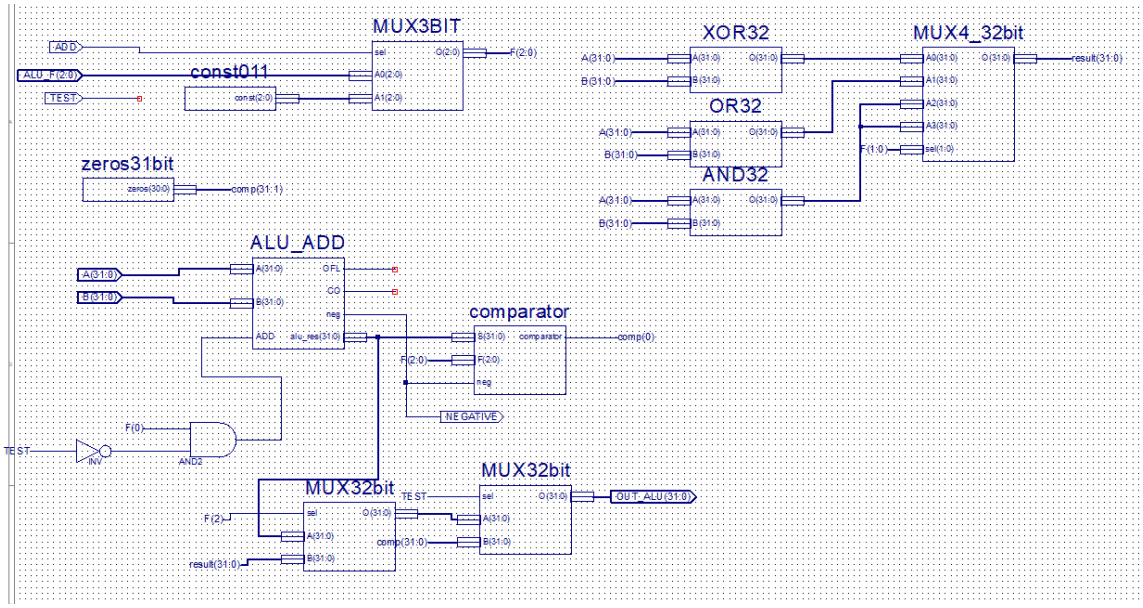
32 bit zero module:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity zeros32bit is
33      Port ( zeros : out  STD_LOGIC_VECTOR (31 downto 0));
34  end zeros32bit;
35
36  architecture Behavioral of zeros32bit is
37
38  begin
39
40  zeros<=(X"00000000");
41
42  end Behavioral;
43
44
```

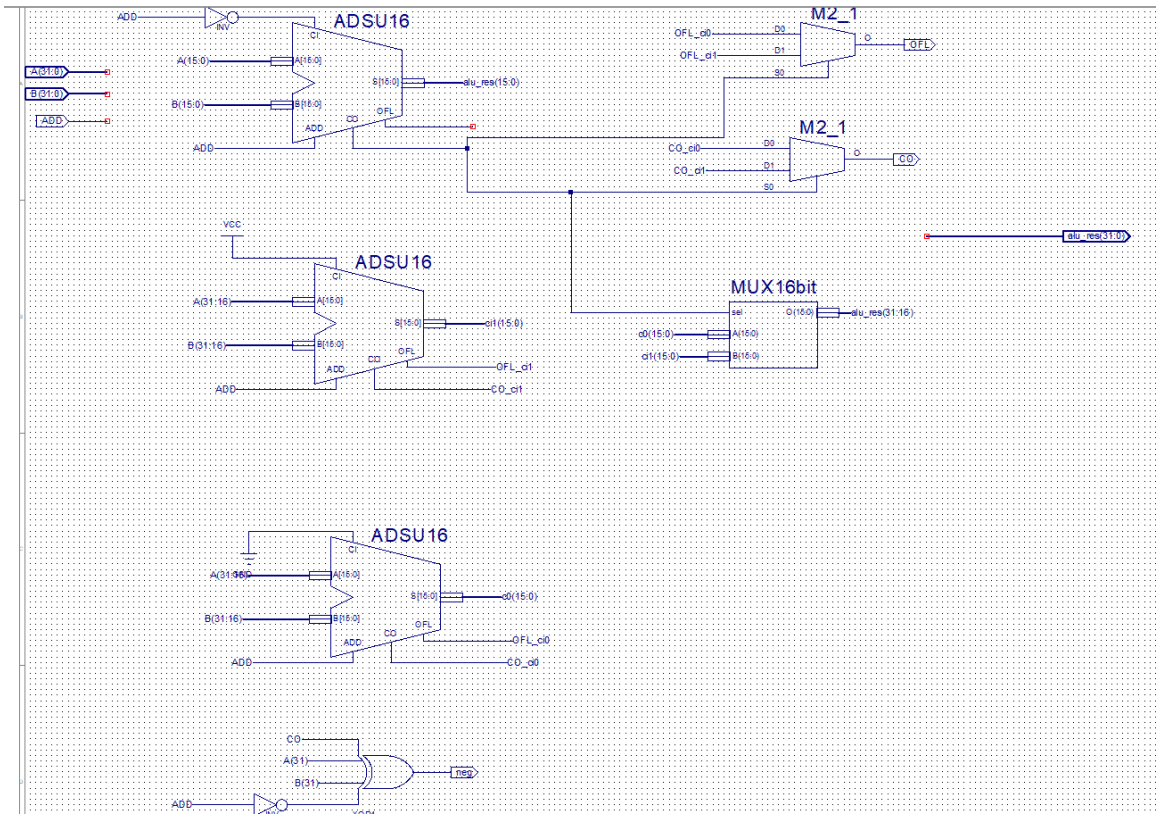
31 bit zero module:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity zeros31bit is
33      Port ( zeros : out  STD_LOGIC_VECTOR (30 downto 0));
34  end zeros31bit;
35
36  architecture Behavioral of zeros31bit is
37
38  begin
39
40  zeros<="0000000000000000000000000000000";
41
42  end Behavioral;
43
44
```

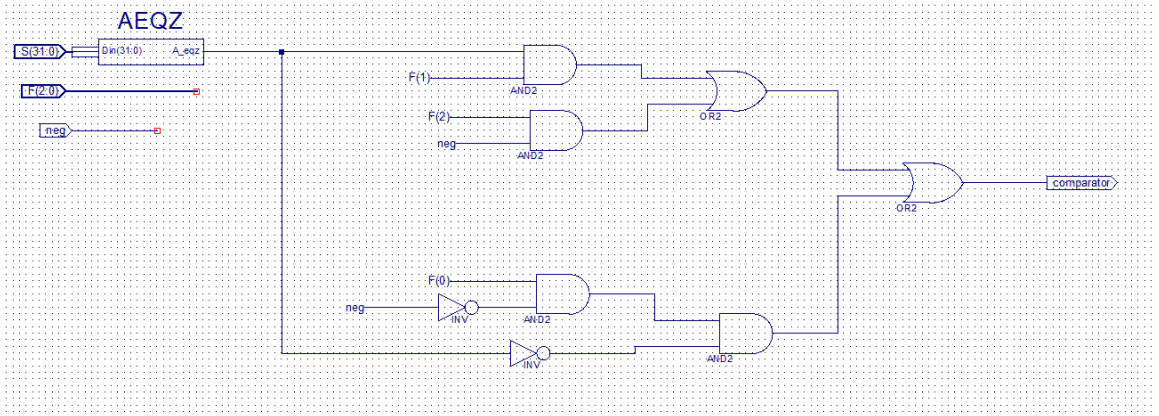
ALU:



ALU_ADD:



Comparator:



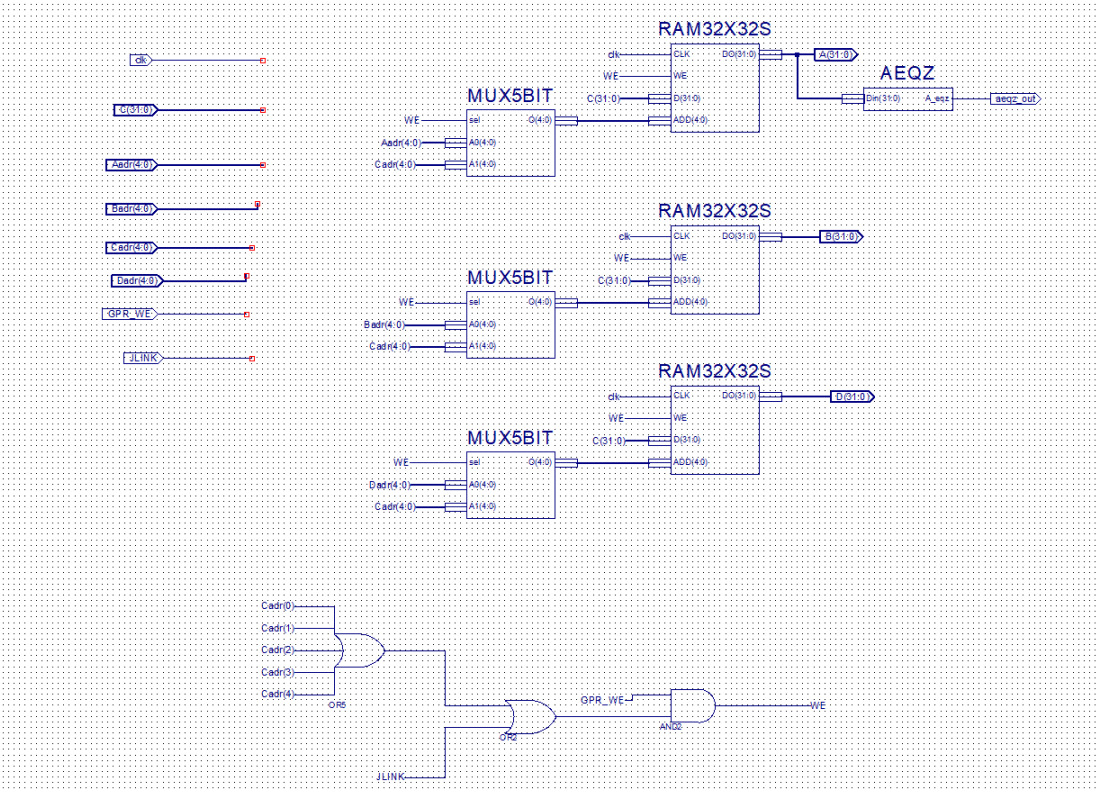
CONST011:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity const011 is
33      Port ( const : out  STD_LOGIC_VECTOR (2 downto 0));
34  end const011;
35
36  architecture Behavioral of const011 is
37
38  begin
39
40  const <="011";
41
42  end Behavioral;
43
44
```

Shift:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity shift is
33      Port ( shift : in  STD_LOGIC;
34             R : in  STD_LOGIC;
35             DIN : in  STD_LOGIC_VECTOR (31 downto 0);
36             DOUT : out  STD_LOGIC_VECTOR (31 downto 0));
37  end shift;
38
39  architecture Behavioral of shift is
40
41  begin
42
43  DOUT<= ('0' & DIN(31 downto 1)) when ((R = '1') and (shift = '1')) else (DIN(30 downto 0) & '0') when ((R = '0') and (shift='1')) else DIN;
44
45  end Behavioral;
46
```

GPR:



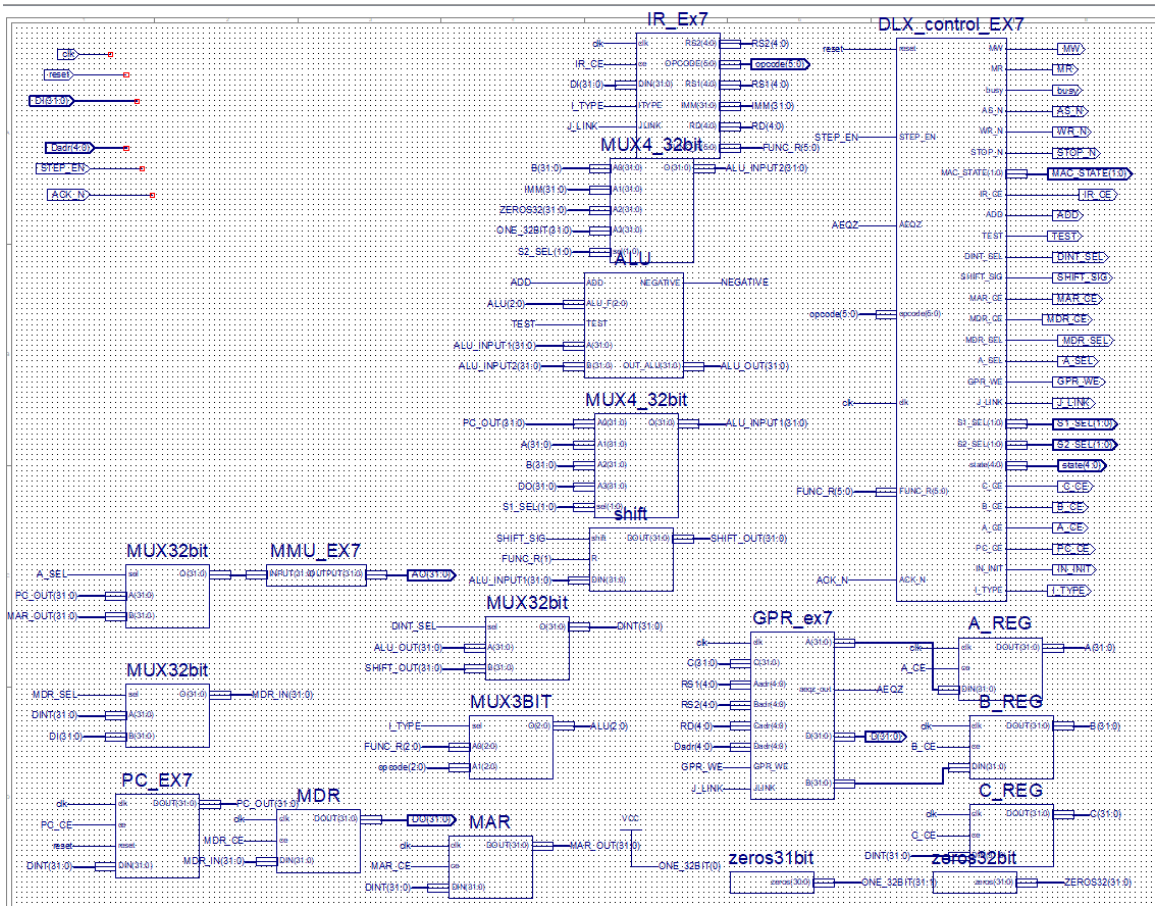
AEQZ:

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity AEQZ is
33      Port ( Din : in  STD_LOGIC_VECTOR (31 downto 0);
34             A_eqz : out  STD_LOGIC);
35  end AEQZ;
36
37  architecture Behavioral of AEQZ is
38
39  begin
40  A_eqz <= '1' when Din = X"00000000" else '0';
41
42  end Behavioral;
43
44
```

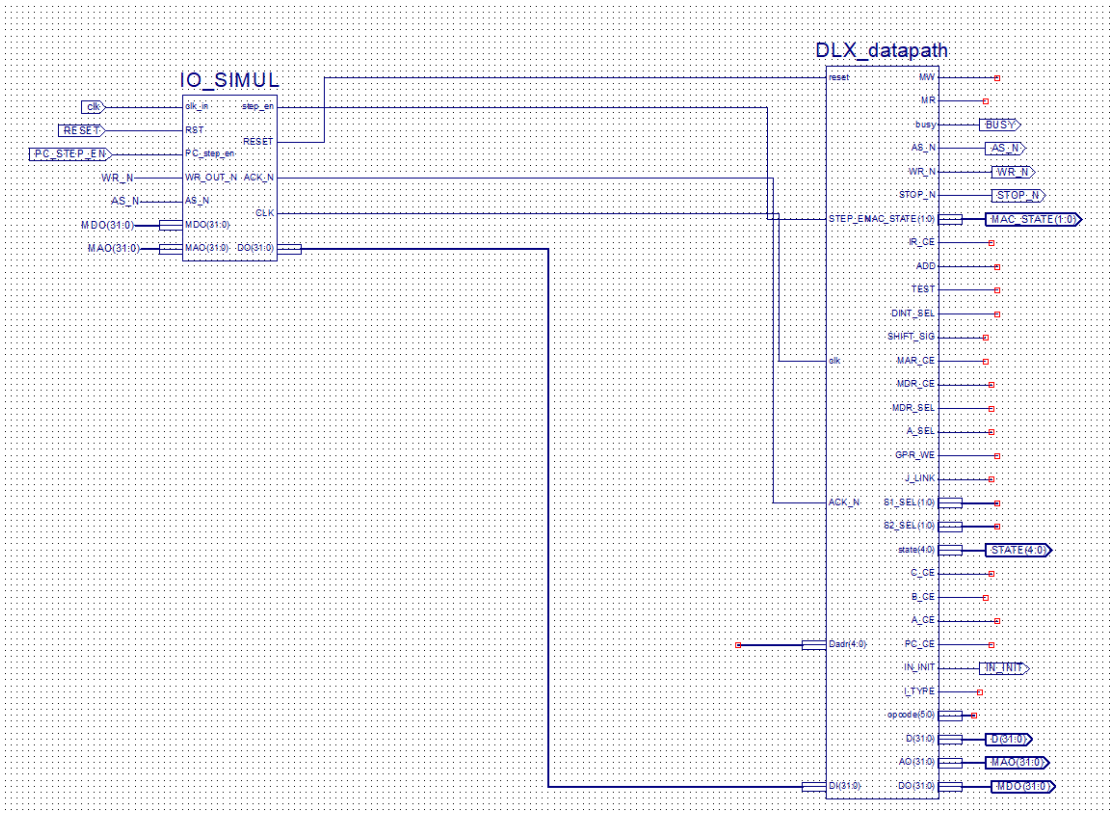
MMU:

```
----------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:25:40 12/12/2021
-- Design Name:
-- Module Name:    MMU_EX7 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MMU_EX7 is
    Port ( INPUT : in  STD_LOGIC_VECTOR (31 downto 0);
           OUTPUT : out  STD_LOGIC_VECTOR (31 downto 0));
end MMU_EX7;

architecture Behavioral of MMU_EX7 is

begin

OUTPUT <= x"00" & INPUT (23 downto 0);

end Behavioral;


```

Datapath:


IR_Ex7
DLX_control_EX7
MUX4_32bit
ALU
MUX4_32bit
shift
MUX32bit
MMU_EX7
MUX32bit
MUX3BIT
GPR_ex7
A_REG
B_REG
C_REG
PC_EX7
MDR
MAR
zeros31bit
zeros32bit

IO SIMUL:


IO_SIMUL
clk
RESET
PC_STEP_EN
WR_N
AS_N
MDO(31:0)
MAO(31:0)
clk_in
RST
PC_step_en
WR_OUT_N
AS_N
MDO(31:0)
MAO(31:0)
step_en
RESET
ACK_N
CLK
DO(31:0)
DLX_datapath
reset
clk
ACK_N
Dadr(4:0)
STEP-EMAC_STATE(1:0)
Di(31:0)
Di(31:0)
MW
MR
busy
AS_N
WR_N
STOP_N
IR_CE
ADD
TEST
DINT_SEL
SHIFT_SIG
MAR_CE
MDR_CE
MDR_SEL
A_SEL
GPR_WE
J_LINK
B1_SEL(1:0)
B2_SEL(1:0)
state(4:0)
C_CE
B_CE
A_CE
PC_CE
IN_INIT
LTYPE
opcode(5:0)
DI(31:0)
AO(31:0)
DO(31:0)
BUSY
AS_N
WR_N
STOP_N
MAC_STATE(1:0)
STATE(4:0)
IN_INIT
DI(31:0)
MAO(31:0)
MDO(31:0)

Test Vectors:

ALU:

ALU														
I/O	Signals	Expected Values												
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"100000"	"100000"	"100000"	"100000"	"100000"	"100000"	"100000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	AEQZ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Output	DLX STATE (4:0)	"00000"	"00000"	"00001"	"00001"	"00001"	"00001"	"00001"	"00010"	"00100"	"01100"	"00000"	"00000"	"00000"
		INIT	INIT	FETCH	FETCH	FETCH	FETCH	FETCH	DECODE	ALU	WBR	INIT	INIT	INIT
Output	MAC STATE (1:0)	"00"	"00"	"00"	"01"	"01"	"01"	"01"	"10"	"00"	"00"	"00"	"00"	"00"
		wait4req	wait4req	wait4req	wait4ack	wait4ack	wait4ack	wait4ack	next	wait4req	wait4req	wait4req	wait4req	wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	1	1	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	1	0	0	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	1	0	0	0
Output	MR	0	0	1	1	1	1	1	0	0	0	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	0	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	1	1	1
Output	AS_N	1	1	1	0	0	0	0	1	1	1	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"01"	"00"	"00"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"00"	"00"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	0	0	0	0

BTAKEN:

BTAKEN														
I/O	Signals	Expected Values												
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"000101"	"000101"	"000101"	"000101"	"000101"	"000101"	"000101"
Input	AEQZ	NA	NA	NA	NA	NA	NA	0	0	0	0	0	0	0
Output	DLX STATE (4:0)	"00000"	"00000"	"00001"	"00001"	"00001"	"00001"	"00001"	"00010"	"01011"	"10001"	"00000"	"00000"	"00000"
		INIT	INIT	FETCH	FETCH	FETCH	FETCH	FETCH	DECODE	BRANCH	BTAKEN	INIT	INIT	INIT
Output	MAC STATE (1:0)	"00"	"00"	"00"	"01"	"01"	"01"	"01"	"10"	"00"	"00"	"00"	"00"	"00"
		wait4req	wait4req	wait4req	wait4ack	wait4ack	wait4ack	wait4ack	next	wait4req	wait4req	wait4req	wait4req	wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	1	1	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"000101"	"000101"	"000101"	"000101"	"000101"	"000101"	"000101"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	1	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MR	0	0	1	1	1	1	1	0	0	0	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	0	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	1	1	1
Output	AS_N	1	1	1	0	0	0	0	1	1	1	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"00"	"01"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	0	1	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	0	0	0	0

JALR:

JALR														
I/O	Signals	Expected Values												
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"010111"	"010111"	"010111"	"010111"	"010111"	"010111"	"010111"
Input	AEQZ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Output	DLX STATE (4:0)	"00000"	"00000"	"00001"	"00001"	"00001"	"00001"	"00001"	"00010"	"01010"	"10000"	"00000"	"00000"	"00000"
		INIT	INIT	FETCH	FETCH	FETCH	FETCH	FETCH	DECODE	SAVEPC	JALR	INIT	INIT	INIT
Output	MAC STATE (1:0)	"00"	"00"	"00"	"01"	"01"	"01"	"01"	"10"	"00"	"00"	"00"	"00"	"00"
		wait4req	wait4req	wait4req	wait4ack	wait4ack	wait4ack	wait4ack	next	wait4req	wait4req	wait4req	wait4req	wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	1	1	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"010111"	"010111"	"010111"	"010111"	"010111"	"010111"	"010111"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	1	0	0	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	1	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MR	0	0	1	1	1	1	1	0	0	0	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	0	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	1	1	1
Output	AS_N	1	1	1	0	0	0	0	1	1	1	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"01"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"10"	"10"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	1	1	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	1	0	0	0

LOAD:

LOAD																			
I/O	Signals	Expected Values																	
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"
Input	AEQZ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Output	DLX STATE (4:0)	"00000"	"00000"	"00001"	"00001"	"00001"	"00001"	"00001"	"00010"	"01000"	"01110"	"01110"	"01110"	"01110"	"01110"	"01110"	"10010"	"01101"	"00000"
		INIT	INIT	FETCH	FETCH	FETCH	FETCH	FETCH	DECODE	ADDRESSCMP	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	COPYMDR2C	WBI	INIT
Output	MAC STATE (1:0)	"00"	"00"	"00"	"01"	"01"	"01"	"01"	"10"	"00"	"00"	"01"	"01"	"01"	"01"	"01"	"10"	"00"	"00"
		wait4req	wait4req	wait4req	wait4ack	wait4ack	wait4ack	wait4ack	next	wait4req	wait4req	wait4ack	wait4ack	wait4ack	wait4ack	wait4ack	next	wait4req	wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"	"100011"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Output	MR	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Output	AS_N	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	0	0	0	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"01"	"00"	"00"	"00"	"00"	"00"	"11"	"00"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"10"	"00"	"00"	"00"	"00"	"00"	"10"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STORE:

STORE																		
I/O	Signals	Expected Values																
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"
Input	AEQZ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Output	DLX STATE (4:0)	"00000" INIT	"00000" INIT	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00010" DECODE	"01000" ADDRESSCMP	"01111" COPY1GPR2MD	"10011" STORE	"10011" STORE	"10011" STORE	"10011" STORE	"10011" STORE	"00000" INIT	"00000" INIT	"00000" INIT
Output	MAC STATE (1:0)	"00" wait4req	"00" wait4req	"00" wait4req	"01" wait4ack	"01" wait4ack	"01" wait4ack	"01" wait4ack	"10" next	"00" wait4req	"00" wait4ack	"00" wait4req	"01" wait4ack	"01" wait4ack	"01" wait4ack	"01" wait4ack	"10" next	"00" wait4req	"00" wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"	"101011"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MR	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1
Output	AS_N	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"01"	"10"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"01"	"10"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TESTI:

TESTI														
I/O	Signals	Expected Values												
Input	#CC	0	1	2	3	4	5	6	7	8	9	10	11	12
Input	RESET	1	0	0	0	0	0	0	0	0	0	0	0	0
Input	STEP_EN	0	1	0	0	0	0	0	0	0	0	0	0	0
Input	ACK_N	1	1	1	1	1	1	0	1	1	1	1	1	1
Input	FUNC_R(5:0)	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"	"000000"
Input	opcode(5:0)	NA	NA	NA	NA	NA	NA	"011011"	"011011"	"011011"	"011011"	"011011"	"011011"	"011011"
Input	AEQZ	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Output	DLX STATE (4:0)	"00000" INIT	"00000" INIT	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00001" FETCH	"00010" DECODE	"00111" TESTI	"01101" WBI	"00000" INIT	"00000" INIT	"00000" INIT
Output	MAC STATE (1:0)	"00" wait4req	"00" wait4req	"00" wait4req	"01" wait4ack	"01" wait4ack	"01" wait4ack	"01" wait4ack	"10" next	"00" wait4req	"00" wait4req	"00" wait4req	"00" wait4req	"00" wait4req
Output	IN_INIT	1	1	0	0	0	0	0	0	0	0	1	1	1
Output	opcode(5:0)	NA	NA	NA	NA	NA	NA	"011011"	"011011"	"011011"	"011011"	"011011"	"011011"	"011011"
Output	IR_CE	0	0	0	0	0	0	1	0	0	0	0	0	0
Output	A_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	B_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	C_CE	0	0	0	0	0	0	0	0	1	0	0	0	0
Output	PC_CE	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	GPR_WE	0	0	0	0	0	0	0	0	0	1	0	0	0
Output	MR	0	0	1	1	1	1	1	0	0	0	0	0	0
Output	MW	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	BUSY	0	0	1	1	1	1	0	0	0	0	0	0	0
Output	WR_N	1	1	1	1	1	1	1	1	1	1	1	1	1
Output	AS_N	1	1	1	0	0	0	0	1	1	1	1	1	1
Output	STOP_N	1	1	1	1	0	0	1	1	1	1	1	1	1
Output	S1_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"01"	"00"	"00"	"00"	"00"
Output	S2_SEL(1:0)	"00"	"00"	"00"	"00"	"00"	"00"	"00"	"11"	"01"	"00"	"00"	"00"	"00"
Output	ADD	0	0	0	0	0	0	0	1	0	0	0	0	0
Output	TEST	0	0	0	0	0	0	0	0	1	0	0	0	0
Output	I_TYPE	0	0	0	0	0	0	0	0	1	1	0	0	0
Output	DINT_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	SHIFT_SIG	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MAR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_CE	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	MDR_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	A_SEL	0	0	0	0	0	0	0	0	0	0	0	0	0
Output	JLINK	0	0	0	0	0	0	0	0	0	1	0	0	0

DLX Control Testbench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY DLX_control_EX7_DLX_control_EX7_sch_tb IS
END DLX_control_EX7_DLX_control_EX7_sch_tb;
ARCHITECTURE behavioral OF DLX_control_EX7_DLX_control_EX7_sch_tb IS

   COMPONENT DLX_control_EX7
   PORT( MW :  OUT   STD_LOGIC;
         MR   :  OUT   STD_LOGIC;
         busy :  OUT   STD_LOGIC;
         AS_N :  OUT   STD_LOGIC;
         WR_N :  OUT   STD_LOGIC;
         STOP_N  :  OUT   STD_LOGIC;
         MAC_STATE  :  OUT   STD_LOGIC_VECTOR (1 DOWNTO 0);
         IR_CE   :  OUT   STD_LOGIC;
         ADD  :  OUT   STD_LOGIC;
         TEST :  OUT   STD_LOGIC;
         DINT_SEL   :  OUT   STD_LOGIC;
         SHIFT_SIG  :  OUT   STD_LOGIC;
         MAR_CE  :  OUT   STD_LOGIC;
         MDR_CE  :  OUT   STD_LOGIC;
         MDR_SEL :  OUT   STD_LOGIC;
         A_SEL   :  OUT   STD_LOGIC;
         GPR_WE  :  OUT   STD_LOGIC;
         J_LINK  :  OUT   STD_LOGIC;
         S1_SEL  :  OUT   STD_LOGIC_VECTOR (1 DOWNTO 0);
         S2_SEL  :  OUT   STD_LOGIC_VECTOR (1 DOWNTO 0);
         state   :  OUT   STD_LOGIC_VECTOR (4 DOWNTO 0);
         C_CE :  OUT   STD_LOGIC;
         B_CE :  OUT   STD_LOGIC;
         A_CE :  OUT   STD_LOGIC;
         PC_CE   :  OUT   STD_LOGIC;
         IN_INIT :  OUT   STD_LOGIC;
         reset   :  IN STD_LOGIC;
         STEP_EN :  IN STD_LOGIC;
         AEQZ :  IN STD_LOGIC;
         opcode  :  IN STD_LOGIC_VECTOR (5 DOWNTO 0);
         clk  :  IN STD_LOGIC;
         FUNC_R  :  IN STD_LOGIC_VECTOR (5 DOWNTO 0);
         ACK_N   :  IN STD_LOGIC;
         I_TYPE  :  OUT   STD_LOGIC);
   END COMPONENT;
```

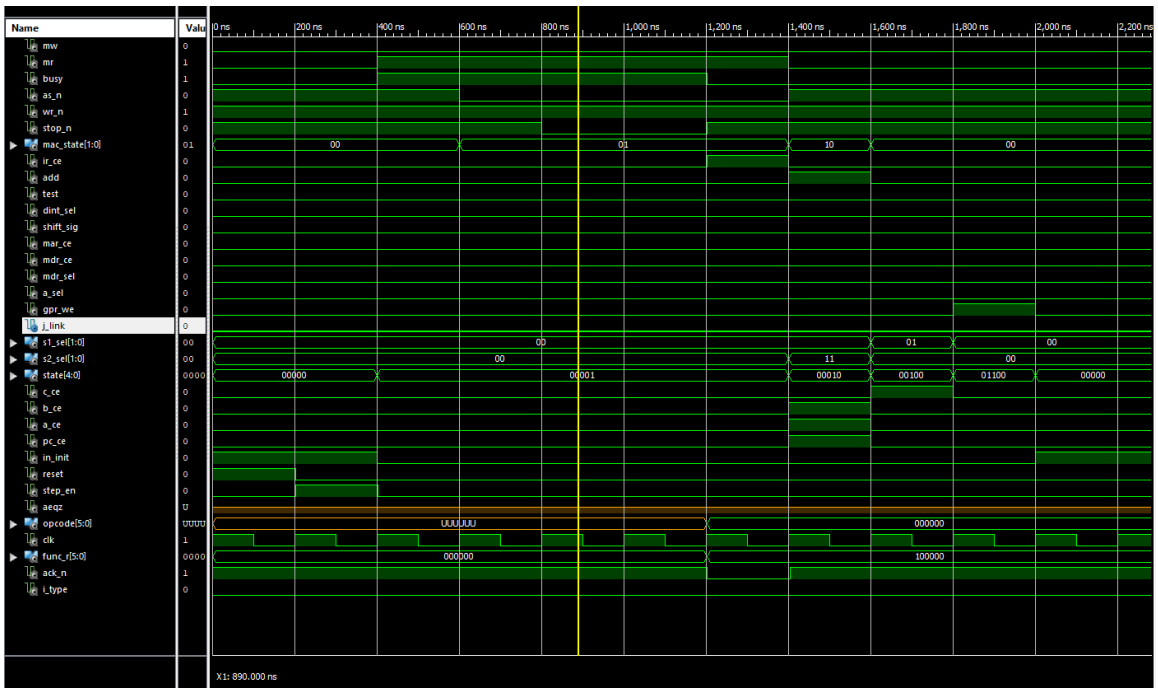
```
61      SIGNAL MW    :  STD_LOGIC;
62      SIGNAL MR    :  STD_LOGIC;
63      SIGNAL busy  :  STD_LOGIC;
64      SIGNAL AS_N  :  STD_LOGIC;
65      SIGNAL WR_N  :  STD_LOGIC;
66      SIGNAL STOP_N   :  STD_LOGIC;
67      SIGNAL MAC_STATE   :  STD_LOGIC_VECTOR (1 DOWNTO 0);
68      SIGNAL IR_CE    :  STD_LOGIC;
69      SIGNAL ADD   :  STD_LOGIC;
70      SIGNAL TEST  :  STD_LOGIC;
71      SIGNAL DINT_SEL    :  STD_LOGIC;
72      SIGNAL SHIFT_SIG   :  STD_LOGIC;
73      SIGNAL MAR_CE   :  STD_LOGIC;
74      SIGNAL MDR_CE   :  STD_LOGIC;
75      SIGNAL MDR_SEL  :  STD_LOGIC;
76      SIGNAL A_SEL    :  STD_LOGIC;
77      SIGNAL GPR_WE   :  STD_LOGIC;
78      SIGNAL J_LINK   :  STD_LOGIC;
79      SIGNAL S1_SEL   :  STD_LOGIC_VECTOR (1 DOWNTO 0);
80      SIGNAL S2_SEL   :  STD_LOGIC_VECTOR (1 DOWNTO 0);
81      SIGNAL state    :  STD_LOGIC_VECTOR (4 DOWNTO 0);
82      SIGNAL C_CE  :  STD_LOGIC;
83      SIGNAL B_CE  :  STD_LOGIC;
84      SIGNAL A_CE  :  STD_LOGIC;
85      SIGNAL PC_CE    :  STD_LOGIC;
86      SIGNAL IN_INIT  :  STD_LOGIC;
87      SIGNAL reset    :  STD_LOGIC;
88      SIGNAL STEP_EN  :  STD_LOGIC;
89      SIGNAL AEQZ  :  STD_LOGIC;
90      SIGNAL opcode   :  STD_LOGIC_VECTOR (5 DOWNTO 0);
91      SIGNAL clk   :  STD_LOGIC;
92      SIGNAL FUNC_R   :  STD_LOGIC_VECTOR (5 DOWNTO 0);
93      SIGNAL ACK_N    :  STD_LOGIC;
94      SIGNAL I_TYPE   :  STD_LOGIC;
95
```

```
 96  BEGIN
 97
 98     UUT: DLX_control_EX7 PORT MAP(
 99        MW => MW,
100        MR => MR,
101        busy => busy,
102        AS_N => AS_N,
103        WR_N => WR_N,
104        STOP_N => STOP_N,
105        MAC_STATE => MAC_STATE,
106        IR_CE => IR_CE,
107        ADD => ADD,
108        TEST => TEST,
109        DINT_SEL => DINT_SEL,
110        SHIFT_SIG => SHIFT_SIG,
111        MAR_CE => MAR_CE,
112        MDR_CE => MDR_CE,
113        MDR_SEL => MDR_SEL,
114        A_SEL => A_SEL,
115        GPR_WE => GPR_WE,
116        J_LINK => J_LINK,
117        S1_SEL => S1_SEL,
118        S2_SEL => S2_SEL,
119        state => state,
120        C_CE => C_CE,
121        B_CE => B_CE,
122        A_CE => A_CE,
123        PC_CE => PC_CE,
124        IN_INIT => IN_INIT,
125        reset => reset,
126        STEP_EN => STEP_EN,
127        AEQZ => AEQZ,
128        opcode => opcode,
129        clk => clk,
130        FUNC_R => FUNC_R,
131        ACK_N => ACK_N,
132        I_TYPE => I_TYPE
133     );
134
135  -- *** Test Bench - User Defined Section ***
136
137  CLKp: process
138  begin
139  CLK<='1';
140  wait for 100 ns;
141  CLK<='0';
142  wait for 100 ns;
143  end process;
```

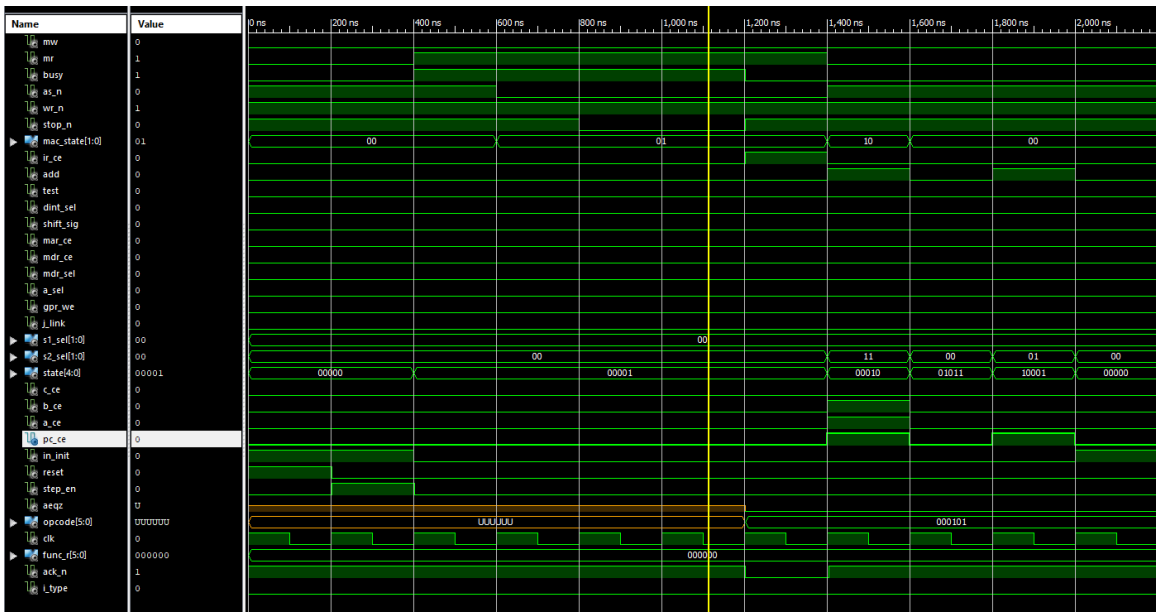
```
145  tb: process
146  begin
147  --initialization
148  reset<='1';
149  STEP_EN<='0';
150  ACK_N<='1';
151  FUNC_R<="000000";
152  wait for 202 ns;
153  STEP_EN<='1';
154  reset<='0';
155  wait for 200 ns;
156  STEP_EN<='0';
157  wait for 800 ns;
158  ACK_N<='0';
159
160  --opcode templates
161
162  --opcode<="101011"; --STORE
163
164  --opcode<="100011"; --LOAD
165
166  --opcode<="000000"; --ALU
167  --FUNC_R<="100000"; --ALU
168
169  --opcode<="000101"; --BTAKEN
170  --AEQZ<='0';        --BTAKEN
171
172  --opcode<="010111"; --JALR
173
174  --opcode<="011011"; --TESTI
175
176  wait for 202 ns;
177  ACK_N <= '1';
178  wait for 1400 ns;
179  ACK_N <= '0';
180  wait for 200 ns;
181  ACK_N <= '1';
182
183  wait;
184  end process;
185
186  -- *** End Test Bench - User Defined Section ***
187
188  END;
```

Testbench results:

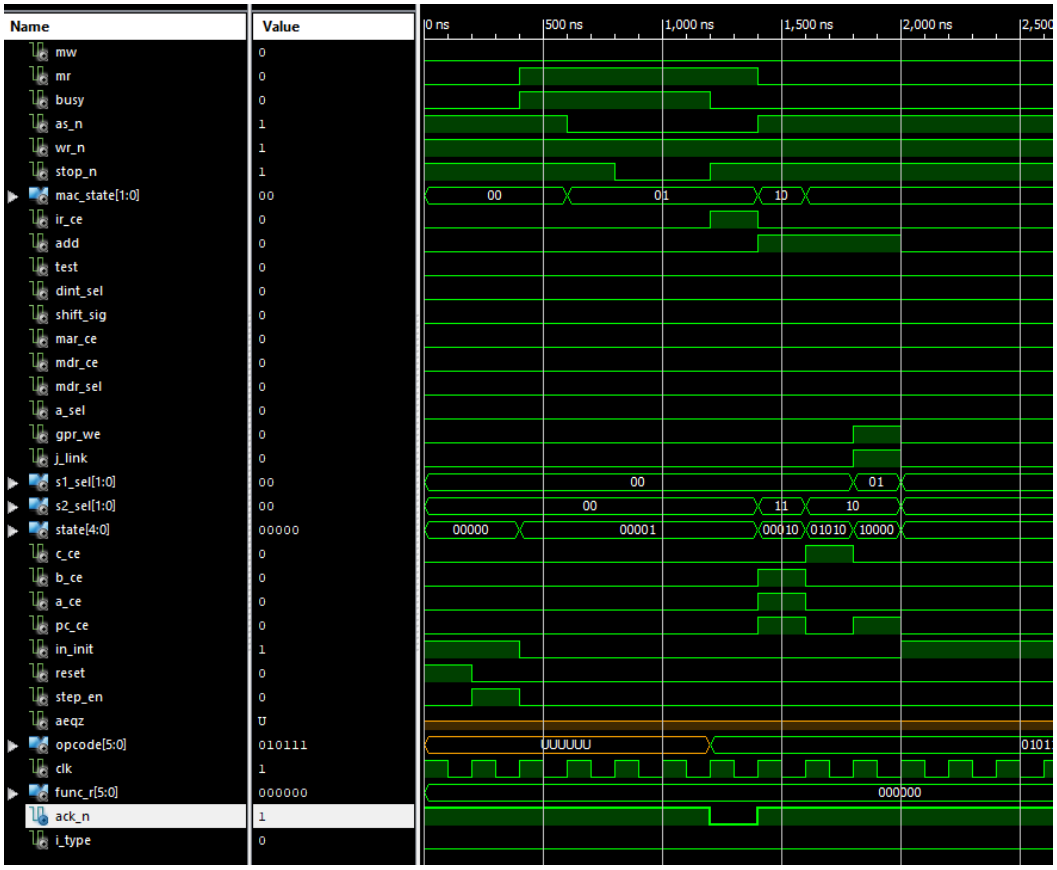
ALU:



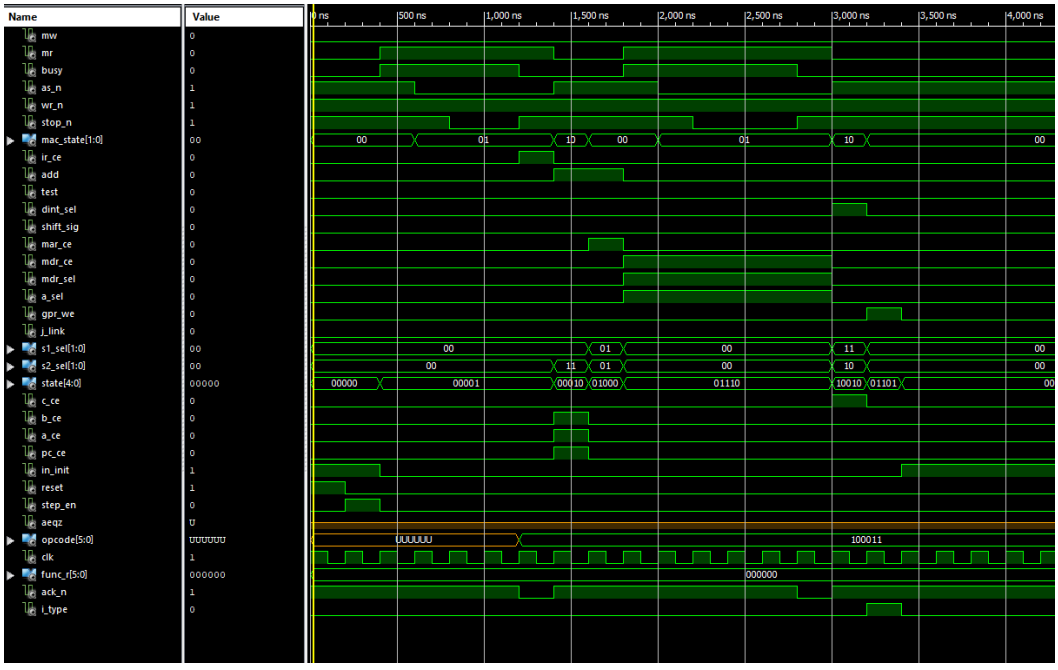
BTAKEN:



JALR:

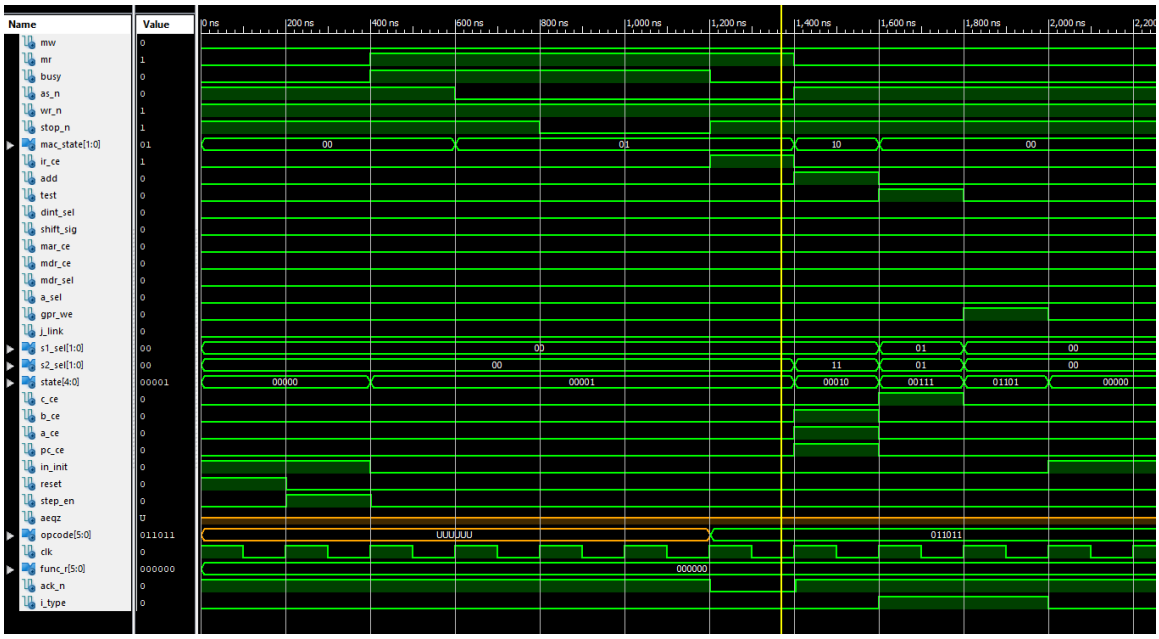

Name
Value
mw 0
mr 0
busy 0
as_n 1
wr_n 1
stop_n 1
mac_state[1:0] 00
ir_ce 0
add 0
test 0
dint_sel 0
shift_sig 0
mar_ce 0
mdr_ce 0
mdr_sel 0
a_sel 0
gpr_we 0
j_link 0
s1_sel[1:0] 00
s2_sel[1:0] 00
state[4:0] 00000
c_ce 0
b_ce 0
a_ce 0
pc_ce 0
in_init 1
reset 0
step_en 0
aeqz U
opcode[5:0] 010111
clk 1
func_r[5:0] 000000
ack_n 1
i_type 0
0 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
00
01
10
00
01
00
11
10
00000
00001
00010
01010
10000
UUUUUU
0101
000000

LOAD:


Name
Value
mw 0
mr 0
busy 0
as_n 1
wr_n 1
stop_n 1
mac_state[1:0] 00
ir_ce 0
add 0
test 0
dint_sel 0
shift_sig 0
mar_ce 0
mdr_ce 0
mdr_sel 0
a_sel 0
gpr_we 0
j_link 0
s1_sel[1:0] 00
s2_sel[1:0] 00
state[4:0] 00000
c_ce 0
b_ce 0
a_ce 0
pc_ce 0
in_init 1
reset 1
step_en 0
aeqz U
opcode[5:0] UUUUUU
clk 1
func_r[5:0] 000000
ack_n 1
i_type 0
0 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
2,500 ns
3,000 ns
3,500 ns
4,000 ns
00
01
10
00
01
10
00
00
01
00
11
00
00
11
01
00
10
00
00001
00010
01000
01110
10010
01101
UUUUUU
100011
000000

STORE:

TESTI:



The simulation results match the test vectors.

DLX Assembly Program:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

-- Package for SRAM pre-initialization data
package sram_data is

   -- Size of pre instantiated data
   constant data_size: integer := 33;

   type pre_inst_data is array(0 to data_size-1) of std_logic_vector(31 downto 0);
   constant pre_inst_mem : pre_inst_data := (
   -- The actual data :
X"2C010001", --   0x00000000:                    addi R1 R0 1
X"2C020001", --   0x00000001:                    addi R2 R0 1
X"00411023", --   0x00000002:                    add R2 R2 R1
X"AC02001F", --   0x00000003:                    sw R2 R0 addr1
X"8C030020", --   0x00000004:                    lw R3 R0 data1
X"00620822", --   0x00000005:                    sub R1 R3 R2
X"103F0001", --   0x00000006:                    beqz R1 BR1
X"2C01000A", --   0x00000007:                    addi R1 R0 10
X"143F0015", --   0x00000008:    BR1:            bnez R1 BR2
X"145F0001", --   0x00000009:                    bnez R2 BR3
X"2D21000A", --   0x0000000A:                    addi R1 R9 10
X"105F0012", --   0x0000000B:    BR3:            beqz R2 BR2
X"007F2000", --   0x0000000C:                    slli R4 R3
X"007F2802", --   0x0000000D:                    srli R5 R3
X"00822824", --   0x0000000E:                    xor R5 R4 R2
X"00820825", --   0x0000000F:                    or R1 R4 R2
X"00820826", --   0x00000010:                    and R1 R4 R2
X"70410002", --   0x00000011:                    slti R1 R2 2
X"68410002", --   0x00000012:                    seqi R1 R2 2
X"64410002", --   0x00000013:                    sgti R1 R2 2
X"78410002", --   0x00000014:                    slei R1 R2 2
X"6C220000", --   0x00000015:                    sgei R2 R1 0
X"74220001", --   0x00000016:                    snei R2 R1 1
X"2C06001B", --   0x00000017:                    addi R6 R0 27
X"2C07001E", --   0x00000018:                    addi R7 R0 30
X"5CDF0000", --   0x00000019:                    jalr R6
X"2C080001", --   0x0000001A:                    addi R8 R0 1
X"58FF0000", --   0x0000001B:                    jr R7
X"2C080001", --   0x0000001C:                    addi R8 R0 1
X"2C080001", --   0x0000001D:                    addi R8 R0 1
X"FFFF0000", --   0x0000001E:    BR2:            halt

X"00000000", --   0x0000001F:    addr1:          dc 0x0
X"00000002"  --   0x00000020:    data1:          dc 2
                      );

end sram_data;

package body sram_data is


end sram_data;

```

Testbench:

```
15  LIBRARY ieee;
16  USE ieee.std_logic_1164.ALL;
17  USE ieee.numeric_std.ALL;
18  LIBRARY UNISIM;
19  USE UNISIM.Vcomponents.ALL;
20  ENTITY IO_SIMUL_EX7_IO_SIMUL_EX7_sch_tb IS
21  END IO_SIMUL_EX7_IO_SIMUL_EX7_sch_tb;
22  ARCHITECTURE behavioral OF IO_SIMUL_EX7_IO_SIMUL_EX7_sch_tb IS
23
24     COMPONENT IO_SIMUL_EX7
25     PORT( clk   :  IN STD_LOGIC;
26           RESET   :  IN STD_LOGIC;
27           PC_STEP_EN :  IN STD_LOGIC;
28           WR_N :  OUT   STD_LOGIC;
29           AS_N :  OUT   STD_LOGIC;
30           MDO  :  OUT   STD_LOGIC_VECTOR (31 DOWNTO 0);
31           MAO  :  OUT   STD_LOGIC_VECTOR (31 DOWNTO 0);
32           BUSY :  OUT   STD_LOGIC;
33           STOP_N  :  OUT   STD_LOGIC;
34           MAC_STATE  :  OUT   STD_LOGIC_VECTOR (1 DOWNTO 0);
35           STATE   :  OUT   STD_LOGIC_VECTOR (4 DOWNTO 0);
36           IN_INIT :  OUT   STD_LOGIC;
37           D :  OUT   STD_LOGIC_VECTOR (31 DOWNTO 0));
38     END COMPONENT;
39
40     SIGNAL clk  :  STD_LOGIC;
41     SIGNAL RESET   :  STD_LOGIC;
42     SIGNAL PC_STEP_EN :  STD_LOGIC;
43     SIGNAL WR_N :  STD_LOGIC;
44     SIGNAL AS_N :  STD_LOGIC;
45     SIGNAL MDO  :  STD_LOGIC_VECTOR (31 DOWNTO 0);
46     SIGNAL MAO  :  STD_LOGIC_VECTOR (31 DOWNTO 0);
47     SIGNAL BUSY :  STD_LOGIC;
48     SIGNAL STOP_N  :  STD_LOGIC;
49     SIGNAL MAC_STATE  :  STD_LOGIC_VECTOR (1 DOWNTO 0);
50     SIGNAL STATE   :  STD_LOGIC_VECTOR (4 DOWNTO 0);
51     SIGNAL IN_INIT :  STD_LOGIC;
52     SIGNAL D :  STD_LOGIC_VECTOR (31 DOWNTO 0);
53
54  BEGIN
```

```
56      UUT: IO_SIMUL_EX7 PORT MAP(
57         clk => clk,
58         RESET => RESET,
59         PC_STEP_EN => PC_STEP_EN,
60         WR_N => WR_N,
61         AS_N => AS_N,
62         MDO => MDO,
63         MAO => MAO,
64         BUSY => BUSY,
65         STOP_N => STOP_N,
66         MAC_STATE => MAC_STATE,
67         STATE => STATE,
68         IN_INIT => IN_INIT,
69         D => D
70      );
71
72  -- *** Test Bench - User Defined Section ***
73  clk_proc: process
74  begin
75  CLK<='1';
76  wait for 100 ns;
77  CLK<='0';
78  wait for 100 ns;
79  end process;
80
81  tb: process
82  begin
83  RESET<='1';
84  PC_STEP_EN<='0';
85  wait for 202 ns;
86  RESET<='0';
87  PC_STEP_EN<='1'; --start instruction 1
88  wait for 200 ns;
89  PC_STEP_EN<='0';
90  wait for 3000 ns;
91  PC_STEP_EN<='1'; --start instruction 2
92  wait for 200 ns;
93  PC_STEP_EN<='0';
94  wait for 3000 ns;
95  PC_STEP_EN<='1'; --start instruction 3
96  wait for 200 ns;
97  PC_STEP_EN<='0';
98  wait for 3000 ns;
99  PC_STEP_EN<='1'; --start instruction 4
100 wait for 200 ns;
101 PC_STEP_EN<='0';
102 wait for 3000 ns;
103 PC_STEP_EN<='1'; --start instruction 5
104 wait for 200 ns;
105 PC_STEP_EN<='0';
106 wait for 3000 ns;
107 PC_STEP_EN<='1'; --start instruction 6
108 wait for 200 ns;
109 PC_STEP_EN<='0';
110 wait for 3000 ns;
111 PC_STEP_EN<='1'; --start instruction 7
112 wait for 200 ns;
113 PC_STEP_EN<='0';
114 wait for 3000 ns;
```

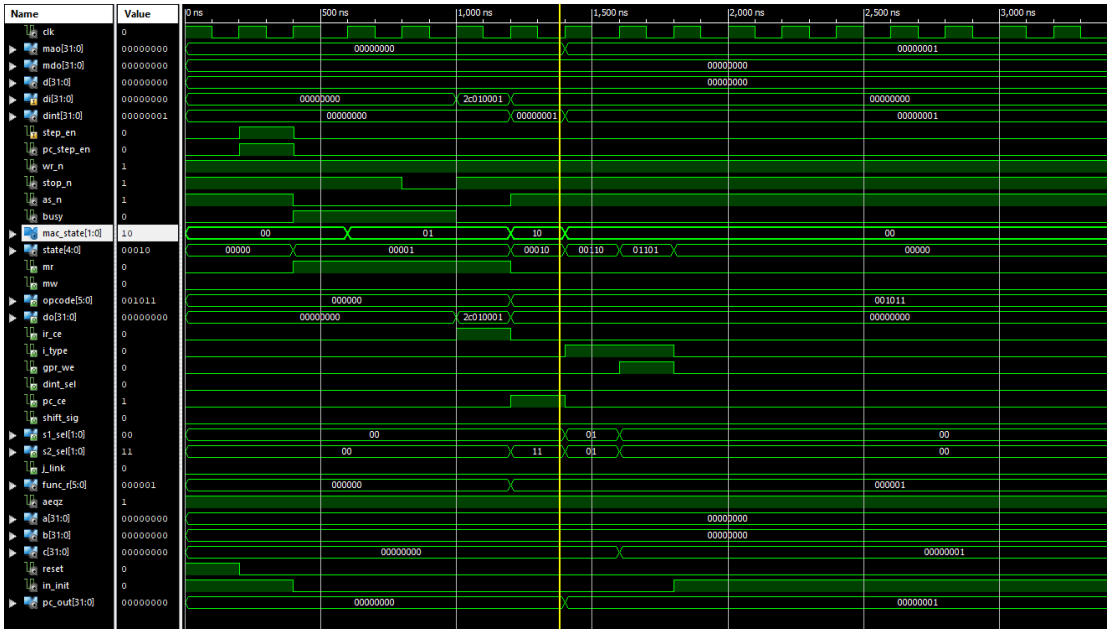
```
115  PC_STEP_EN<='1'; --start instruction 8
116  wait for 200 ns;
117  PC_STEP_EN<='0';
118  wait for 3000 ns;
119  PC_STEP_EN<='1'; --start instruction 9
120  wait for 200 ns;
121  PC_STEP_EN<='0';
122  wait for 3000 ns;
123  PC_STEP_EN<='1'; --start instruction 10
124  wait for 200 ns;
125  PC_STEP_EN<='0';
126  wait for 3000 ns;
127  PC_STEP_EN<='1'; --start instruction 11
128  wait for 200 ns;
129  PC_STEP_EN<='0';
130  wait for 3000 ns;
131  PC_STEP_EN<='1'; --start instruction 12
132  wait for 200 ns;
133  PC_STEP_EN<='0';
134  wait for 3000 ns;
135  PC_STEP_EN<='1'; --start instruction 13
136  wait for 200 ns;
137  PC_STEP_EN<='0';
138  wait for 3000 ns;
139  PC_STEP_EN<='1'; --start instruction 14
140  wait for 200 ns;
141  PC_STEP_EN<='0';
142  wait for 3000 ns;
143  PC_STEP_EN<='1'; --start instruction 15
144  wait for 200 ns;
145  PC_STEP_EN<='0';
146  wait for 3000 ns;
147  PC_STEP_EN<='1'; --start instruction 16
148  wait for 200 ns;
149  PC_STEP_EN<='0';
150  wait for 3000 ns;
151  PC_STEP_EN<='1'; --start instruction 17
152  wait for 200 ns;
153  PC_STEP_EN<='0';
154  wait for 3000 ns;
155  PC_STEP_EN<='1'; --start instruction 18
156  wait for 200 ns;
157  PC_STEP_EN<='0';
158  wait for 3000 ns;
159  PC_STEP_EN<='1'; --start instruction 19
160  wait for 200 ns;
161  PC_STEP_EN<='0';
162  wait for 3000 ns;
163  PC_STEP_EN<='1'; --start instruction 20
164  wait for 200 ns;
165  PC_STEP_EN<='0';
166  wait for 3000 ns;
167  PC_STEP_EN<='1'; --start instruction 21
168  wait for 200 ns;
169  PC_STEP_EN<='0';
170  wait for 3000 ns;
171  PC_STEP_EN<='1'; --start instruction 22
172  wait for 200 ns;
173  PC_STEP_EN<='0';
174  wait for 3000 ns;
```

```
175   PC_STEP_EN<='1'; --start instruction 23
176   wait for 200 ns;
177   PC_STEP_EN<='0';
178   wait for 3000 ns;
179   PC_STEP_EN<='1'; --start instruction 24
180   wait for 200 ns;
181   PC_STEP_EN<='0';
182   wait for 3000 ns;
183   PC_STEP_EN<='1'; --start instruction 25
184   wait for 200 ns;
185   PC_STEP_EN<='0';
186   wait for 3000 ns;
187   PC_STEP_EN<='1'; --start instruction 26
188   wait for 200 ns;
189   PC_STEP_EN<='0';
190   wait for 3000 ns;
191
192   wait;
193   end process;
194   -- *** End Test Bench - User Defined Section ***
195
196   END;
197
```

Simulation results:

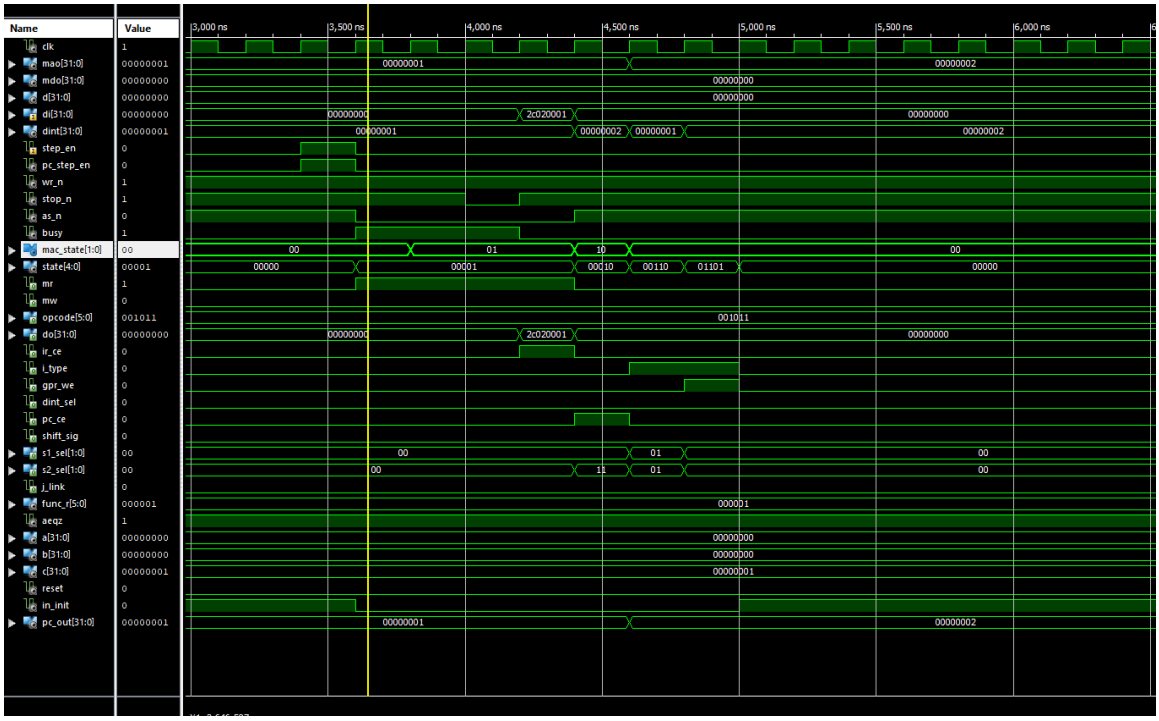
PC = 0x0 : addi R1 R0 0x1 (R1 = 0x1)

States: INIT, FETCH, DECODE, ALUI, WBI



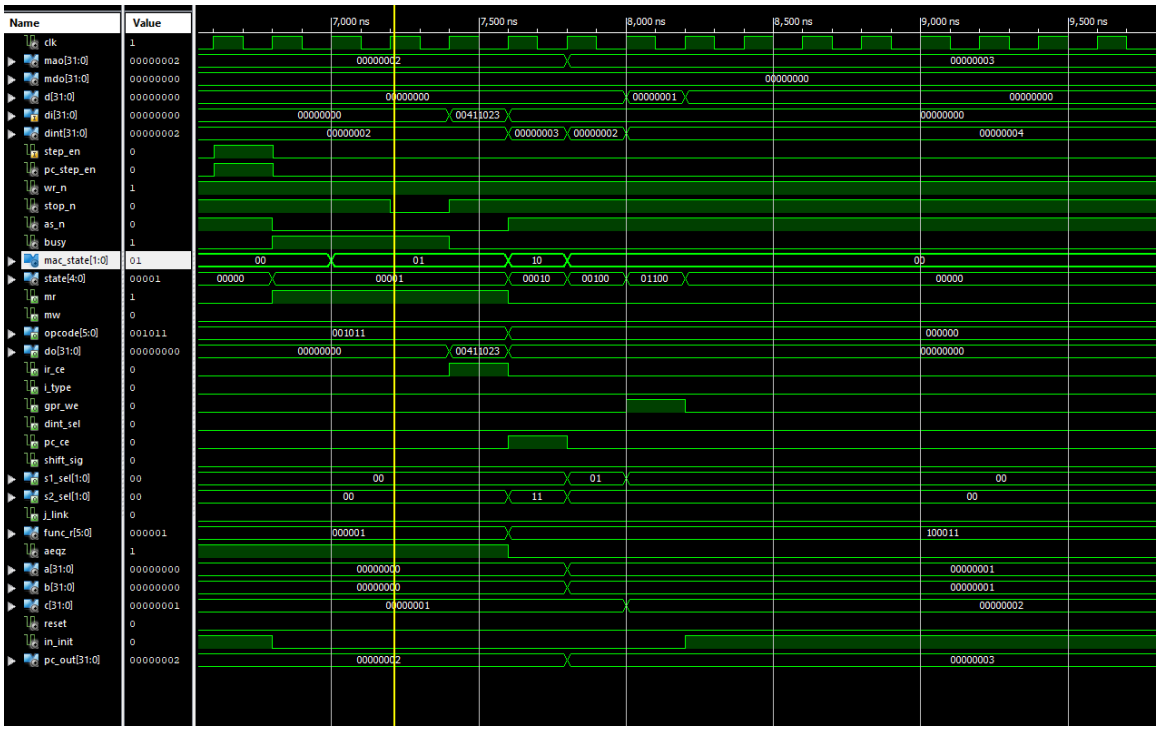
PC = 0x1 : addi R2 R0 0x1 (R2 = 0x1)

States: INIT, FETCH, DECODE, ALUI, WBI



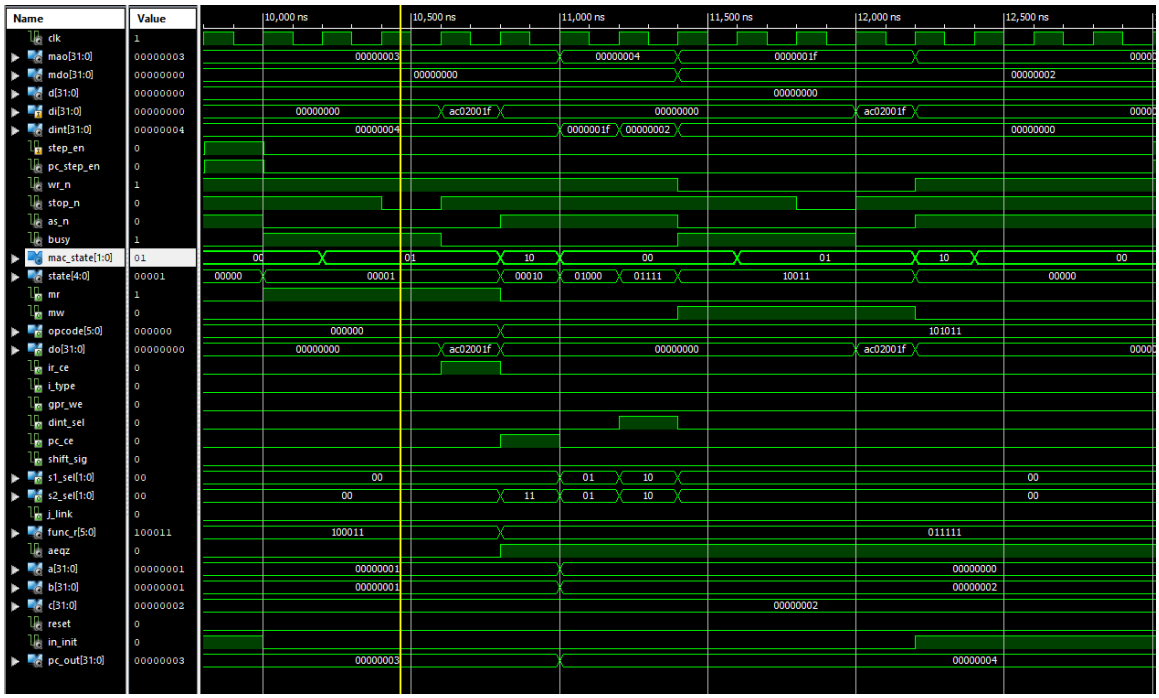
PC = 0x2 : add R2 R2 R1 (R2 = R2 + R1 = 0x2)

States: INIT, FETCH, DECODE, ALU, WBR



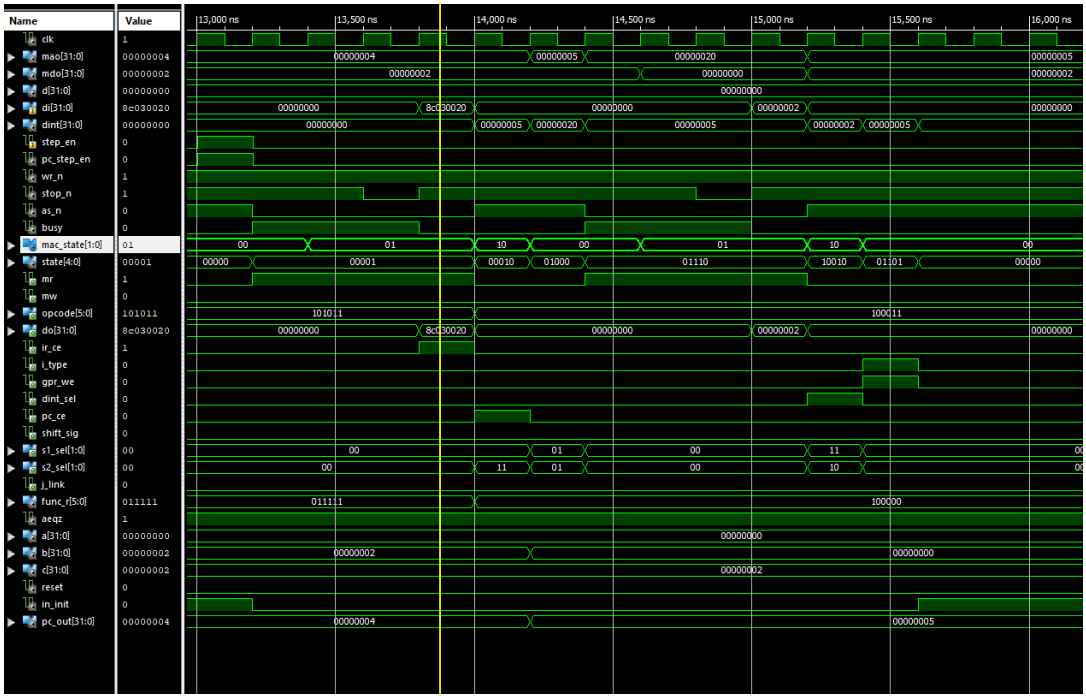
PC = 0x3 : sw R2 R0 addr1 (M[R0+addr1] = M[0x1F] = R2 = 0x2)

States: INIT, FETCH, DECODE, ADDRESSCMP, COPYGPR2MDR,STORE



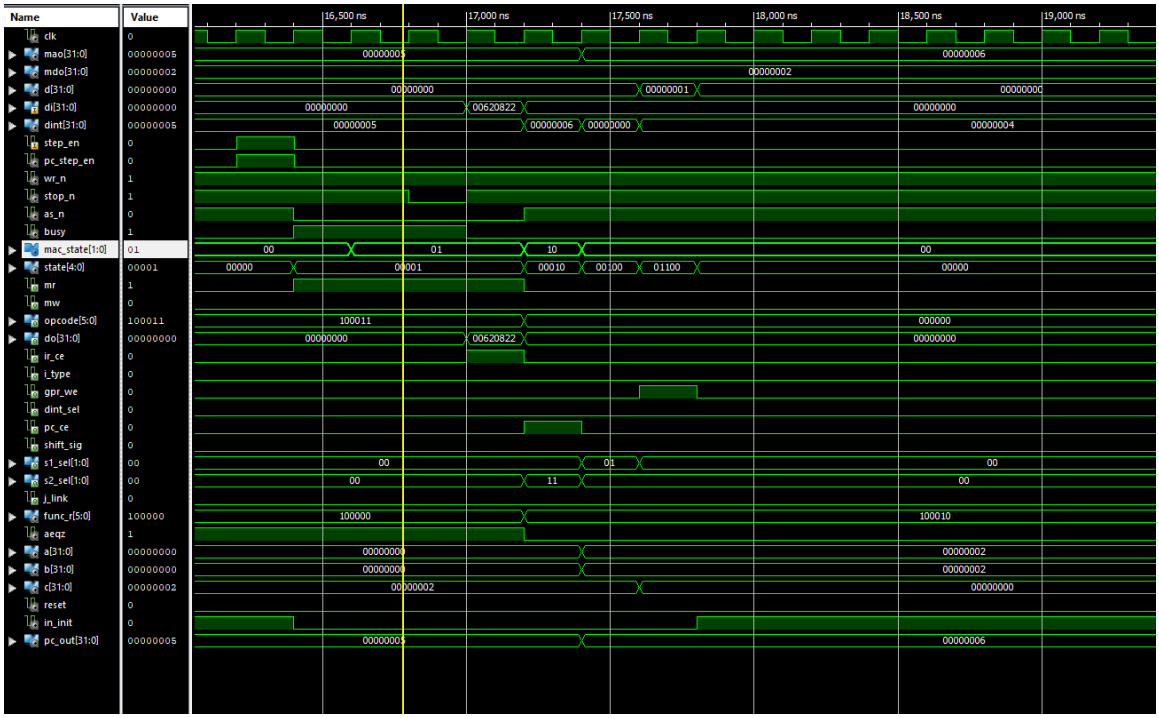
PC = 0x4 : lw R3 R0 data1 (R3 = M[R0+data1] = M[0x20] = 0x2)

States: INIT, FETCH, DECODE, ADDRESSCMP ,LOAD,COPYMDR2C,WBI



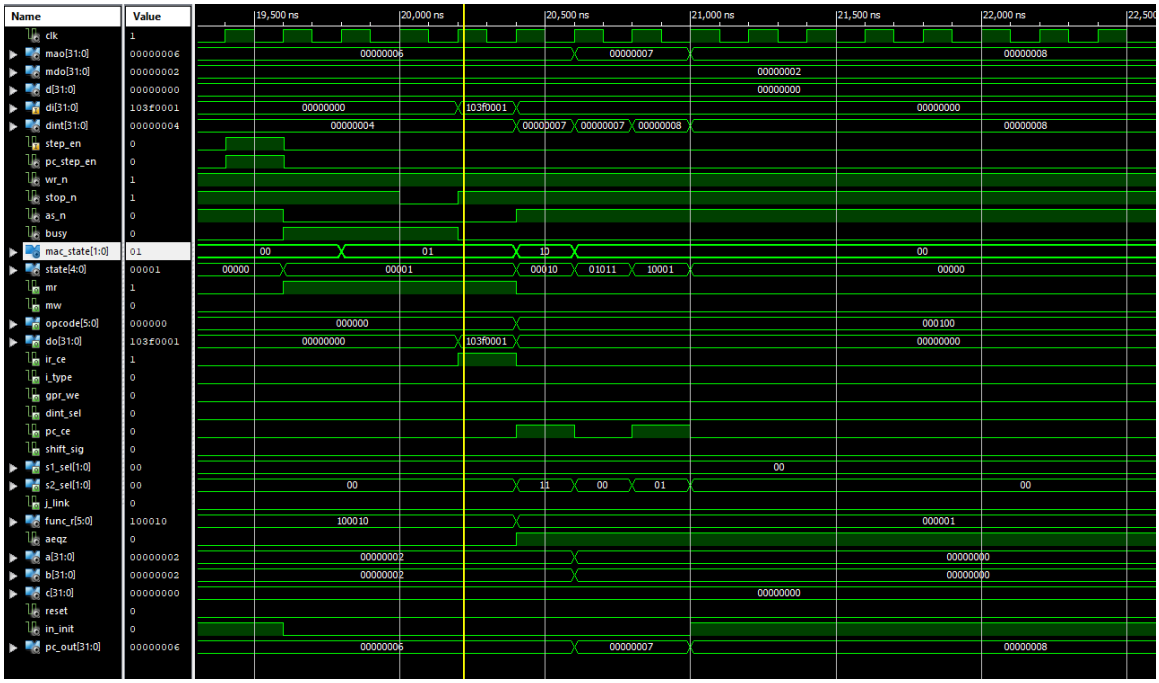
PC = 0x5 : sub R1 R3 R2 (R1 = R3 – R2 = 2 – 2 = 0)

States: INIT, FETCH, DECODE, ALU, WBR



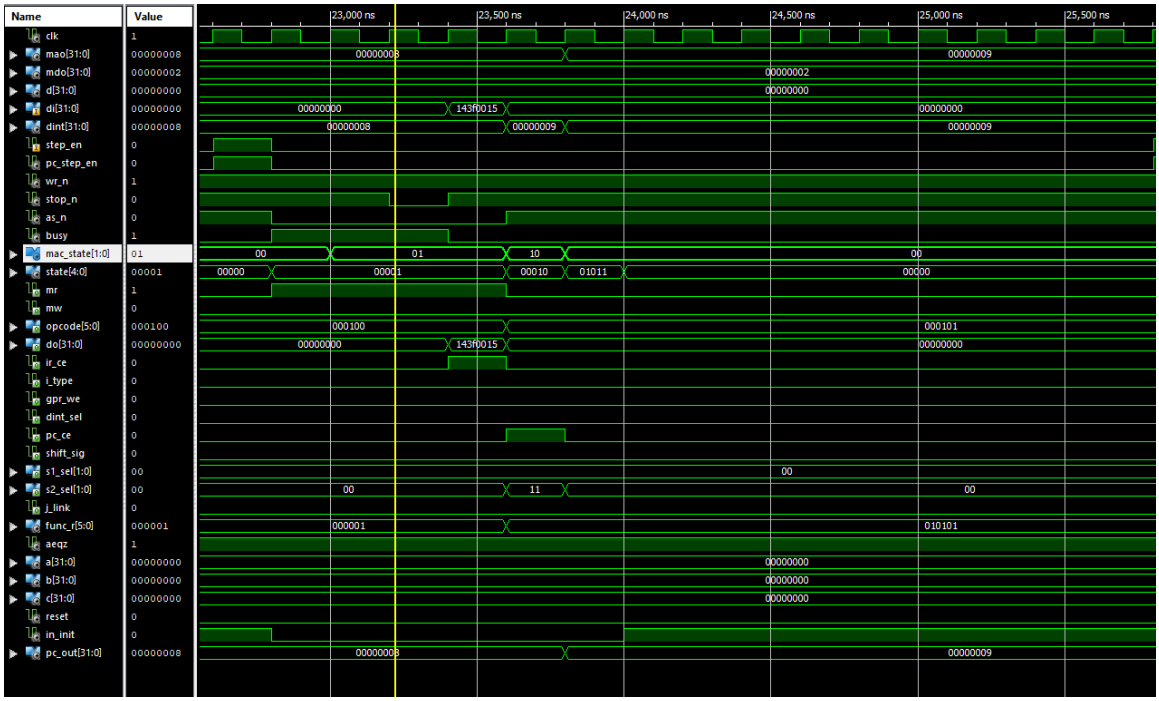
PC = 0x6 : beqz R1 BR1 (branch taken because R1 = 0)

States: INIT, FETCH, DECODE, BRANCH, BTAKEN



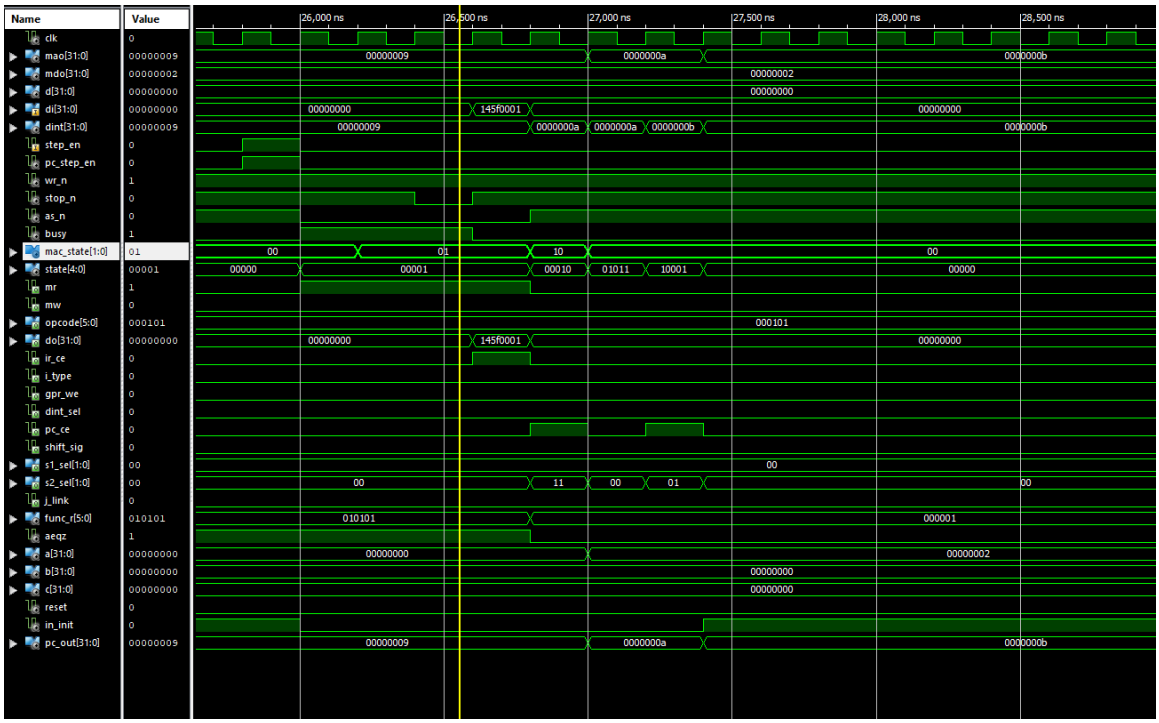
PC = 0x8 : bnez R1 BR2 (branch not taken because R1 = 0)

States: INIT, FETCH, DECODE, BRANCH



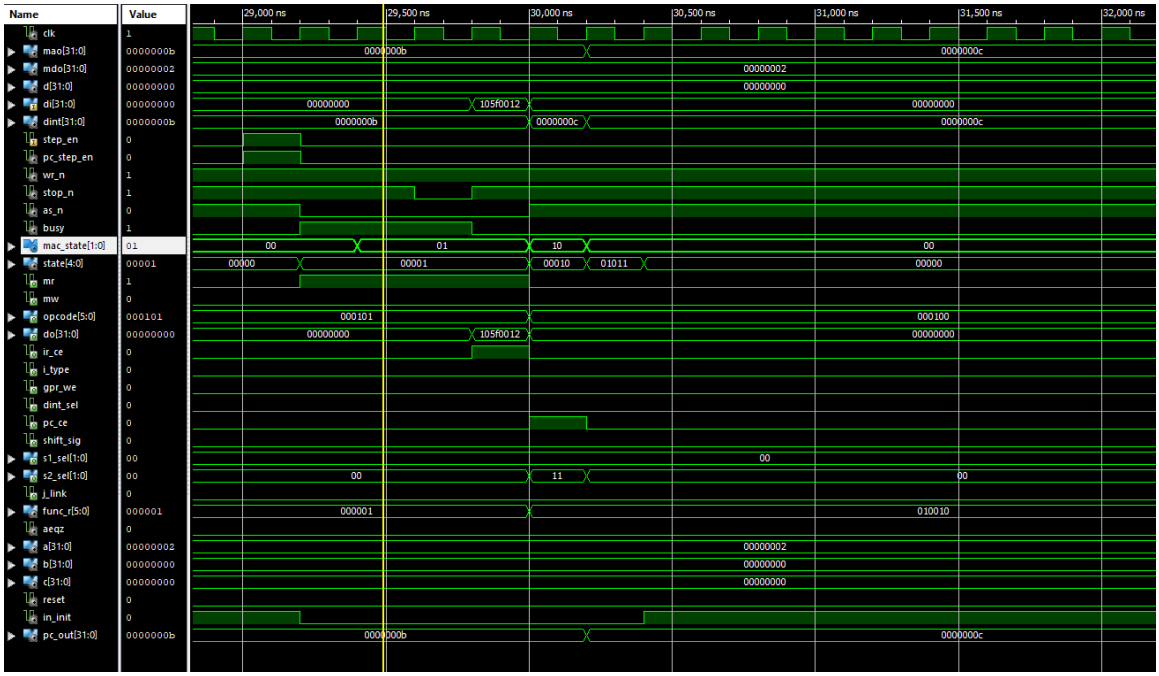
PC = 0x9 : bnez R2 BR3 (branch taken because R2 = 2 ≠ 0)

States: INIT, FETCH, DECODE, BRANCH, BTAKEN



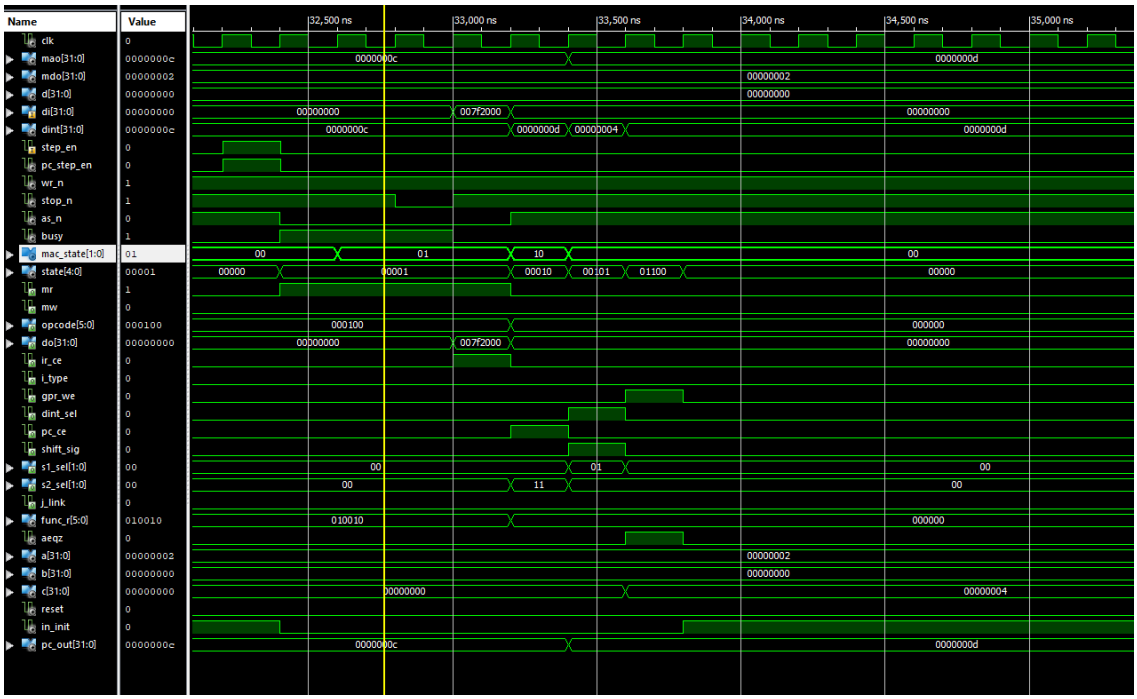
PC = 0xB : beqz R2 BR2 (branch not taken because R2 = 2 ≠ 0)

States: INIT, FETCH, DECODE, BRANCH



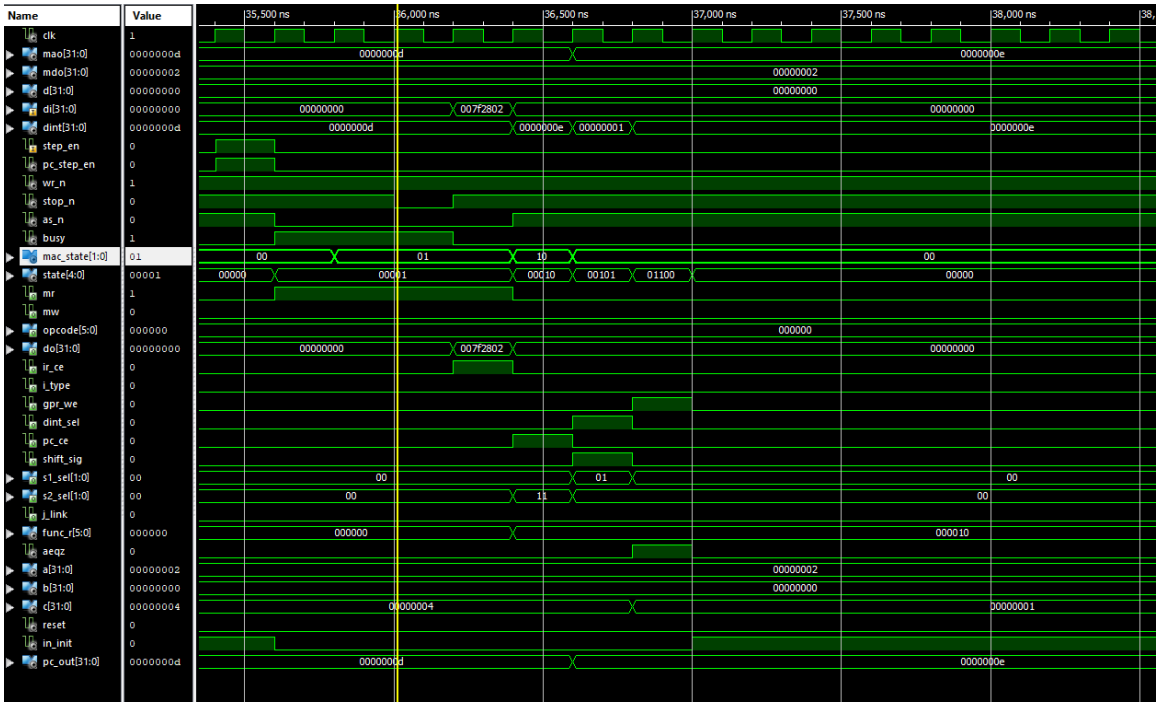
PC = 0xC : slli R4 R3 (R4 = R3 << 1 = 0x2 << 1 = 0x4)

States: INIT, FETCH, DECODE, SHIFT, WBR



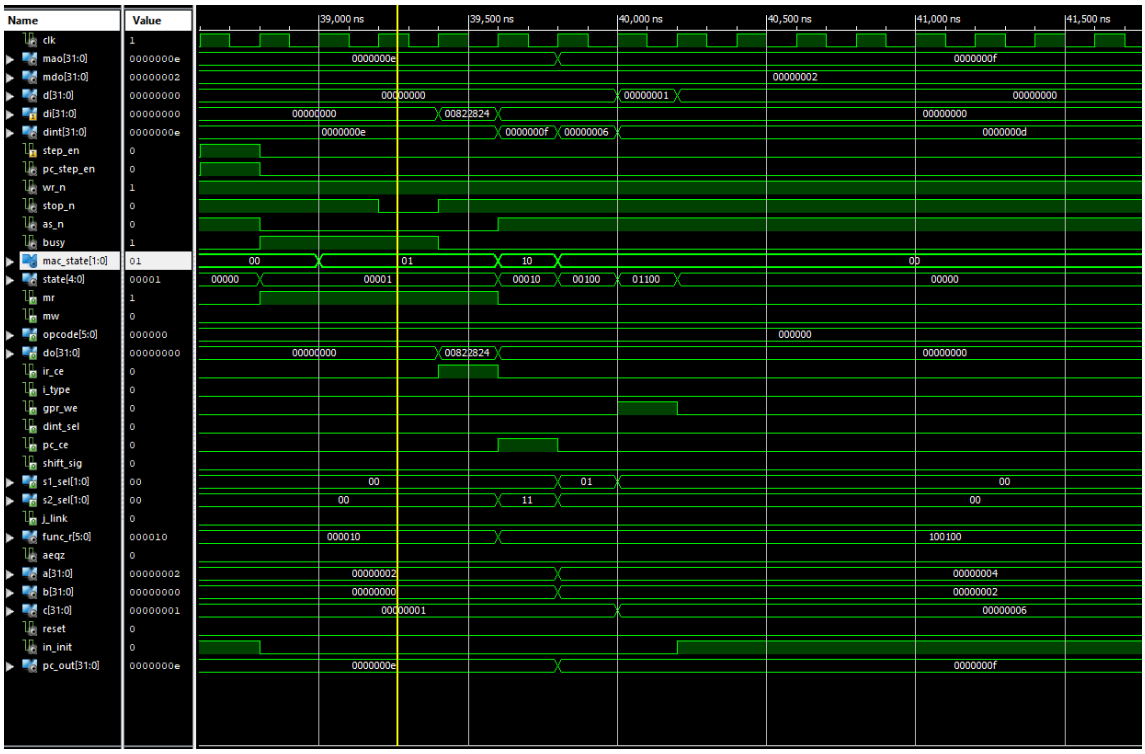
PC = 0xD : srli R5 R3 (R5 = R3 >> 1 = 0x2 >> 1 = 0x1)

States: INIT, FETCH, DECODE, SHIFT, WBR



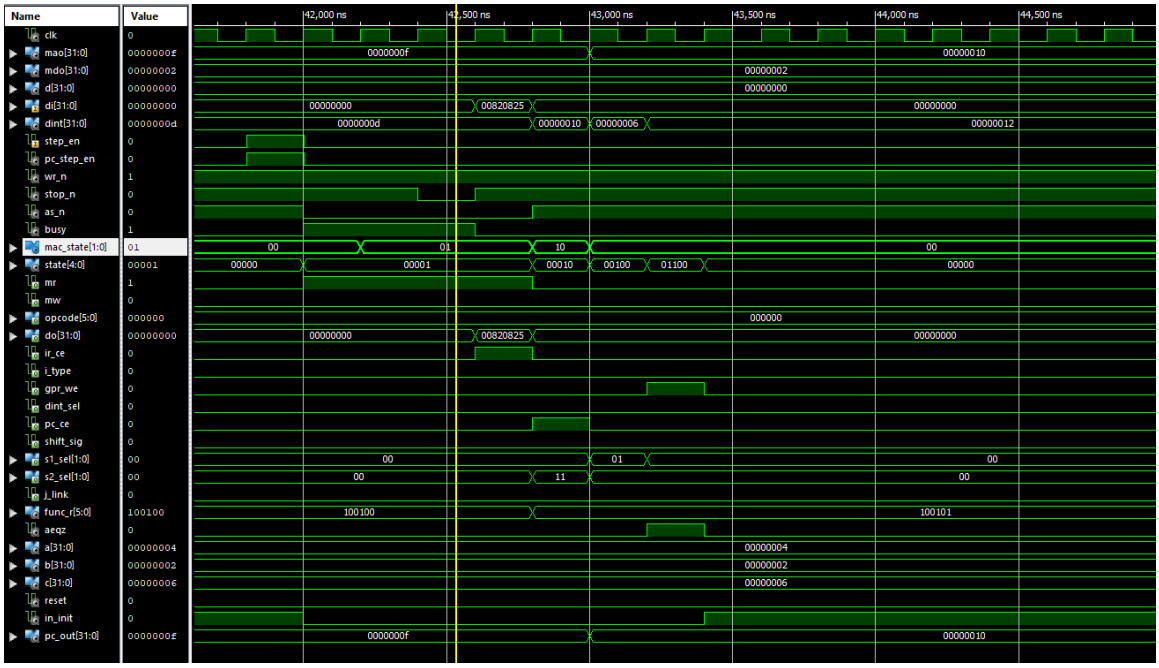
PC = 0xE : xor R5 R4 R2 (R5 = R4 xor R2 = 0x4 xor 0x2 = 0x6)

States: INIT, FETCH, DECODE, ALU, WBR



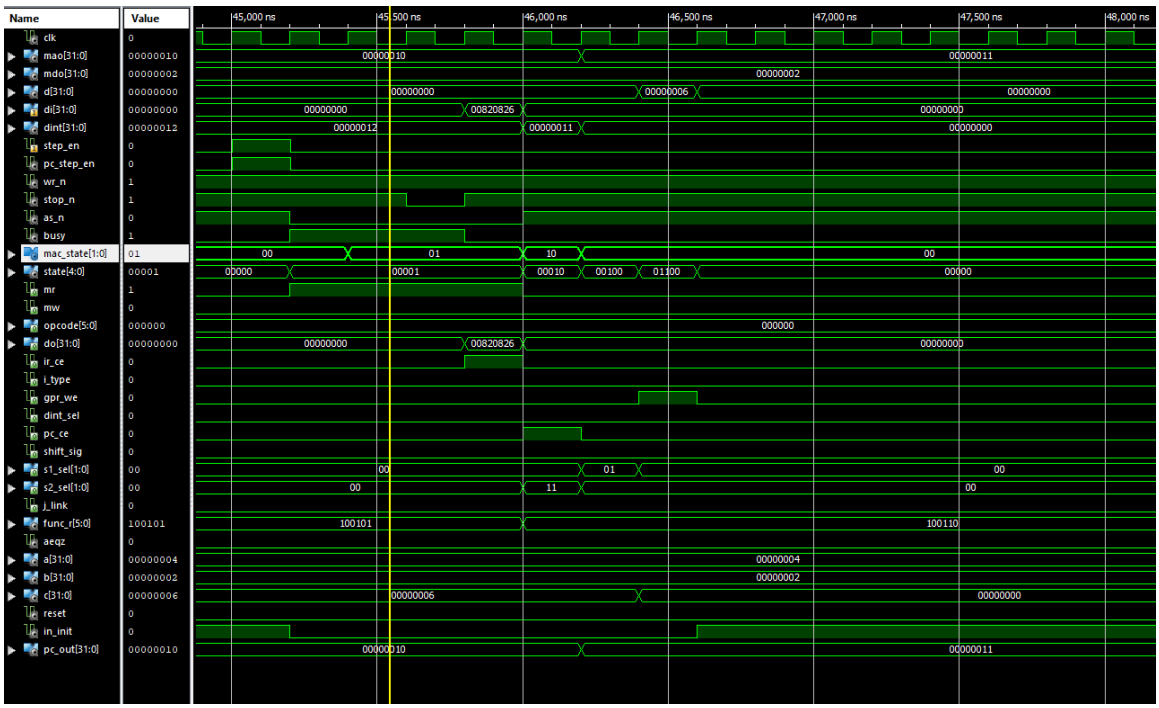
PC = 0xF : or R1 R4 R2 (R1 = R4 or R2 = 0x4 or 0x2 = 0x6)

States: INIT, FETCH, DECODE, ALU, WBR



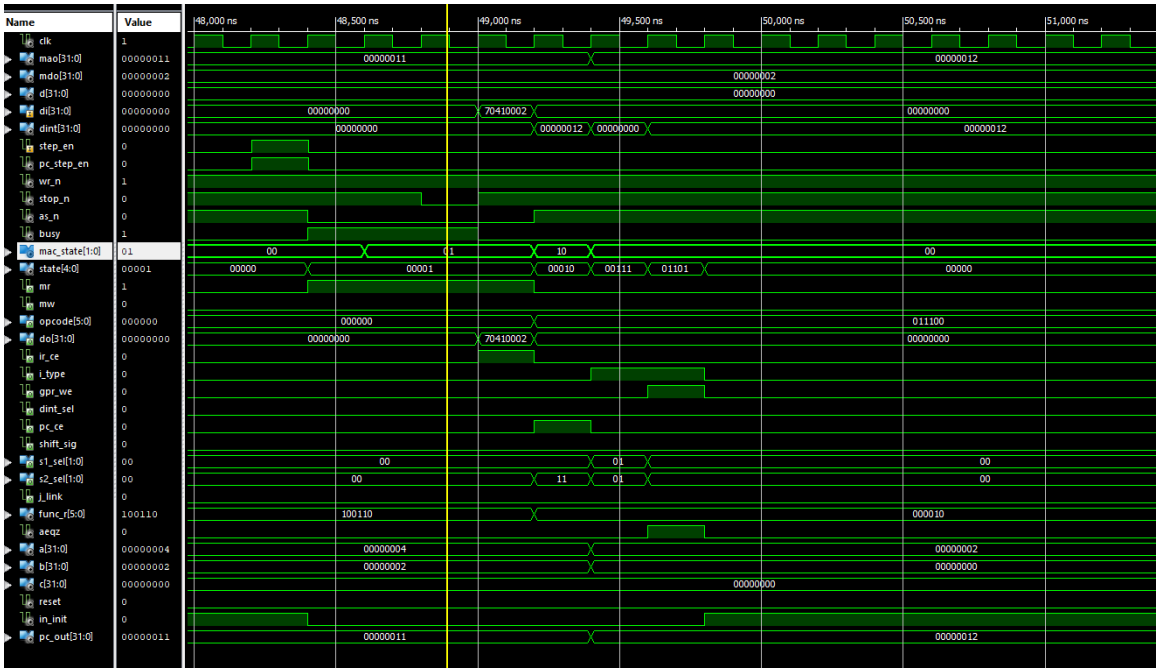
PC = 0x10 : and R1 R4 R2 (R1 = R4 and R2 = 0x4 and 0x2 = 0x0)

States: INIT, FETCH, DECODE, ALU, WBR



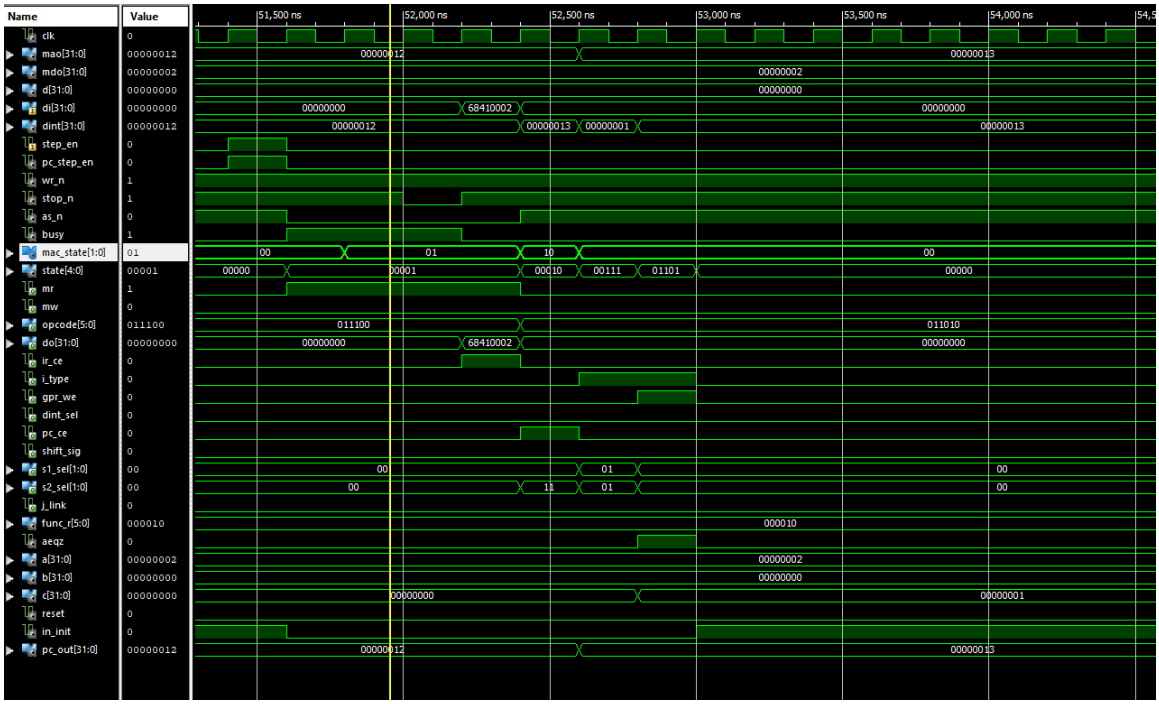
PC = 0x11 : slti R1 R2 0x2 (R1 = 0, because R2 = 2)

States: INIT, FETCH, DECODE, TESTI, WBI



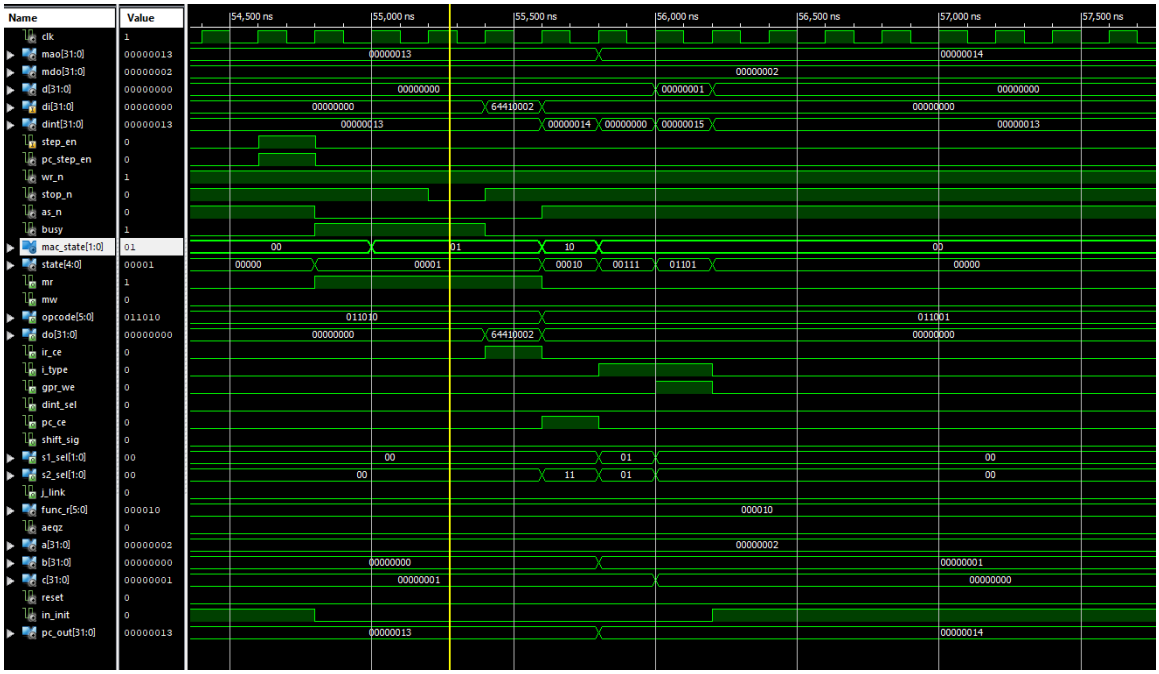
PC = 0x12 : seqi R1 R2 0x2 (R1 = 1, because R2 = 2)

States: INIT, FETCH, DECODE, TESTI, WBI



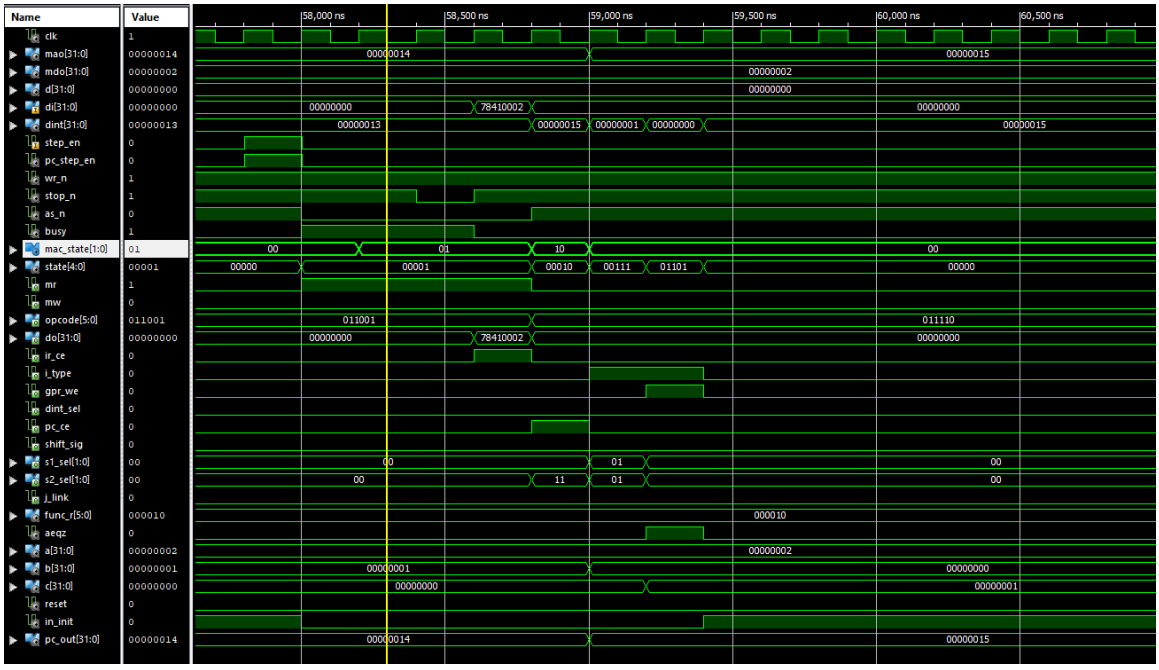
PC = 0x13 : sgti R1 R2 0x2 (R1 = 0, because R2 = 2)

States: INIT, FETCH, DECODE, TESTI, WBI



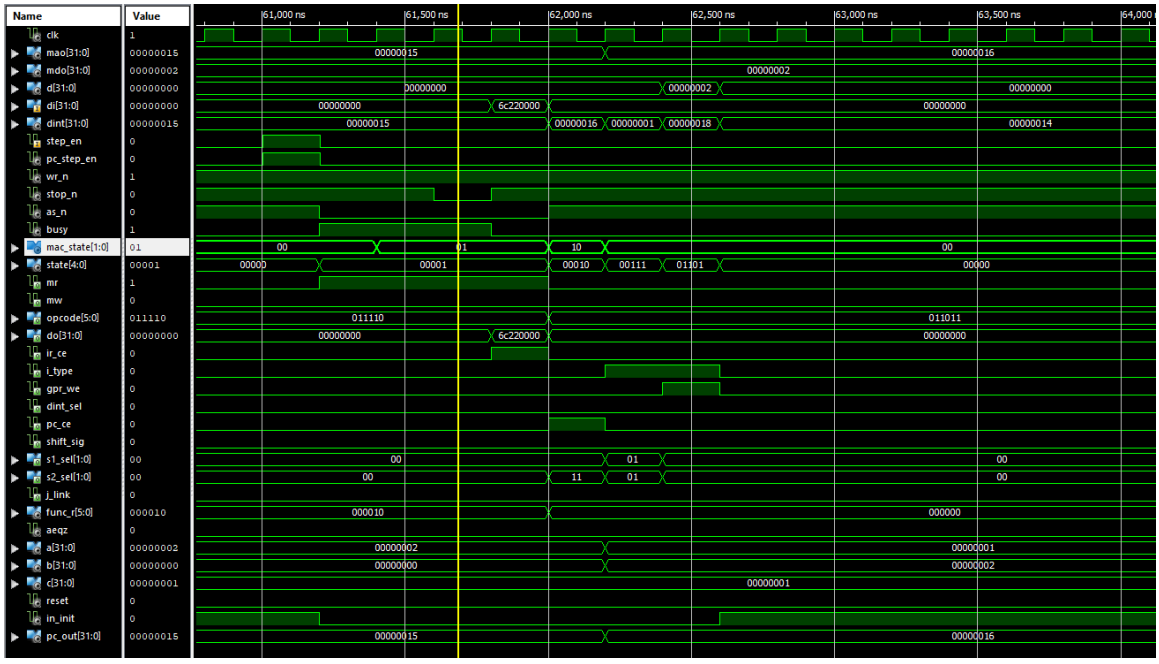
PC = 0x14 : slei R1 R2 0x2 (R1 = 1, because R2 = 2)

States: INIT, FETCH, DECODE, TESTI, WBI



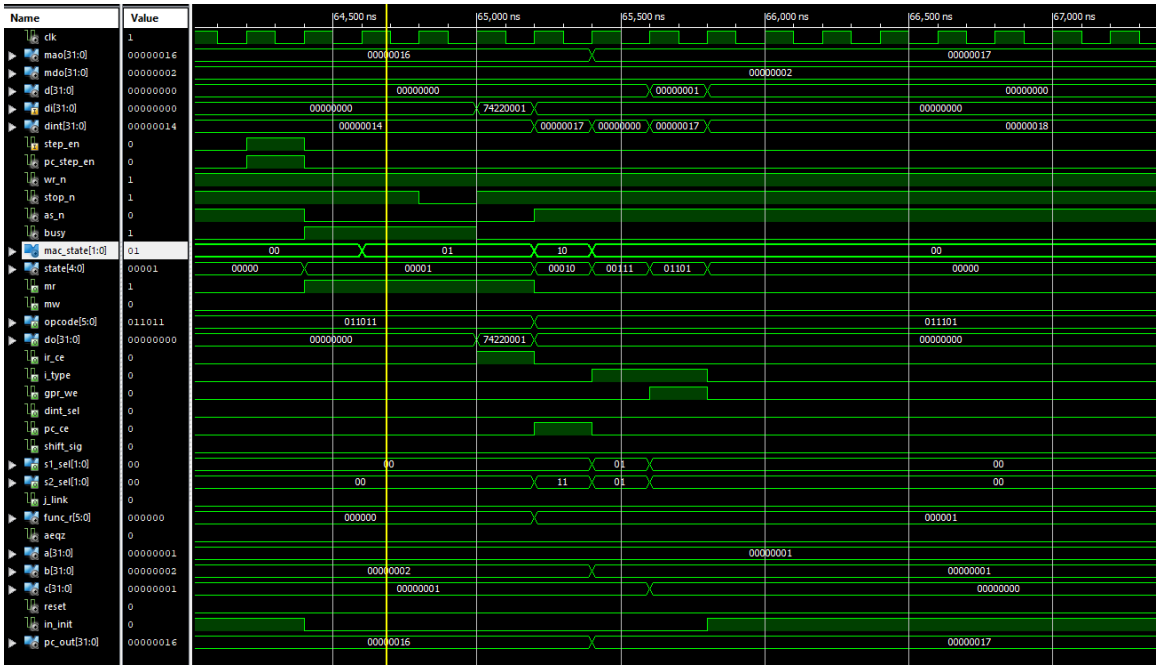
PC = 0x15 : sgei R2 R1 0x0 (R2 = 1, because R1 = 1 ≥ 0)

States: INIT, FETCH, DECODE, TESTI, WBI



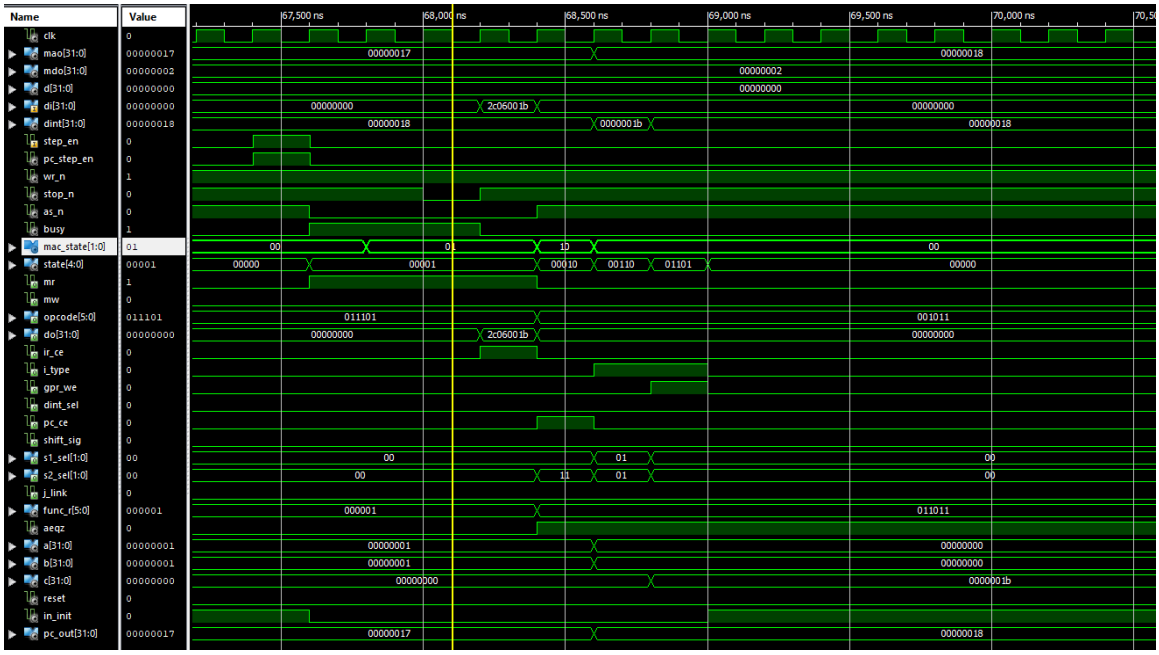
PC = 0x16 : snei R2 R1 0x1 (R2 = 0, because R1 = 1)

States: INIT, FETCH, DECODE, TESTI, WBI



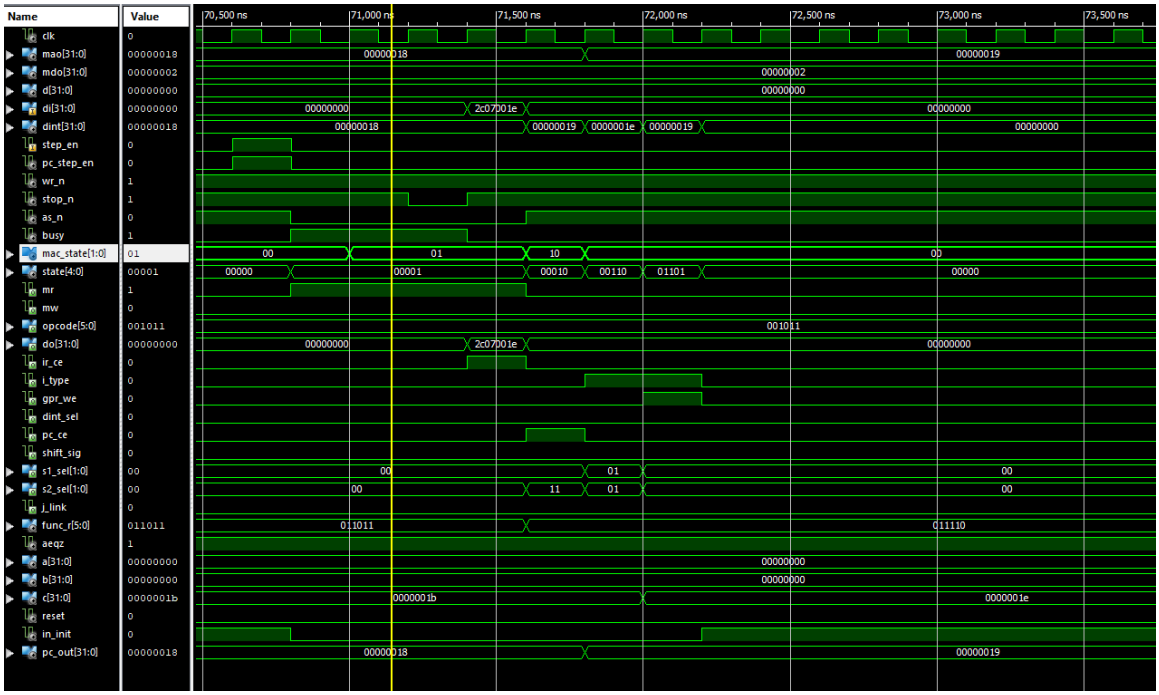
PC = 0x17 : addi R6 R0 0x1B (R6 = 0x1B)

States: INIT, FETCH, DECODE, ALUI, WBI



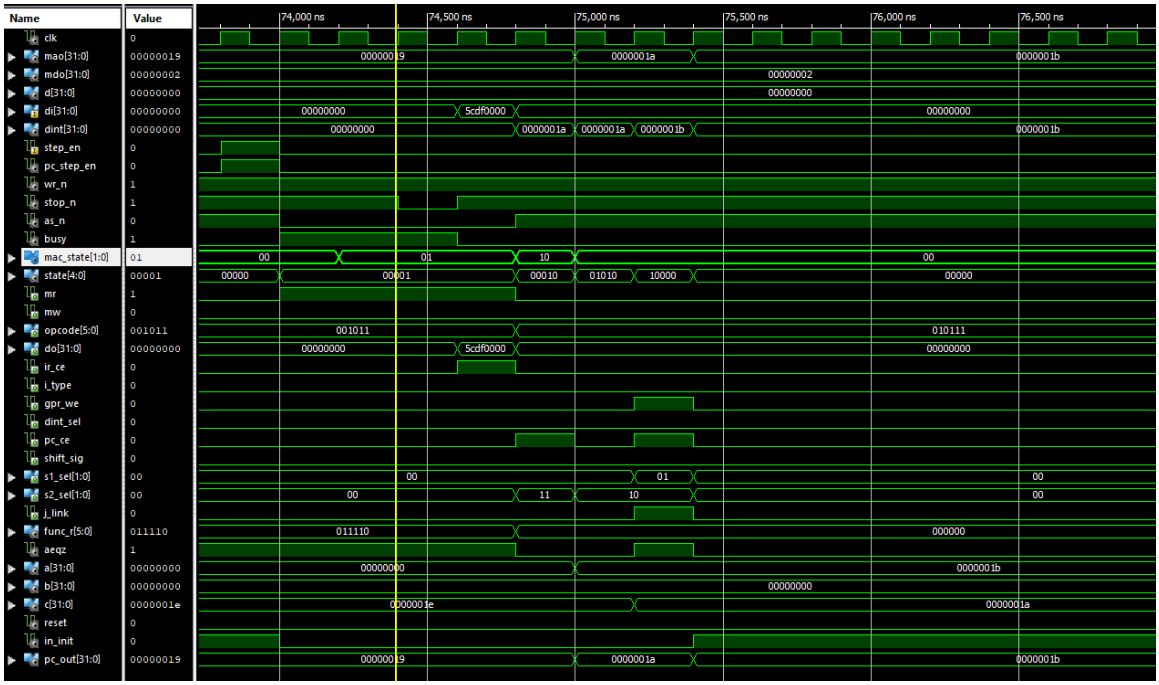
PC = 0x18 : addi R7 R0 0x1E (R7 = 0x1E)

States: INIT, FETCH, DECODE, ALUI, WBI



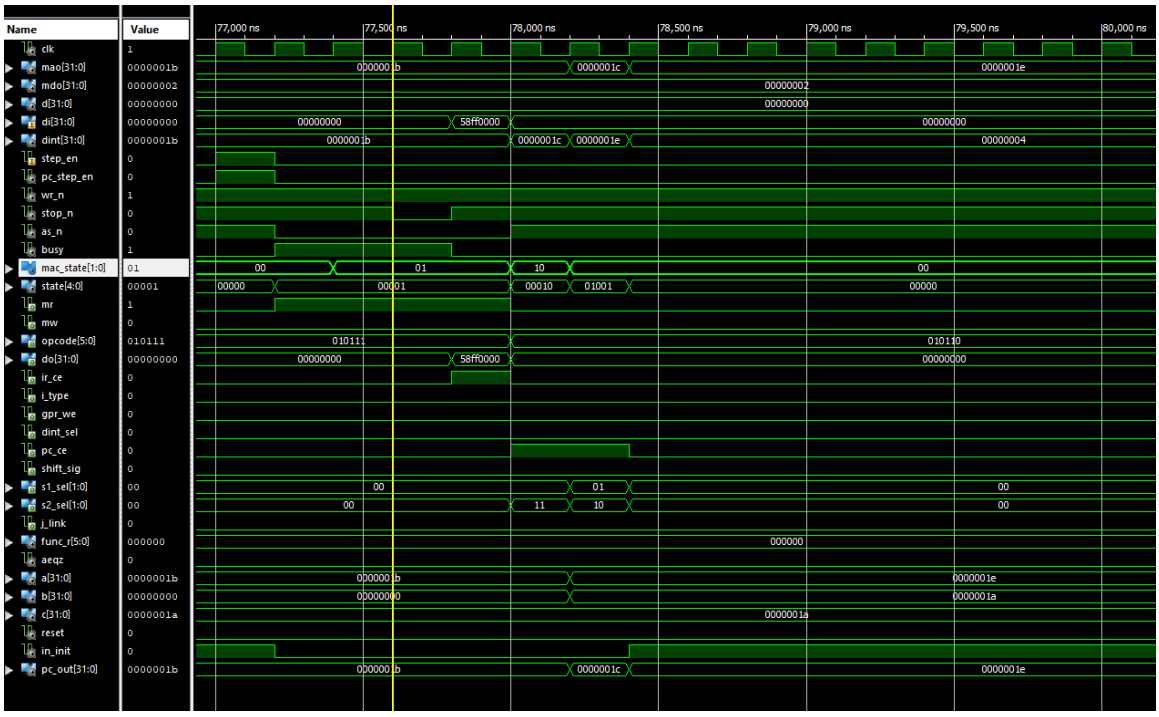
PC = 0x19 : jalr R6 (PC = R6 = 0x1B, R31 = 0x1A)

States: INIT, FETCH, DECODE, SAVEPC, JALR



PC = 0x1B : jr R7 (PC = R7 = 0x1E)

States: INIT, FETCH, DECODE, JR



PC = 0x1E : halt (finish program)

States: INIT, FETCH, DECODE, HALT