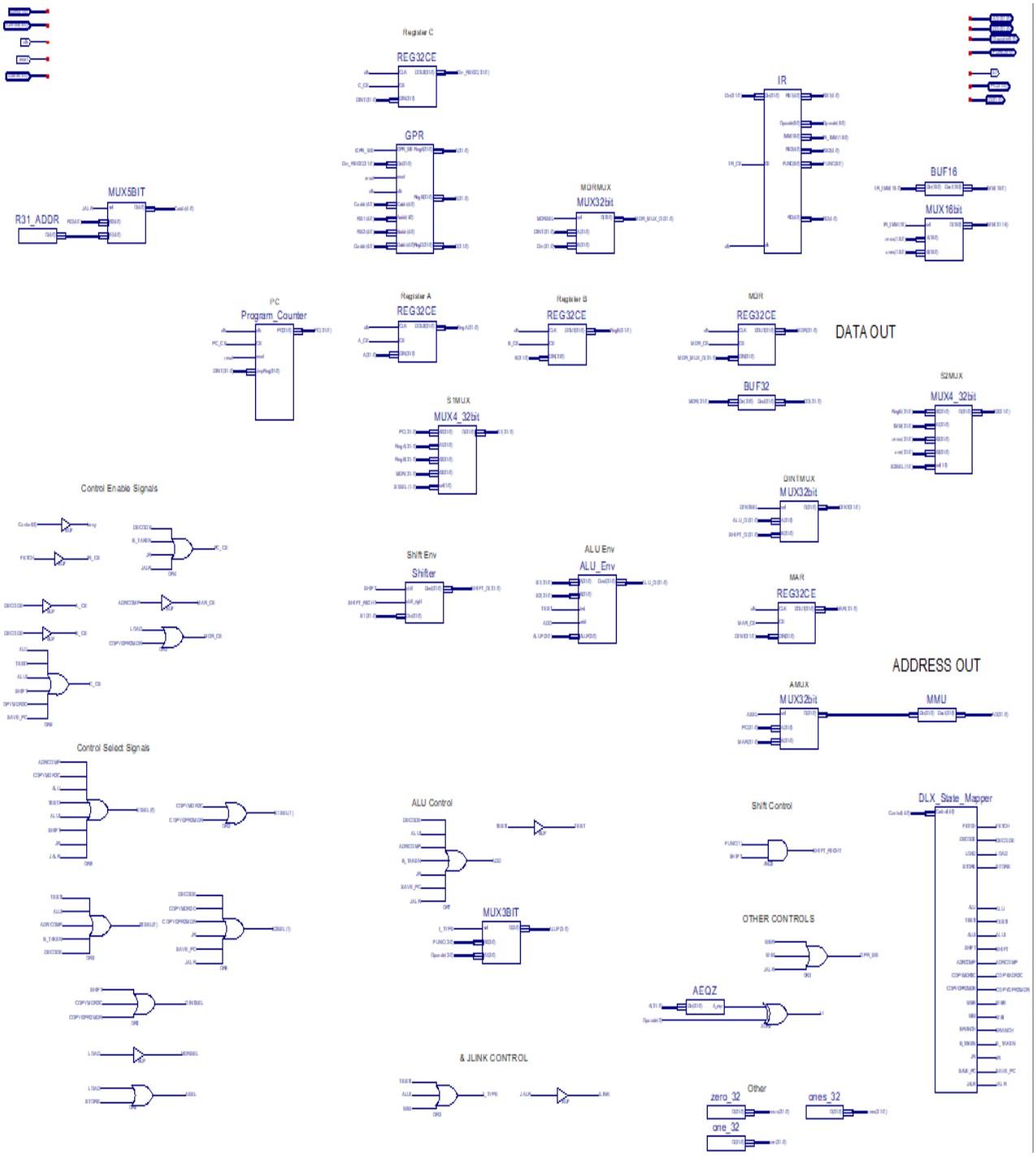
Handout #7: A simplified DLX: design, testing, timing, and programming

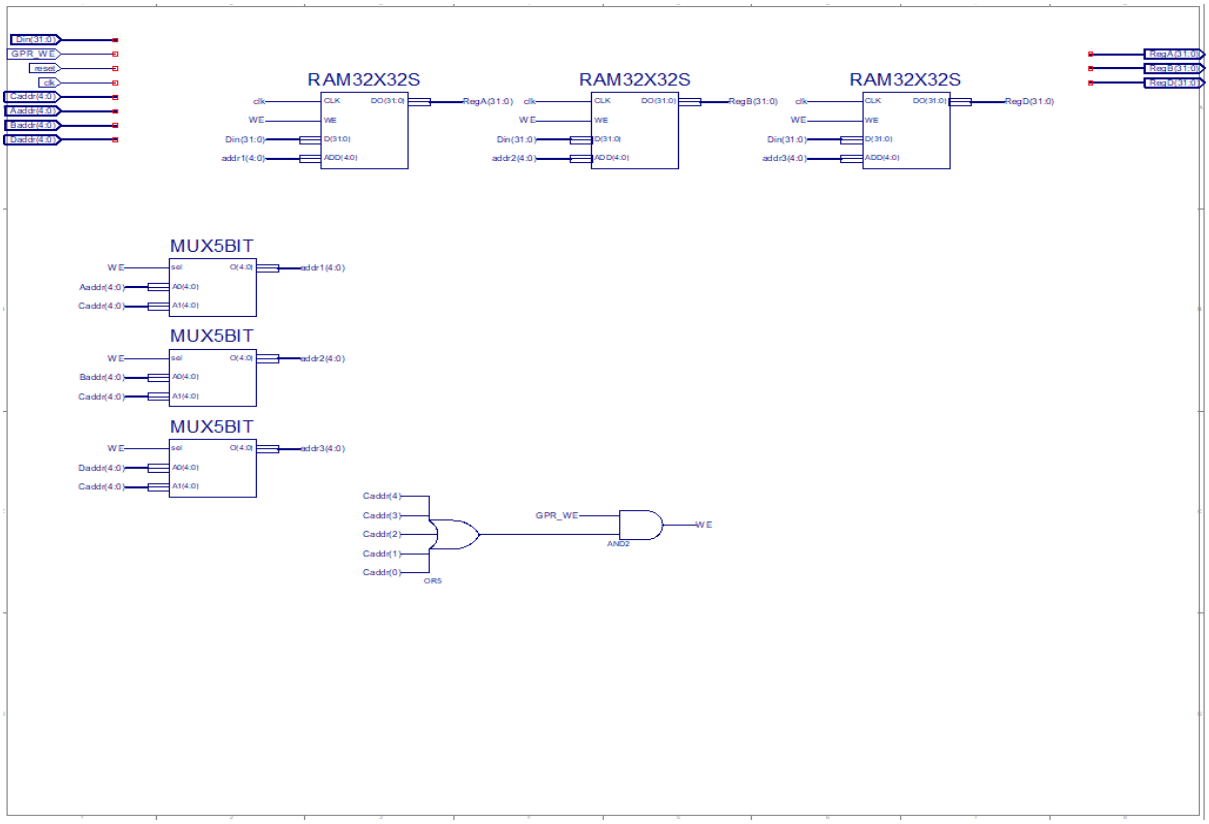
7.1. Design and test the simplified DLX

Part1 – DLX datapath:

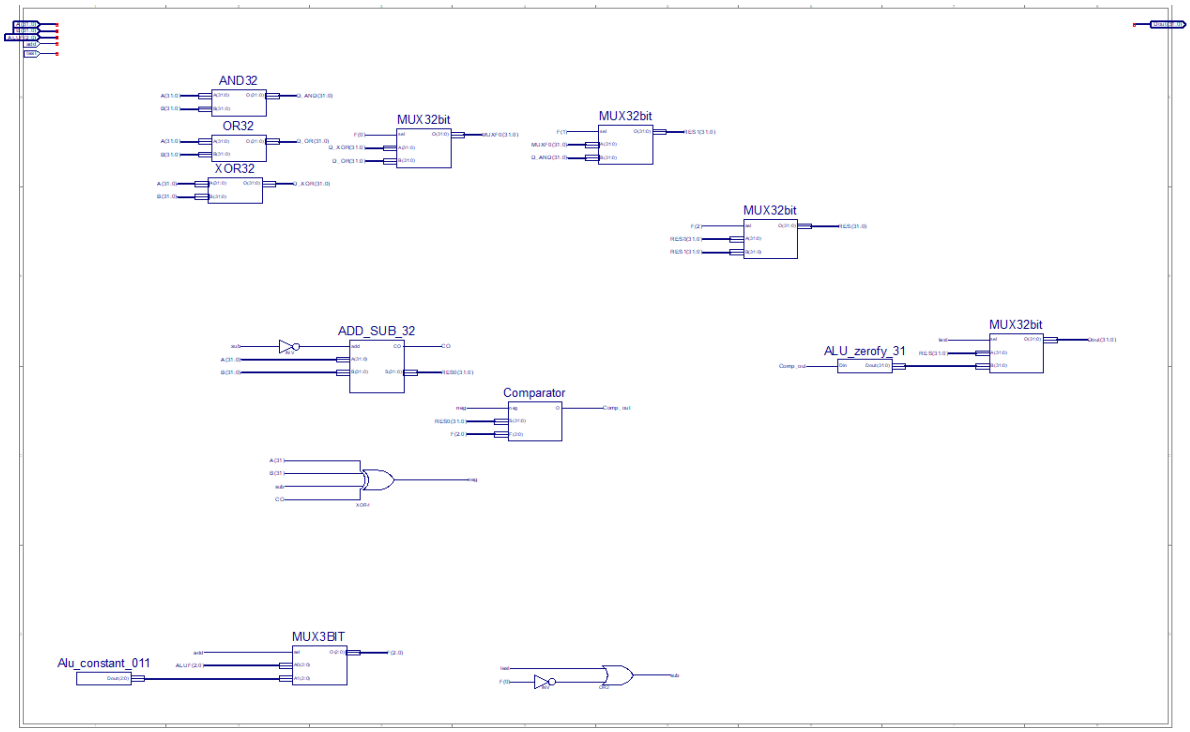
We designed the simplified DLX, and this is the datapath and the components inside it, and how they are built:



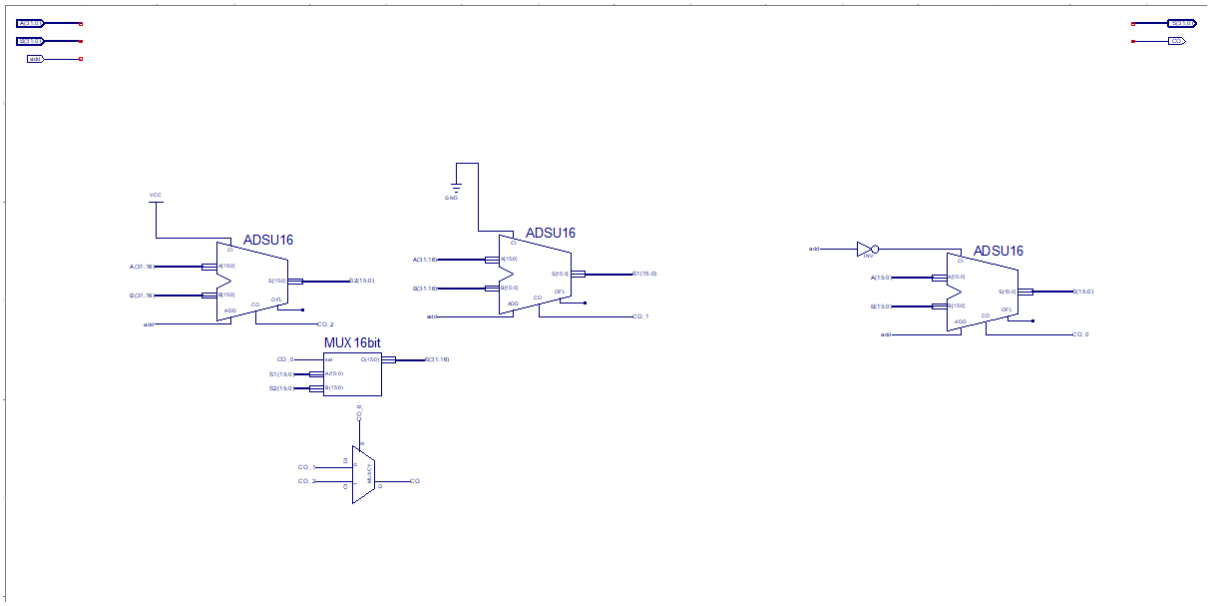
The GPR:



The ALU_ENV:



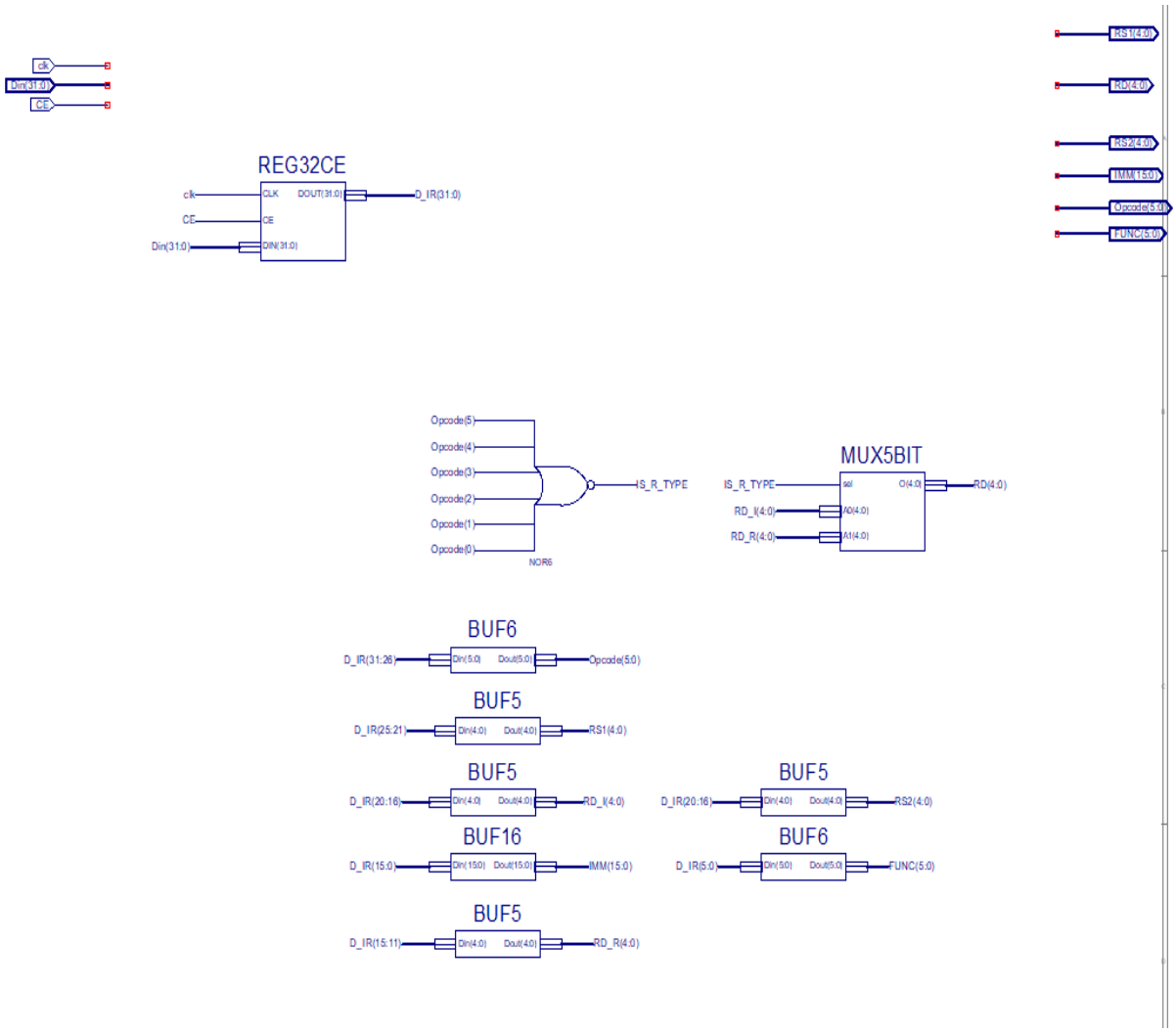
The ADD_SUB:



The comparator:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    13:51:03 06/15/2023
-- Design Name:
-- Module Name:    Comparator - 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 Comparator is
    Port ( S : in  STD_LOGIC_VECTOR (31 downto 0);
           neg : in  STD_LOGIC;
           F : in  STD_LOGIC_VECTOR (2 downto 0);
           O : out  STD_LOGIC);
end Comparator;

architecture Behavioral of Comparator is
signal bZero : STD_LOGIC;
signal op1 : STD_LOGIC;
signal op2 : STD_LOGIC;

begin

bZero <= '1' when (S=X"00000000") else '0';

op1 <= '1' when (bZero='0') and (neg='0' and F(0) = '1') else '0';

op2 <= '1' when ((F(2)='1' AND neg='1') OR (F(1)='1' AND bZero='1')) else '0';

O<= op1 or op2;
end Behavioral;
```

The IR environment:



the shifter:

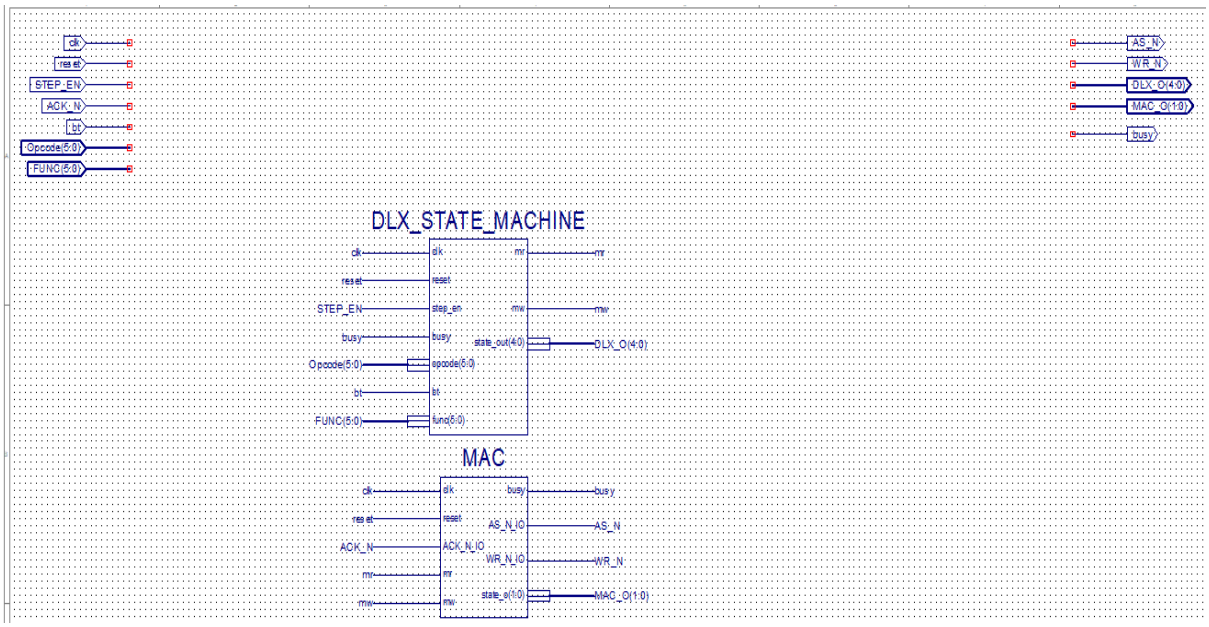
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    09:23:54 06/12/2023
-- Design Name:
-- Module Name:    Shifter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.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 Shifter is
    Port ( Din : in  STD_LOGIC_VECTOR (31 downto 0);
           shift : in  STD_LOGIC;
           shift_right : in  STD_LOGIC;
           Dout : out  STD_LOGIC_VECTOR (31 downto 0));
end Shifter;

architecture Behavioral of Shifter is
signal D : STD_LOGIC_VECTOR(31 downto 0);

begin

MAIN : PROCESS(Din,shift,shift_right)
begin
   if(shift='1') then
      if(shift_right='1') then
         D<= "0"&Din(31 downto 1);
      else
         D<= Din(30 downto 0) &"0";
      end if;
   else
      D<=Din;
   end if;

end PROCESS MAIN;

Dout <= D;

end Behavioral;
```

The DLX_state_mapper:

```
-- Company:
-- Engineer:
--
-- Create Date:    11:52:43 05/21/2023
-- Design Name:
-- Module Name:    DLX_State_Mapper - 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 DLX_State_Mapper is
    Port ( Control : in  STD_LOGIC_VECTOR (4 downto 0);
           FETCH : out STD_LOGIC;
           DECODE : out STD_LOGIC;
           ALU : out STD_LOGIC;
           TESTI : out STD_LOGIC;
           ALUI : out STD_LOGIC;
           SHIFT : out STD_LOGIC;
           ADRCOMP : out STD_LOGIC;
           LOAD : out STD_LOGIC;
           STORE : out STD_LOGIC;
           COPYMDR2C : out STD_LOGIC;
           COPYGPR2MDR : out STD_LOGIC;
           WBR : out STD_LOGIC;
           WBI : out STD_LOGIC;
           BRANCH : out STD_LOGIC;
           B_TAKEN : out STD_LOGIC;
           JR : out STD_LOGIC;
           SAVE_PC : out STD_LOGIC;
           JALR : out STD_LOGIC
           );
```

```
53  end DLX_State_Mapper;
54
55  architecture Behavioral of DLX_State_Mapper is
56     -- states
57     constant FETCH_S : std_logic_vector(4 downto 0) := "00001";
58     constant DECODE_S : std_logic_vector(4 downto 0) := "00010";
59     constant ALU_S : std_logic_vector(4 downto 0) := "00011";
60     constant TESTI_S : std_logic_vector(4 downto 0) := "00100";
61     constant ALUI_S : std_logic_vector(4 downto 0) := "00101";
62     constant SHIFT_S : std_logic_vector(4 downto 0) := "00110";
63     constant ADRCOMP_S : std_logic_vector(4 downto 0) := "00111";
64     constant LOAD_S : std_logic_vector(4 downto 0) := "01000";
65     constant STORE_S : std_logic_vector(4 downto 0) := "01001";
66     constant COPYMDR2C_S : std_logic_vector(4 downto 0) := "01010";
67     constant COPYGPR2MDR_S : std_logic_vector(4 downto 0) := "01011";
68     constant WBR_S : std_logic_vector(4 downto 0) := "01100";
69     constant WBI_S : std_logic_vector(4 downto 0) := "01101";
70     constant BRANCH_S : std_logic_vector(4 downto 0) := "01110";
71     constant B_TAKEN_S  : std_logic_vector(4 downto 0) := "01111";
72     constant JR_S : std_logic_vector(4 downto 0) := "10000";
73     constant SAVE_PC_S : std_logic_vector(4 downto 0) := "10001";
74     constant JALR_S : std_logic_vector(4 downto 0) := "10010";
75
76  begin
77     FETCH <= '1' when (Control=FETCH_S) else '0';
78     DECODE <= '1' when (Control=DECODE_S) else '0';
79     ALU <= '1' when (Control=ALU_S) else '0';
80     TESTI <= '1' when (Control=TESTI_S) else '0';
81     ALUI <= '1' when (Control=ALUI_S) else '0';
82     SHIFT <= '1' when (Control=SHIFT_S) else '0';
83     ADRCOMP <= '1' when (Control=ADRCOMP_S) else '0';
84     LOAD <= '1' when (Control=LOAD_S) else '0';
85     STORE <= '1' when (Control=STORE_S) else '0';
86     COPYMDR2C <= '1' when (Control=COPYMDR2C_S) else '0';
87     COPYGPR2MDR <= '1' when (Control=COPYGPR2MDR_S) else '0';
88     WBR <= '1' when (Control=WBR_S) else '0';
89     WBI <= '1' when (Control=WBI_S) else '0';
90     BRANCH <= '1' when (Control=BRANCH_S) else '0';
91     B_TAKEN <= '1' when (Control=B_TAKEN_S) else '0';
92     JR <= '1' when (Control=JR_S) else '0';
93     SAVE_PC <= '1' when (Control=SAVE_PC_S) else '0';
94     JALR <= '1' when (Control=JALR_S) else '0';
95
96  end Behavioral;
97
98
```

Part2– DLX control:



The MAC module (VHDL code below) is exactly the same as the one in handout 6. The control module (VHDL code attached below) implements the state machine explained in the recitation. The above schematic is the final DLX_control module which simply produced the states (MAC and DLX), where we then use them in the Data Path module to process control signals.

control_t (the test bench of the control module):

The test was carried out by changing the incoming signals of the control module:

1. opcode(5:0)

2. func(5:0)

3. ACK_N

4. step_en

5. reset

6. bt

Our test vectors :

	opcode	func
ALU	000000	100000
TESTI	011000	000000
LOAD	100011	000000
STORE	100111	000000
JALR	010111	000000
BTAKEN(bt=1)	000100	000000
BTAKEN(bt=0)	000100	000000

Our DLX_STATE_MACHINE module (in VHDL) that we wrote was :

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    10:15:40 05/15/2023
-- Design Name:
-- Module Name:    DLX_STATE_MACHINE - 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 DLX_STATE_MACHINE is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           step_en : in  STD_LOGIC;
           busy : in  STD_LOGIC;
           opcode : in  STD_LOGIC_VECTOR (5 downto 0);
           func : in STD_LOGIC_VECTOR (5 downto 0);
           bt : in   STD_LOGIC;
           mr : out  STD_LOGIC;
           mw : out  STD_LOGIC;
           state_out : out  STD_LOGIC_VECTOR (4 downto 0));
end DLX_STATE_MACHINE;

architecture Behavioral of DLX_STATE_MACHINE is
```

```
46
47  signal state : std_logic_vector(4 downto 0);
48  signal opcode4 : std_logic_vector(3 downto 0);
49  signal opcode3 : std_logic_vector(2 downto 0);
50  signal opcode2 : std_logic_vector(1 downto 0);
51
52     -- states
53     constant INIT : std_logic_vector(4 downto 0)       := "00000";
54     constant FETCH : std_logic_vector(4 downto 0)      := "00001";
55     constant DECODE : std_logic_vector(4 downto 0)     := "00010";
56     constant ALU : std_logic_vector(4 downto 0)        := "00011";
57     constant TESTI : std_logic_vector(4 downto 0)      := "00100";
58     constant ALUI : std_logic_vector(4 downto 0)       := "00101";
59     constant SHIFT : std_logic_vector(4 downto 0)      := "00110";
60     constant ADRCOMP : std_logic_vector(4 downto 0)    := "00111";
61     constant LOAD : std_logic_vector(4 downto 0)       := "01000";
62     constant STORE : std_logic_vector(4 downto 0)      := "01001";
63     constant COPYMDR2C : std_logic_vector(4 downto 0) := "01010";
64     constant COPYGPR2MDR : std_logic_vector(4 downto 0) := "01011";
65     constant WBR : std_logic_vector(4 downto 0) := "01100";
66     constant WBI : std_logic_vector(4 downto 0) := "01101";
67     constant BRANCH : std_logic_vector(4 downto 0) := "01110";
68     constant B_TAKEN  : std_logic_vector(4 downto 0) := "01111";
69     constant JR : std_logic_vector(4 downto 0) := "10000";
70     constant SAVE_PC : std_logic_vector(4 downto 0) := "10001";
71     constant JALR : std_logic_vector(4 downto 0) := "10010";
72     constant HALT : std_logic_vector(4 downto 0) := "10011";
73  -- opcodes
74     --DATA TRANSFER
75     constant LW_OPCODE : std_logic_vector(5 downto 0)    := "100011";
76     constant SW_OPCODE : std_logic_vector(5 downto 0)    := "101011";
77     --ARITHMETIC
78     constant ADDI_OPCODE : std_logic_vector(5 downto 0) := "001011";
79     --TEST
80     constant SGTI_OPCODE : std_logic_vector(5 downto 0) := "011001";
81     constant SEQI_OPCODE : std_logic_vector(5 downto 0) := "011010";
82     constant SGEI_OPCODE : std_logic_vector(5 downto 0) := "011011";
83     constant SLTI_OPCODE : std_logic_vector(5 downto 0) := "011100";
84     constant SNEI_OPCODE : std_logic_vector(5 downto 0) := "011101";
85     constant SLEI_OPCODE : std_logic_vector(5 downto 0) := "011110";
86     --CONTROL
87     constant BEQZ_OPCODE : std_logic_vector(5 downto 0) := "000100";
88     constant BNEZ_OPCODE : std_logic_vector(5 downto 0) := "000101";
89     constant JR_OPCODE : std_logic_vector(5 downto 0)    := "010110";
90     constant JALR_OPCODE : std_logic_vector(5 downto 0) := "010111";
91     --MISC
```

```
   constant NOP_OPCODE_FORMAT : std_logic_vector(2 downto 0)  := "110";
   constant HALT_OPCODE : std_logic_vector(5 downto 0) := "111111";
   --SHIFT
   constant SLLI_OPCODE : std_logic_vector(5 downto 0) := "000000";
   constant SRLI_OPCODE : std_logic_vector(5 downto 0) := "000010";
   --ARITHEMTIC
   constant D1 : STD_LOGIC_VECTOR(2 downto 0) := "110";
   constant D2 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
   constant D4 : STD_LOGIC_VECTOR(3 downto 0) := "0000";
   constant D5 : STD_LOGIC_VECTOR(2 downto 0) := "001";
   constant D6 : STD_LOGIC_VECTOR(2 downto 0) := "011";
   constant D7 : STD_LOGIC_VECTOR(1 downto 0) := "10";
   constant D8 : STD_LOGIC_VECTOR(2 downto 0) := "010";
   constant D9 : STD_LOGIC_VECTOR(2 downto 0) := "010";
   constant D8_last : STD_LOGIC := '1';
   constant D9_last : STD_LOGIC := '0';
   constant D12 : STD_LOGIC_VECTOR(3 downto 0) := "0001";
   constant D13 : STD_LOGIC := '1'; -- index 2

-- FUNCS
   constant D2_FUNC : std_logic := '1'; -- D2
   constant D4_FUNC : std_logic := '0'; -- D4


begin


STATE_PROC : PROCESS(clk, reset, step_en, bt)
begin

   if(reset='1') then state<=INIT;
   elsif(clk'event and clk='1') then
      case state is
         when INIT =>
            if(step_en='1') then state<=FETCH;
            end if;
         when FETCH =>
            --get instr from IR
            if(busy='0') then state<=DECODE;
            end if;
         when DECODE =>
            --decode
            -- DATA TRANSFER
```

```
135             if(opcode3=D1) then
136                if(step_en='0') then state<=INIT;
137                else state<=FETCH;
138                end if;
139             elsif(opcode4=D2 and func(5)=D2_FUNC) then state<=ALU;
140             elsif(opcode4=D4 and func(5)=D4_FUNC)  then state<=SHIFT;
141             elsif(opcode3=D5)  then state<=ALUI;
142             elsif(opcode3=D6)  then state<=TESTI;
143             elsif(opcode2=D7)  then state<=ADRCOMP;
144             elsif(opcode3=D8 and opcode(0)='0')  then state<=JR;
145             elsif(opcode3=D9 and opcode(0)='1')  then state<=SAVE_PC;
146             elsif(opcode4=D12)  then state<=BRANCH;
147             --elsif(opcode(3)=D13)  then state<=COPYGPR2MDR;
148             --elsif(opcode(3)/=D13) then state<=LOAD;
149             else state<=HALT;
150             end if;
151          when LOAD =>
152             if(busy='0') then state<=COPYMDR2C;
153             end if;
154          when STORE =>
155             if(busy='0') then
156                --if(step_en='0') then state<=INIT;
157                --else state<=FETCH;
158                state<=INIT;
159                --end if;
160             end if;
161          when HALT=>
162             if(reset='1') then state<=INIT;
163             end if;
164          -- new states
165          when SHIFT =>
166             if(busy='0') then state<=WBR;
167             end if;
168          when ALU =>
169             if(busy='0') then state<=WBR;
170             end if;
171          -- TO BE IMPLEMENTED!
172          when TESTI =>
173             if(busy='0') then state<=WBI;
174             end if;
175          when ALUI =>
176             if(busy='0') then state<=WBI;
177             end if;
178
179          when ADRCOMP =>
180             if(opcode(3)=D13)  then state<=COPYGPR2MDR;
181             else state<=LOAD;
182             end if;
```

```
183           when COPYMDR2C =>
184              if(busy='0') then state<=WBI;
185              end if;
186
187           when COPYGPR2MDR =>
188              state<=STORE;
189
190           when WBR =>
191              if(step_en='0') then state<=INIT;
192              else state<=FETCH;
193              end if;
194           when WBI =>
195              if(step_en='0') then state<=INIT;
196              else state<=FETCH;
197              end if;
198           when BRANCH =>
199              if(bt='0') then state<=INIT;
200              else state<=B_TAKEN;
201              end if;
202           when B_TAKEN =>
203              if(step_en='0') then state<=INIT;
204              else state<=FETCH;
205              end if;
206           when JR  =>
207              if(step_en='0') then state<=INIT;
208              else state<=FETCH;
209              end if;
210           when SAVE_PC =>
211              if(busy='0') then state<=JALR;
212              else state<=FETCH;
213              end if;
214           when JALR =>
215              if(step_en='0') then state<=INIT;
216              else state<=FETCH;
217              end if;
218
219           when others => null;
220        end case;
221     end if;
222
223  end PROCESS STATE_PROC;
224
225     mw <= '1' when (state=STORE) else '0';
226     mr <= '1' when (state=FETCH) or (state=LOAD) else '0';
227     state_out <=state;
228
229     -- opcode shortucts
230     opcode4 <= opcode(5 downto 2);
231     opcode3 <= opcode(5 downto 3);
232     opcode2 <= opcode(5 downto 4);
233  end Behavioral;
234
```

Our Mac module that we designed (in VHDL) was :

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:59:46 05/20/2023
-- Design Name:
-- Module Name:    MAC - 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 MAC is
    Port (  clk : in STD_LOGIC;
            reset : in STD_LOGIC;
            ACK_N_IO : in  STD_LOGIC;
           mr : in  STD_LOGIC;
           mw : in  STD_LOGIC;
           busy : out  STD_LOGIC;
           AS_N_IO : out  STD_LOGIC;
           WR_N_IO : out  STD_LOGIC;
           state_o : out  STD_LOGIC_VECTOR (1 downto 0));
end MAC;

architecture Behavioral of MAC is

signal state : std_logic_vector(1 downto 0);
```

```
-- states
constant WAIT4REQ : std_logic_vector(1 downto 0) := "00";
constant WAIT4ACK : std_logic_vector(1 downto 0) := "01";
constant STATE_NEXT : std_logic_vector(1 downto 0) := "10";

begin

STATE_PROC : PROCESS(clk,reset)
begin
   if(reset='1') then state<=WAIT4REQ;
   elsif(clk'event and clk='1') then
      case state is
         when WAIT4REQ =>
            if (mw='1' or mr ='1') then --i.e. req=OR(mw,mr)
               state<=WAIT4ACK;
            end if;
         when WAIT4ACK =>
            if (ACK_N_IO = '0') then
               state<=STATE_NEXT;
            end if;
         when STATE_NEXT =>
            state<=WAIT4REQ;
         when others=>NULL;
      end case;
   end if;
end PROCESS STATE_PROC;


   -- more logic here
   state_o <= state;
   busy <= '1' when ((ACK_N_IO ='1') and ((state=WAIT4ACK) or (state=WAIT4REQ)) and (mw='1' or mr='1')) else '0';
   AS_N_IO <= '0' when (state=WAIT4ACK) else '1';
   WR_N_IO <= '0' when (state=WAIT4ACK and mw='1') else '1';

end Behavioral;
```

Our test bench that we wrote was :

```
1  -- Vhdl test bench created from schematic E:\adlx\Semester_B\Nizar_Rea
2  ┼
3  -- Notes:
4  -- 1) This testbench template has been automatically generated using t
5  -- std_logic and std_logic_vector for the ports of the unit under test
6  -- Xilinx recommends that these types always be used for the top-level
7  -- I/O of a design in order to guarantee that the testbench will bind
8  -- correctly to the timing (post-route) simulation model.
9  -- 2) To use this template as your testbench, change the filename to a
10 -- name of your choice with the extension .vhd, and use the "Source->A
11 -- menu in Project Navigator to import the testbench. Then
12 -- edit the user defined section below, adding code to generate the
13 -- stimulus for your design.
14 --
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 DLX_Control_Module_DLX_Control_Module_sch_tb IS
21 END DLX_Control_Module_DLX_Control_Module_sch_tb;
22 ARCHITECTURE behavioral OF DLX_Control_Module_DLX_Control_Module_sch_t
23
24     COMPONENT DLX_Control_Module
25     PORT( STEP_EN   :  IN STD_LOGIC;
26           busy :  OUT   STD_LOGIC;
27           DLX_O   :  OUT   STD_LOGIC_VECTOR (4 DOWNTO 0);
28           AS_N :  OUT   STD_LOGIC;
29           WR_N :  OUT   STD_LOGIC;
30           clk   :  IN STD_LOGIC;
31           reset   :  IN STD_LOGIC;
32           ACK_N   :  IN STD_LOGIC;
33           bt    :  IN STD_LOGIC;
34           Opcode   :  IN STD_LOGIC_VECTOR (5 DOWNTO 0);
35           FUNC :  IN STD_LOGIC_VECTOR (5 DOWNTO 0);
36           MAC_O   :  OUT   STD_LOGIC_VECTOR (1 DOWNTO 0));
37     END COMPONENT;
38
39     SIGNAL STEP_EN :  STD_LOGIC;
40     SIGNAL busy :  STD_LOGIC;
41     SIGNAL DLX_O   :  STD_LOGIC_VECTOR (4 DOWNTO 0);
42     SIGNAL AS_N :  STD_LOGIC;
43     SIGNAL WR_N :  STD_LOGIC;
44     SIGNAL clk   :  STD_LOGIC;
45     SIGNAL reset   :  STD_LOGIC;
46     SIGNAL ACK_N   :  STD_LOGIC;
47     SIGNAL bt    :  STD_LOGIC;
48     SIGNAL Opcode   :  STD_LOGIC_VECTOR (5 DOWNTO 0);
49     SIGNAL FUNC :  STD_LOGIC_VECTOR (5 DOWNTO 0);
50     SIGNAL MAC_O   :  STD_LOGIC_VECTOR (1 DOWNTO 0);
51
52 -- opcodes
53     --DATA TRANSFER
54     constant LW_OPCODE : std_logic_vector(5 downto 0)    := "100011";
55     constant SW_OPCODE : std_logic_vector(5 downto 0)    := "101011";
56     --ARITHMETIC
57     constant ADDI_OPCODE : std_logic_vector(5 downto 0) := "001011";
58     --TEST
59     constant SGTI_OPCODE : std_logic_vector(5 downto 0) := "011001";
60     constant SEQI_OPCODE : std_logic_vector(5 downto 0) := "011010";
61     constant SGEI_OPCODE : std_logic_vector(5 downto 0) := "011011";
62     constant SLTI_OPCODE : std_logic_vector(5 downto 0) := "011100";
63     constant SNEI_OPCODE : std_logic_vector(5 downto 0) := "011101";
```

```
64    constant SLEI_OPCODE : std_logic_vector(5 downto 0) := "011110";
65    --CONTROL
66    constant BEQZ_OPCODE : std_logic_vector(5 downto 0) := "000100";
67    constant BNEZ_OPCODE : std_logic_vector(5 downto 0) := "000101";
68    constant JR_OPCODE : std_logic_vector(5 downto 0)   := "010110";
69    constant JALR_OPCODE : std_logic_vector(5 downto 0) := "010111";
70    --MISC
71    constant NOP_OPCODE_FORMAT : std_logic_vector(2 downto 0)  := "110";
72    constant HALT_OPCODE : std_logic_vector(5 downto 0) := "111111";
73    --SHIFT
74    constant SLLI_OPCODE : std_logic_vector(5 downto 0) := "000000";
75    constant SRLI_OPCODE : std_logic_vector(5 downto 0) := "000010";
76 --CLOCK
77    constant clk_period : time := 200 ns;
78 BEGIN
79
80    UUT: DLX_Control_Module PORT MAP(
81       STEP_EN => STEP_EN,
82       busy => busy,
83       DLX_O => DLX_O,
84       AS_N => AS_N,
85       WR_N => WR_N,
86       clk => clk,
87       reset => reset,
88       ACK_N => ACK_N,
89       bt => bt,
90       Opcode => Opcode,
91       FUNC => FUNC,
92       MAC_O => MAC_O
93    );
94
95 -- *** Test Bench - User Defined Section ***
96    CLK_PROC : PROCESS
97    BEGIN
98       clk<='1';
99       wait for clk_period/2;
100      clk<='0';
101      wait for clk_period/2;
102   END PROCESS;
103
104   tb : PROCESS
105   BEGIN
106      -- initialization
107      reset<='0';
108      step_en<='0';
109      ACK_N<='1';
110      bt<='0';
111      opcode<= "000000";
112      func<= "000000";
113      wait for clk_period;
114      reset<='1';
115      wait for clk_period;
116      reset<='0';
117      wait for clk_period;
118      step_en<='1';
119      wait for clk_period;
120      step_en<='0';
121
122      -- new command ALU
123      wait for clk_period*5;
124      ACK_N<='0';
```

```
124         ACK_N<= '0';
125         wait for clk_period;
126         ACK_N<='1';
127         -- wait for clk_period;
128         --fetch
129         Opcode<="000000";
130         FUNC<="100000";
131         wait for clk_period;
132         -- decode
133         wait for clk_period;
134
135         -- followup
136         wait for clk_period*5;
137         ACK_N<='0';
138         wait for clk_period;
139         ACK_N<='1';
140         wait for clk_period;
141
142         --reset
143         wait for clk_period*5;
144         step_en<='1';
145         wait for clk_period;
146         step_en<='0';
147
148         -- new command TESTI
149         wait for clk_period*5;
150         ACK_N<='0';
151         wait for clk_period;
152         ACK_N<='1';
153         -- wait for clk_period;
154         --fetch
155         Opcode<="011000";
156         FUNC<="000000";
157         wait for clk_period;
158         -- decode
159         wait for clk_period;
160
161         -- followup
162         wait for clk_period*5;
163         ACK_N<='0';
164         wait for clk_period;
165         ACK_N<='1';
166         wait for clk_period;
167
168         --reset
169         wait for clk_period*5;
170         step_en<='1';
171         wait for clk_period;
172         step_en<='0';
173
174         -- new command LOAD
175         wait for clk_period*5;
176         ACK_N<='0';
177         wait for clk_period;
178         ACK_N<='1';
179         -- wait for clk_period;
180         --fetch
181         Opcode<="100011";
```

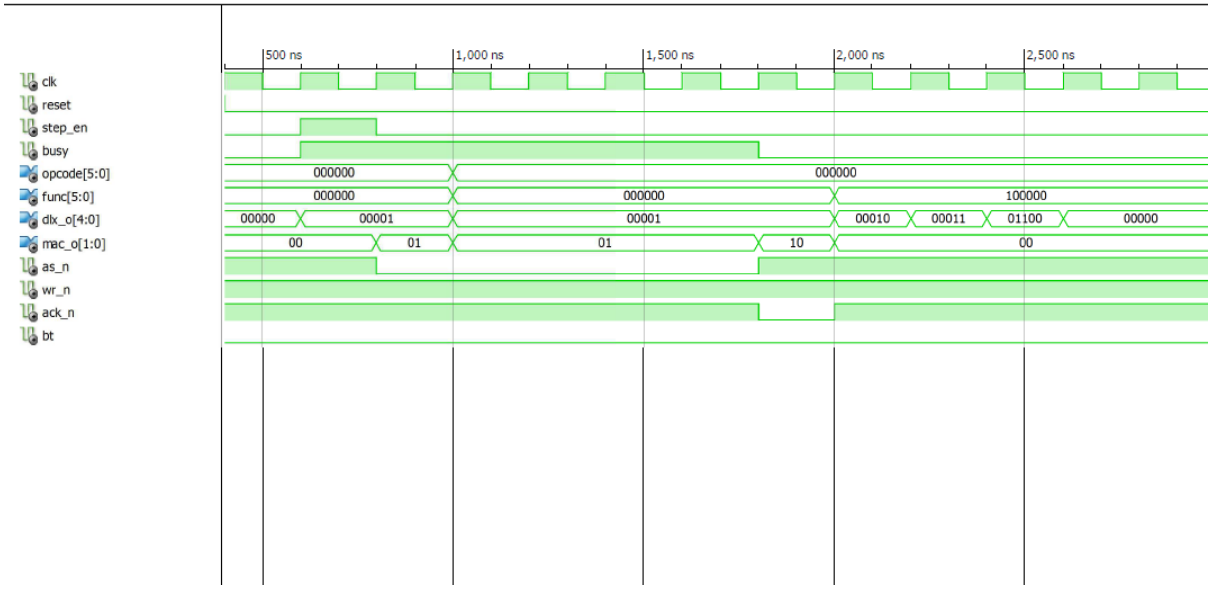
```
180         --fetch
181         Opcode<="100011";
182         FUNC<="000000";
183         wait for clk_period;
184         -- decode
185         wait for clk_period;
186
187         -- followup
188         wait for clk_period*5;
189         ACK_N<='0';
190         wait for clk_period;
191         ACK_N<='1';
192         wait for clk_period;
193
194               --reset
195         wait for clk_period*5;
196         step_en<='1';
197         wait for clk_period;
198         step_en<='0';
199
200         -- new command ADRCMP-STORE
201         wait for clk_period*5;
202         ACK_N<='0';
203         wait for clk_period;
204         ACK_N<='1';
205         -- wait for clk_period;
206         --fetch
207         Opcode<="100111";
208         FUNC<="000000";
209         wait for clk_period;
210         -- decode
211         wait for clk_period;
212
213         -- followup
214         wait for clk_period*5;
215         ACK_N<='0';
216         wait for clk_period;
217         ACK_N<='1';
218         wait for clk_period;
219               -- followup
220         wait for clk_period*5;
221         ACK_N<='0';
222         wait for clk_period;
223         ACK_N<='1';
224         wait for clk_period;
225
226               --reset
227         wait for clk_period*5;
228         step_en<='1';
229         wait for clk_period;
230         step_en<='0';
231
232         -- new command - JALR
233         wait for clk_period*5;
234         ACK_N<='0';
235         wait for clk_period;
236         ACK_N<='1';
237         -- wait for clk_period;
238         --fetch
239         Opcode<="010111";
240         FUNC<="000000";
```

```
239         Opcode<="010111";
240         FUNC<="000000";
241         wait for clk_period;
242         -- decode
243         wait for clk_period;
244 
245         -- followup
246         wait for clk_period*5;
247         ACK_N<='0';
248         wait for clk_period;
249         ACK_N<='1';
250         wait for clk_period;
251 
252               --reset
253         wait for clk_period*5;
254         step_en<='1';
255         wait for clk_period;
256         step_en<='0';
257 
258 
259         -- new command - BTAKEN
260         wait for clk_period*5;
261         ACK_N<='0';
262         wait for clk_period;
263         ACK_N<='1';
264         -- wait for clk_period;
265         --fetch
266         Opcode<="000100";
267         FUNC<="000000";
268         bt<='1';
269         wait for clk_period;
270         -- decode
271         wait for clk_period;
272 
273         -- followup
274         wait for clk_period*5;
275         ACK_N<='0';
276         wait for clk_period;
277         ACK_N<='1';
278         wait for clk_period;
279 
280         --reset
281         wait for clk_period*5;
282         step_en<='1';
283         wait for clk_period;
284         step_en<='0';
285 
286 
287         -- new command - BTAKEN
288         wait for clk_period*5;
289         ACK_N<='0';
290         wait for clk_period;
291         ACK_N<='1';
292         -- wait for clk_period;
293         --fetch
294         Opcode<="000100";
295         FUNC<="000000";
296         bt<='0';
297         wait for clk_period;
298         -- decode
299         wait for clk_period;
300 
```

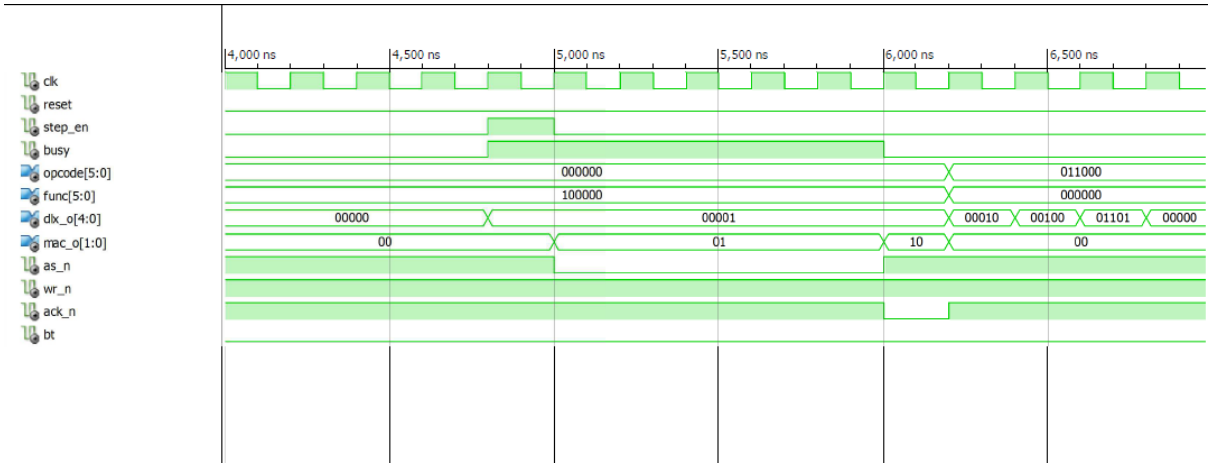
```
297         wait for clk_period;
298         -- decode
299         wait for clk_period;
300 
301         -- followup
302         wait for clk_period*5;
303         ACK_N<='0';
304         wait for clk_period;
305         ACK_N<='1';
306         wait for clk_period;
307         WAIT;
308      END PROCESS;
309  -- *** End Test Bench - User Defined Section ***
310 
311  END;
312 
```

Now onto our control module simulations:

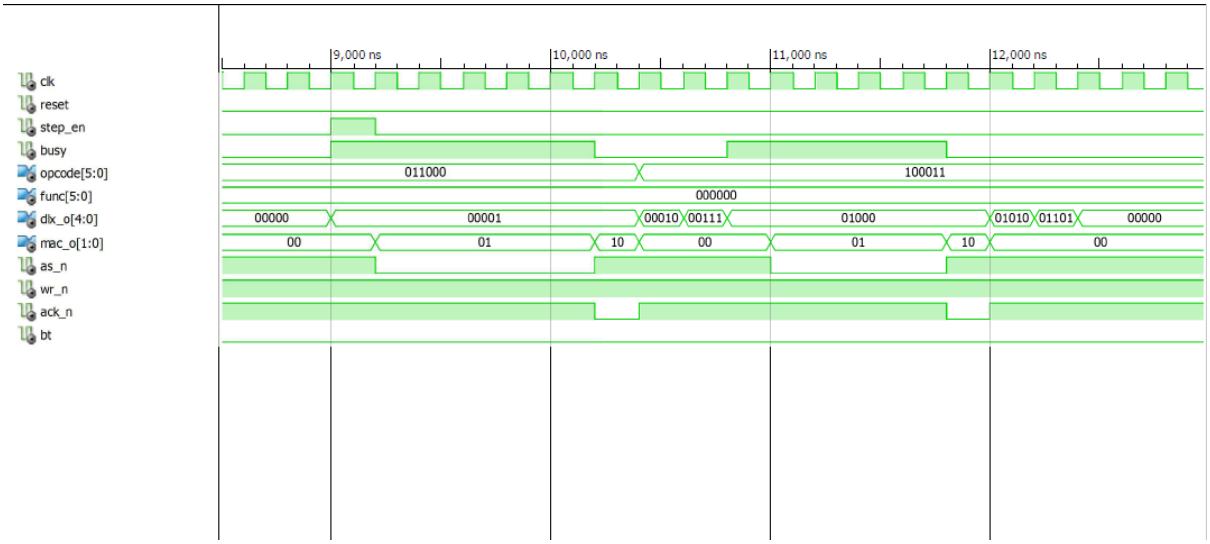
ALU:



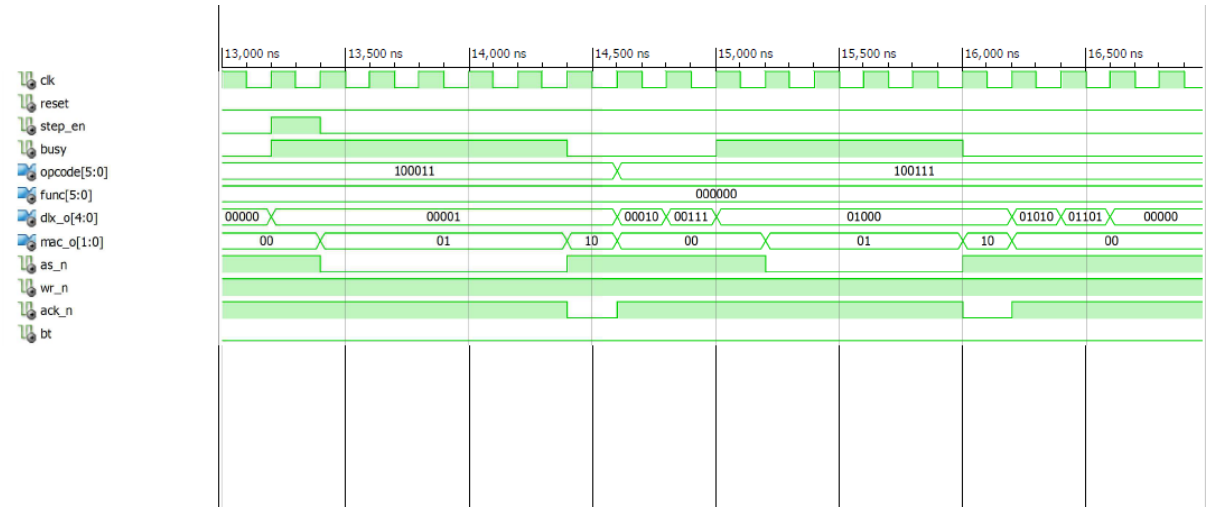
TESTI:



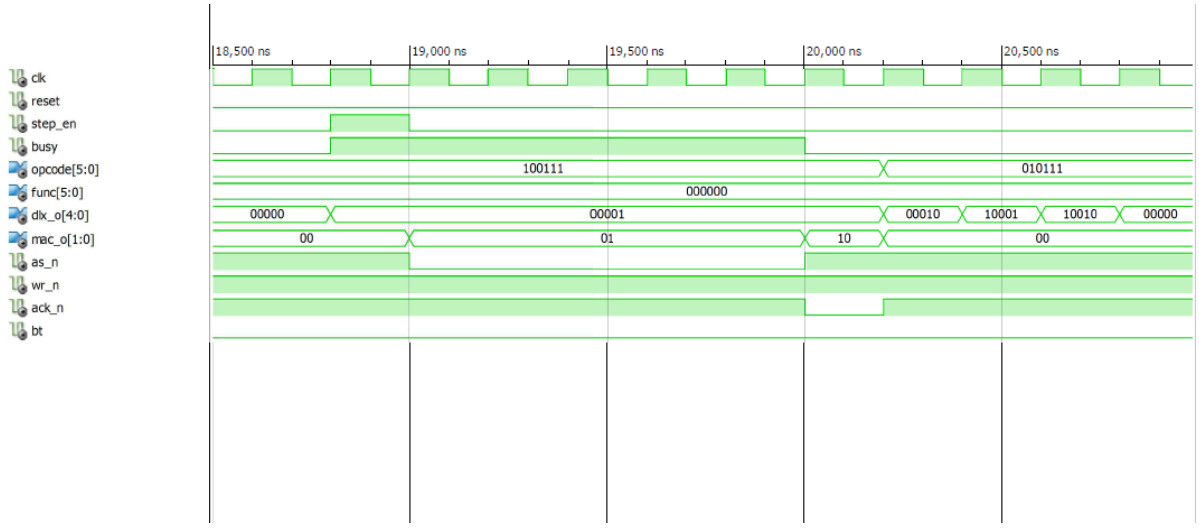
LOAD:


9,000 ns
10,000 ns
11,000 ns
12,000 ns
clk
reset
step_en
busy
opcode[5:0]
func[5:0]
dix_o[4:0]
mac_o[1:0]
as_n
wr_n
ack_n
bt
011000
100011
000000
00000
00001
00010
00111
01000
01010
01101
00000
00
01
10
00
01
10
00

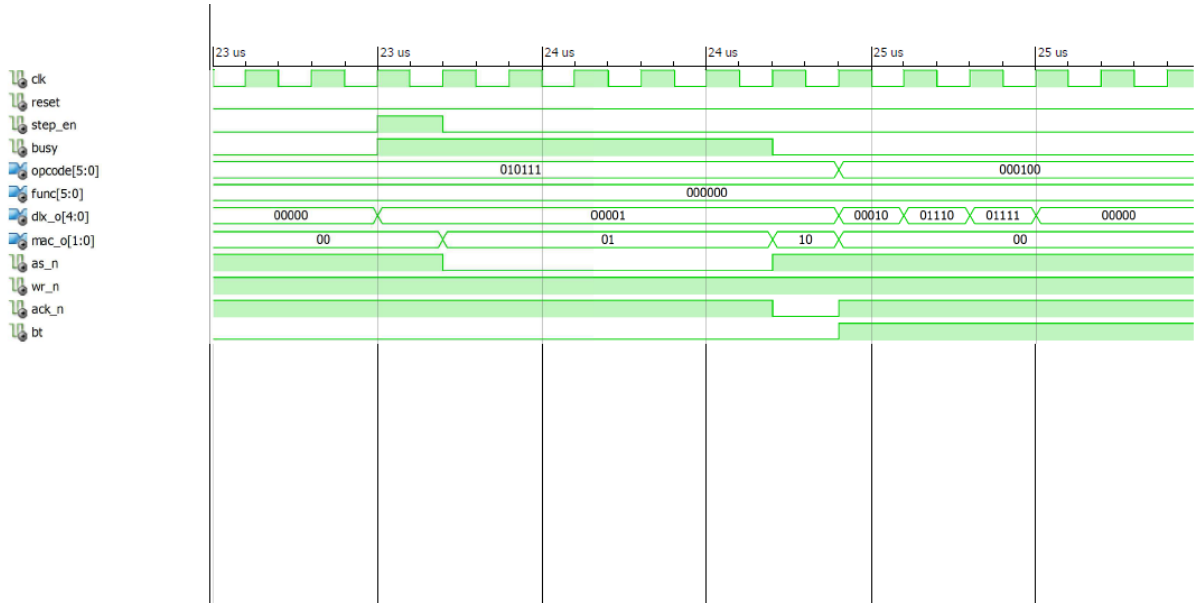
STORE:


13,000 ns
13,500 ns
14,000 ns
14,500 ns
15,000 ns
15,500 ns
16,000 ns
16,500 ns
clk
reset
step_en
busy
opcode[5:0]
func[5:0]
dix_o[4:0]
mac_o[1:0]
as_n
wr_n
ack_n
bt
100011
100111
000000
00000
00001
00010
00111
01000
01010
01101
00000
00
01
10
00
01
10
00

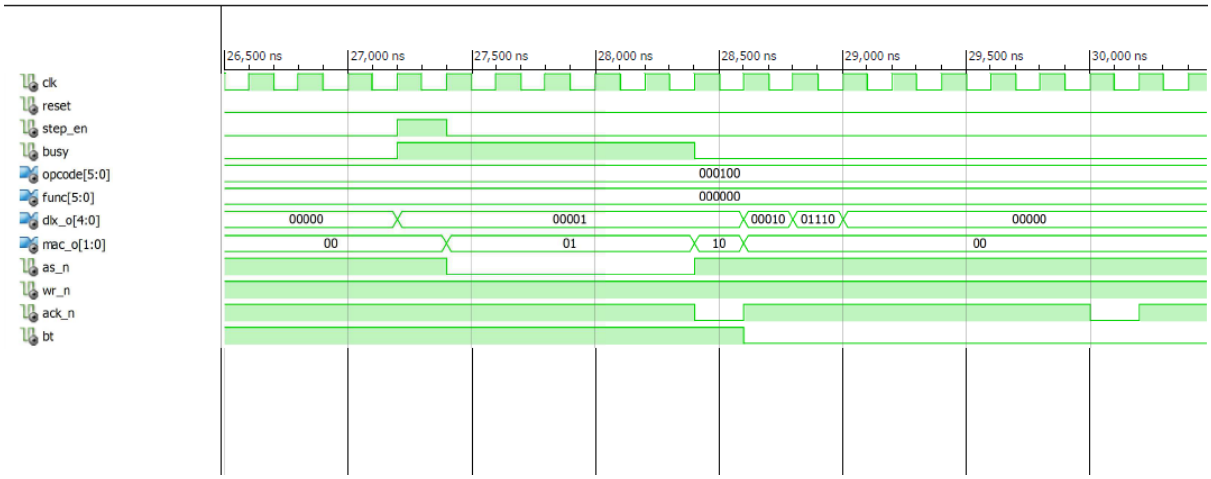
JALR:



BTAKEN (bt=1):



BTAKEN (bt=0):


26,500 ns
27,000 ns
27,500 ns
28,000 ns
28,500 ns
29,000 ns
29,500 ns
30,000 ns
clk
reset
step_en
busy
opcode[5:0]
func[5:0]
dlx_o[4:0]
mac_o[1:0]
as_n
wr_n
ack_n
bt
000100
000000
00000
00001
00010
01110
00000
00
01
10
00

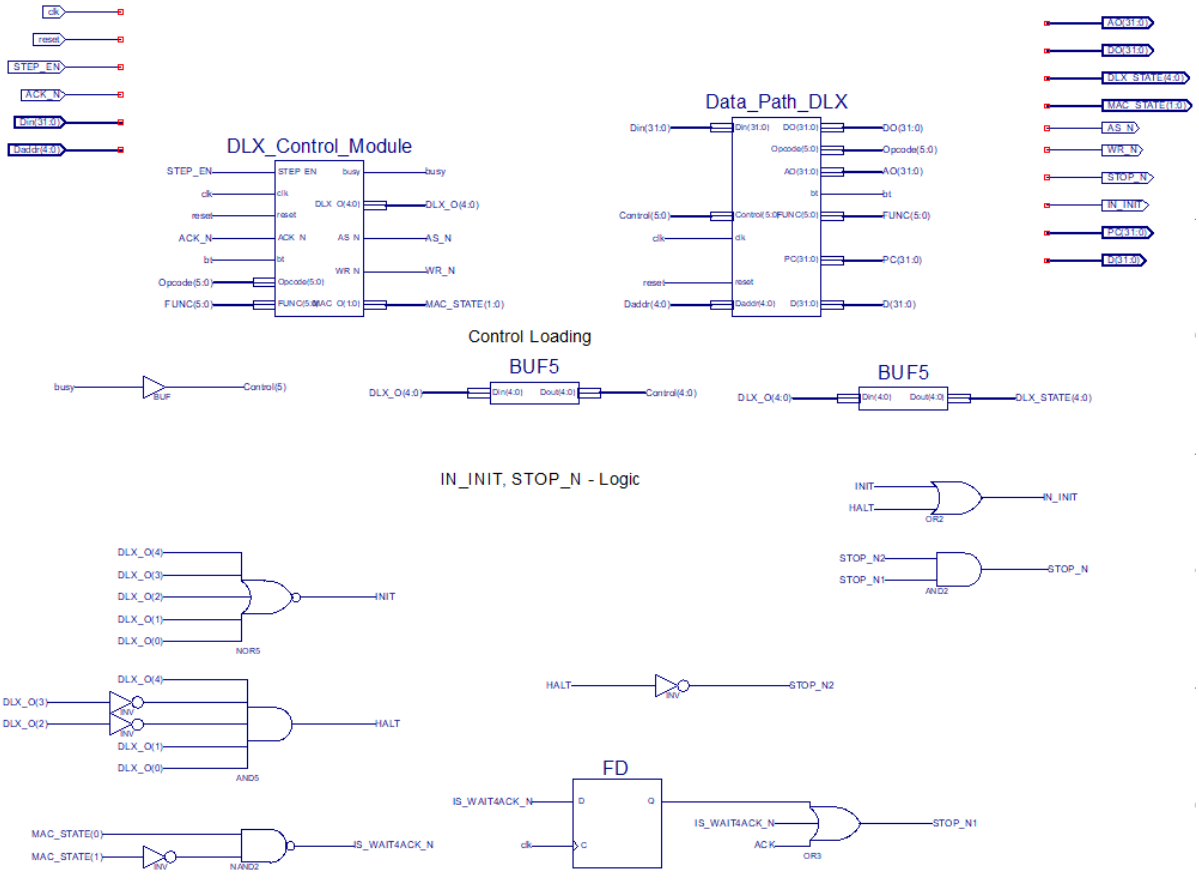
And as we can see from the simulations above our control module is behaving as desired according to the tables below :

Name	RTL Instruction	Active Control Signals
Fetch	*IR=M(PC)*	MR, IRce
Decode	*A = R(RS1)* *B = R(RS2)* *PC = PC + 1*	Ace, Bce, S2se1[1], S2sel[0] PCce, add
Alu	*C = A* op *B*	S1sel[0], Cce
TestI	C = *(A* rel *imm)*	S1se1[0], S2se1[0], Cce, test, Itype
AluI(add)	C = A + sext*(imm)*	S1se1[0], S2se1[0], Cce, add, Itype
Shift	C = A shift *sa* *Sa = 1,IR[1]*	S1se1[0], Cce DINTse1, shift (,right)
Adr.Comp	*MAR = A +sext(imm)*	S1se1[0], S2se1[0].MARce, add
Load	*MDR = M(MAR)*	MDRce, Ase1, MR, MDRse1
Store	*M(MAR) = MDR*	Ase1, MW
CopyMDR2C	*C=MDR(>>0)*	S1sel[0], S1se1[1], S2se1[1], DINTse1, Cce
CopyGPR2MDR	*MDR = B(<<0)*	S1se1[1], S2se1[1], DINTse1, MDRce
WBR	*R(RD) = C* (R-type)	GPR_WE
WBI	*R(RD) = C* (I-type)	GPR_WE, Itype
Branch	Branch taken?	
Btaken	*PC = PC +sext(imm)*	S2se1[0], Add, PCce
JR	*PC = A*	S1se1[0], S2se1[1], add, PCce
Save PC	*C = PC*	S2se1[1], add, CCe
JALR	*PC = A* *R(31) = C*	S1se1[0], S2se1[1], add, PCce GPR_WE, jlink

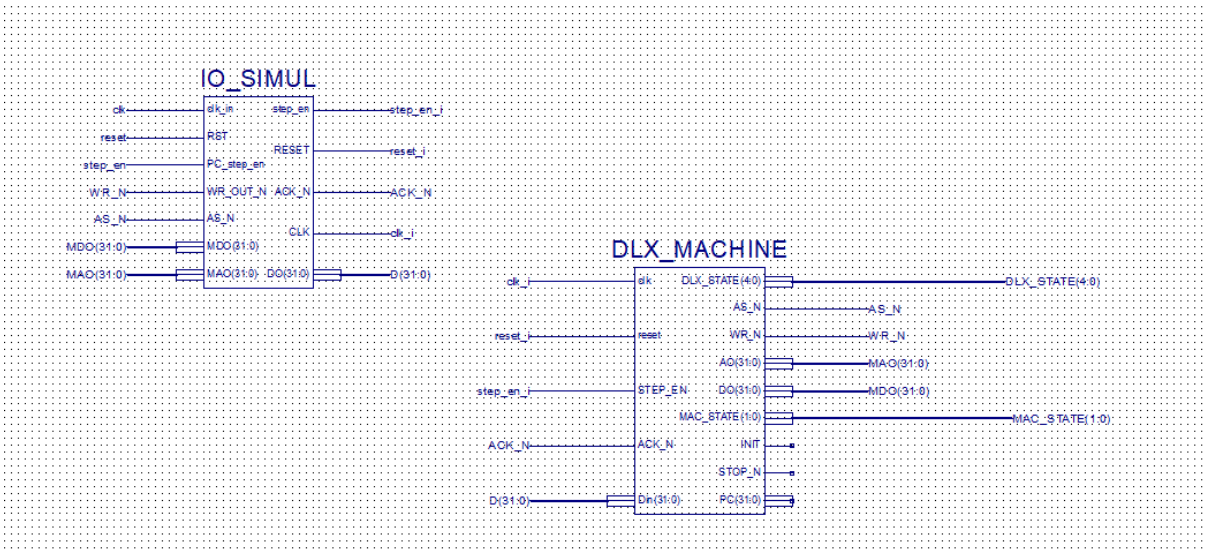
Signal	Value	Semantics
ALUf[2:0]		Controls the functionality of ALU
Rce		Register clock enable
S1sel[1:0]	00	PC
	01	A
	10	B
	11	MDR
S2se1[1:0]	00	B
	01	IR
	10	0
	11	1
DINTse1	0	ALU
	1	Shifter
MDRse1	0	DINT
	1	DI
Ase1	0	PC
	1	MAR
Shift		Explicit Shift-Instruction
Right		Shift to the right
Add		Forces an addition
Test		Forces a test (in the ALU)
MR		Memory Read
MW		Memory Write
GPR_WE		GPR write enable
Itype		Itype-Instruction
Jlink		Jump and link

Part3 – DLX and IOSimul testing:

Our final simplified DLX looked like this :



And Our schematic of the IO_simul and our dlx was as following :



Now for the testing of the RTL instructions we used the following code that we wrote :

```
0x00000000: 0x8C050022                      lw R5 R0 data1
0x00000001: 0xAC050ABC                      sw R5 R0 0xabc

0x00000002: 0x2C051234                      addi R5 R0 0x1234

0x00000003: 0x2C010001                      addi R1 R0 0x1

0x00000004: 0x003F1000                      slli R2 R1
0x00000005: 0x005F1000                      slli R2 R2
0x00000006: 0x005F1002                      srli R2 R2
0x00000007: 0x005F1002                      srli R2 R2

0x00000008: 0x2C010007                      addi R1 R0 0x7
0x00000009: 0x2C02000F                      addi R2 R0 0xf
0x0000000A: 0x00221823                      add R3 R1 R2
0x0000000B: 0x00221822                      sub R3 R1 R2
0x0000000C: 0x00221826                      and R3 R1 R2
0x0000000D: 0x00221825                      or  R3 R1 R2
0x0000000E: 0x00221824                      xor R3 R1 R2

0x0000000F: 0x2C010002                      addi R1 R0 0x2

0x00000010: 0x2C21FFFF                      addi R1 R1 0xffff
0x00000011: 0x143FFFFE                      bnez R1 0xfffe

0x00000012: 0x2C010001                      addi R1 R0 0x1
0x00000013: 0x103F0004                      beqz R1 end
0x00000014: 0x2C010019                      addi R1 R0 testi
0x00000015: 0x5C3F5220                      jalr R1
0x00000016: 0x2C010018                      addi R1 R0 end
0x00000017: 0x583F8800                      jr R1

0x00000018: 0xFFFF1200      end:            halt

0x00000019: 0x2C01FFFF      testi:          addi R1 R0 0xffff
0x0000001A: 0x2C020005                            addi R2 R0 0x5

0x0000001B: 0x7023FFFE                            slti R3 R1 0xfffe

0x0000001C: 0x6823FFFF                            seqi R3 R1 0xffff

0x0000001D: 0x64230001                            sgti R3 R1 0x1

0x0000001E: 0x78430005                            slei R3 R2 0x5

0x0000001F: 0x6C430004                            sgei R3 R2 0x4

0x00000020: 0x74430009                            snei R3 R2 0x9

0x00000021: 0x5BFF0000                            jr R31


0x00000022: 0x0000FFFF      data1:          dc 0xffff
```

```
Label Report:
end: 33 D: 27, 30
testi: 35 D: 28
data1: 53 D: 1
*XML file date: Wed 20/6/2012 6:49:12
```

And the test bench that we used to test our RTL instructions was :

```
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 TOP_LEVEL_IOSIMUL_TOP_LEVEL_IOSIMUL_sch_tb IS
21  END TOP_LEVEL_IOSIMUL_TOP_LEVEL_IOSIMUL_sch_tb;
22  ARCHITECTURE behavioral OF TOP_LEVEL_IOSIMUL_TOP_LEVEL_IOSIMUL_sch_tb IS
23
24     COMPONENT TOP_LEVEL_IOSIMUL
25     PORT( clk    :  IN STD_LOGIC;
26           reset   :  IN STD_LOGIC;
27           step_en :  IN STD_LOGIC);
28     END COMPONENT;
29
30     SIGNAL clk   :  STD_LOGIC;
31     SIGNAL reset   :  STD_LOGIC;
32     SIGNAL step_en :  STD_LOGIC;
33
34     constant clk_period : time := 100 ns;
35  BEGIN
36
37     UUT: TOP_LEVEL_IOSIMUL PORT MAP(
38        clk => clk,
39        reset => reset,
40        step_en => step_en
41     );
42
43     CLK_PROC : PROCESS
44     BEGIN
45        clk<='1';
46        wait for clk_period/2;
47        clk<='0';
48        wait for clk_period/2;
49     END PROCESS;
50  -- *** Test Bench - User Defined Section ***
51     tb : PROCESS
52     BEGIN
53        step_en<='0';
54        reset<='0';
55
56        wait for 101 ns;
57        reset<='1';
58        wait for clk_period;
59        reset<='0';
60
61        FOR i in 1 to 100 loop
62           -- initiate command
63           wait for clk_period;
64           step_en <='1';
65           wait for clk_period;
66           step_en<='0';
67
68           wait for 20*clk_period;
69        end loop;
70
71
72        WAIT; -- will wait forever
73     END PROCESS;
74  -- *** End Test Bench - User Defined Section ***
75
76  END;
77
```

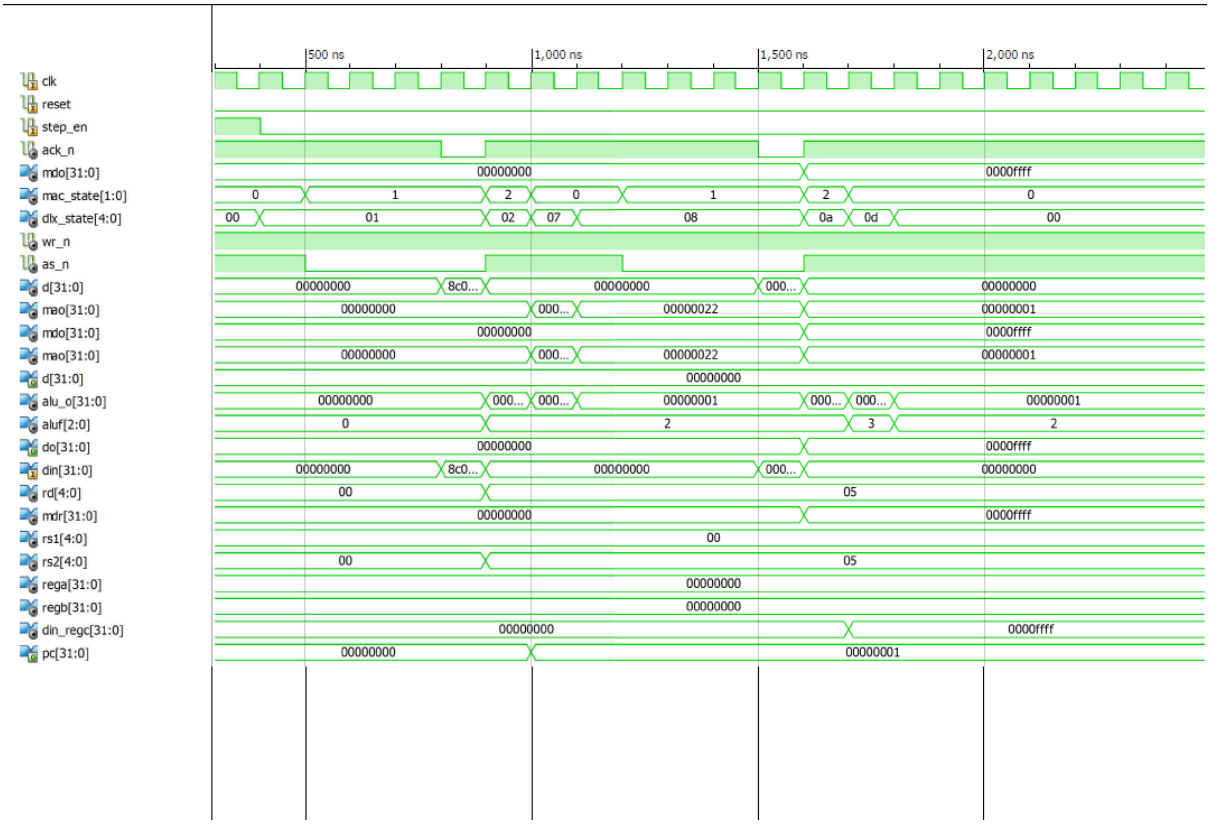
Our states are as following:

state	number
Init	00
Fetch	01
Decode	02
Alu	03
Testi	04
Alui	05
Shift	06
Adrcomp	07
Load	08
Store	09
Copymdr2c	0A
Copygpr2mdr	0B
Wbr	0C
Wbi	0D
Branch	0E
B_taken	0F
Jr	10
Save_pc	11
Jalr	12
halt	13

And from the simulations bellow we can see that the dlx_state_machine is acting as desired.

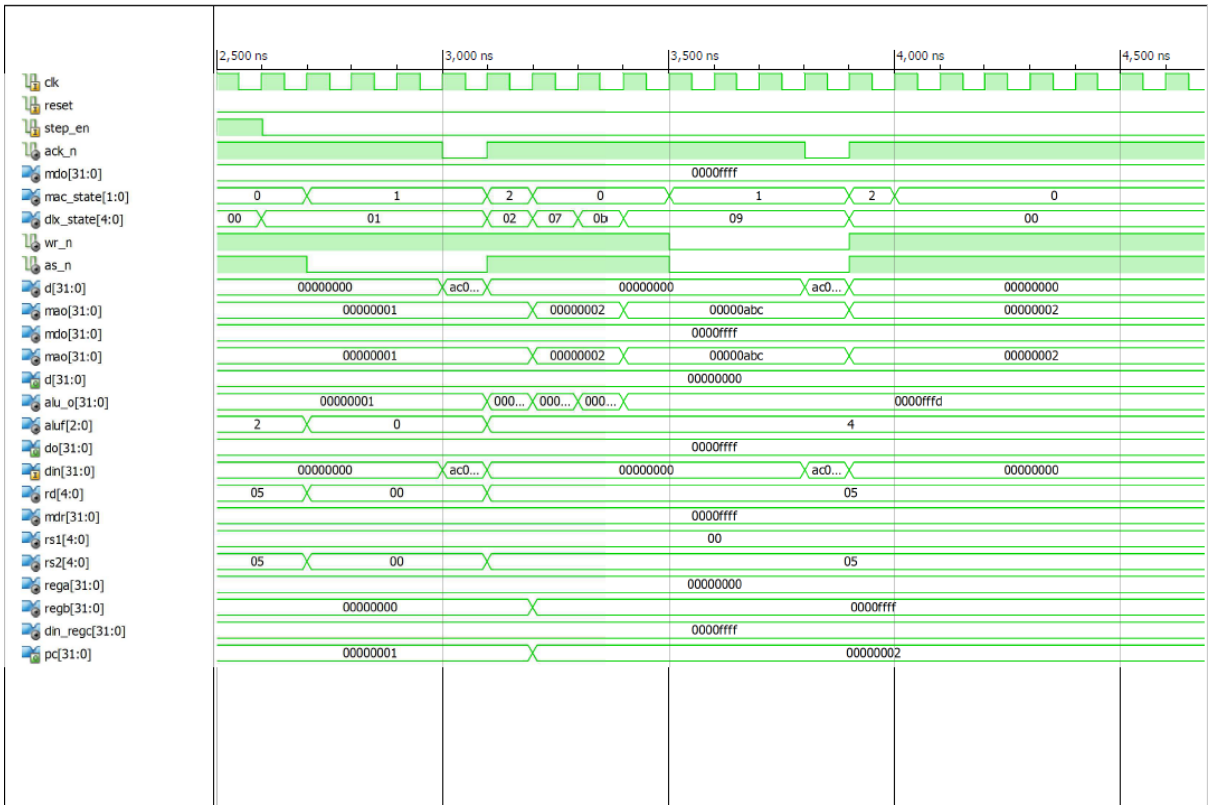
Our IO_simul simulation waveforms:

lw R5 R0 data1:



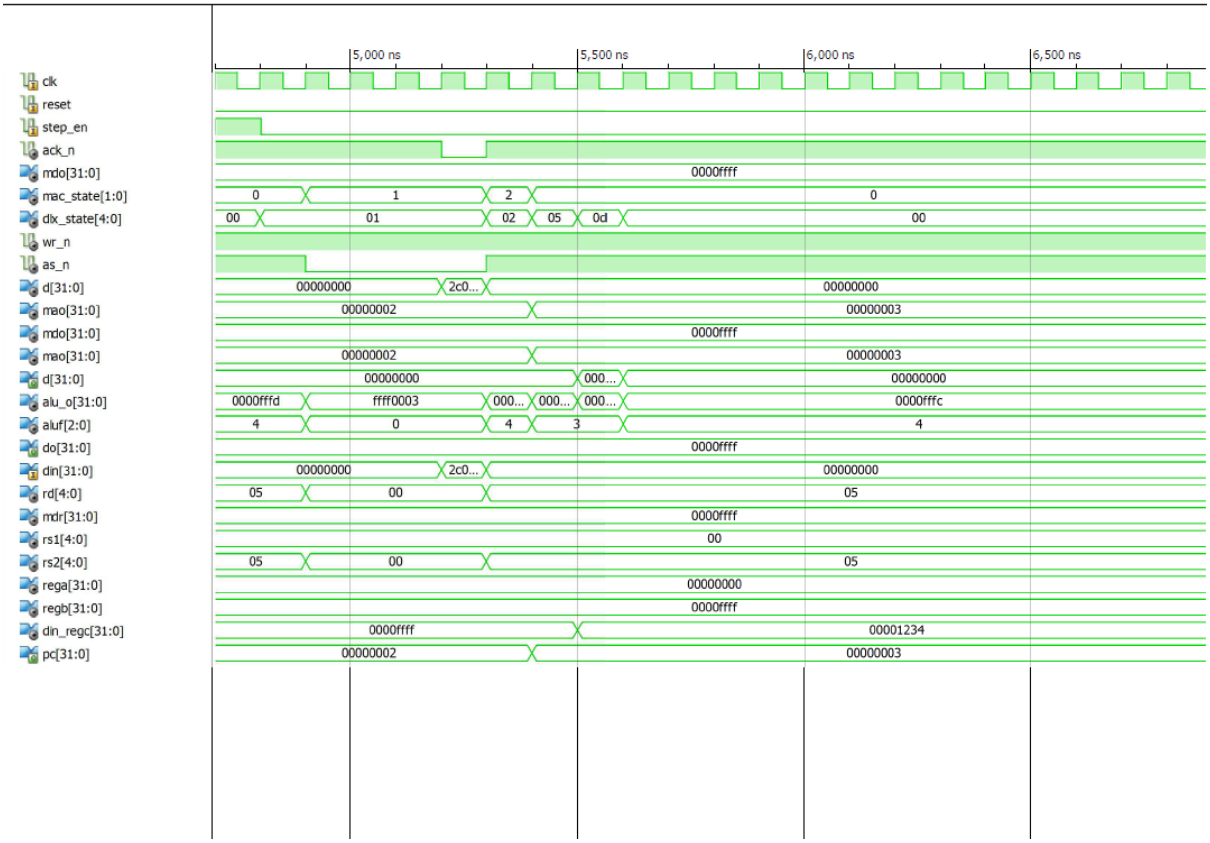
we can see that mao is getting the right address which is 0x00000022, and then getting the value from the address which is 0x0000ffff , and then writing it back in the register R5 ,we also can see din_regc (which Is register c)getting that value to write it back .

sw R5 R0 0xabc:



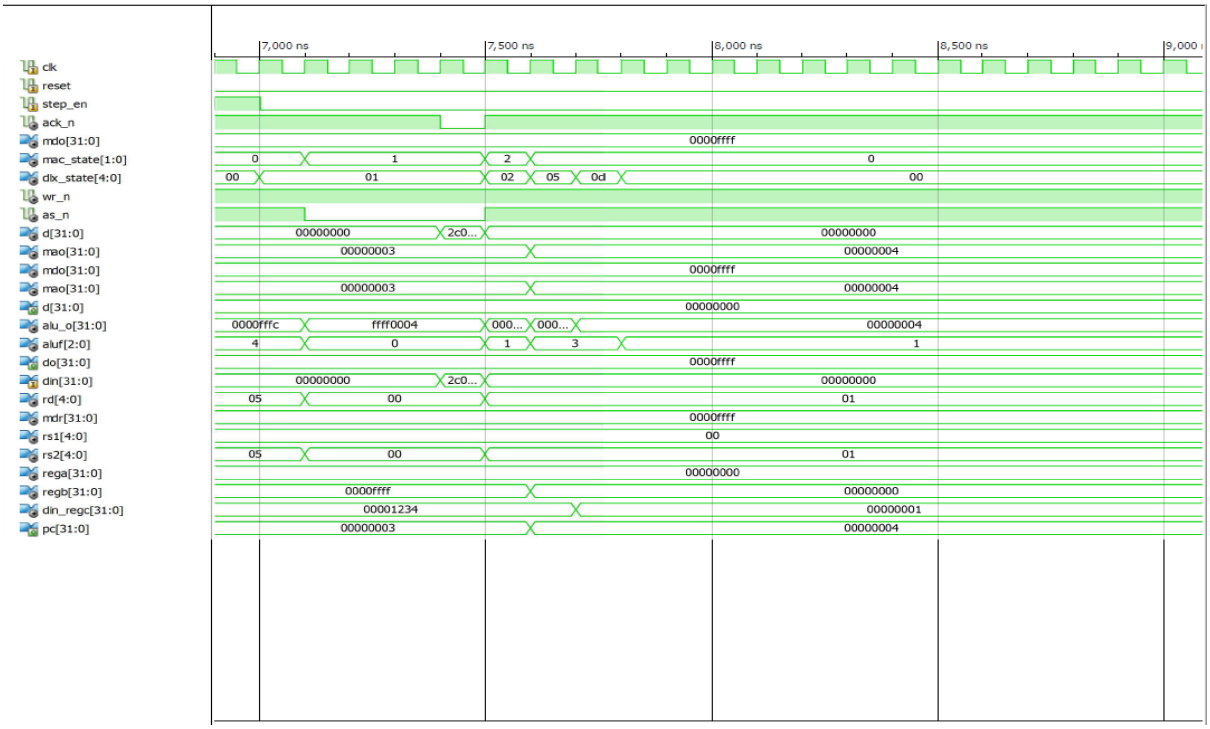
We can see that is getting the right address which is 0x00000abc, and we can see din_regc have the right value to write it in the right address.

addi R5 R0 0x1234:



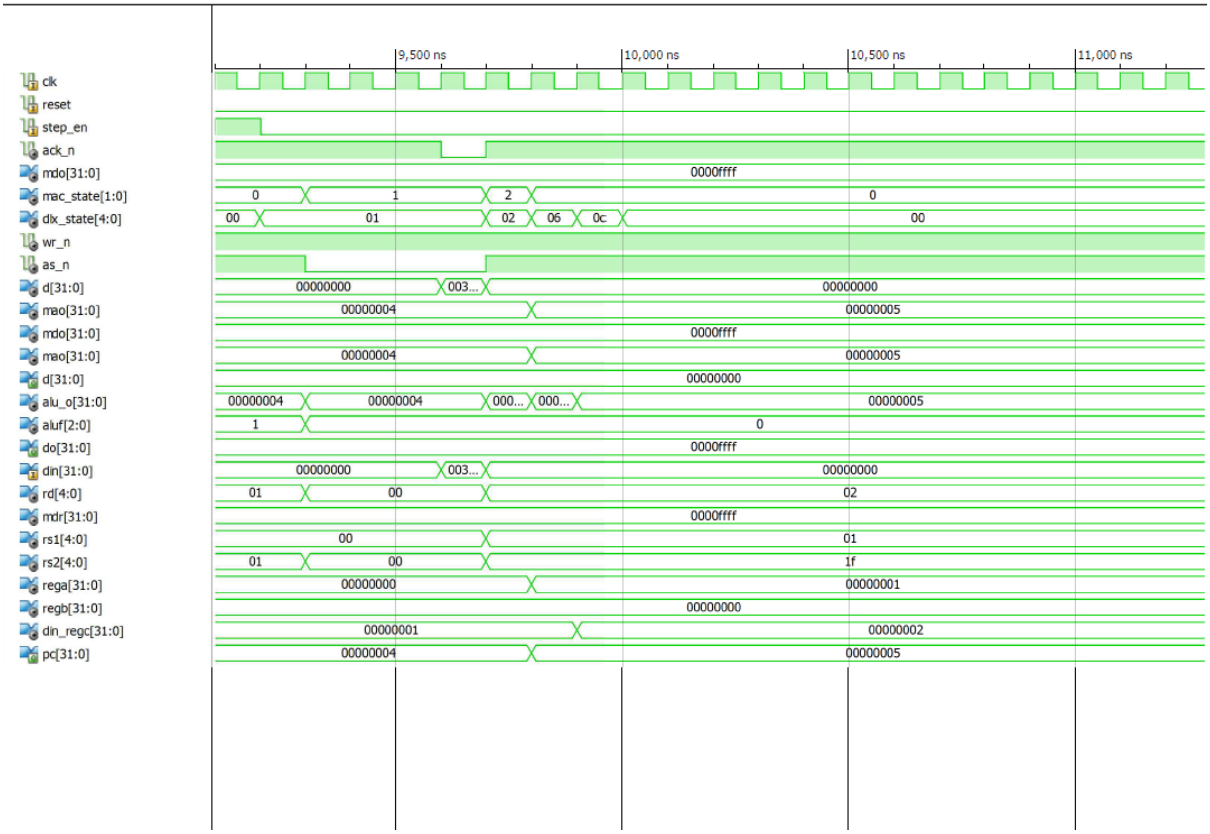
We can see from the waveforms that din_regc is getting the right value.

addi R1 R0 0x1:



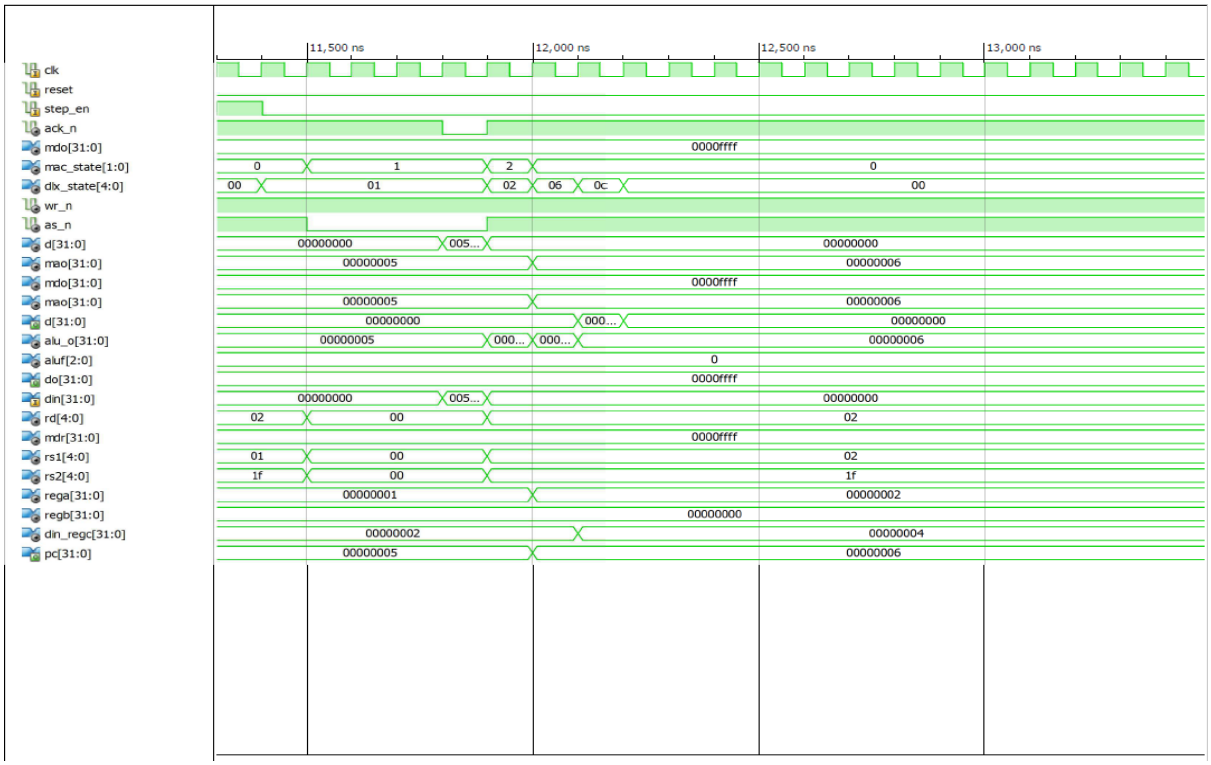
We also can see here that din_regc is getting the right value after all the operations .

slli R2 R1:



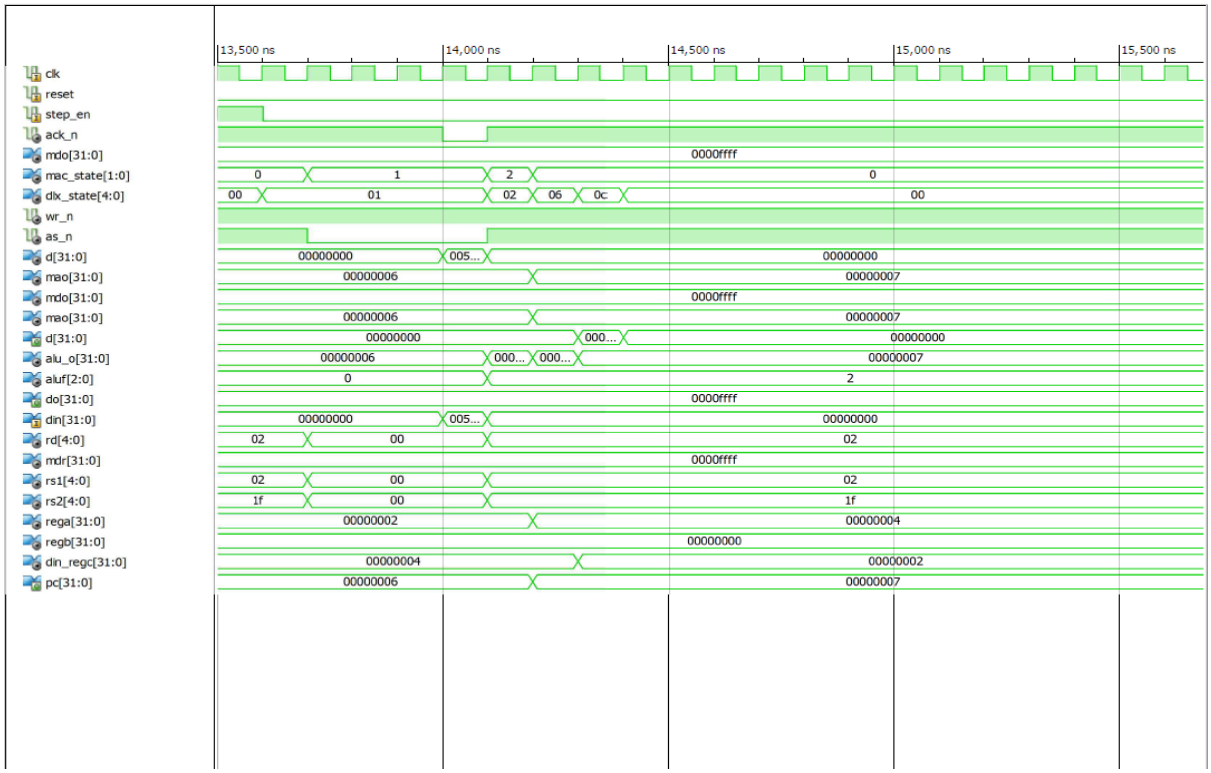
We can see that after doing the shift left instruction , din_regc is getting 2 which is 1 shifted left.

slli R2 R2:



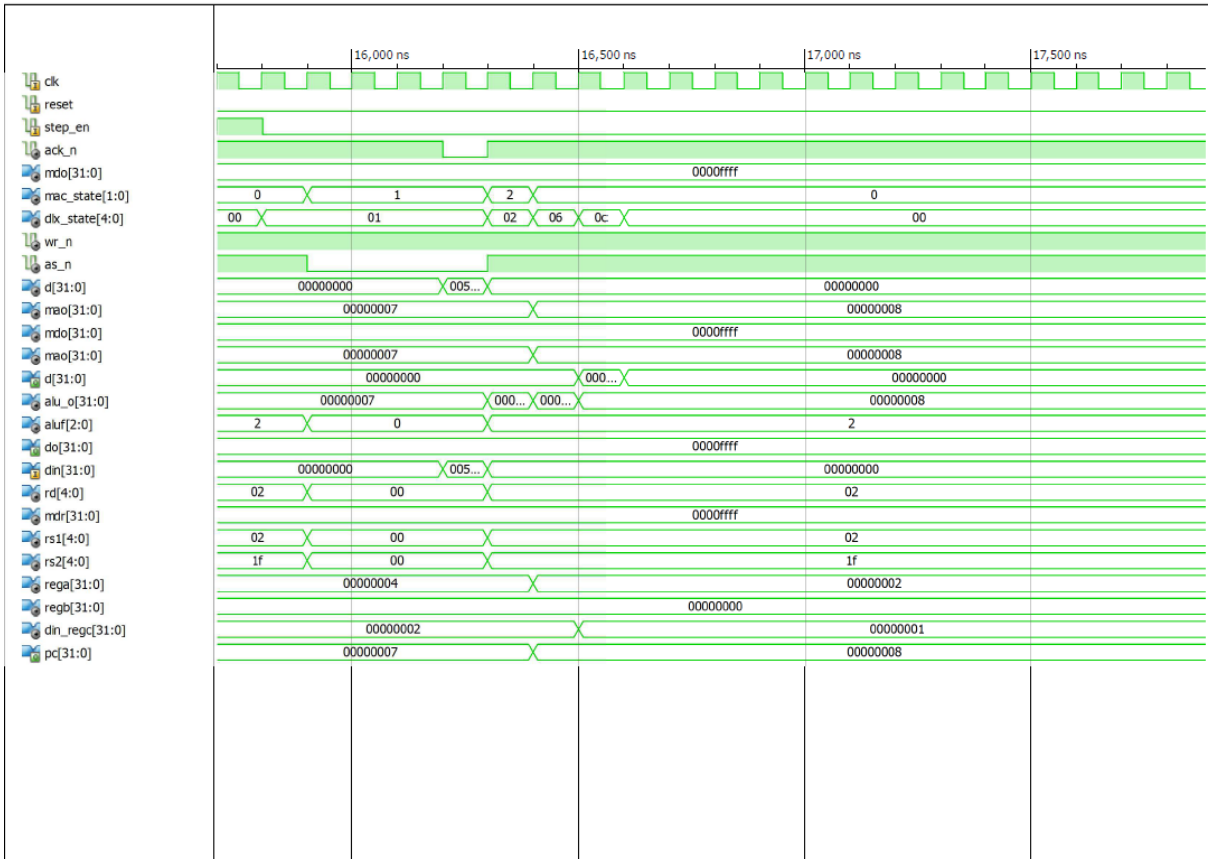
We can see that after doing the shift left instruction , din_regc is getting 4 now, which is the 2 from before shifted left.

srli R2 R2:



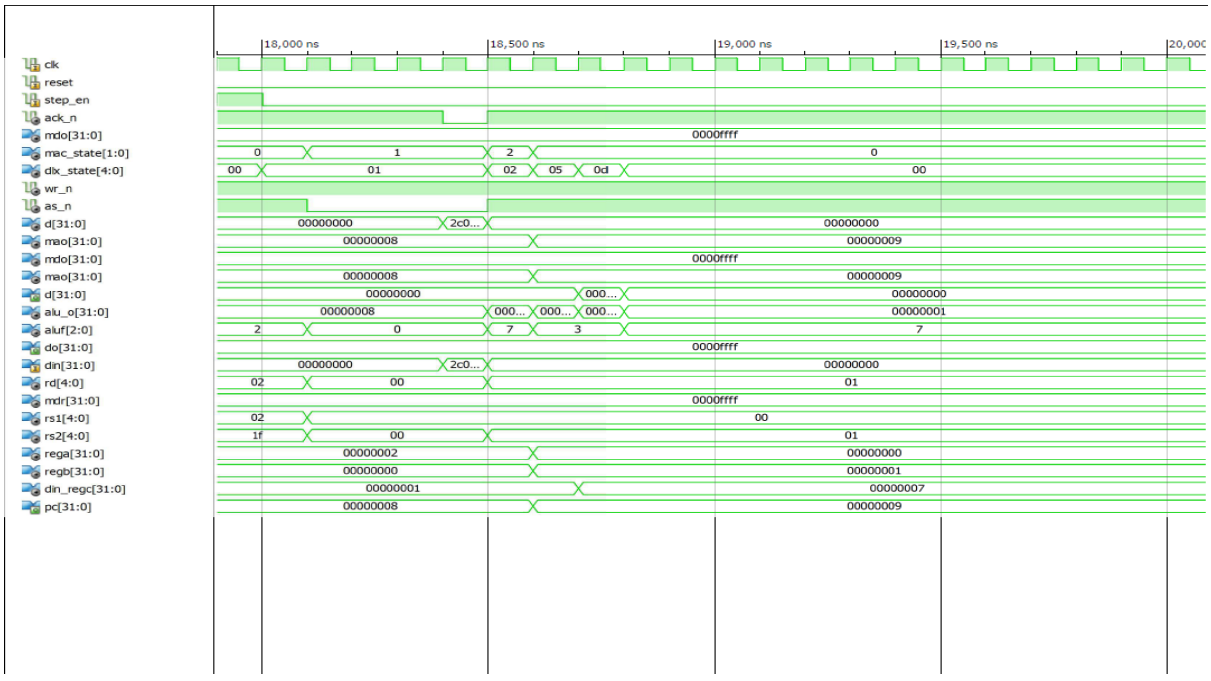
Now We can see that after doing the shift right instruction , din_regc is getting 2 which is 4 shifted right.

srli R2 R2:



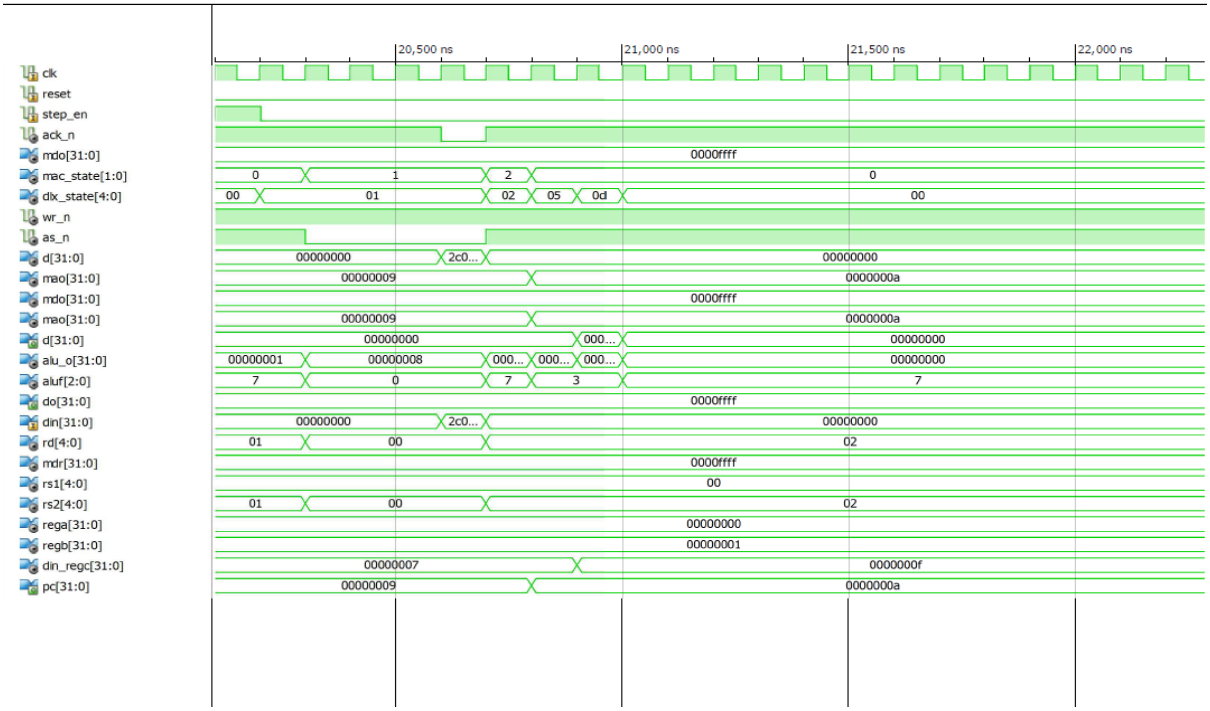
We can see that after doing the shift right instruction , din_regc is getting 1 now, which is the 2 from before shifted right.

addi R1 R0 0x7:



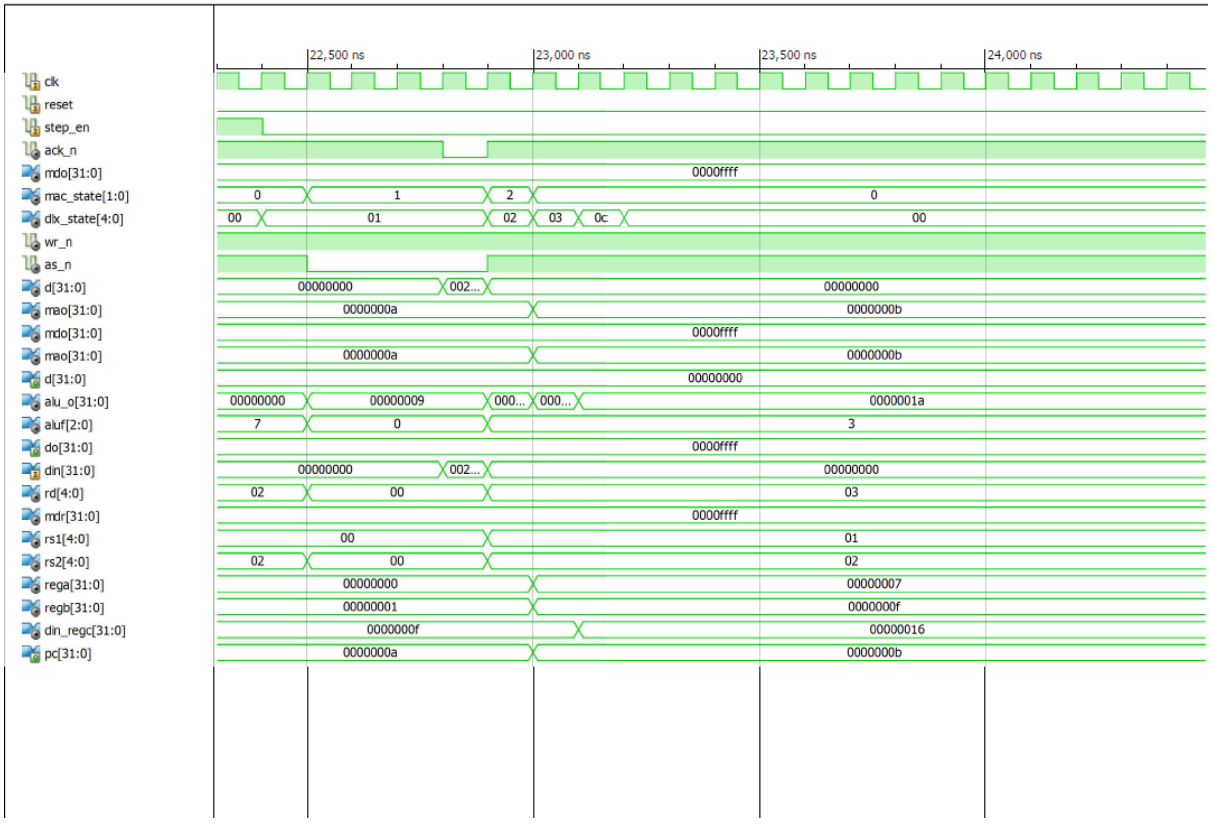
We can see that din_regc is getting the value that should be assigned later by the resa to register R1, and it is 7.

addi R2 R0 0xf:



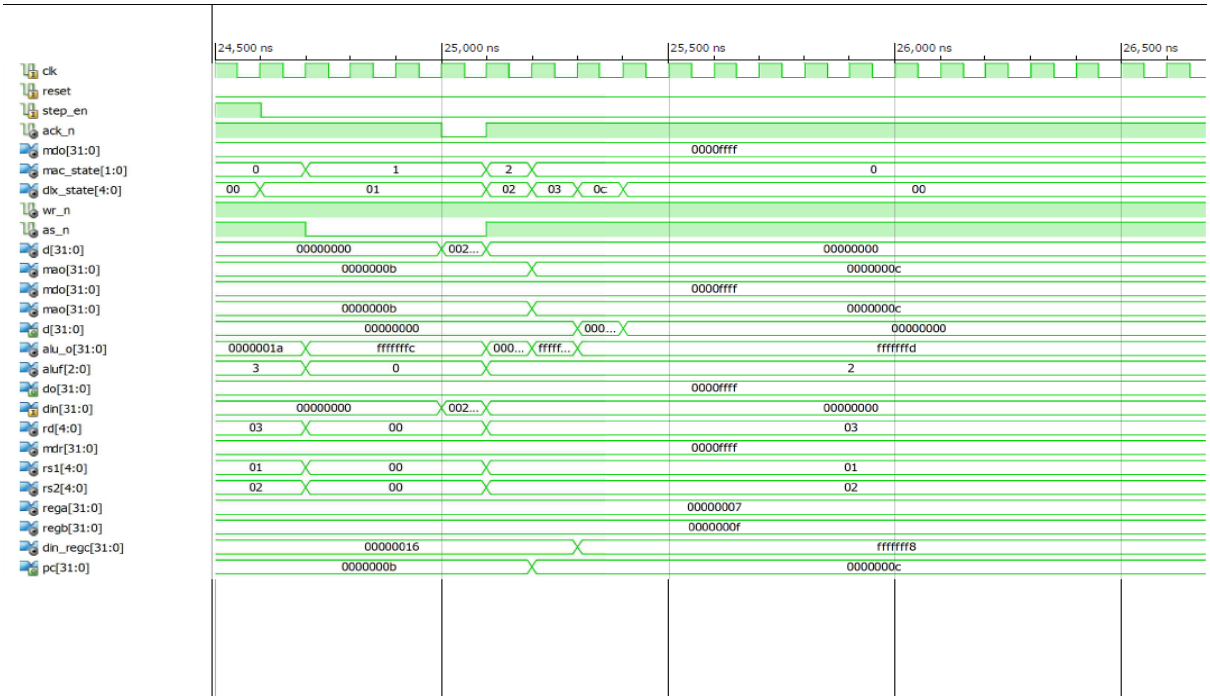
Now we see that after the 7 , din_regc is getting f, which is the value that will be assigned to R2.

add R3 R1 R2:



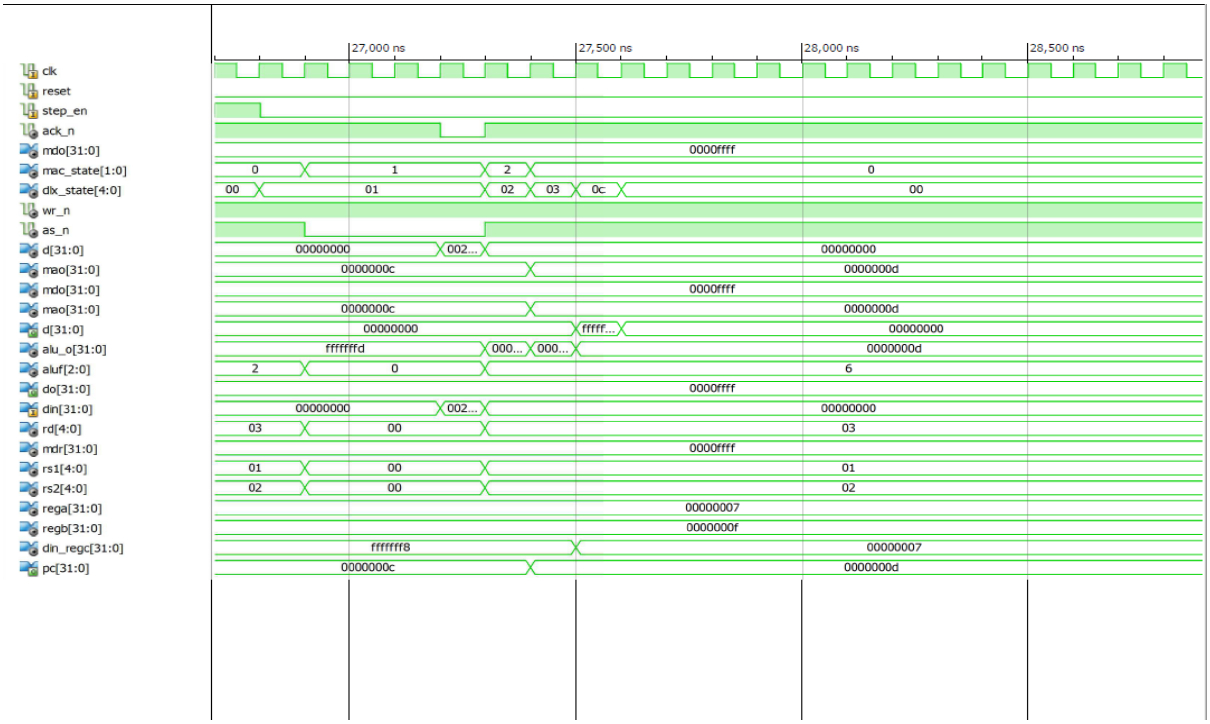
We can see in rega and regb , are saved the values of R1 and R2, which is 0x00000007 and 0x0000000f, and is din_regc, we get the value of the add instruction , which is 0x00000016.

sub R3 R1 R2:



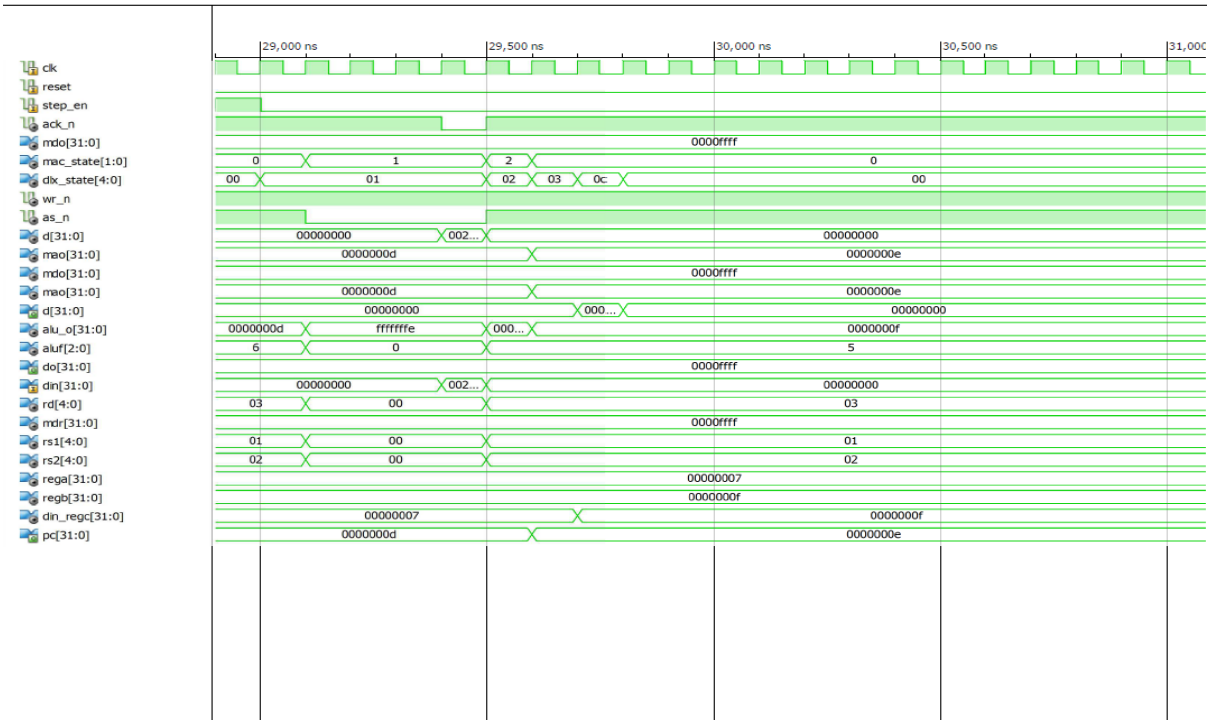
We can see that din_regc is getting the sub between rega and regb, and we got the desired answer and it is a negative number which is 0xfffffff8.

and R3 R1 R2:



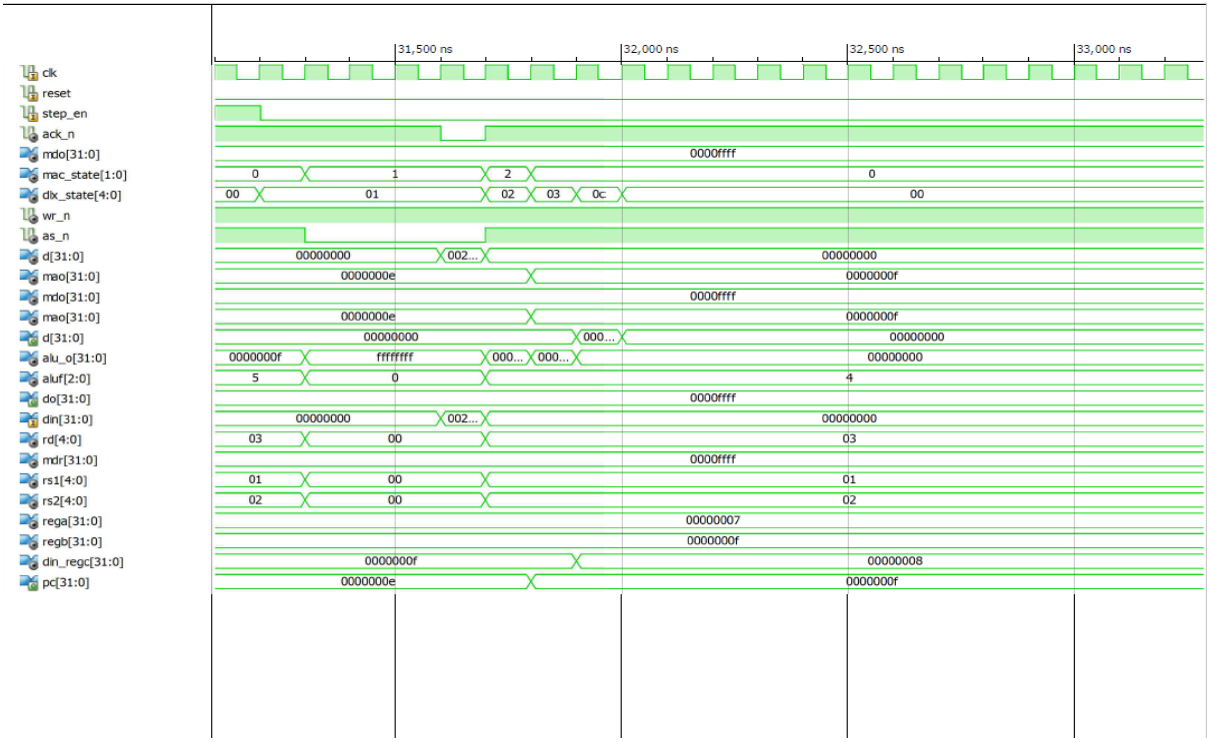
Now we can see that we are getting the and between rega and regb.

or R3 R1 R2:



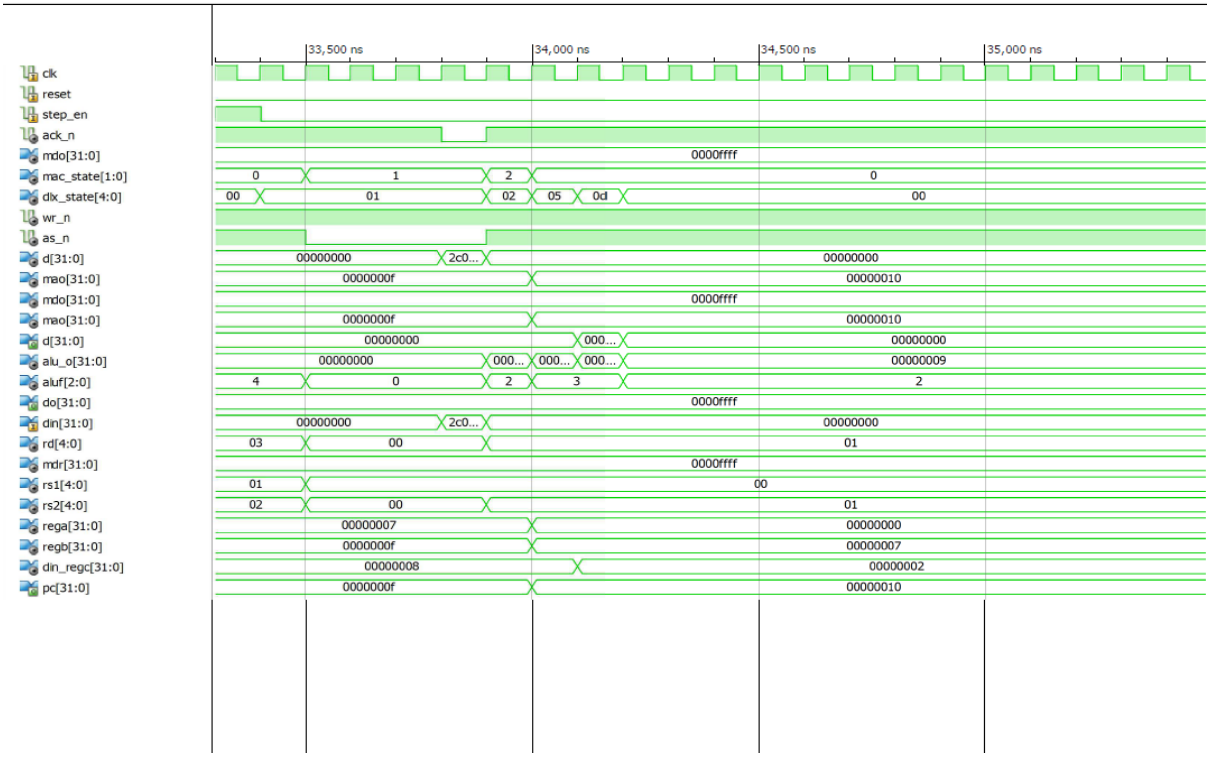
Now we can see that we are getting in din_regc the or between rega and regb.

xor R3 R1 R2:



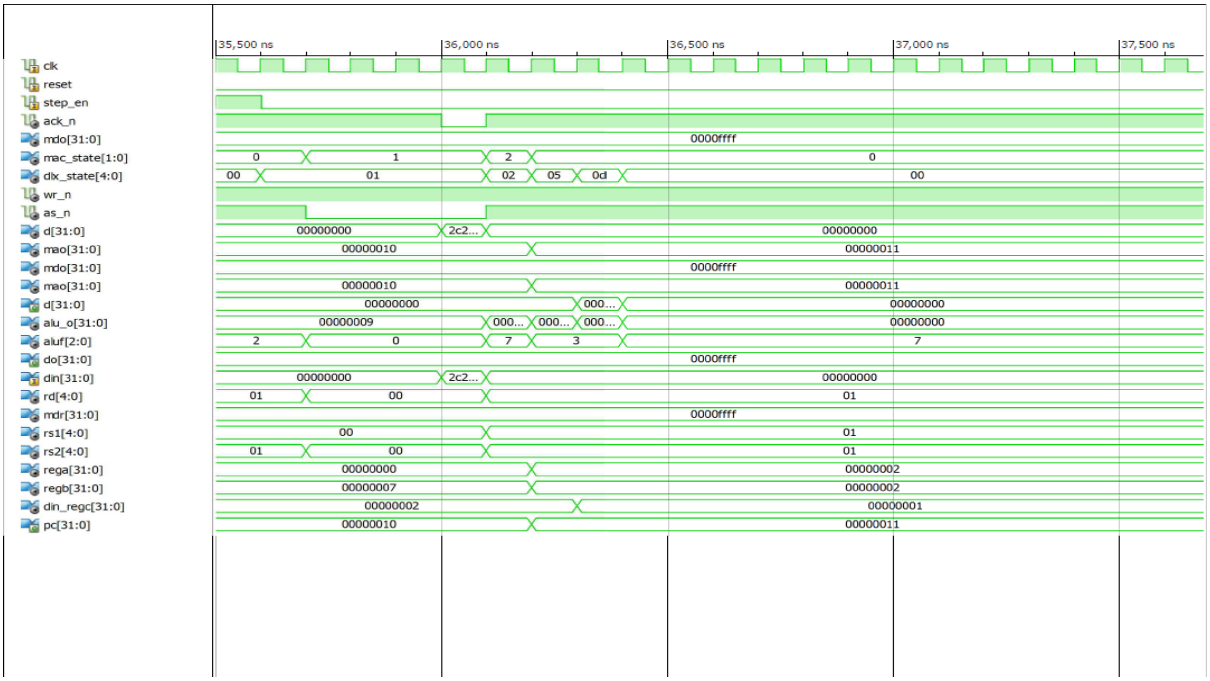
And finally in the xor instruction we got in din_regc the xor between rega and regb.

addi R1 R0 0x2:



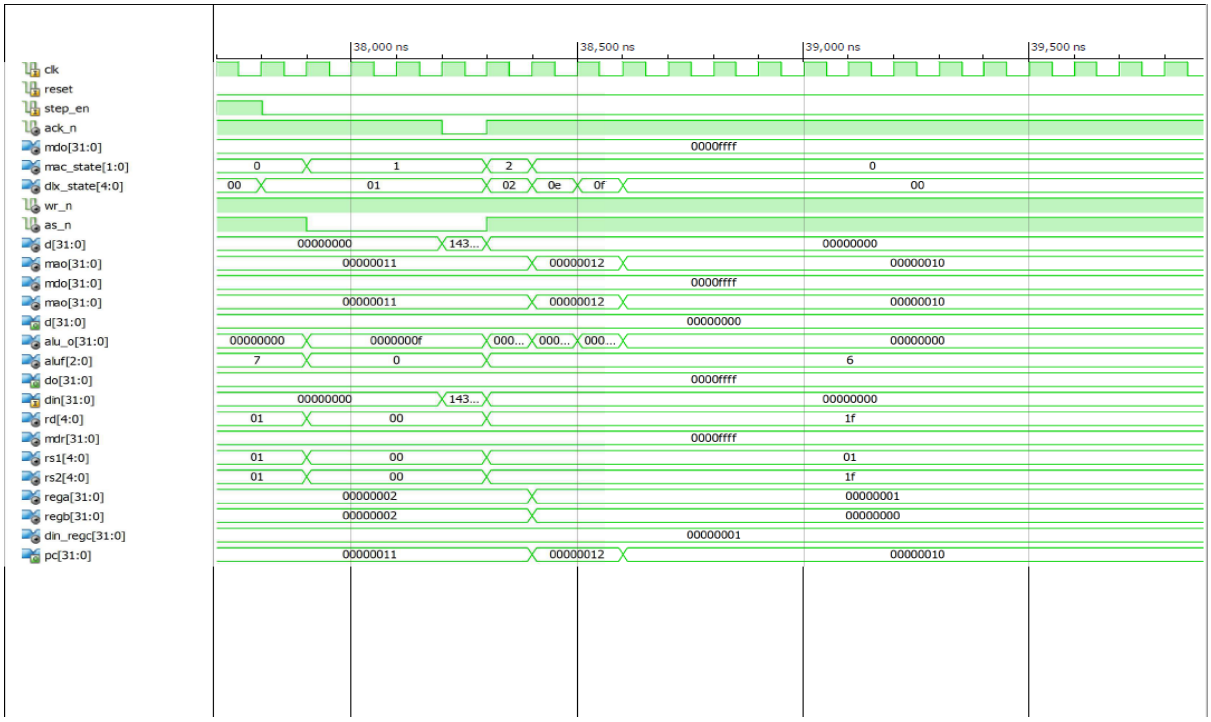
We can see the din_regc is getting 2 as requested .

addi R1 R1 0xffff:



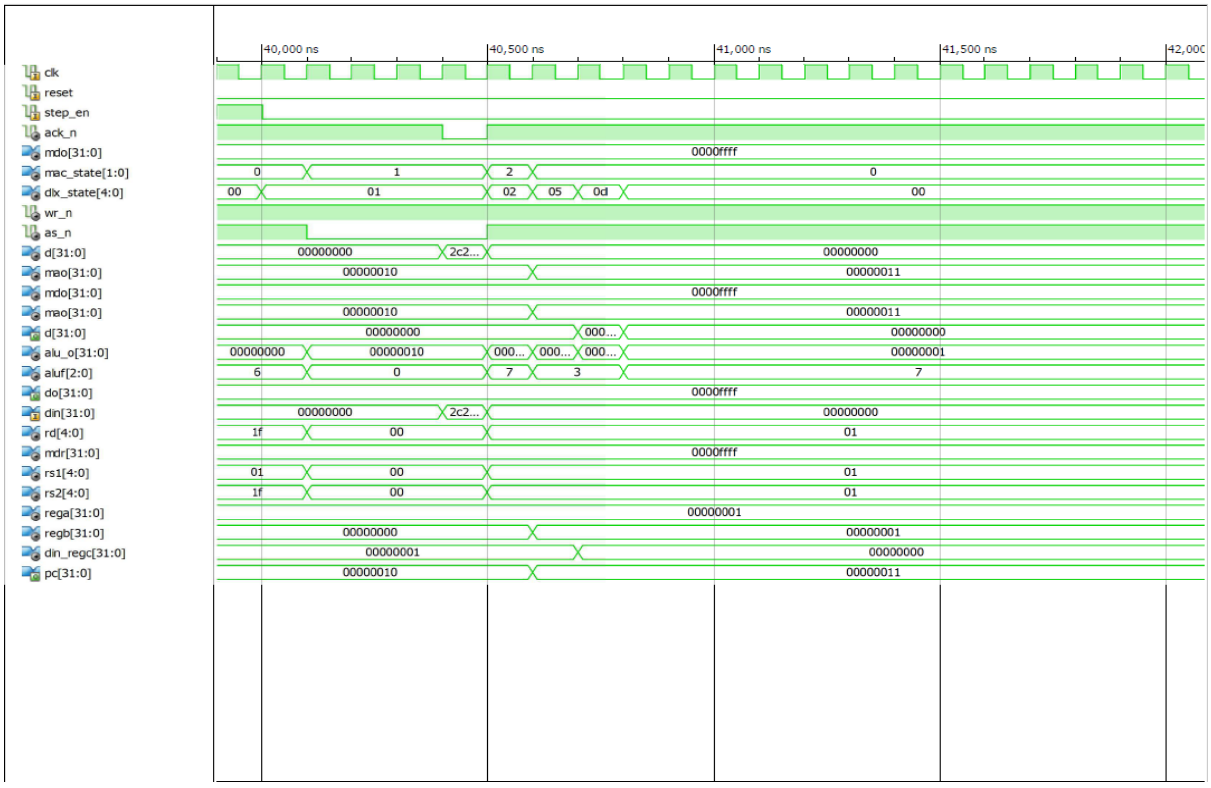
In this instruction we tried adding a negative number which is 0xffffffff (-1), and we can see that it worked is desired, as din_regc is 1 now.

bnez R1 0xfffe:



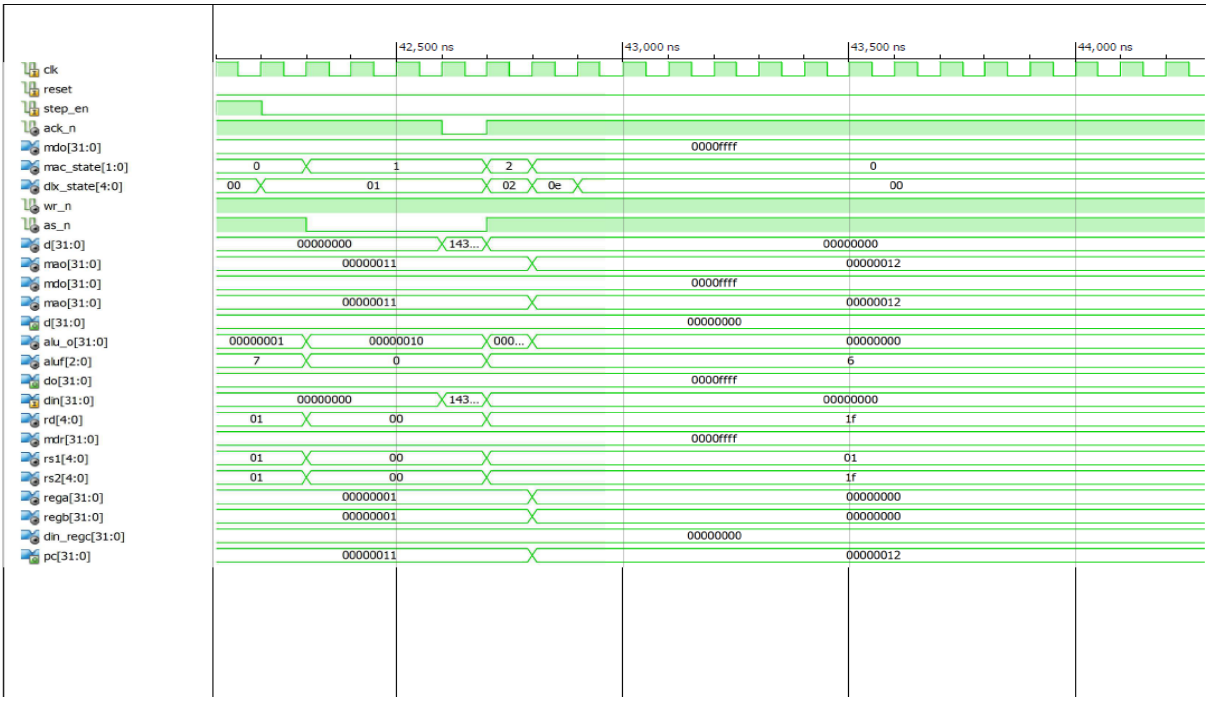
In this instruction we said if R1 (the register from the instruction before), is not 0, then we want to jump to the instruction before which was ”addi R1 R1 0xffff”, and we can see the that pc changed accordingly.

addi R1 R1 0xffff:



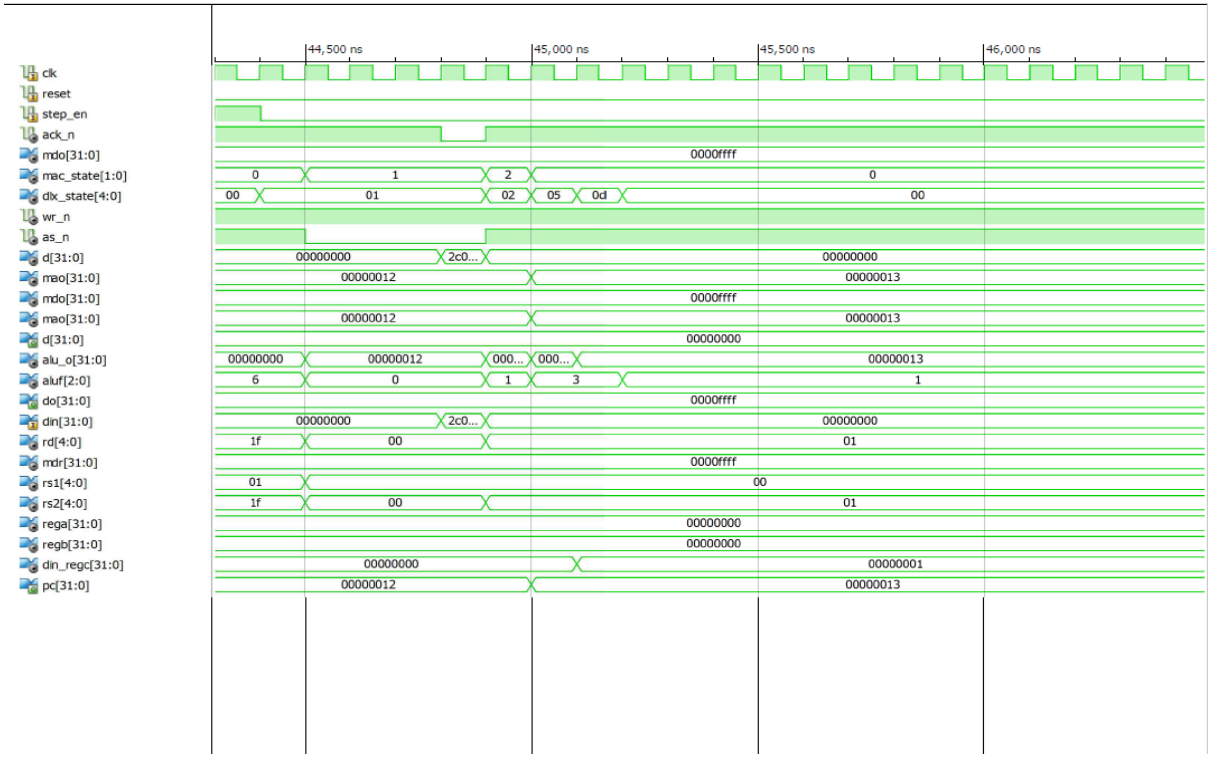
We can see that after doing this instruction again we got R1=0.

bnez R1 0xfffe:



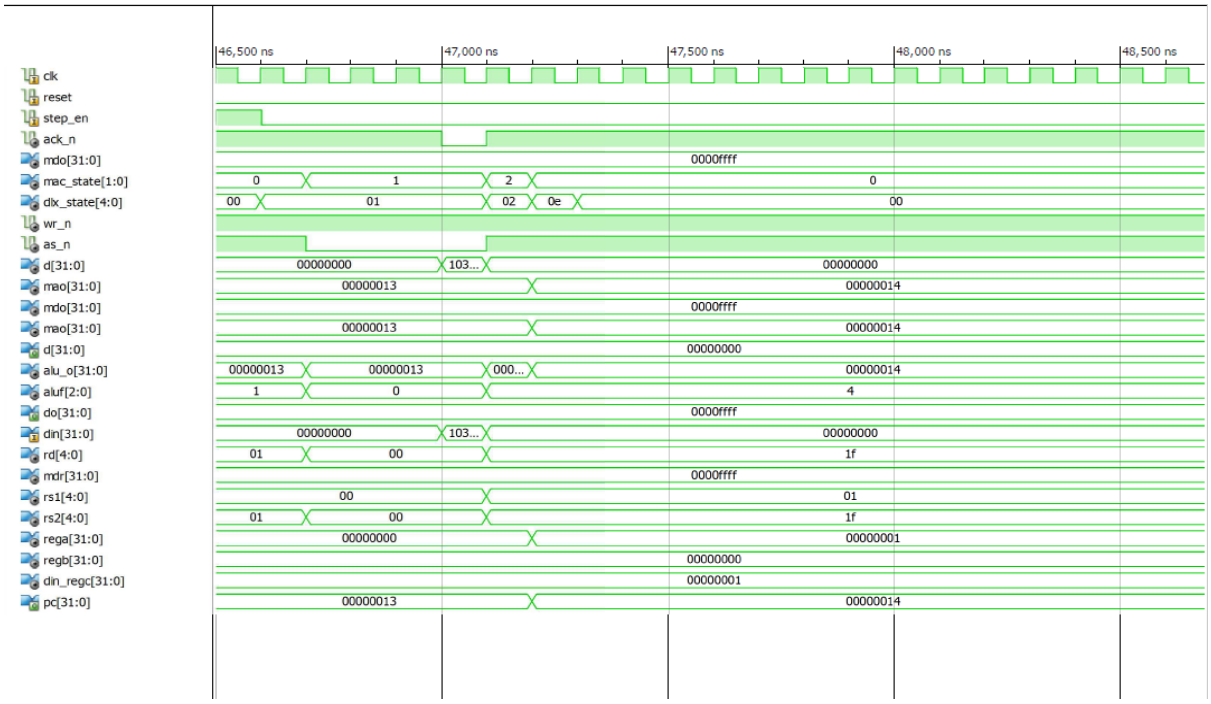
Now because R1=0, the instruction did not do anything and just moved on to the next instruction.

addi R1 R0 0x1:



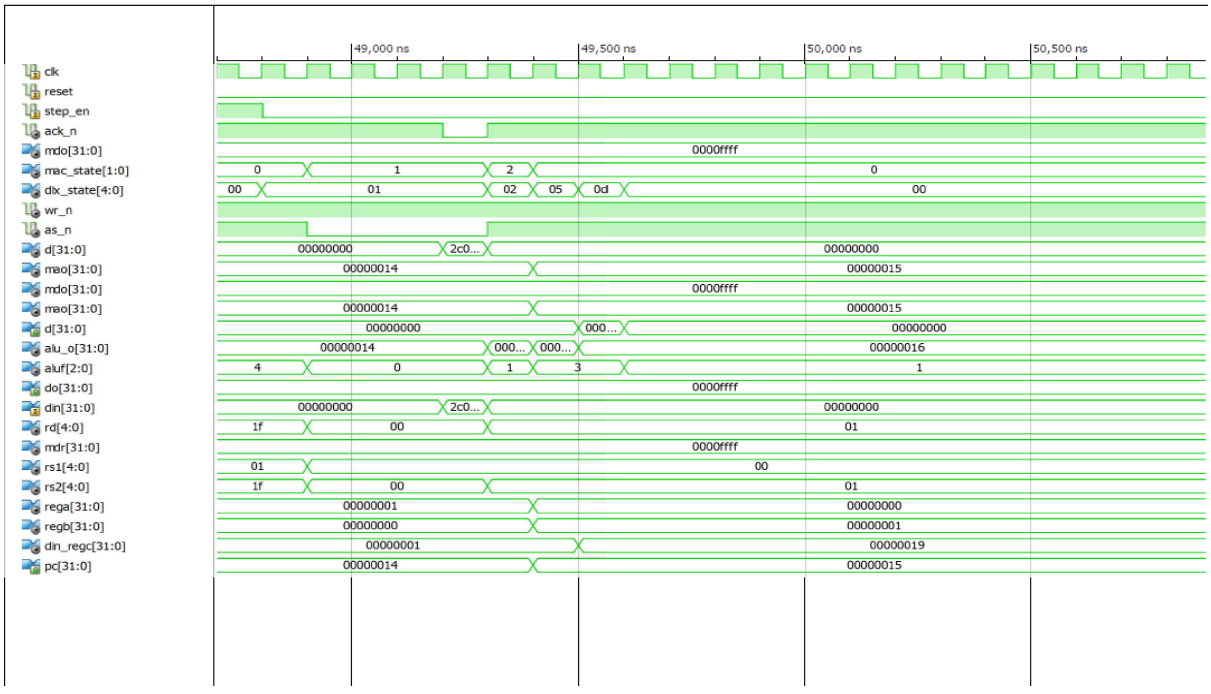
Now we assigned R1 to the value of 1 , and we can see that in the signal din_regc , that it got the value of 1.

beqz R1 end:



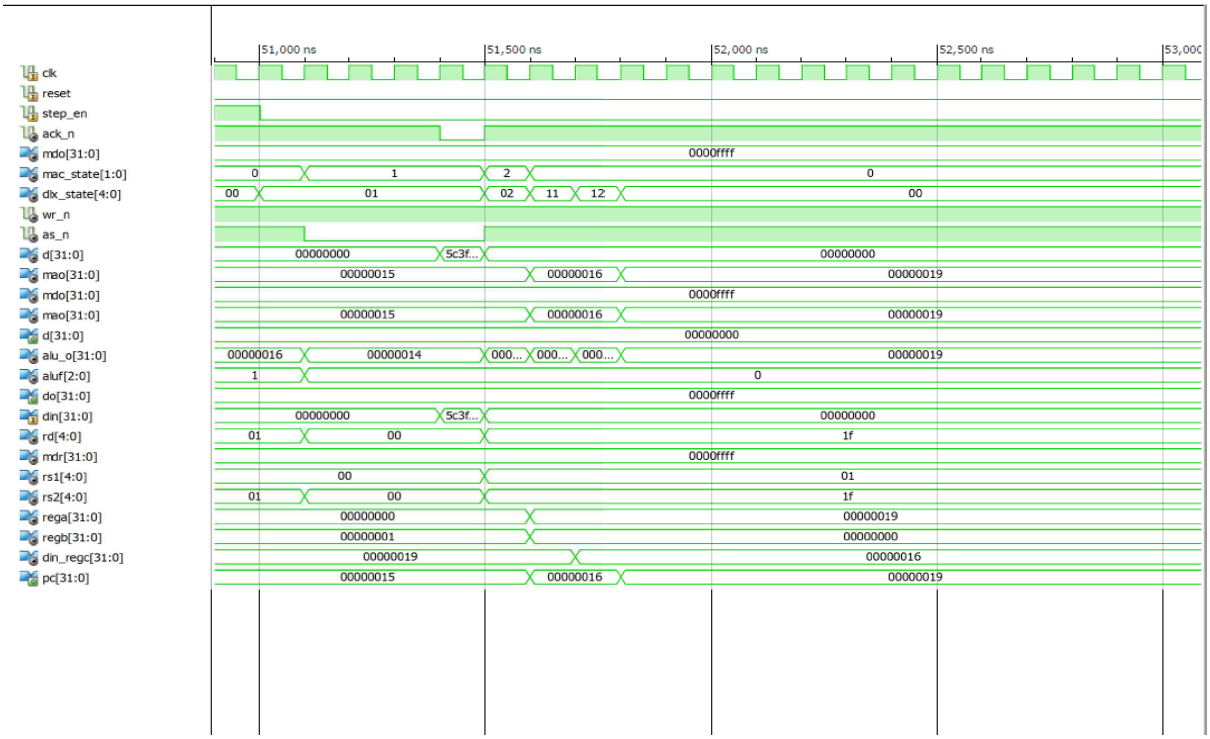
We can see that because R1/=0 , the instruction did not do anything , and it just moved on to the next instruction, as we can see from the pc signal.

addi R1 R0 testi:



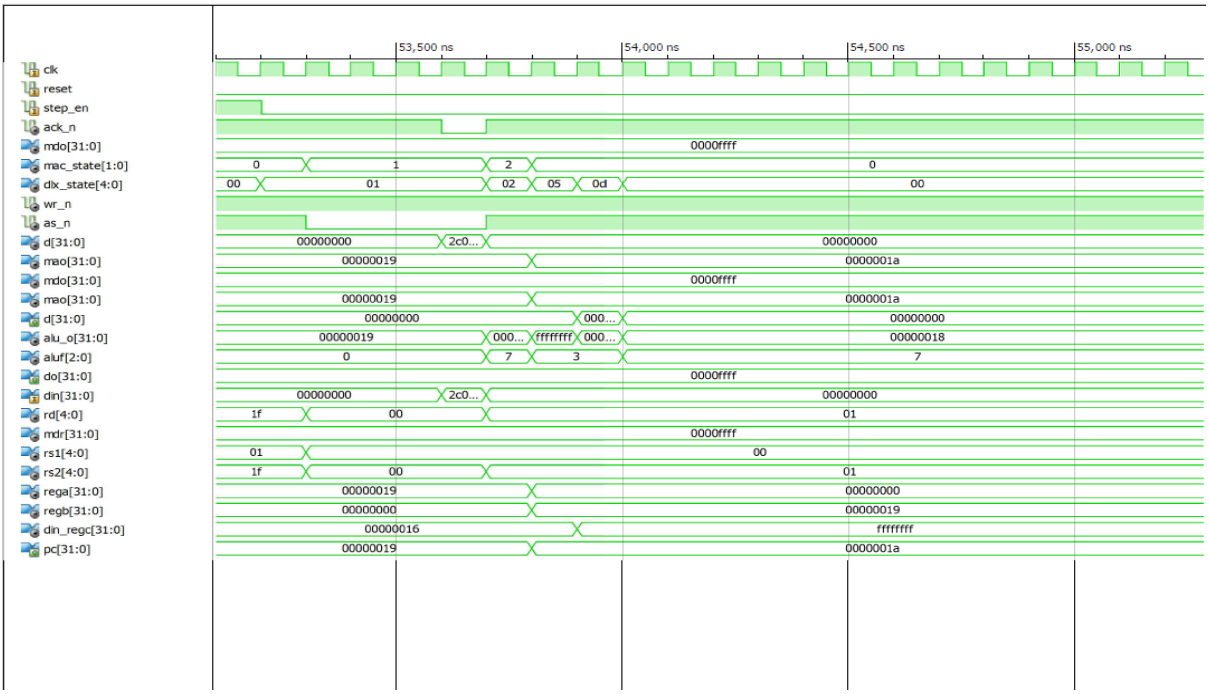
Here we assigned R1 to the value of the address of testi which is 0x00000019, as we can see from din_regc.

jalr R1:



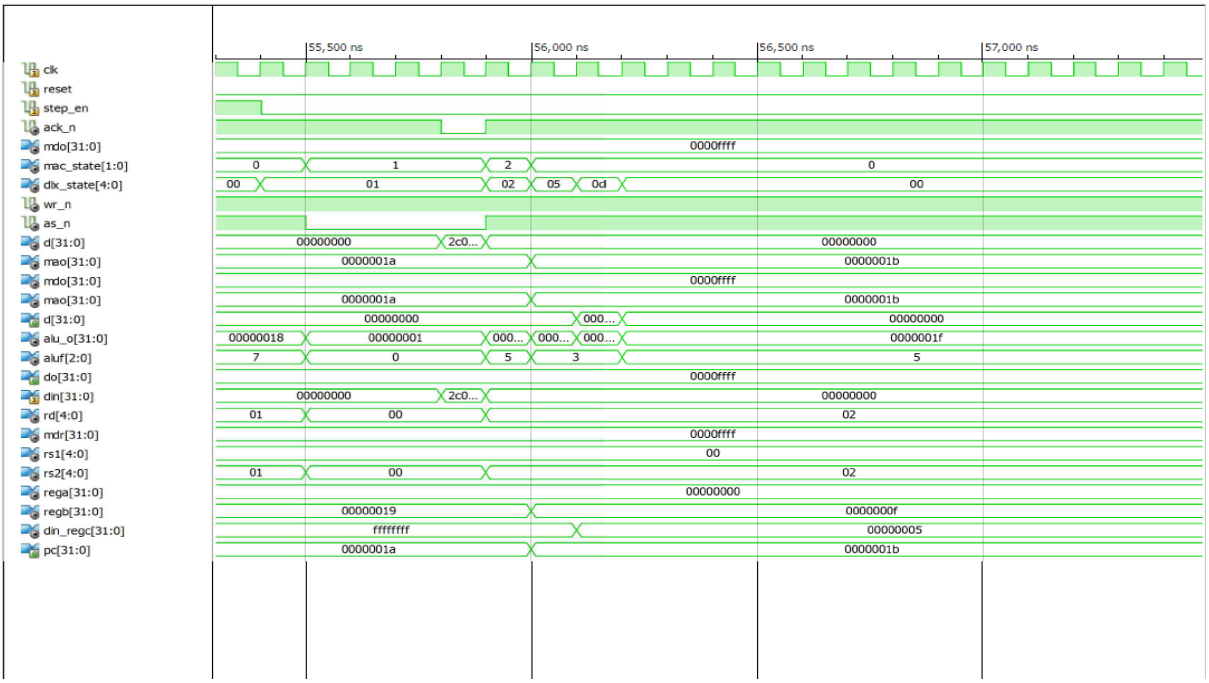
Here we can see that the pc jumped to the address of testi which is 0x00000019, and because R31 in this instruction will get pc+1, we can see that in din_regc that the value of the next pc is going to be assigned to R31.

addi R1 R0 0xffff:



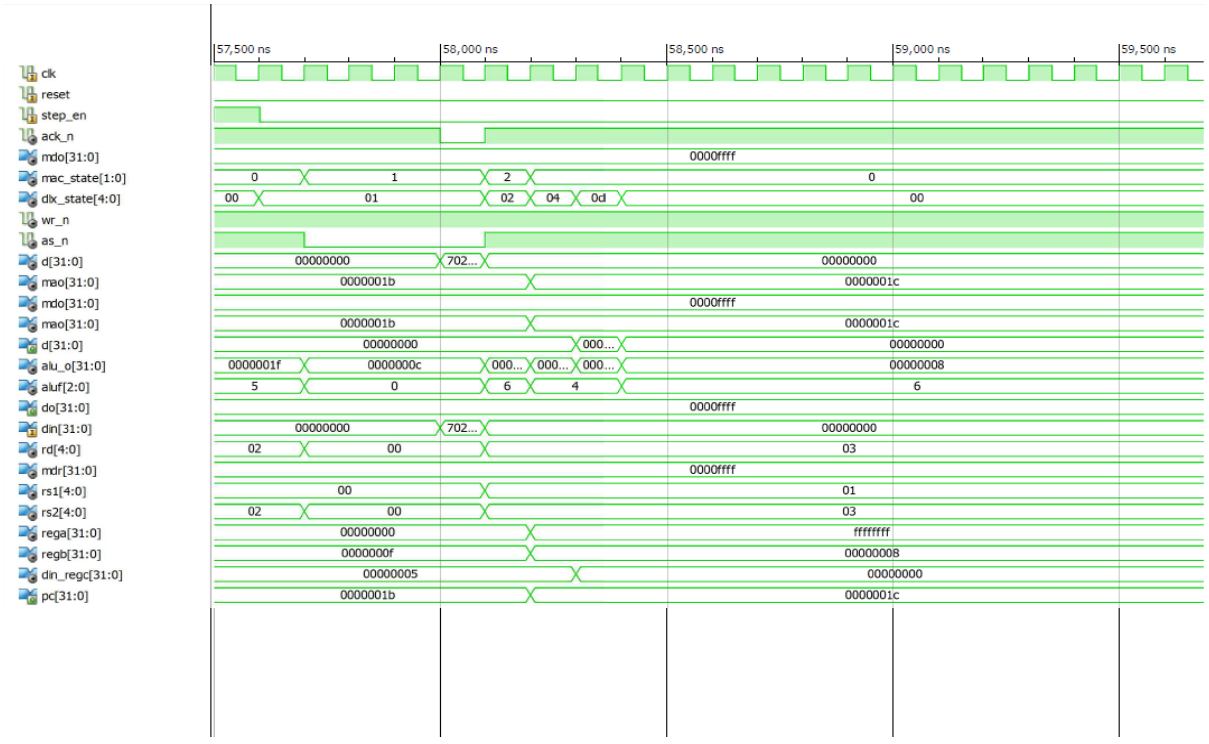
Now we are in pc=0x00000019, and we can see fron din_regc ,that we are assigning 0xffffffff (-1) to R1.

addi R2 R0 0x5:



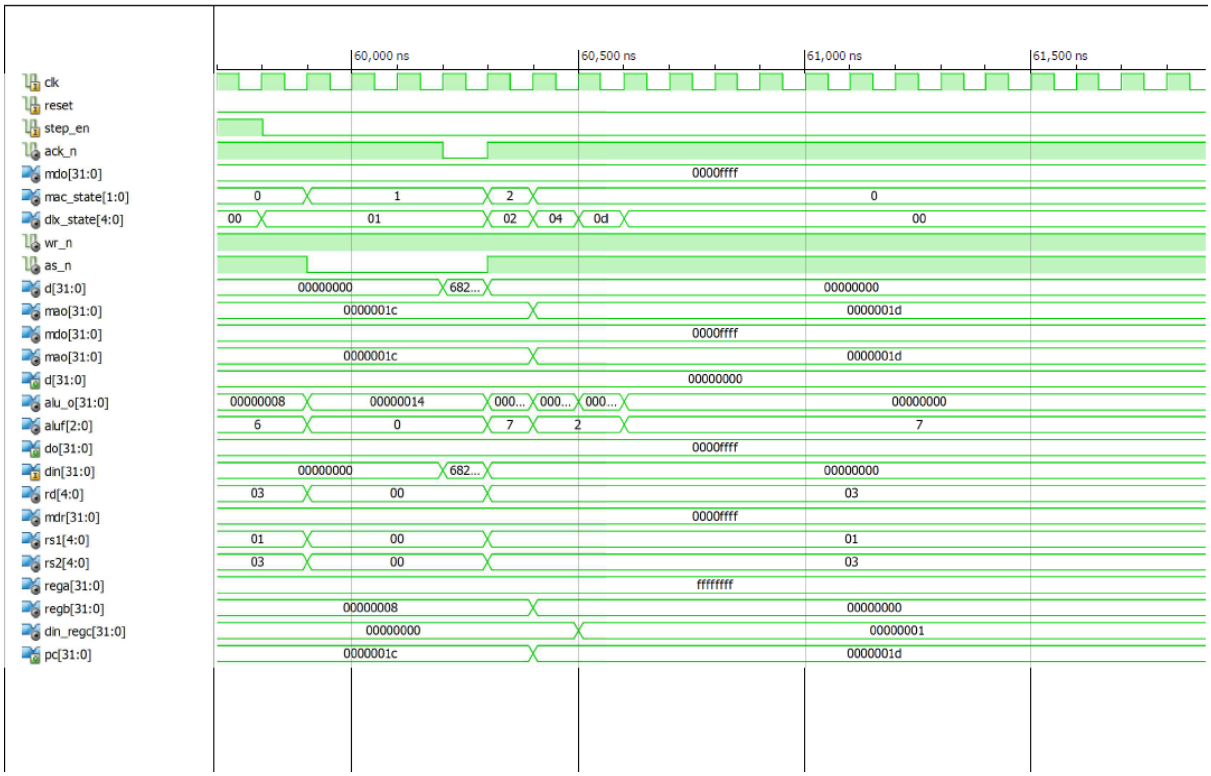
Now we are assigning R2 to the value of 5 as we can see from din_regc.

slti R3 R1 0xfffe:



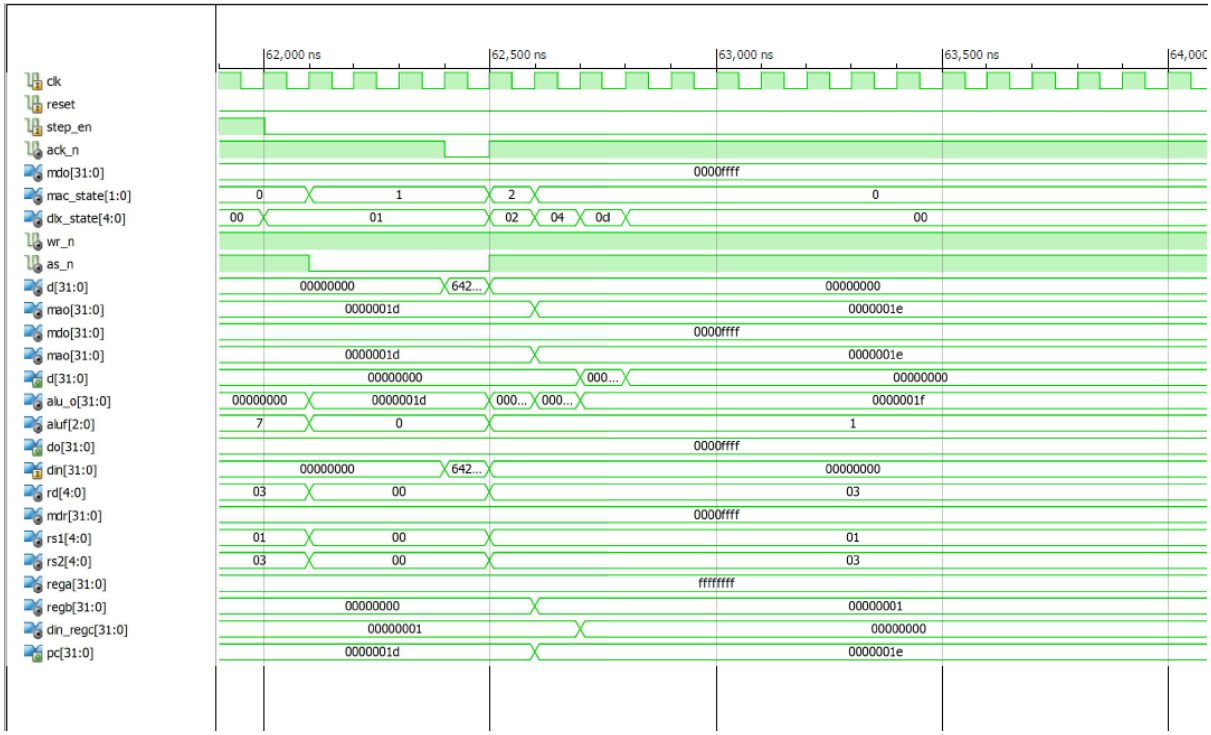
Now we are comparing between R1 (which is -1 as we can see its value in rega), and between the immediate value of 0xfffffffe which is -2, and because -1 is not less than -2 , we got 0 in din_regc which is assigned then to R3.

seqi R3 R1 0xffff:



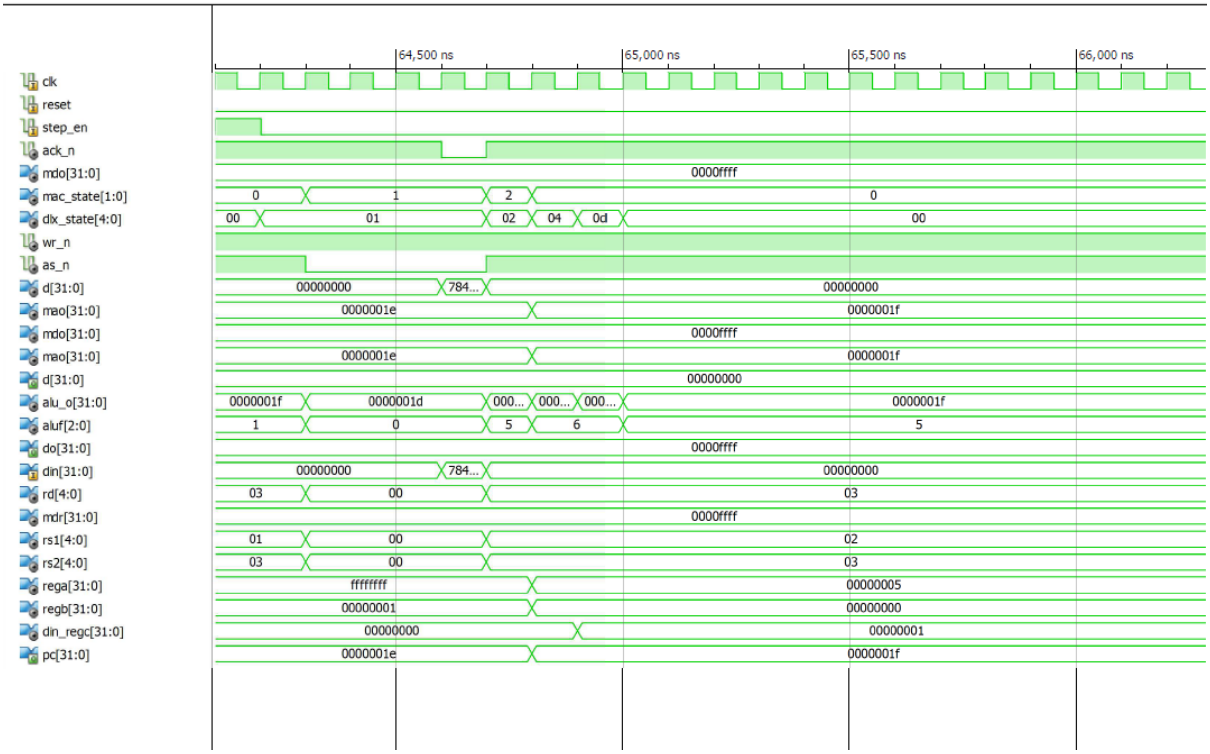
Now because R1 is equal to -1 , we get 1 in din_regc.

sgti R3 R1 0x1:



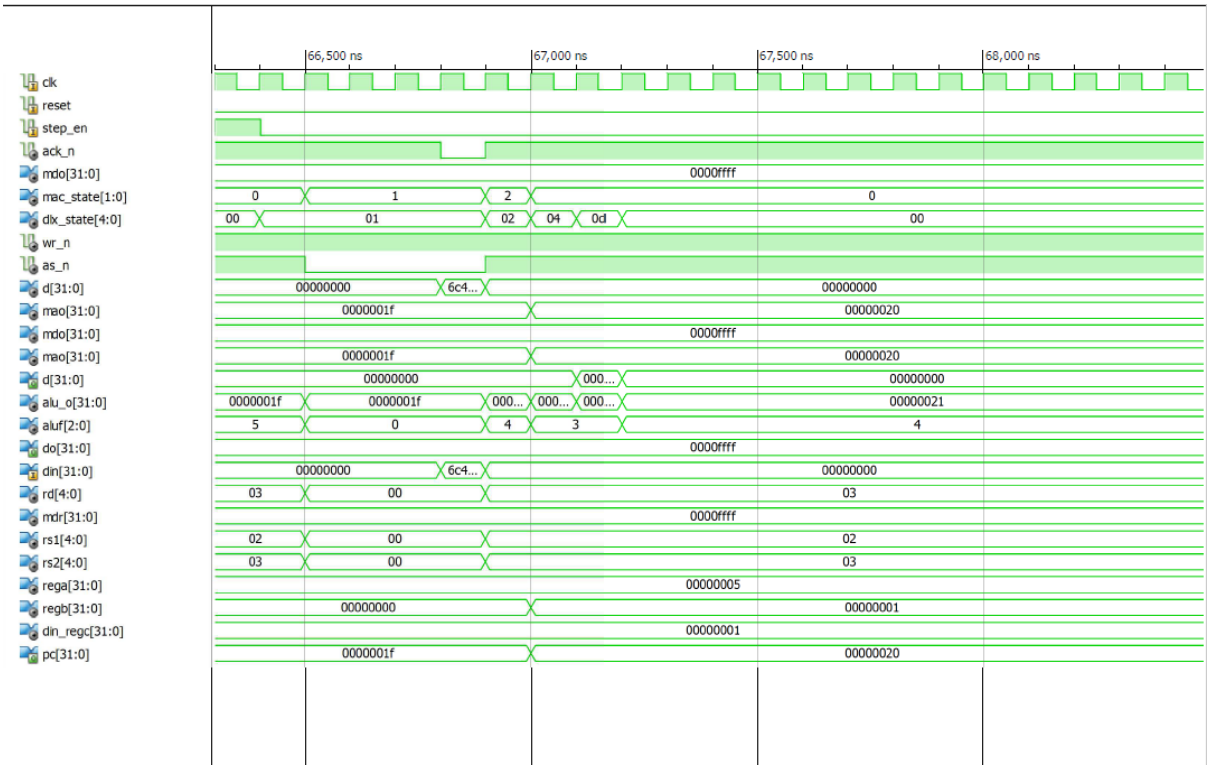
Now because 0x1 the immediate is greater than R1 , we get 0 in din_regc.

slei R3 R2 0x5:



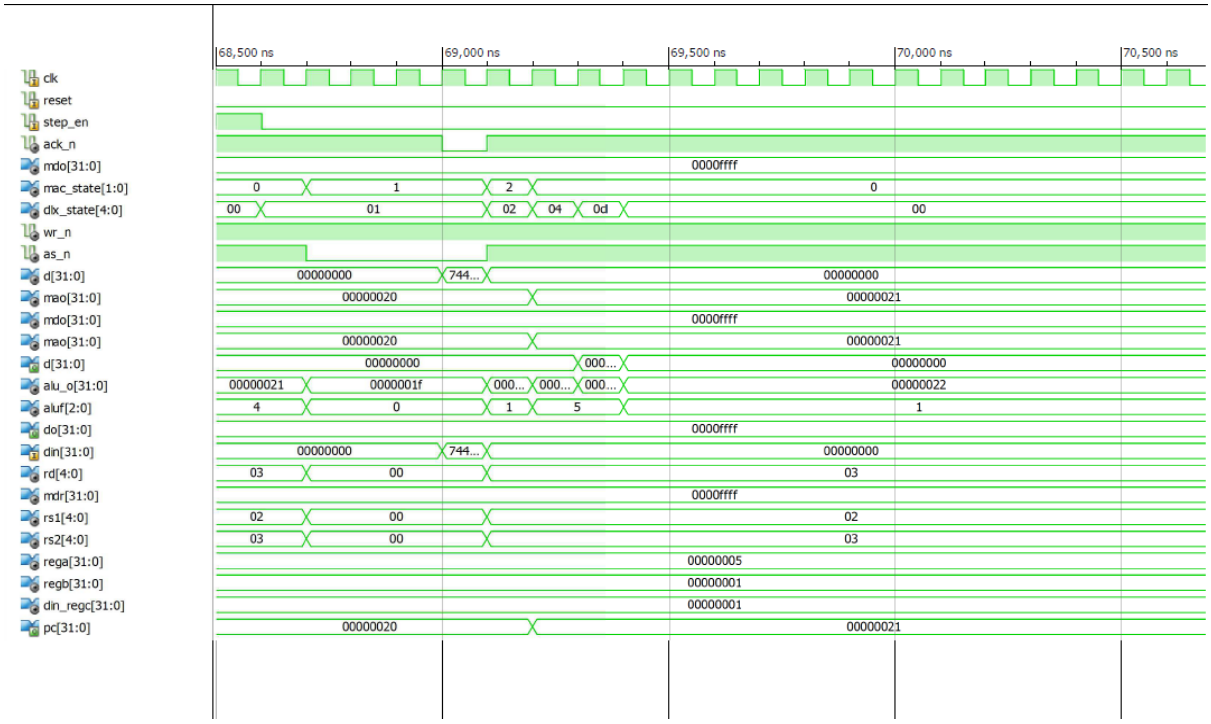
Now we are comparing between R2 (which is 5 as we can see its value in rega), and between the immediate value of also 5, now because the are both equal , we got 1 in din_regc which is assigned then to R3.

sgei R3 R2 0x4:



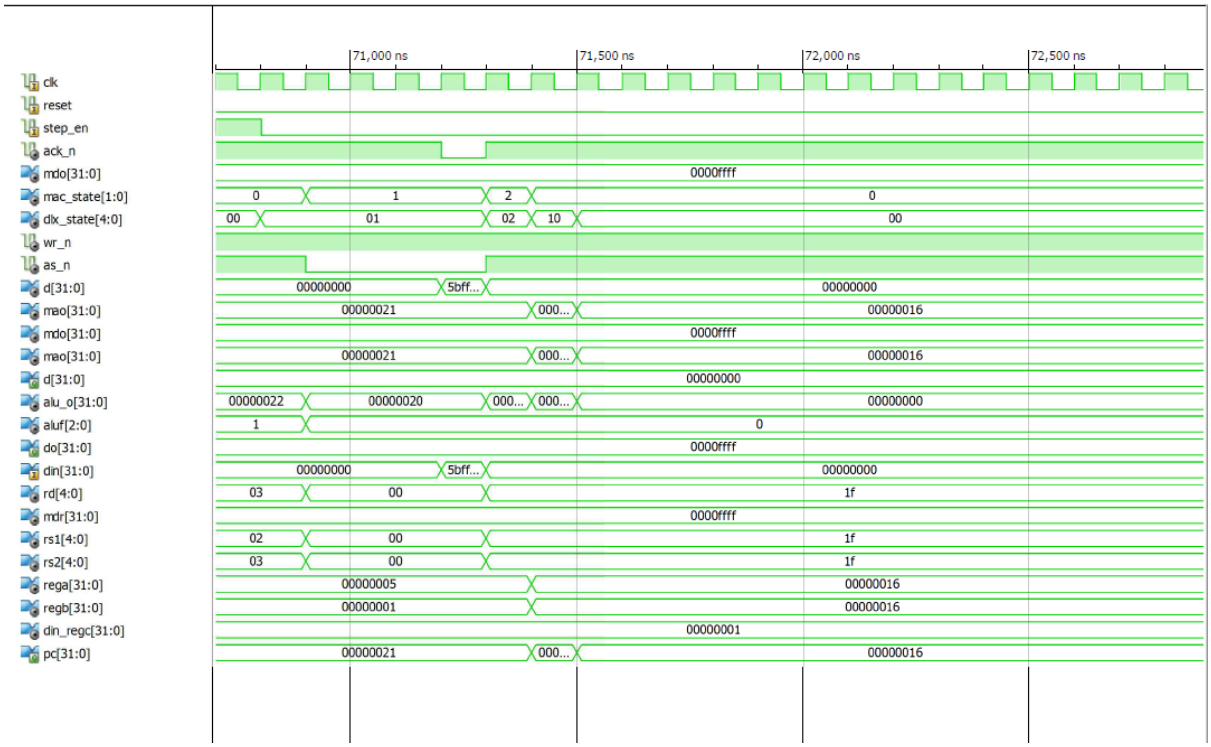
Now because R2 which is 5 , grater than 4 , we got 1 in din_regc.

snei R3 R2 0x9:



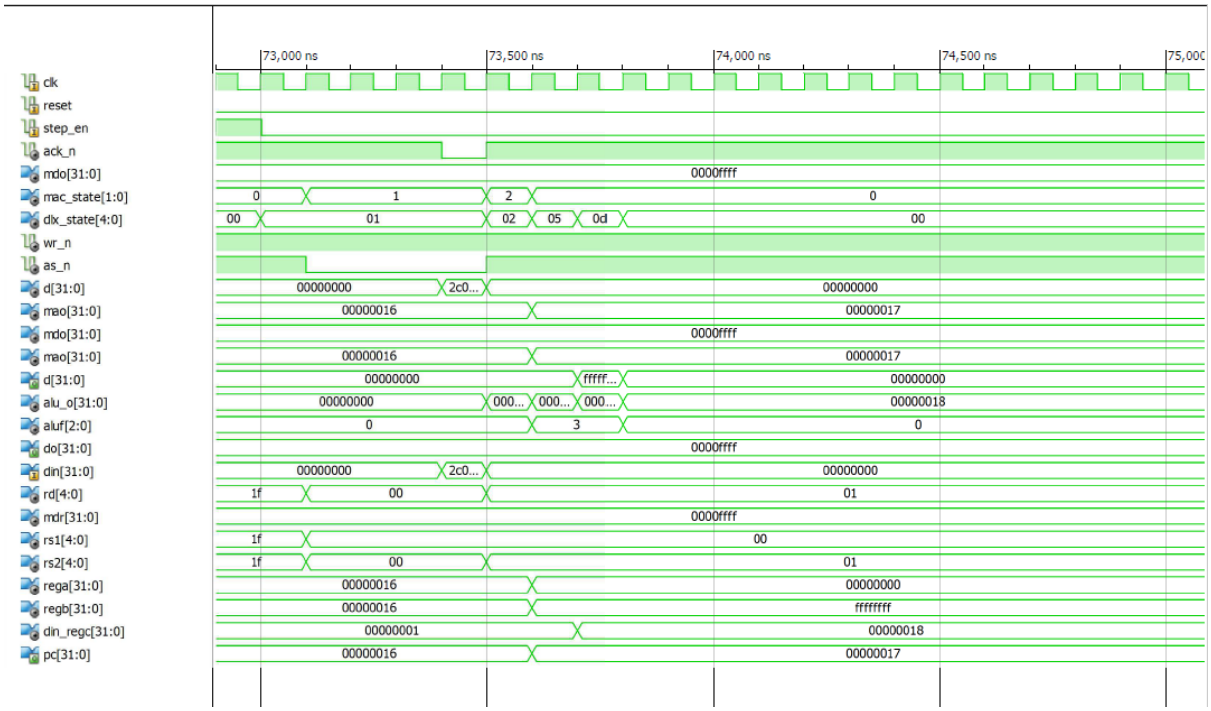
R2 is not equal to 9 so we also get 1 here in din_regc.

jr R31:



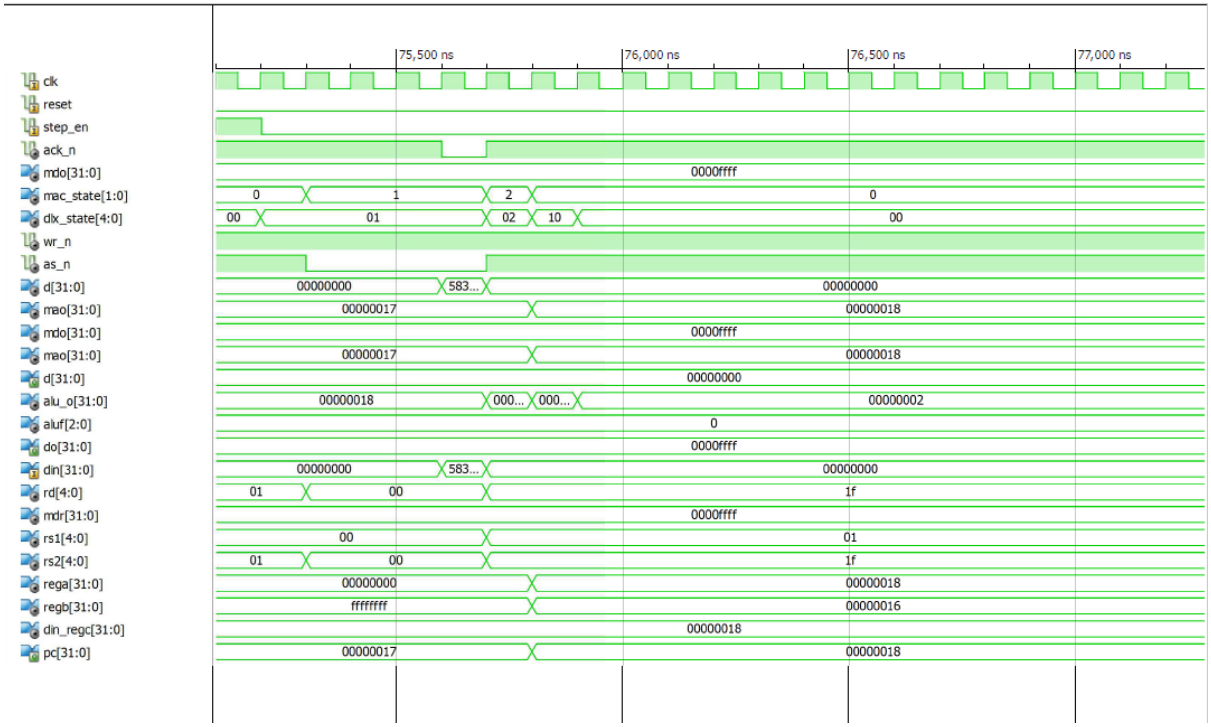
Now we are jumping to the address value of R31, which was saved earlier in it and it was 0x00000016, and we can see that in the pc signal as desired.

addi R1 R0 end:



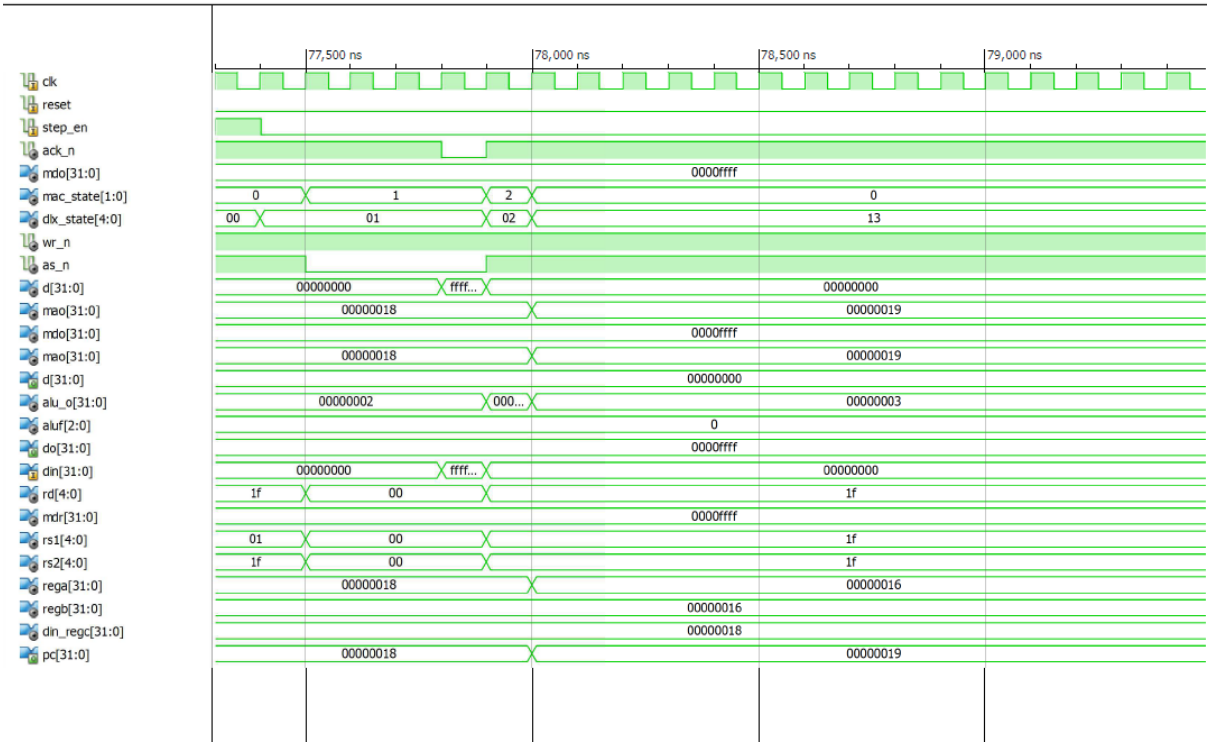
We are assigning R1 to the value of od the address of end which is 0x00000018, and we can see that in din_regc.

jr R1:



We are jumping to the address of end as we can see in the pc signal.

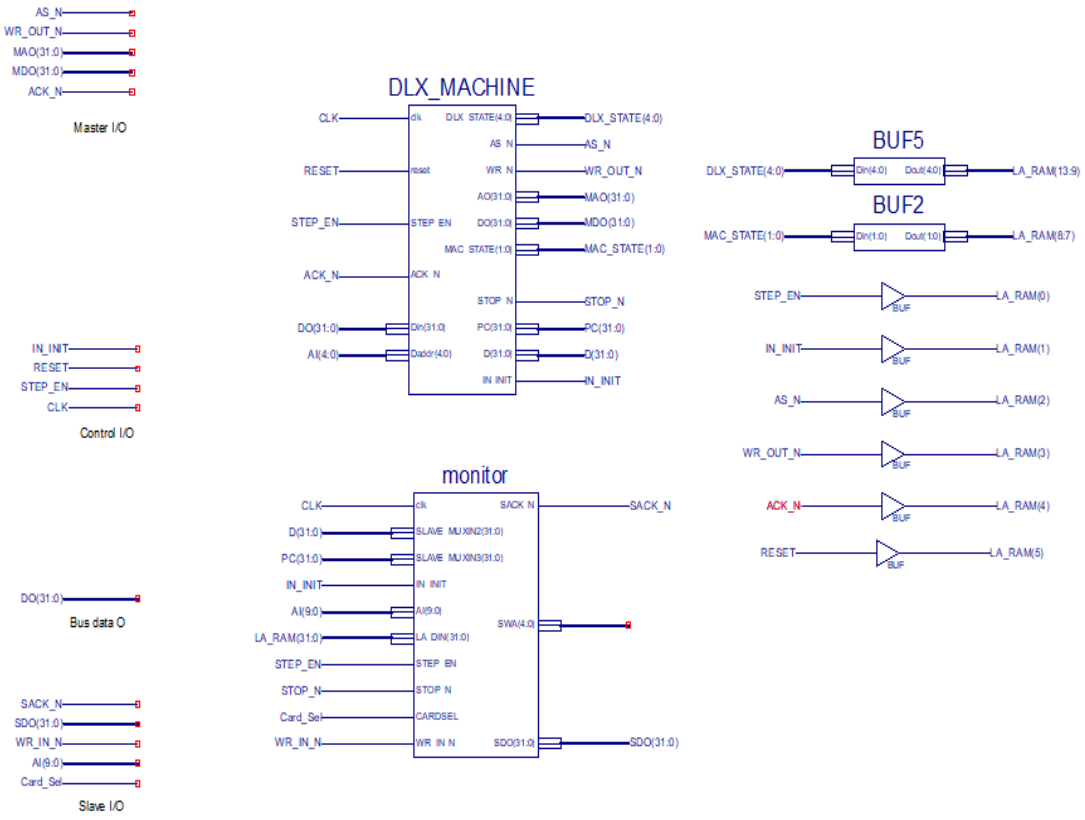
Halt:



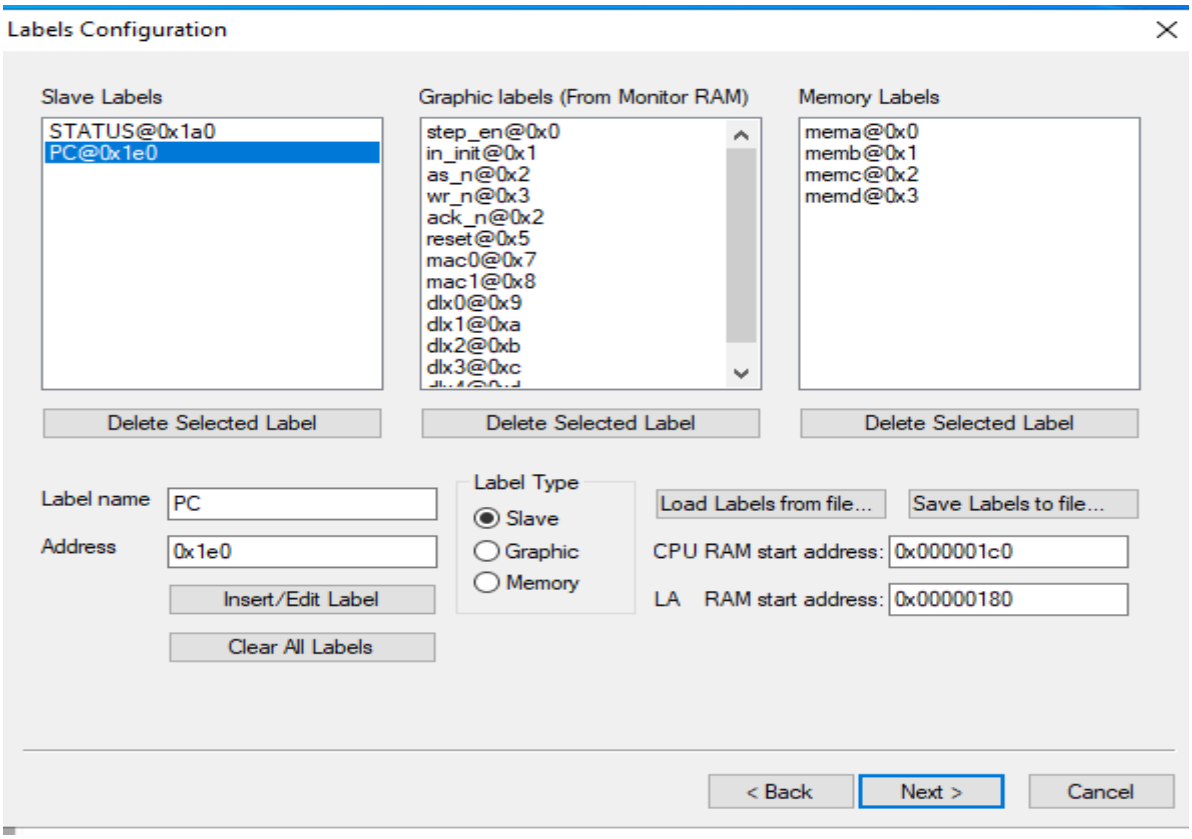
Finally the halt instruction, and we can see that the dlx states are switching as desired , and we stay in halt until we get a reset signal.

7.2 Implement and test your design on the RESA.

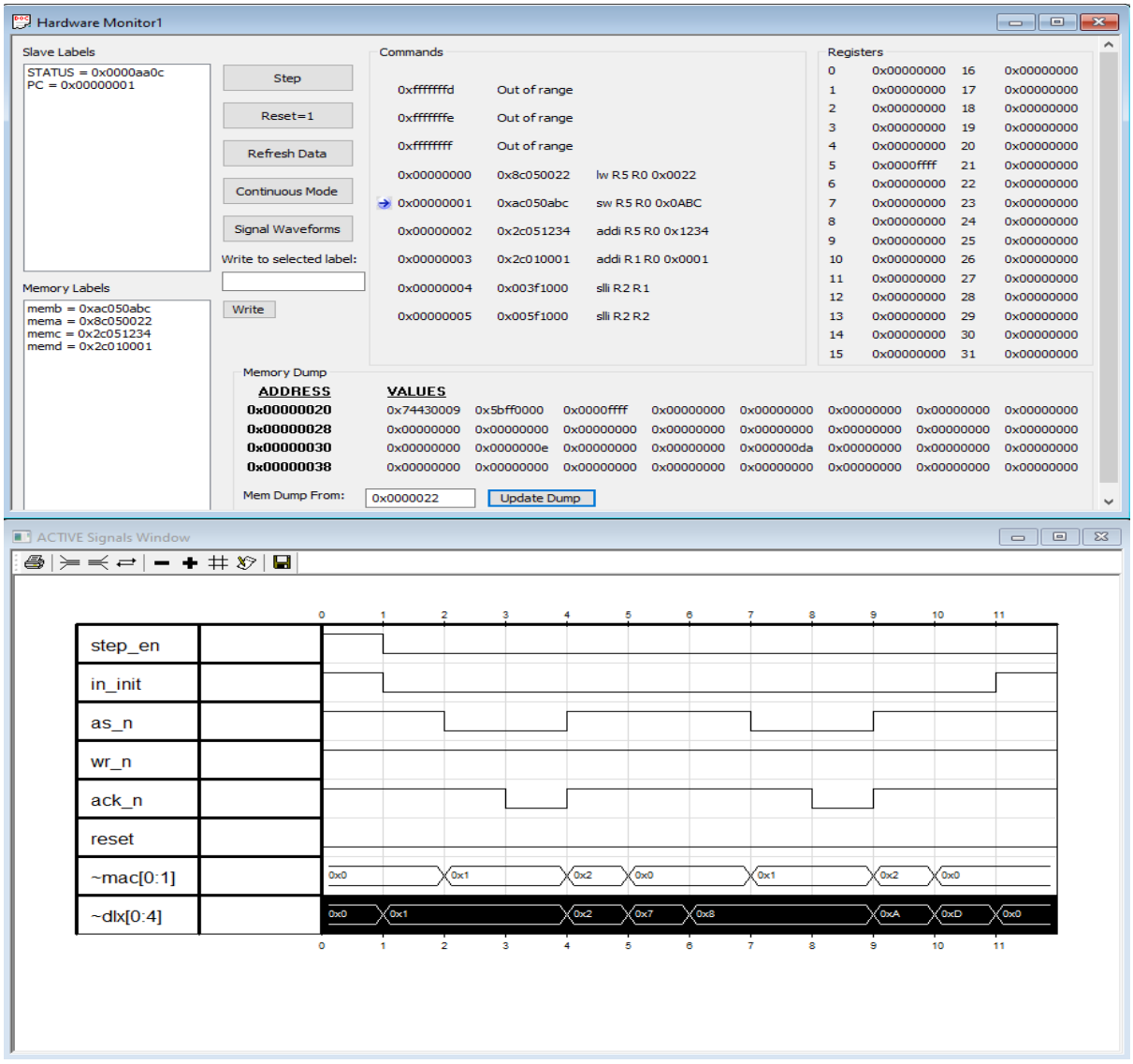
Our top level with the slave monitor and logic analyzer :

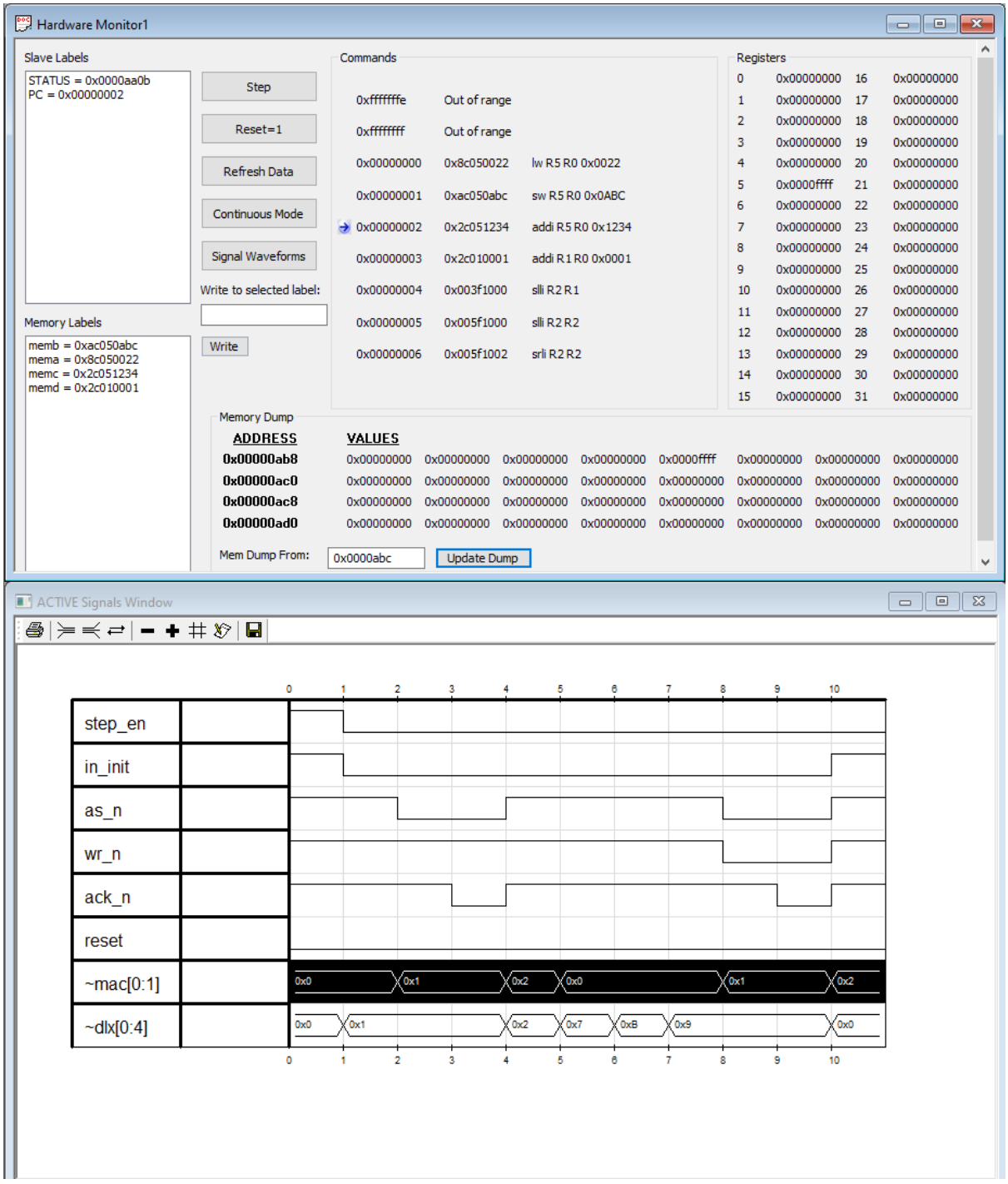


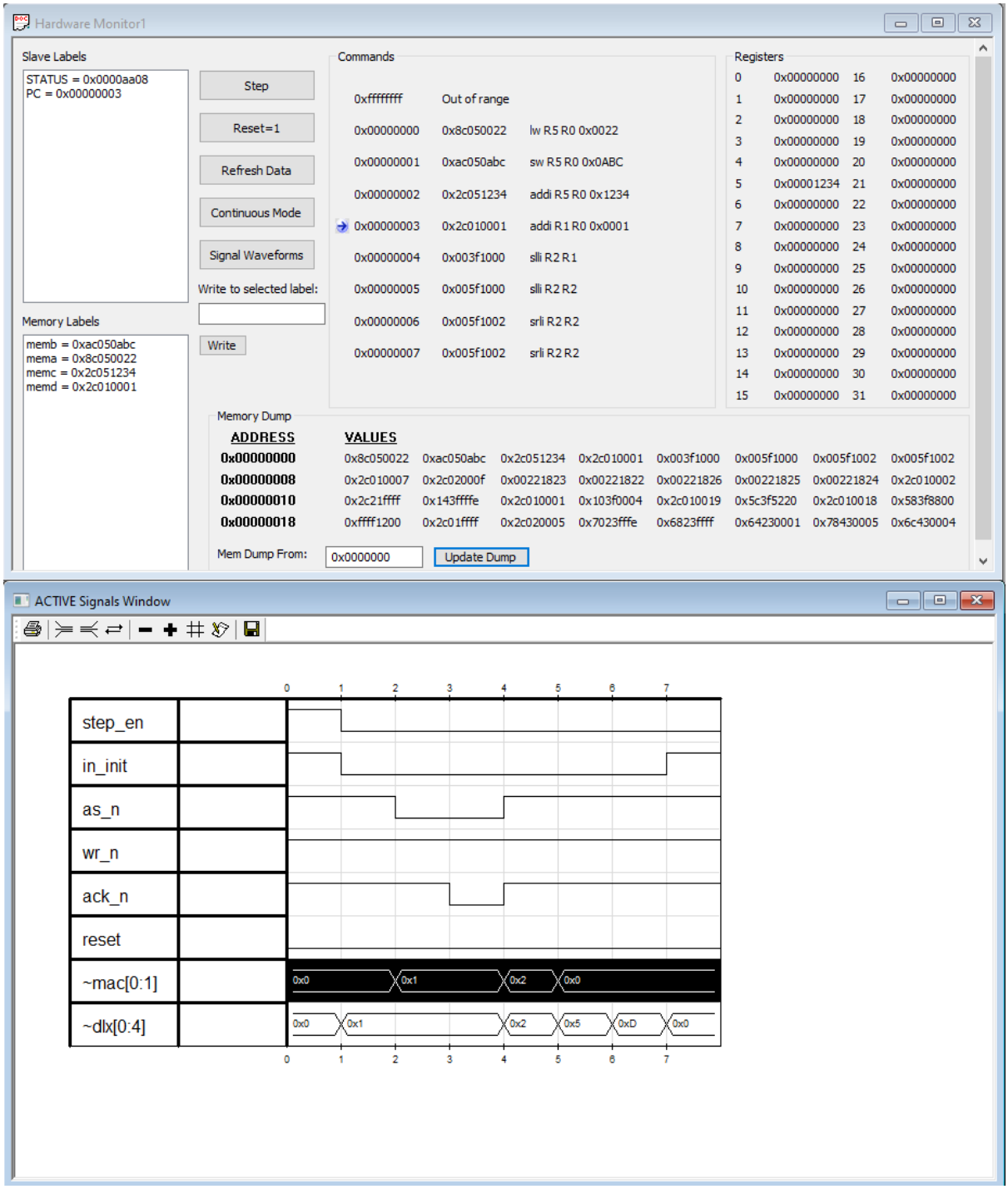
The labels that we used in RESA was as following:



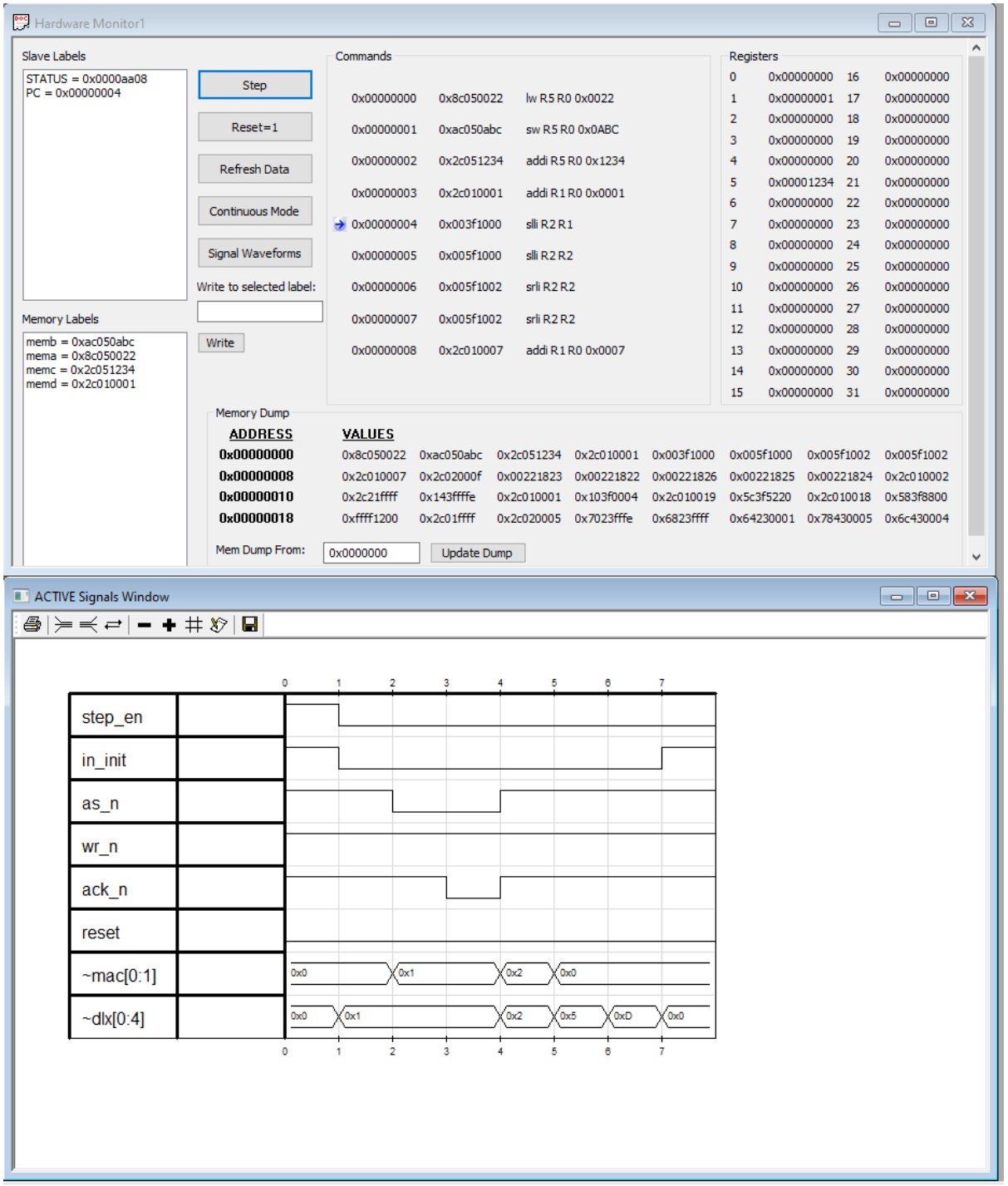
The address is on the PC label, The executed instruction is the one before the arrow, and we see that all instructions are carried out as per our requirements.

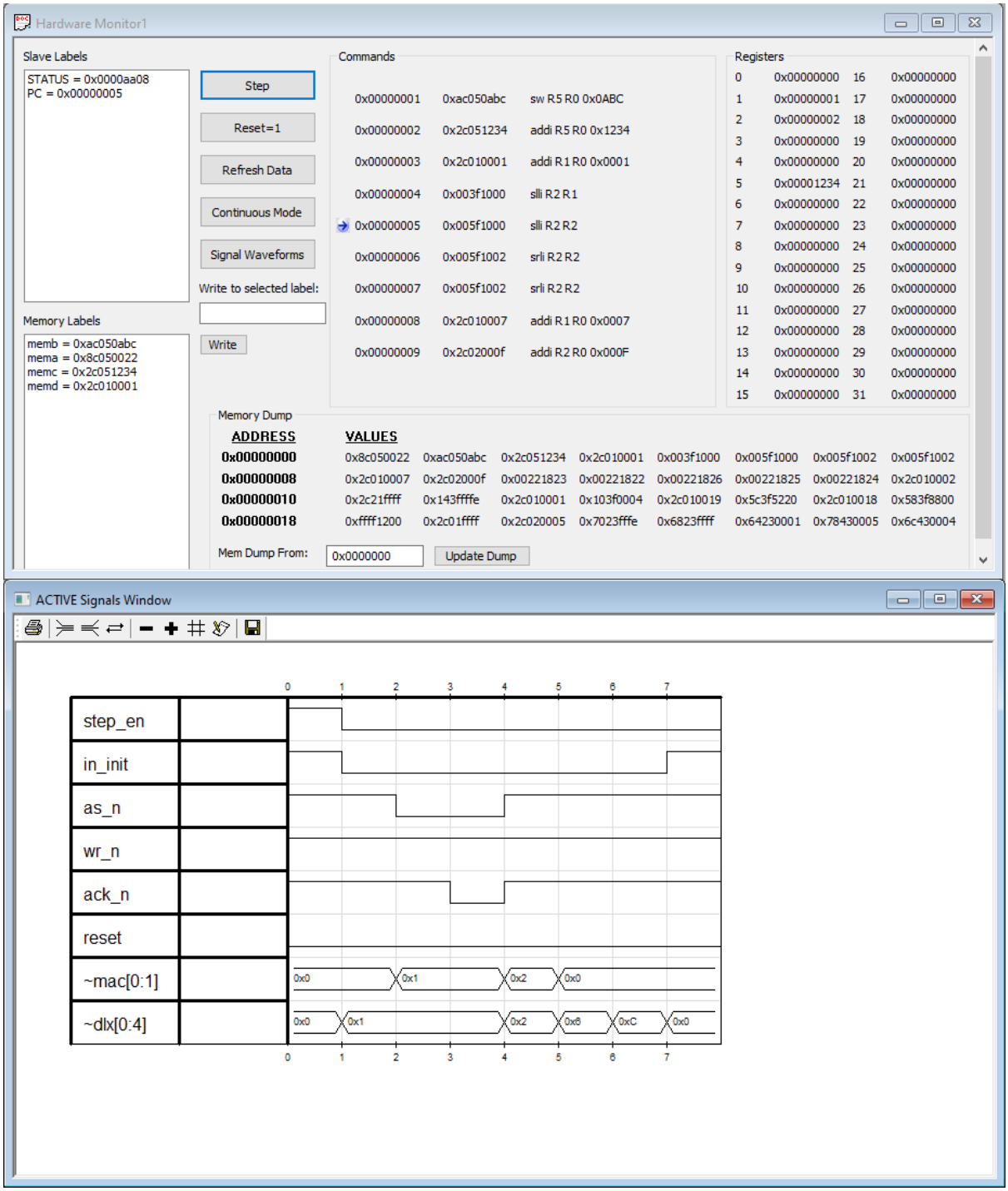


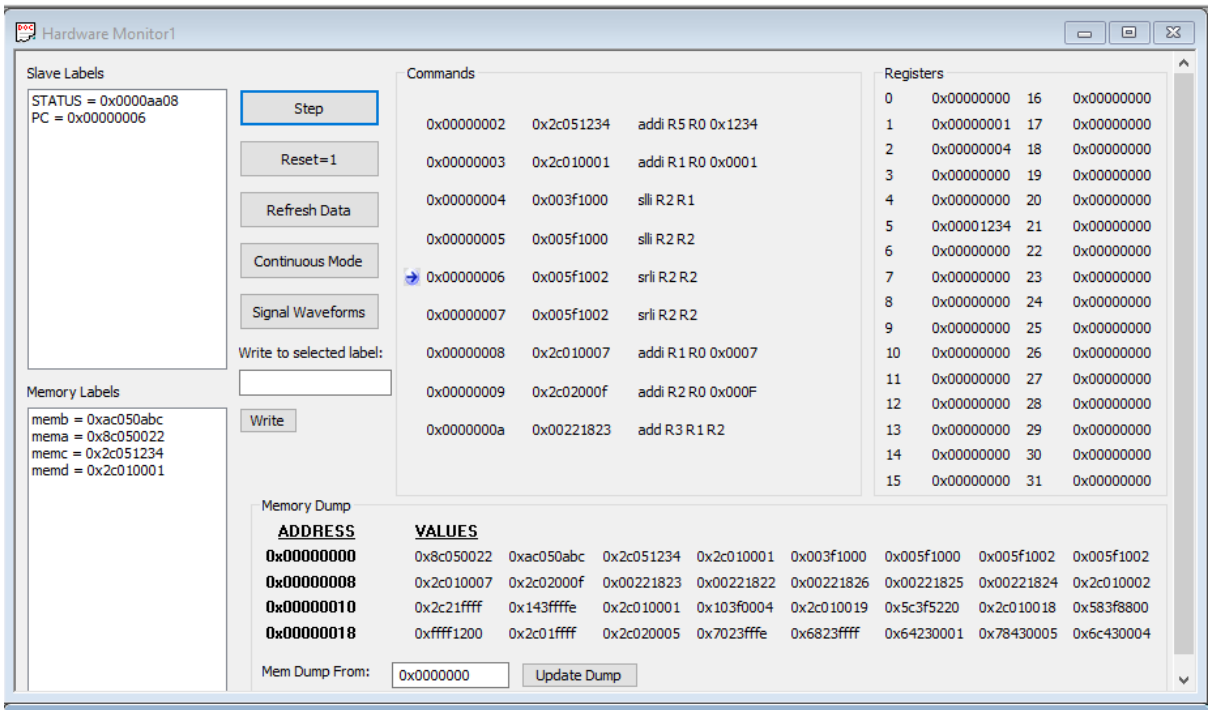

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa0b
PC = 0x00000002
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0xfffffffe Out of range
0xffffffff Out of range
0x00000000 0x8c050022 lw R5 R0 0x0022
0x00000001 0xac050abc sw R5 R0 0x0ABC
0x00000002 0x2c051234 addi R5 R0 0x1234
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000000 17 0x00000000
2 0x00000000 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x0000ffff 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000ab8 0x00000000 0x00000000 0x00000000 0x00000000 0x0000ffff 0x00000000 0x00000000 0x00000000
0x00000ac0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000ac8 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000ad0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
Mem Dump From: 0x0000abc
Update Dump
ACTIVE Signals Window
0 1 2 3 4 5 6 7 8 9 10
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0 0x1 0x2
~dlx[0:4]
0x0 0x1 0x2 0x7 0xB 0x9 0x0
0 1 2 3 4 5 6 7 8 9 10

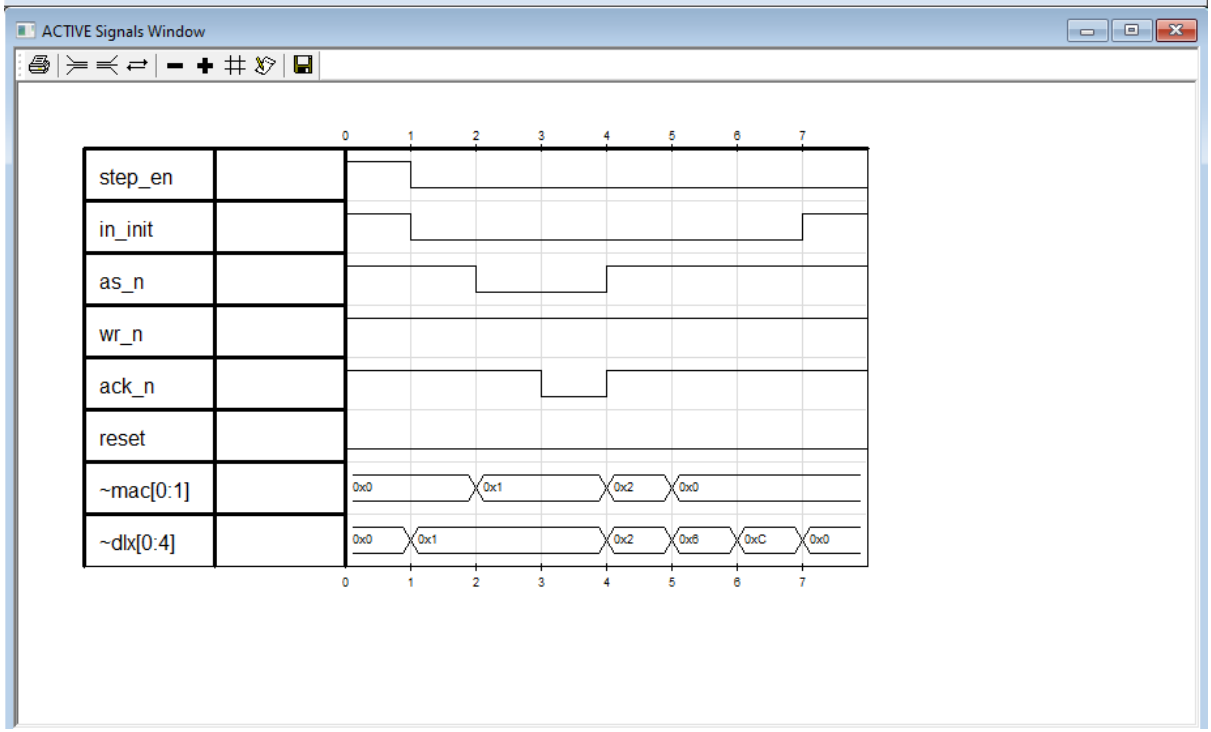

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000003
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0xffffffff Out of range
0x00000000 0x8c050022 lw R5 R0 0x0022
0x00000001 0xac050abc sw R5 R0 0x0ABC
0x00000002 0x2c051234 addi R5 R0 0x1234
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000000 17 0x00000000
2 0x00000000 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

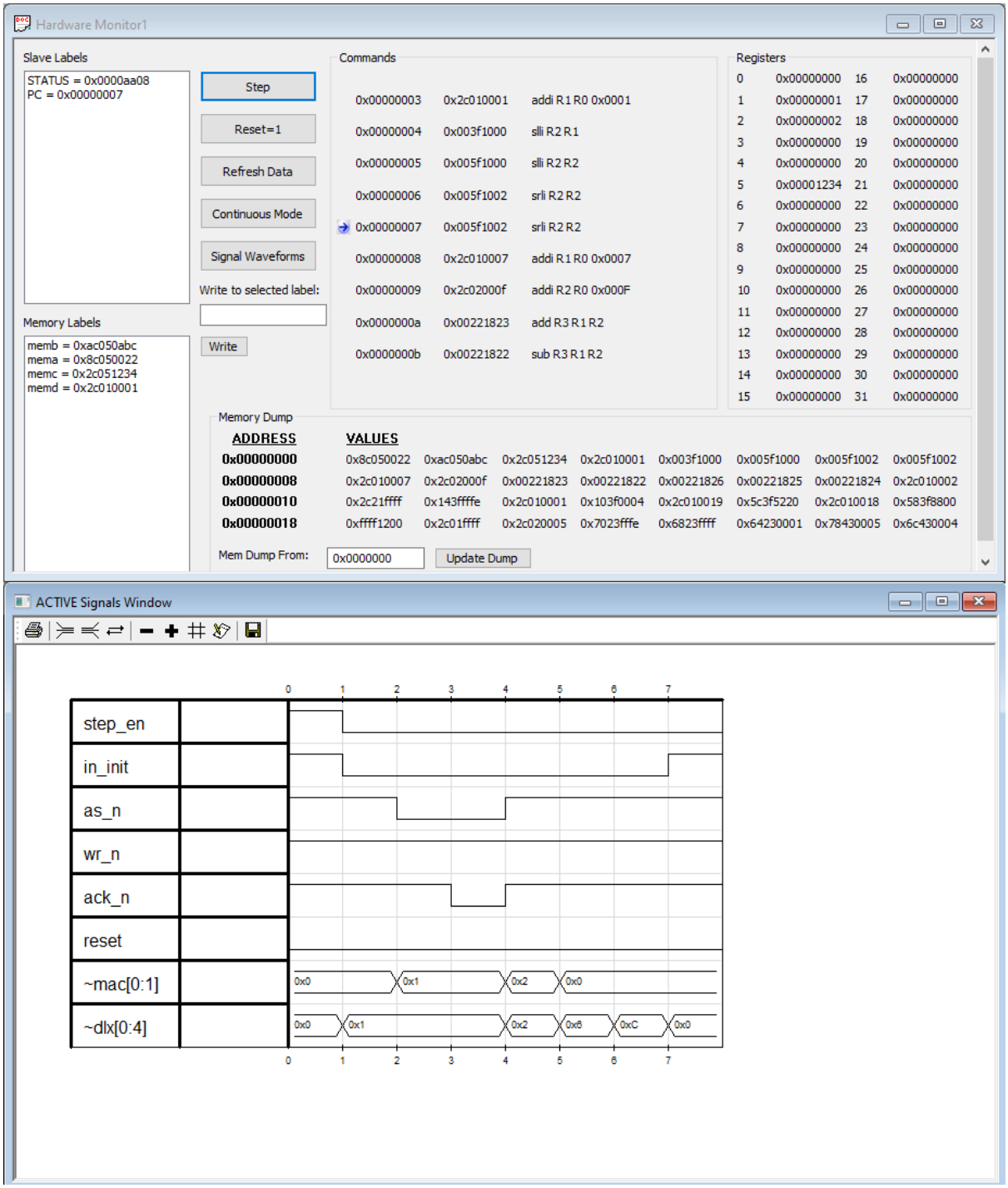
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x5 0xD 0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000004
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000000 0x8c050022 lw R5 R0 0x0022
0x00000001 0xac050abc sw R5 R0 0x0ABC
0x00000002 0x2c051234 addi R5 R0 0x1234
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x00000000 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

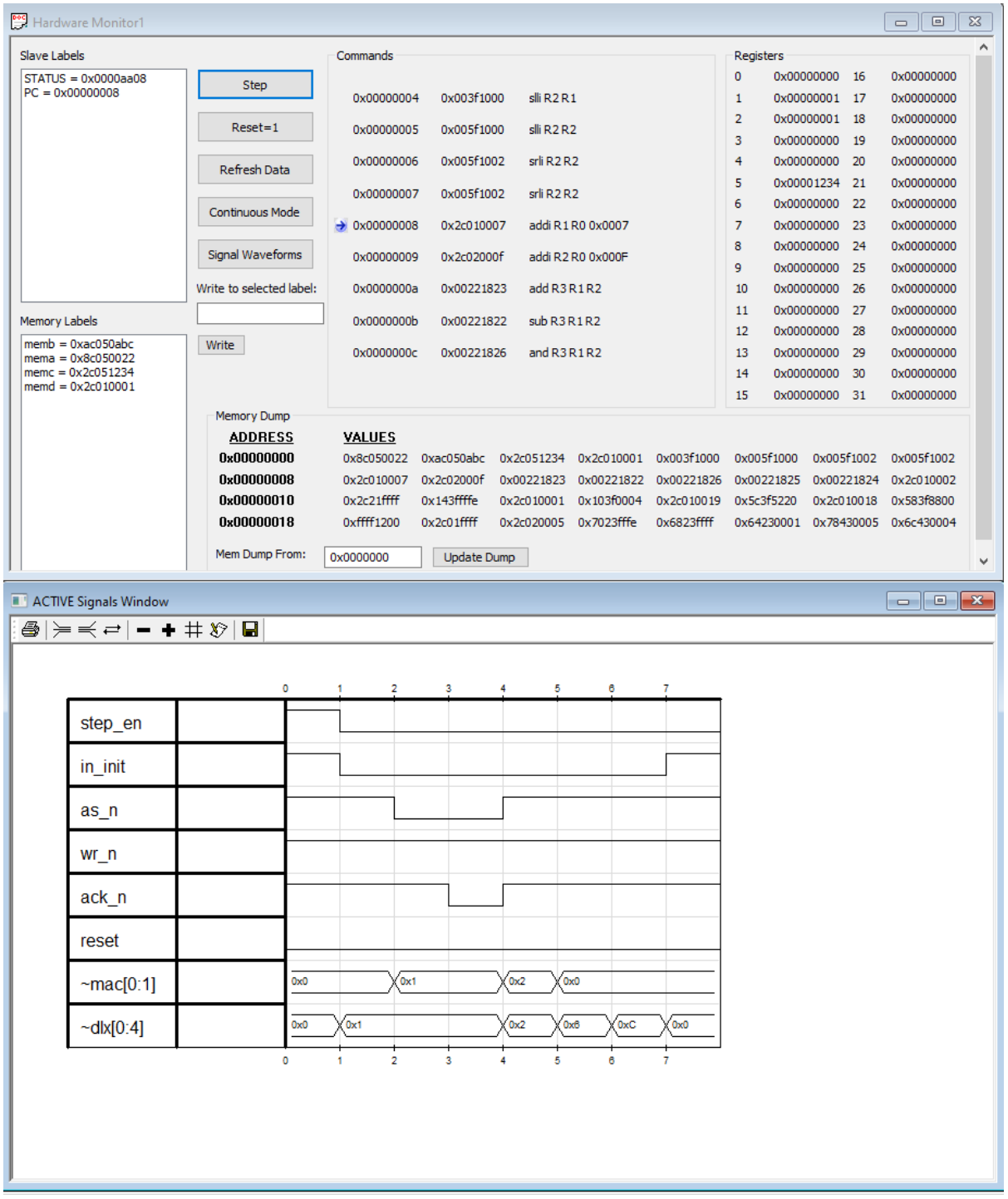

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000005
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000001 0xac050abc sw R5 R0 0x0ABC
0x00000002 0x2c051234 addi R5 R0 0x1234
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x00000002 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x6 0xC 0x0


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000006
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000002 0x2c051234 addi R5 R0 0x1234
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x00000004 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

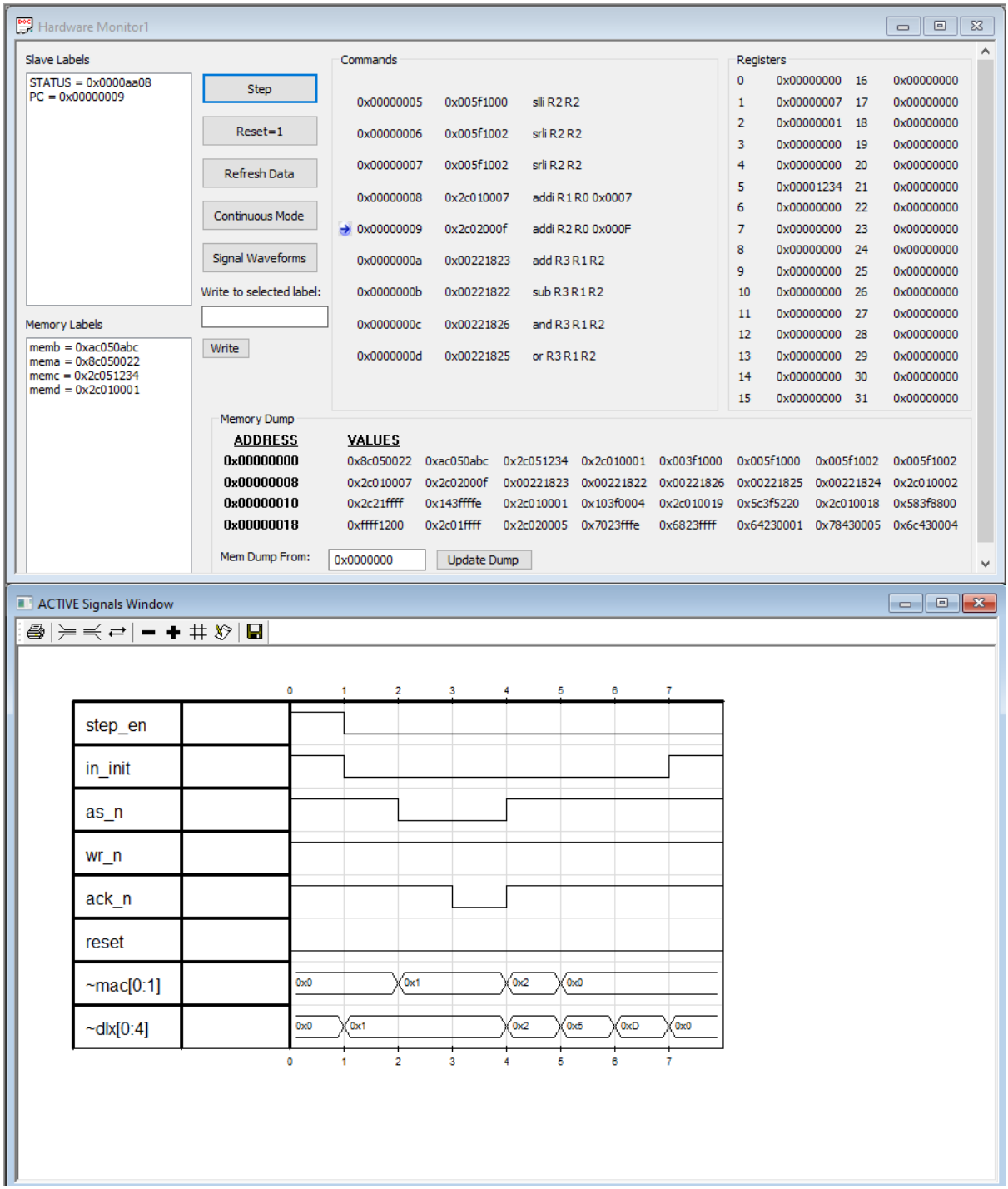

ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x8 0xC 0x0
0 1 2 3 4 5 6 7

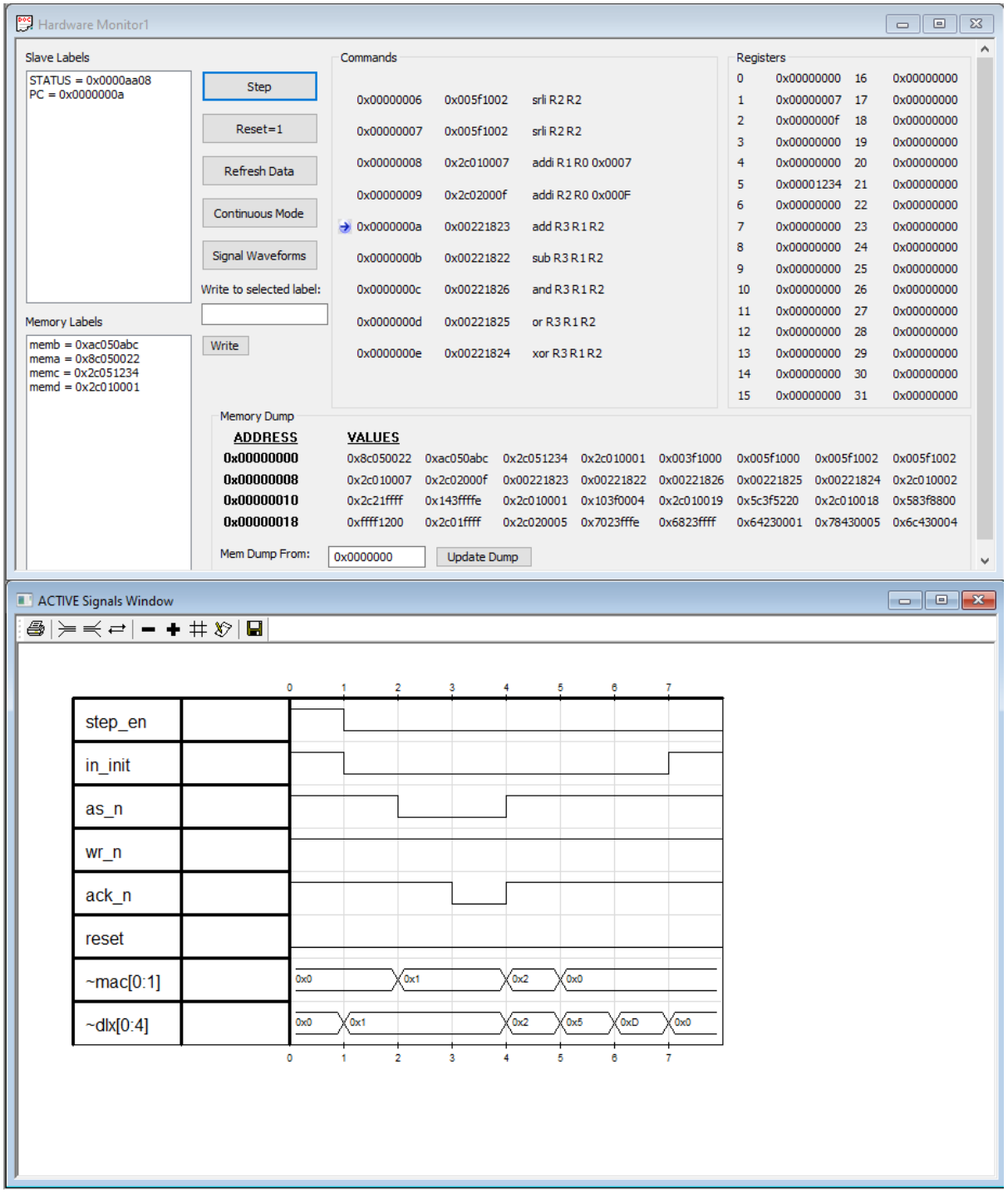

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000007
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000003 0x2c010001 addi R1 R0 0x0001
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x00000002 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x8 0xC 0x0

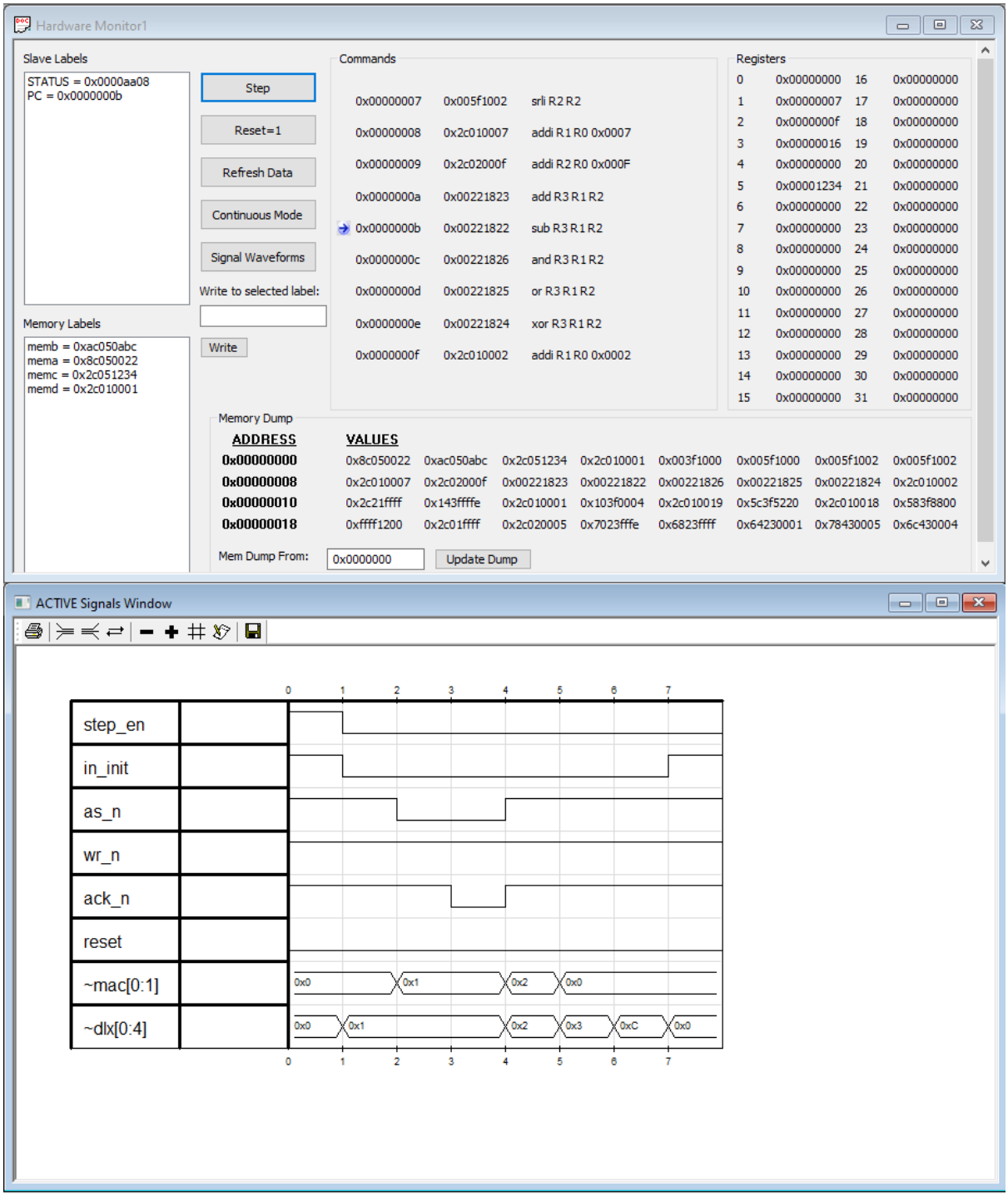

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000008
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000004 0x003f1000 slli R2 R1
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x00000001 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

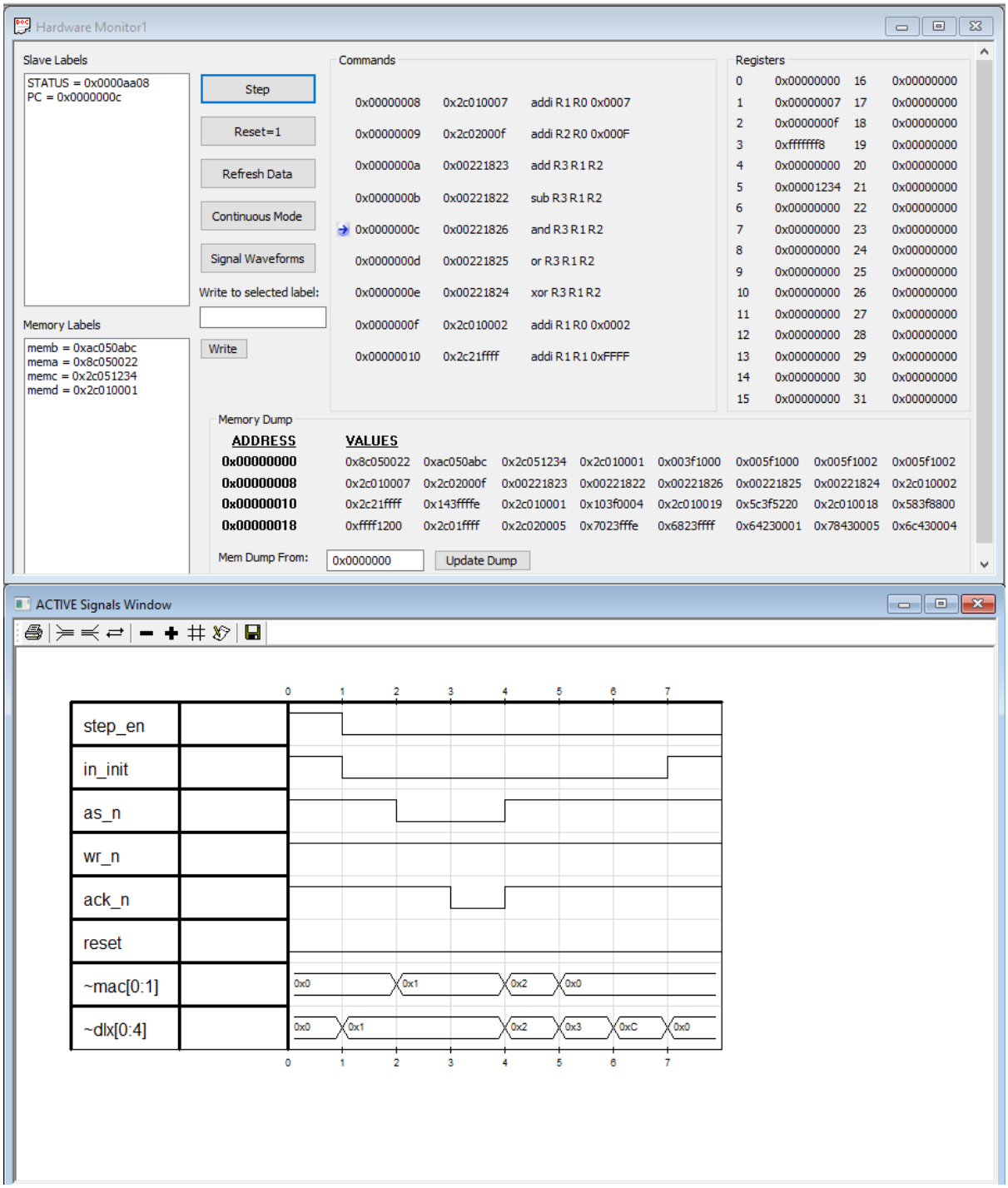
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x8 0xC 0x0

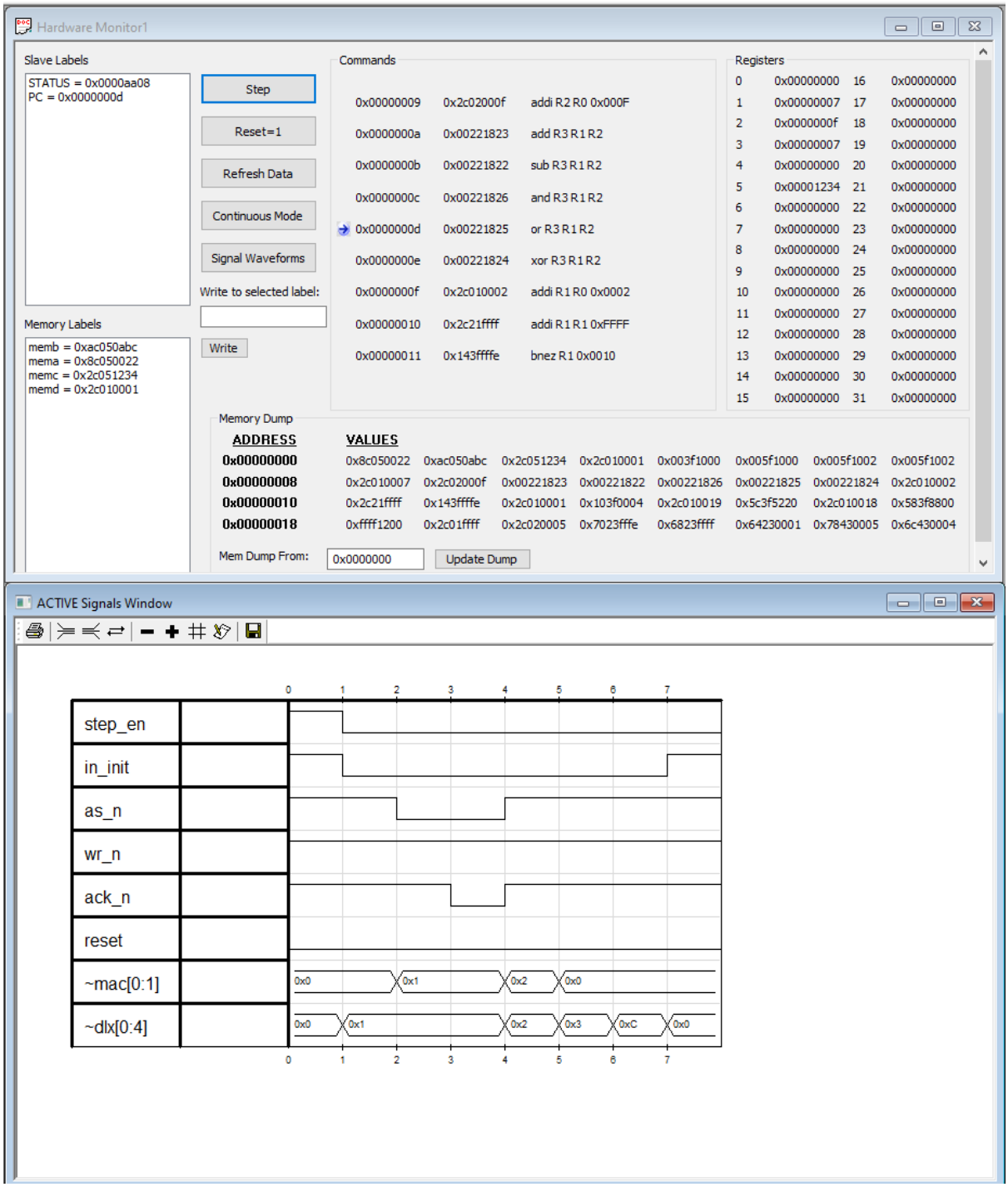

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000009
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000005 0x005f1000 slli R2 R2
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x00000001 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

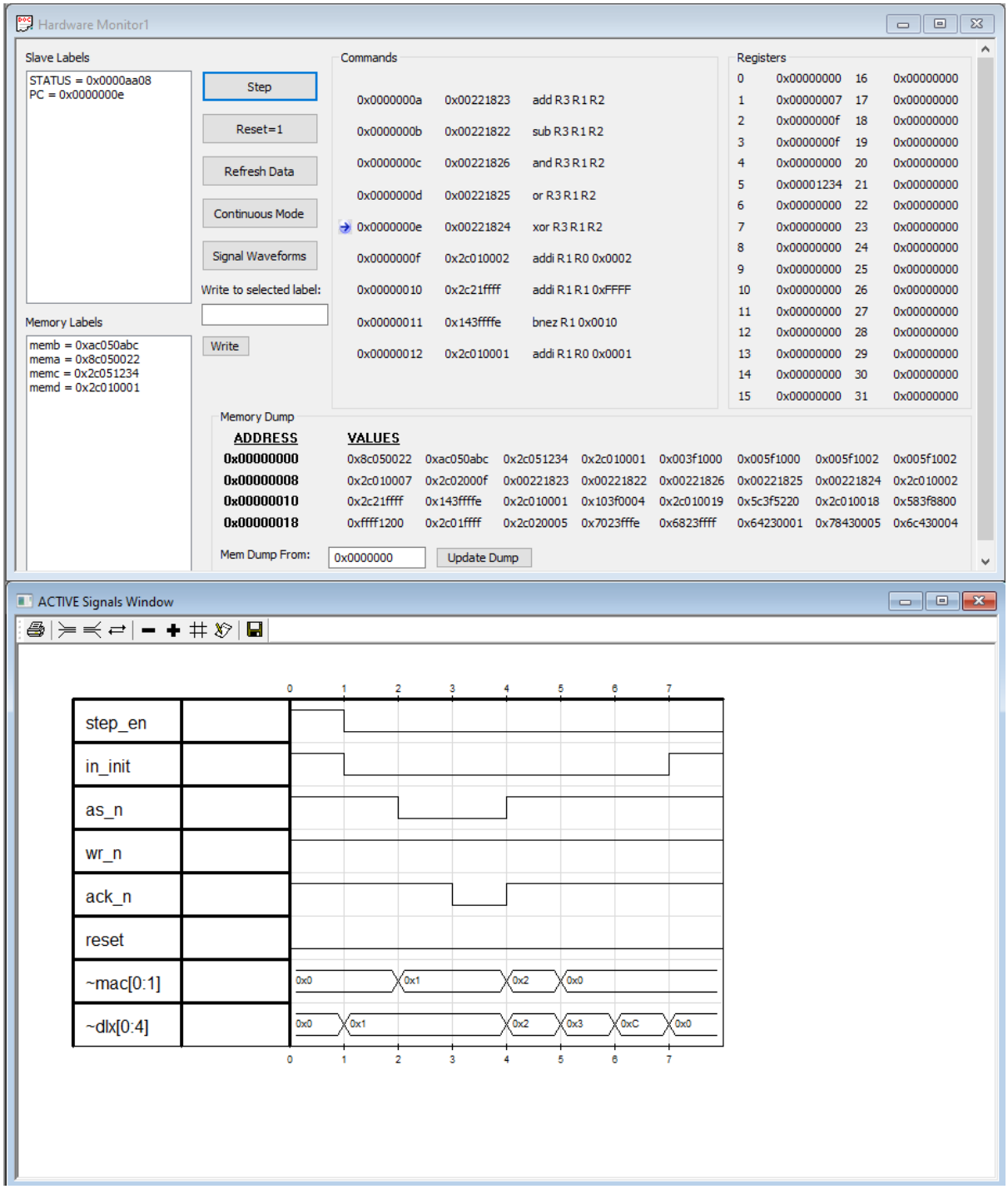

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000a
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000006 0x005f1002 srli R2 R2
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

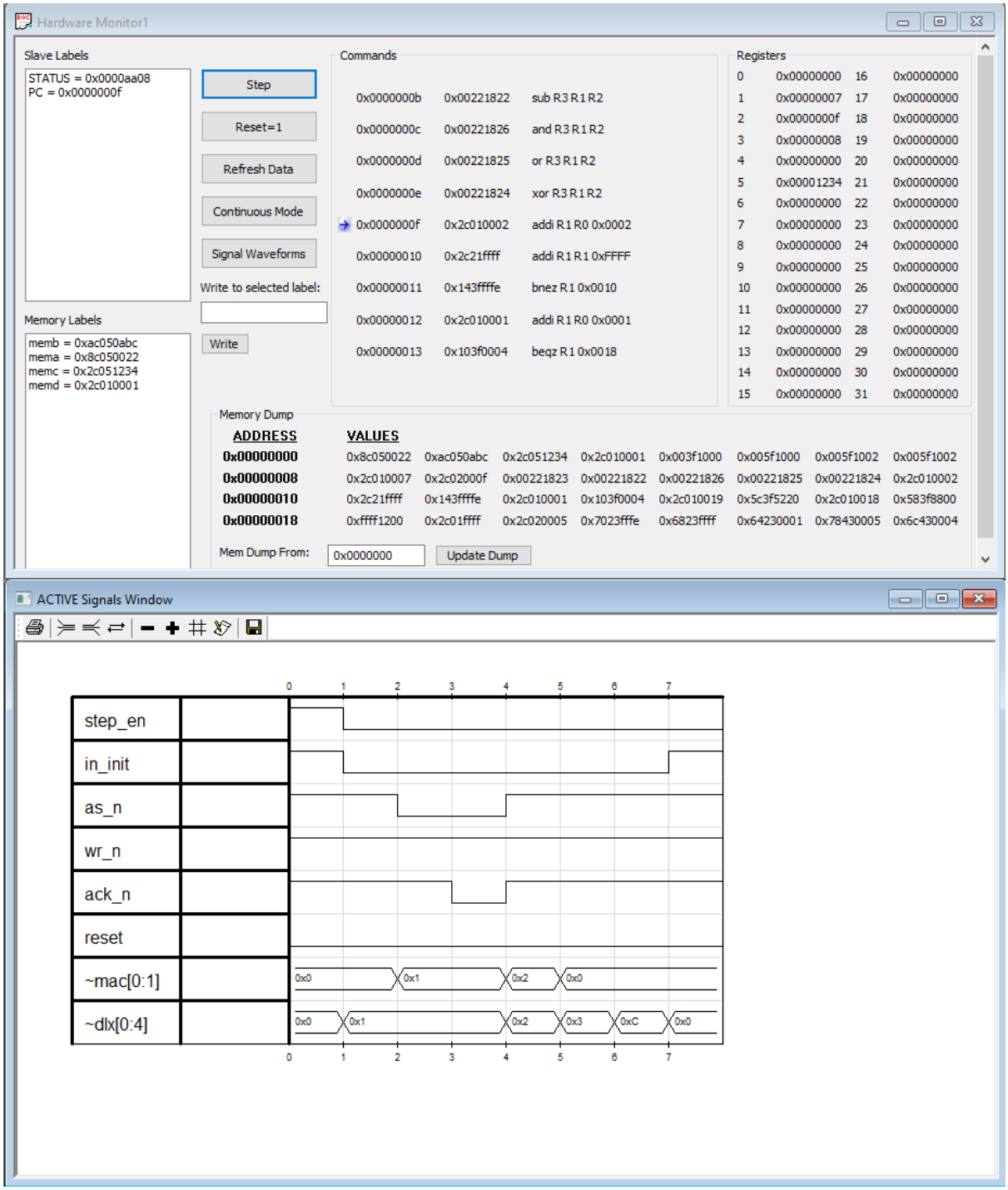
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0
0x1
0x2
0x0
0x0
0x1
0x2
0x5
0xD
0x0
0 1 2 3 4 5 6 7

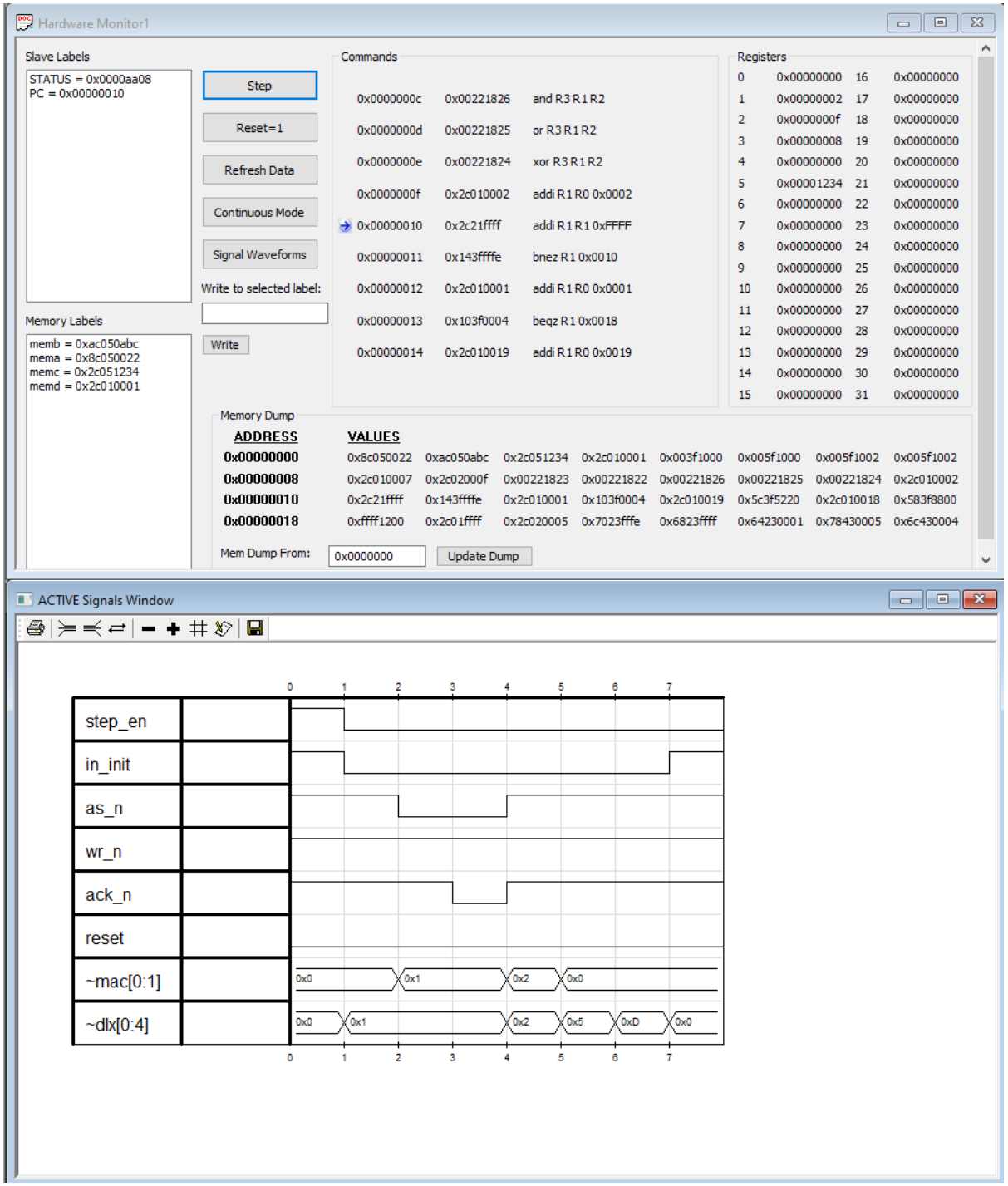

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000b
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000007 0x005f1002 srli R2 R2
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000016 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x3 0xC 0x0


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000c
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000008 0x2c010007 addi R1 R0 0x0007
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0xfffffff8 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x3 0xC 0x0
0 1 2 3 4 5 6 7

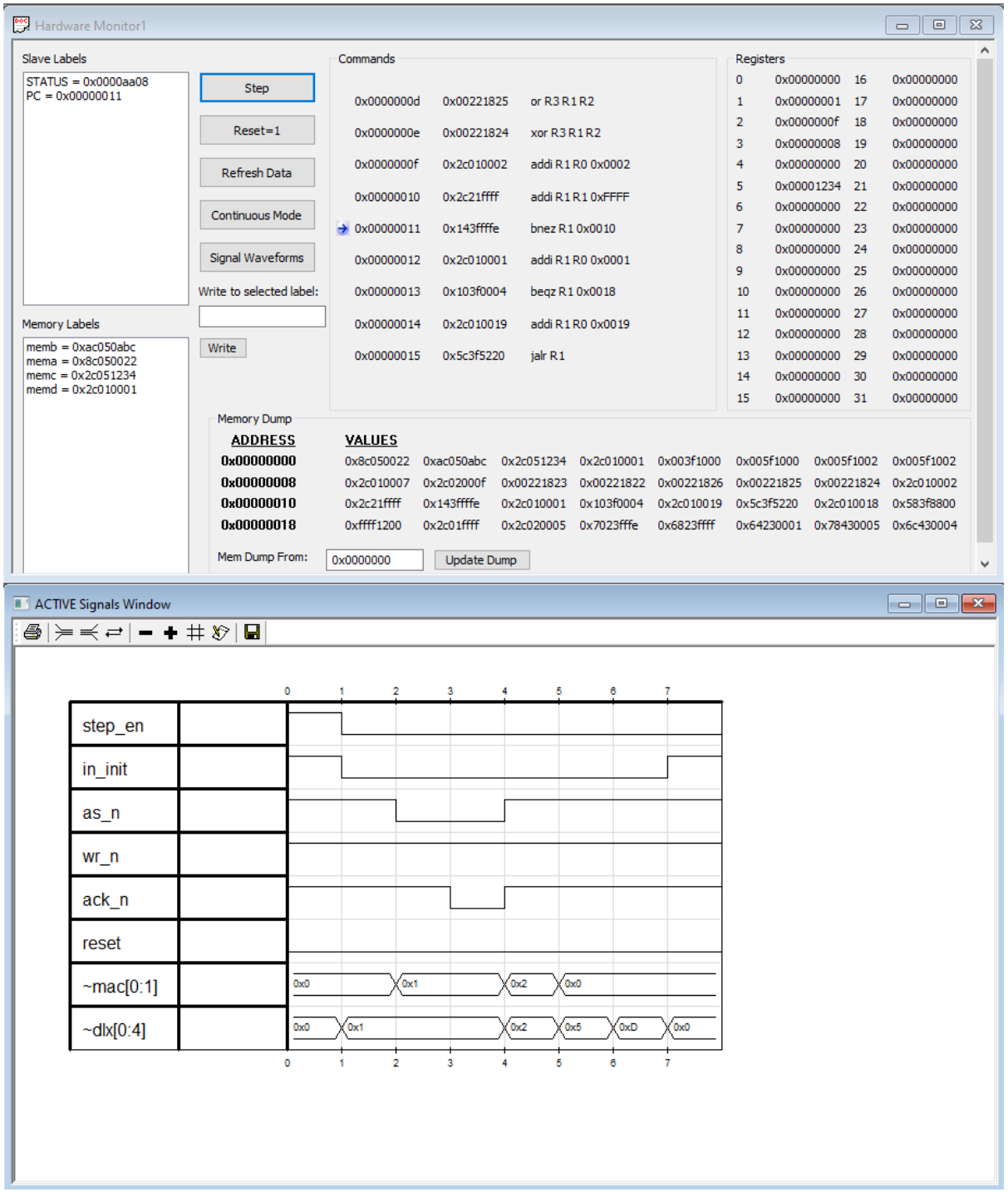

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000d
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000009 0x2c02000f addi R2 R0 0x000F
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000007 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x3 0xC 0x0


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000e
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000a 0x00221823 add R3 R1 R2
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x0000000f 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0
0x1
0x2
0x0
0x0
0x1
0x2
0x3
0xC
0x0
0 1 2 3 4 5 6 7

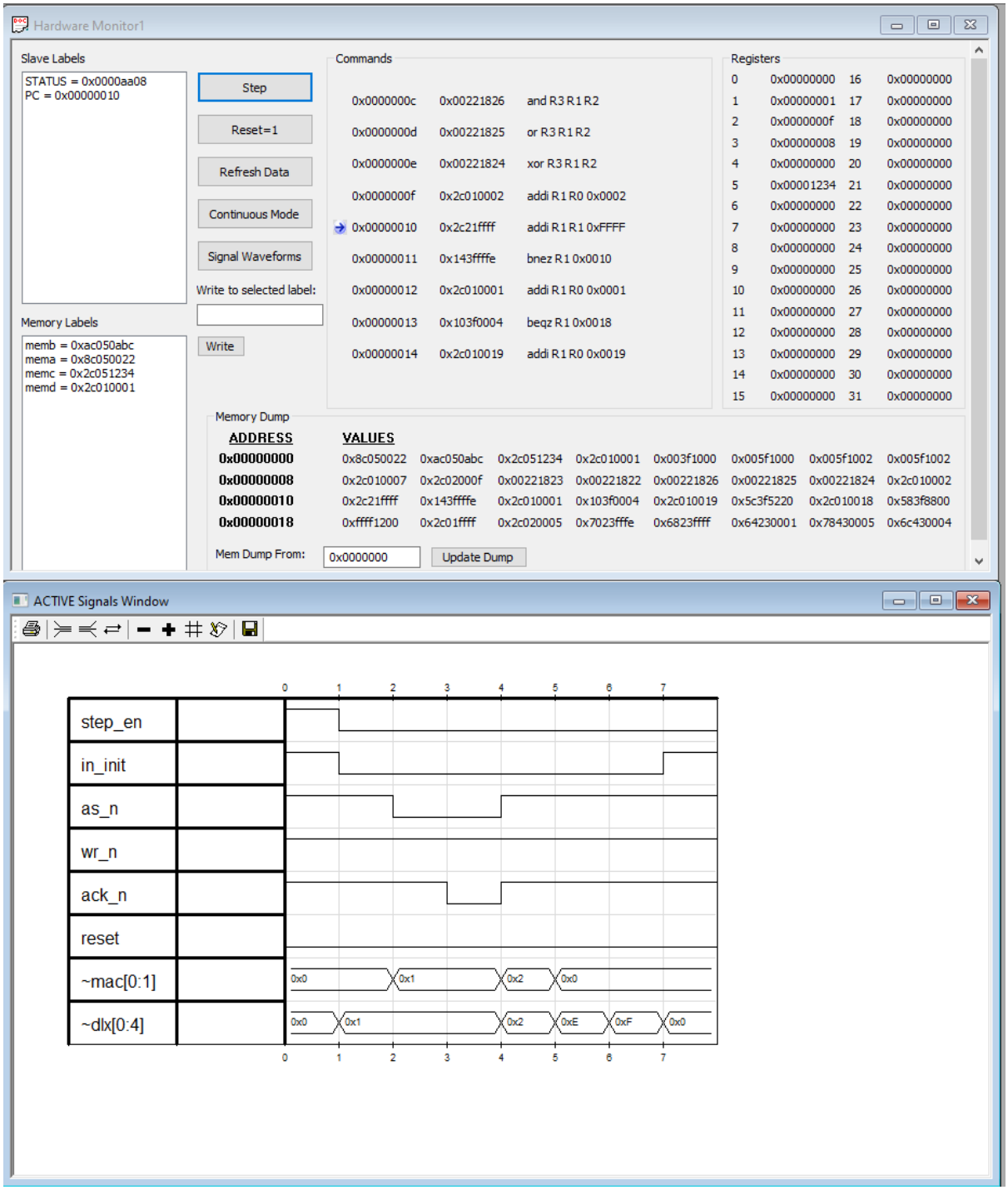

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000000f
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000b 0x00221822 sub R3 R1 R2
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143fffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
Registers
0 0x00000000 16 0x00000000
1 0x00000007 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x3 0xC 0x0

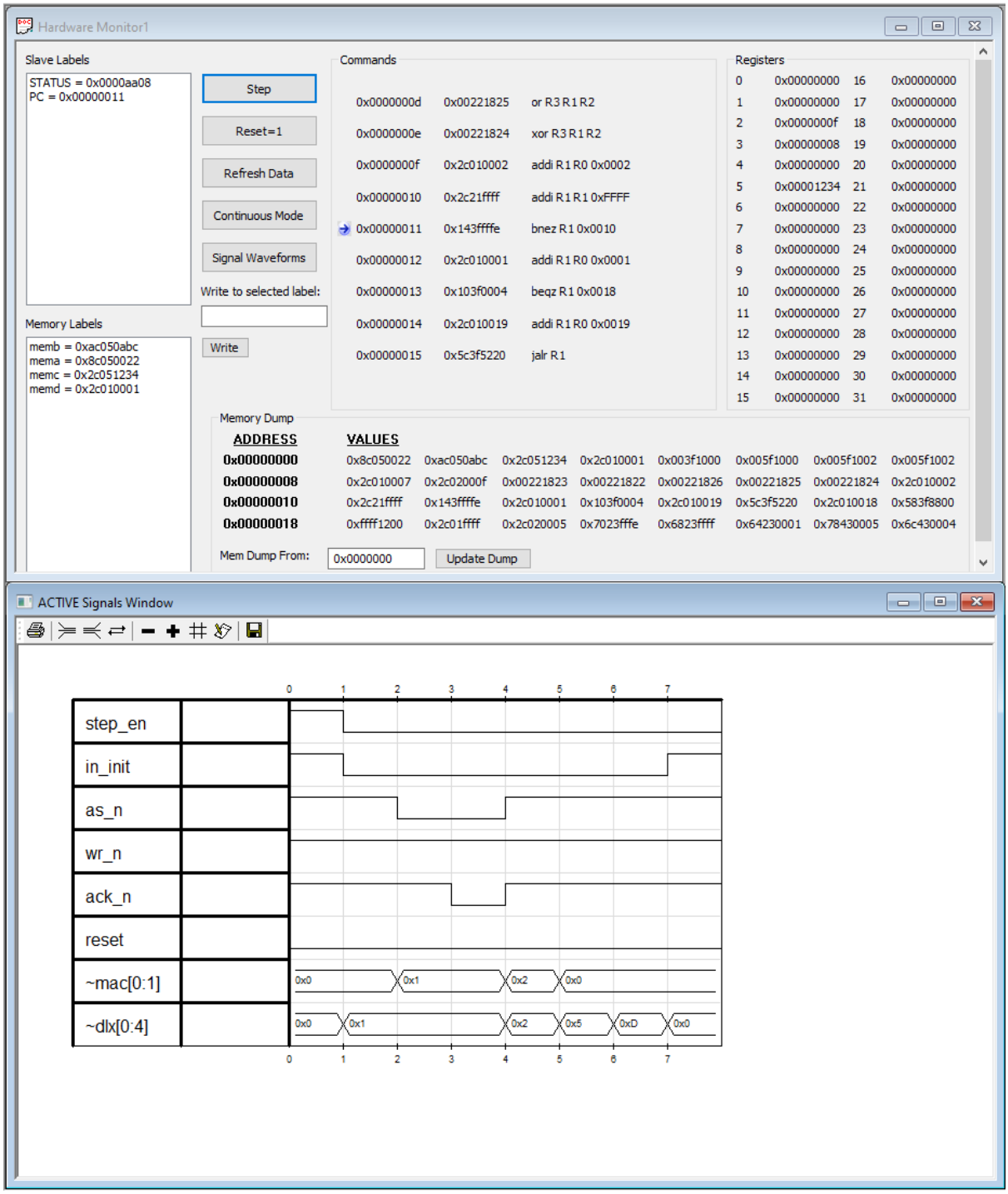

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000010
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
Registers
0 0x00000000 16 0x00000000
1 0x00000002 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

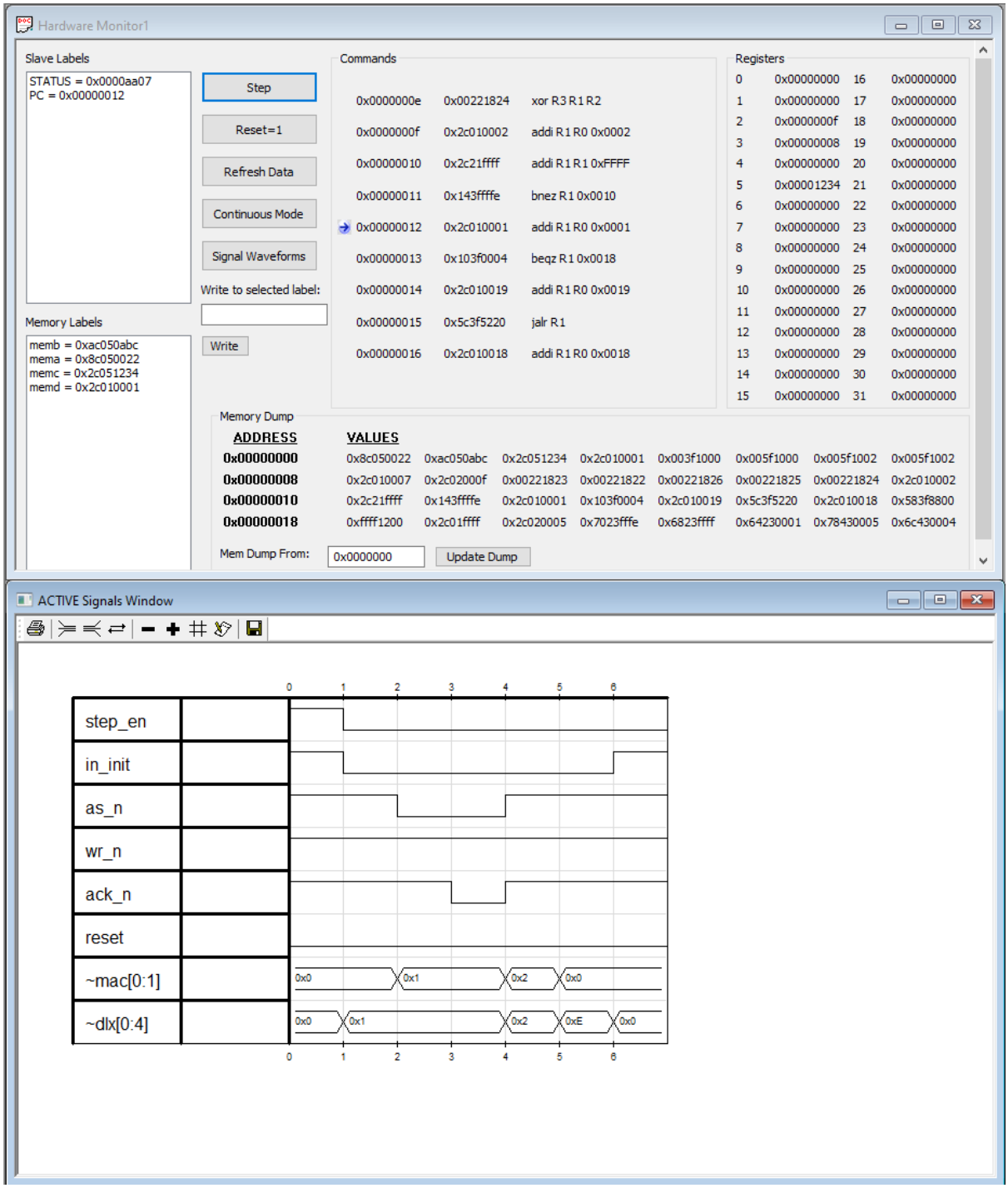
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

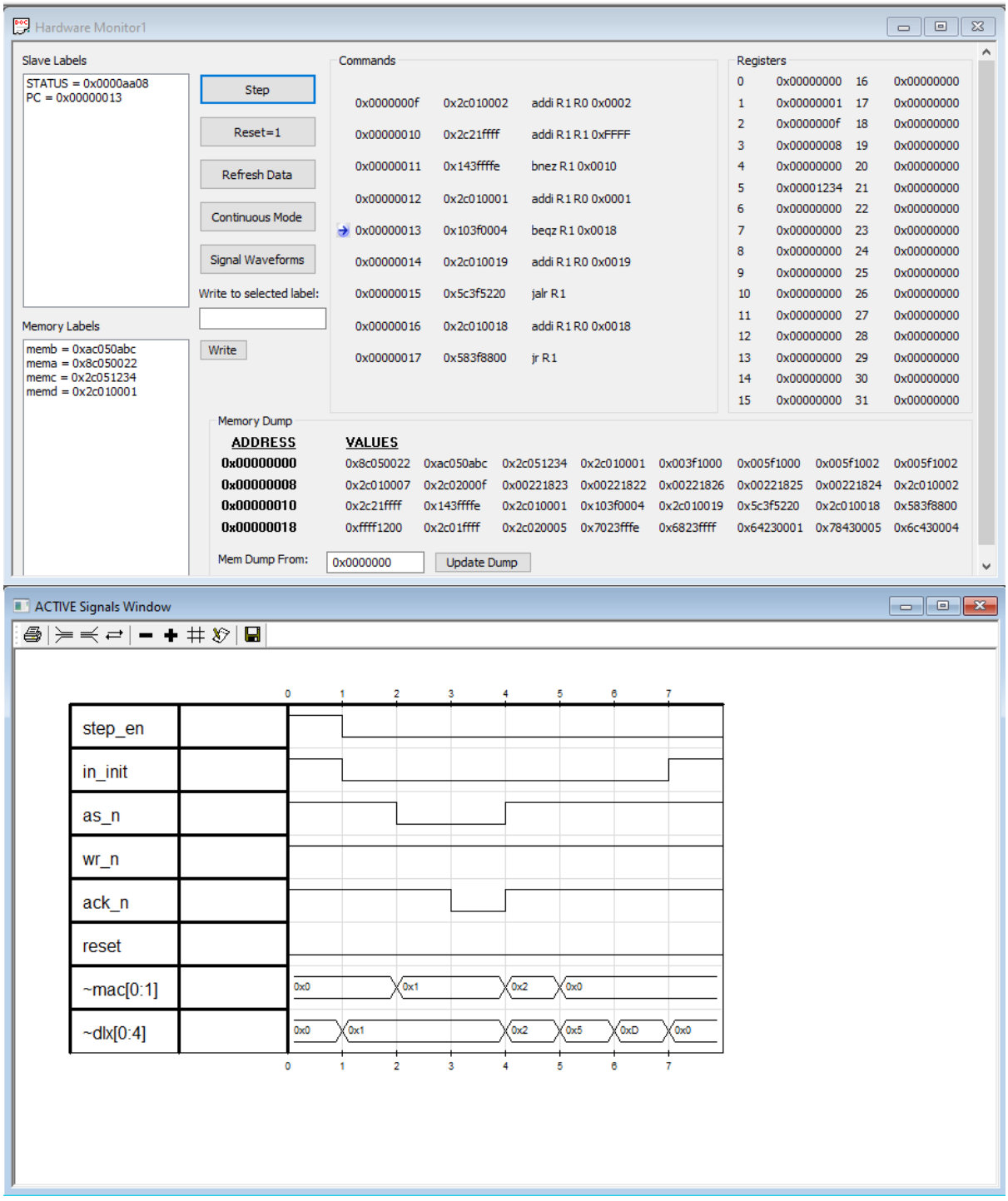

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000011
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

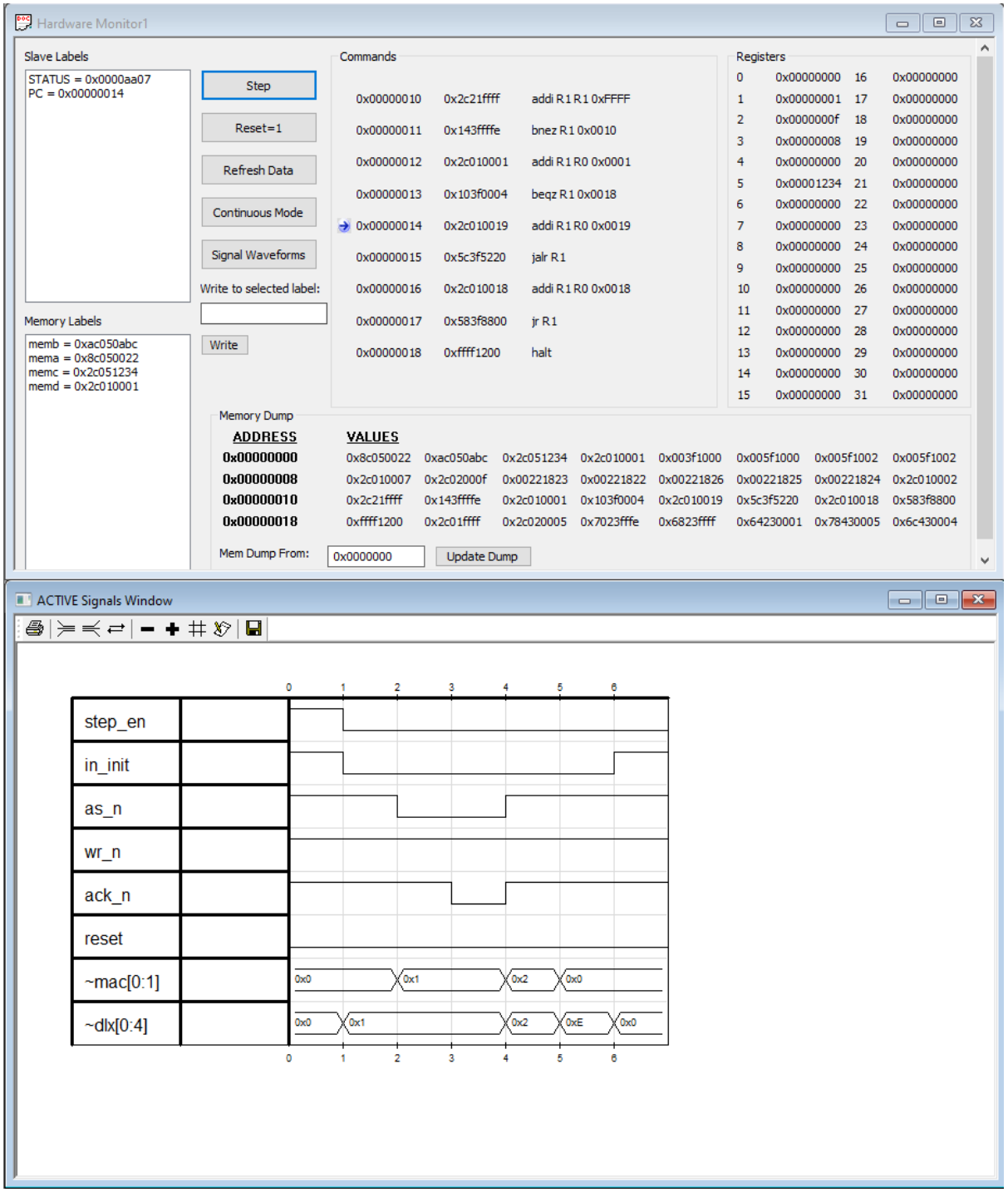
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x5 0xD 0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000010
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000c 0x00221826 and R3 R1 R2
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143fffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0xE 0xF 0x0

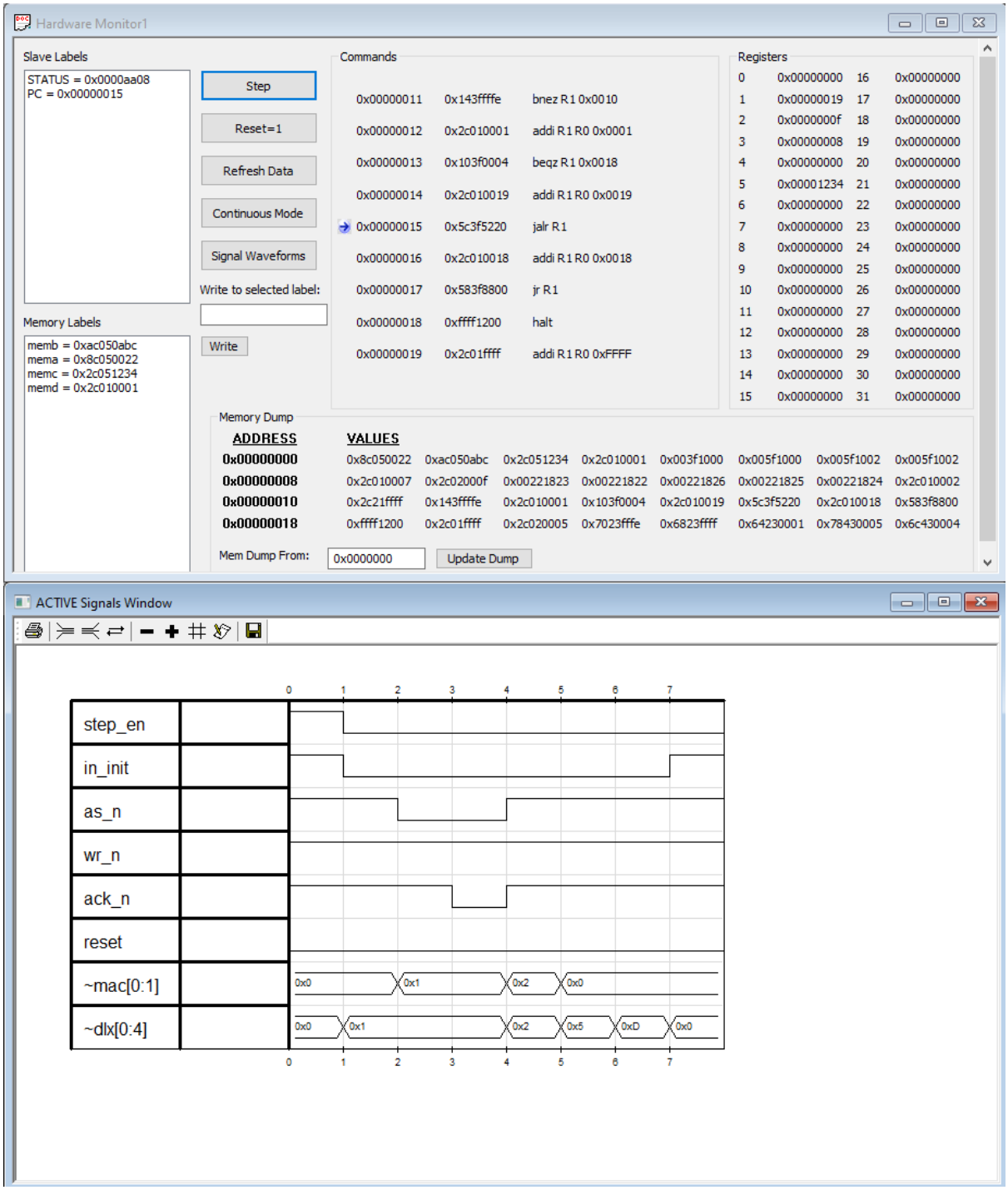

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000011
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000d 0x00221825 or R3 R1 R2
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
Registers
0 0x00000000 16 0x00000000
1 0x00000000 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

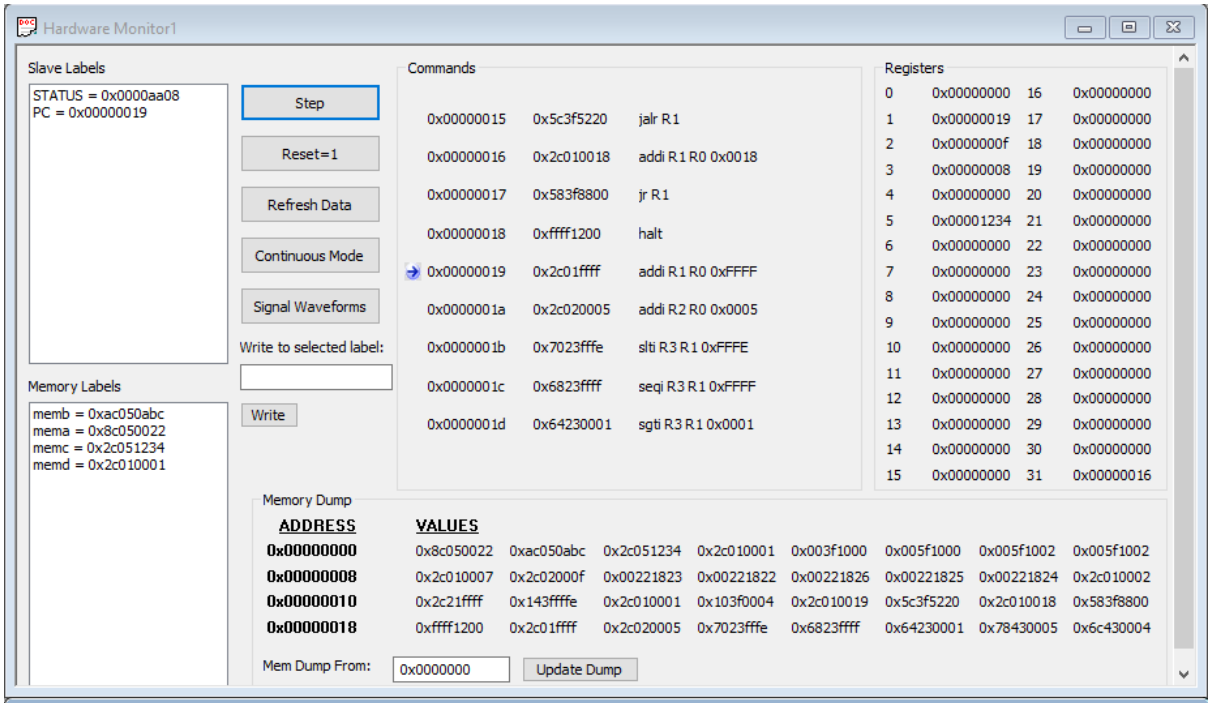

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa07
PC = 0x00000012
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000e 0x00221824 xor R3 R1 R2
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
Registers
0 0x00000000 16 0x00000000
1 0x00000000 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0
0x1
0x2
0x0
0x0
0x1
0x2
0xE
0x0
0 1 2 3 4 5 6

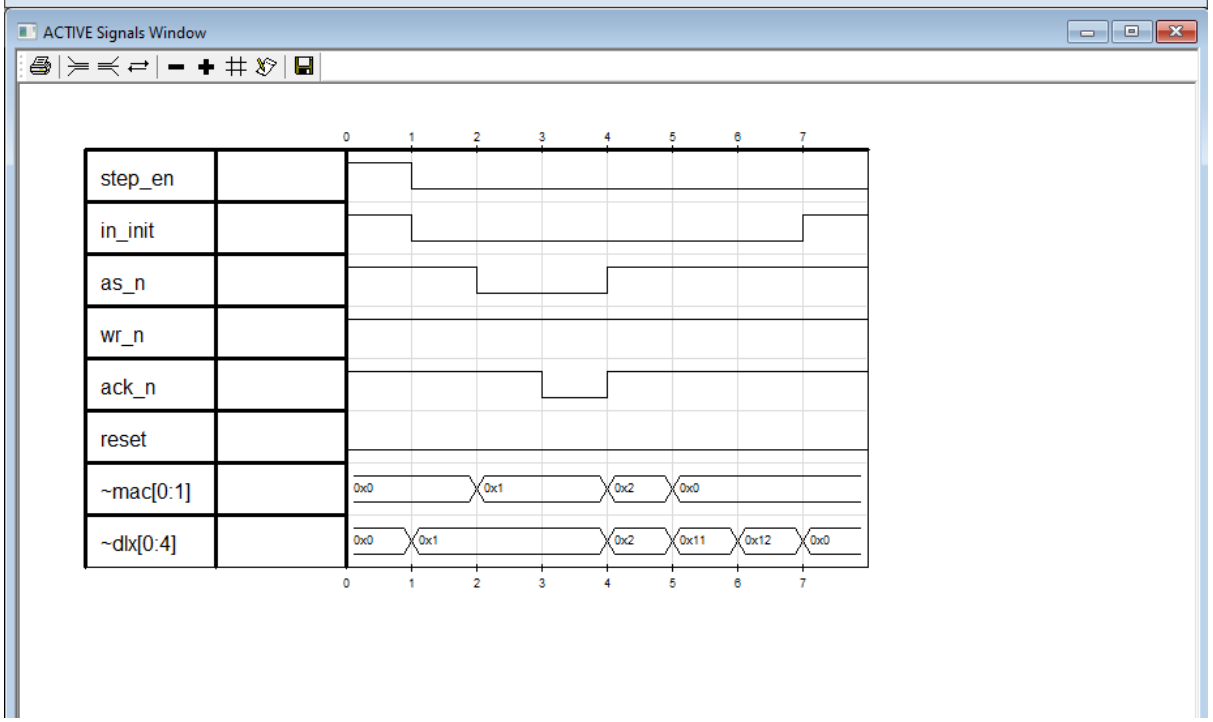

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000013
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000000f 0x2c010002 addi R1 R0 0x0002
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x5 0xD 0x0
0 1 2 3 4 5 6 7

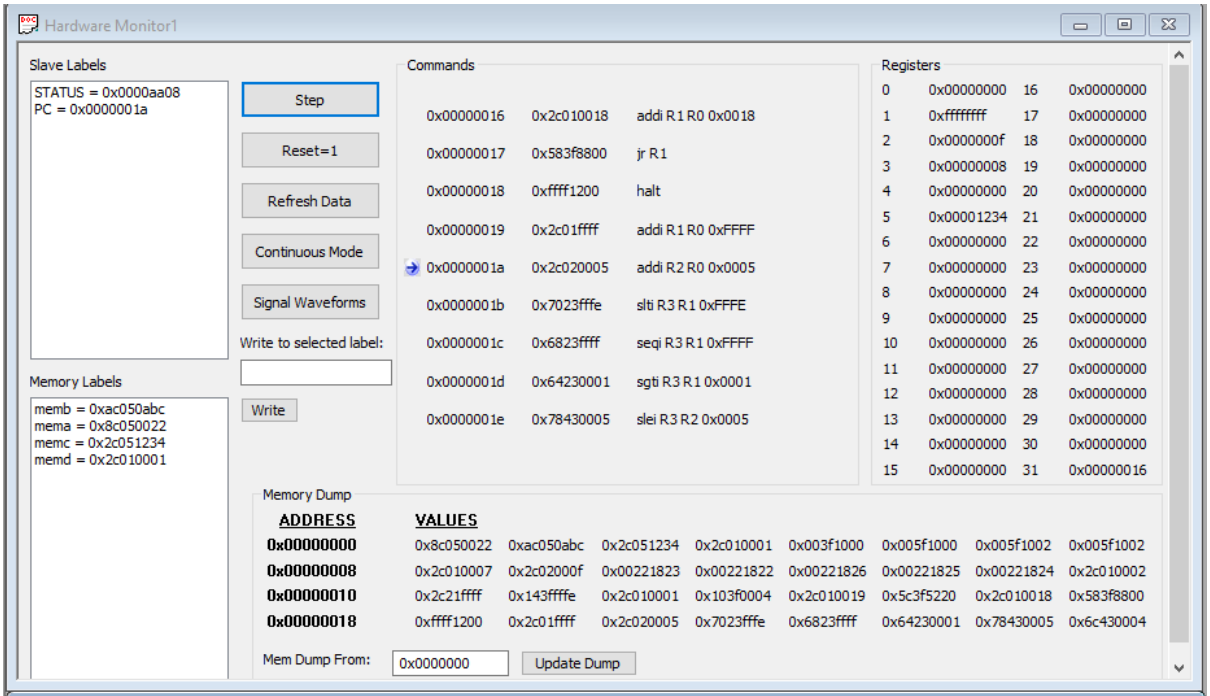

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa07
PC = 0x00000014
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000010 0x2c21ffff addi R1 R1 0xFFFF
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
Registers
0 0x00000000 16 0x00000000
1 0x00000001 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

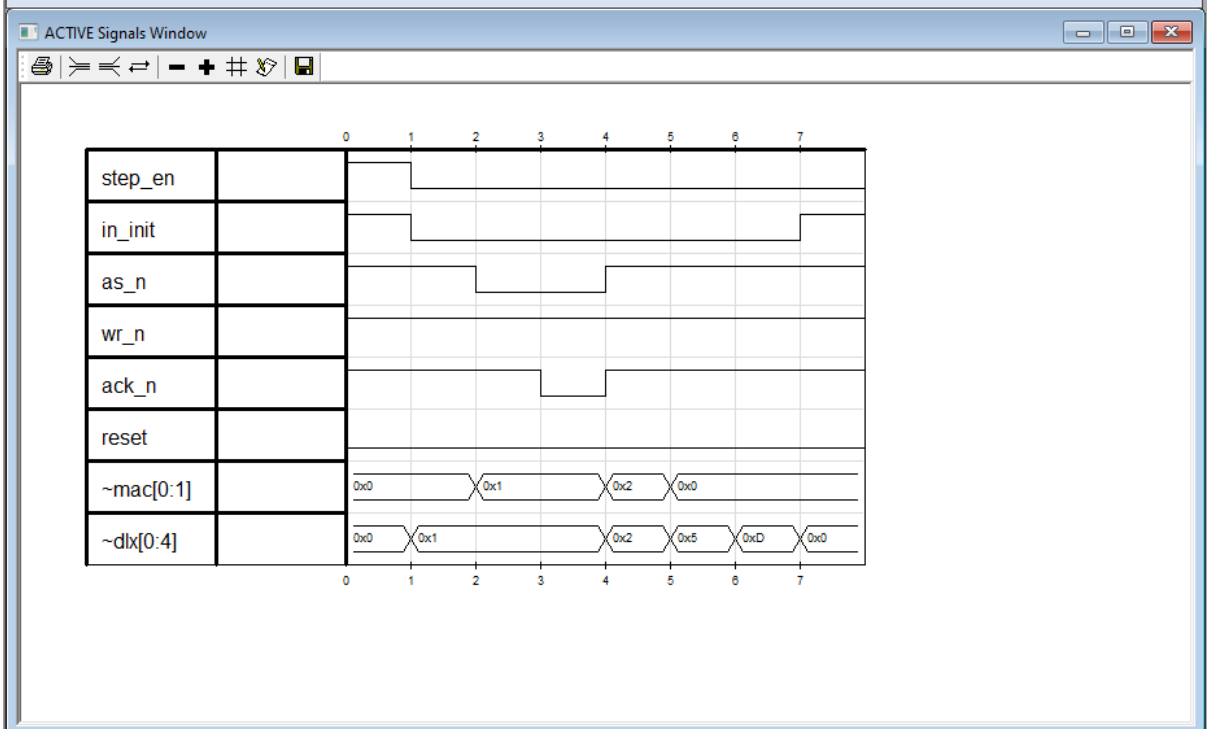
ACTIVE Signals Window
0 1 2 3 4 5 6
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0xE 0x0
0 1 2 3 4 5 6

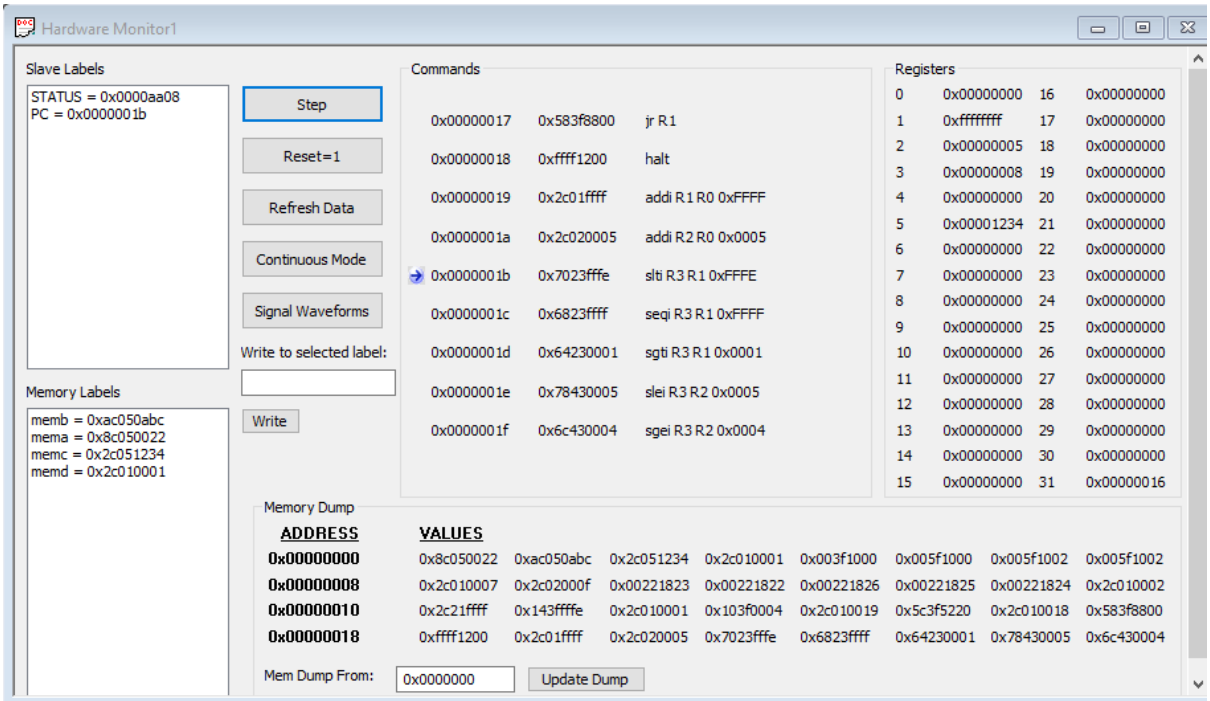

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000015
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000011 0x143ffffe bnez R1 0x0010
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
Registers
0 0x00000000 16 0x00000000
1 0x00000019 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000000
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

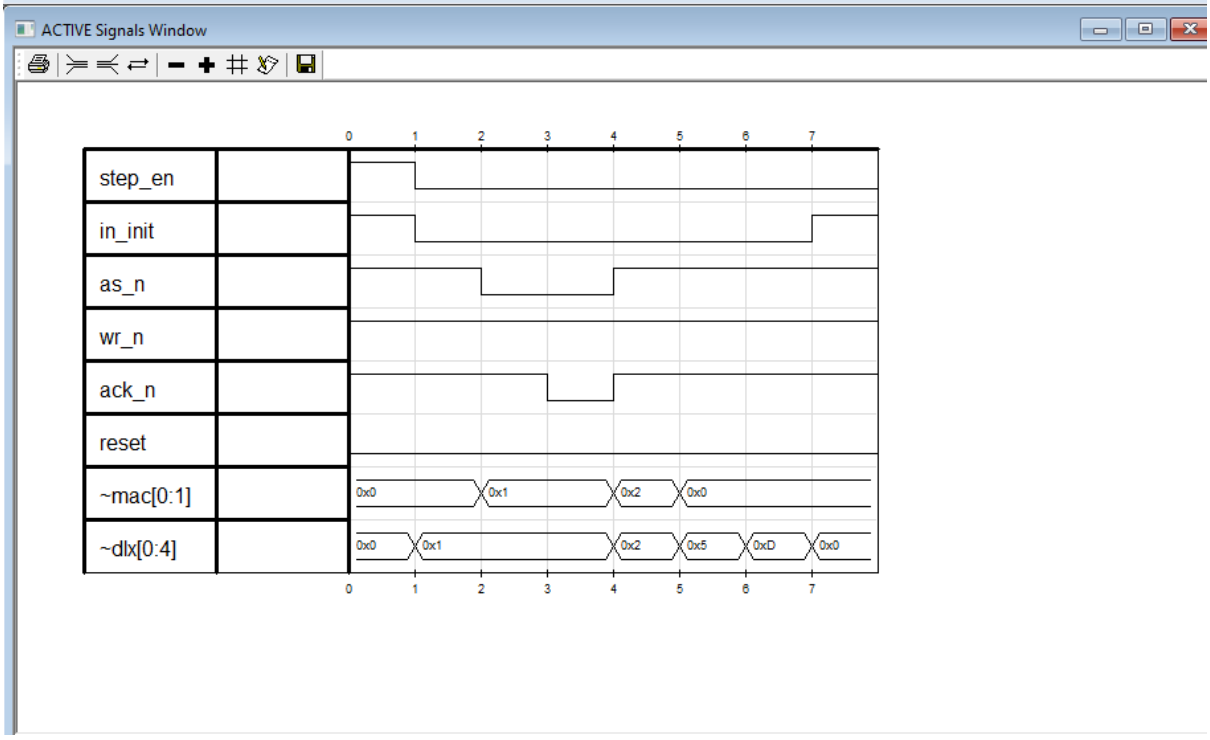

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000019
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
Registers
0 0x00000000 16 0x00000000
1 0x00000019 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From:
0x0000000
Update Dump

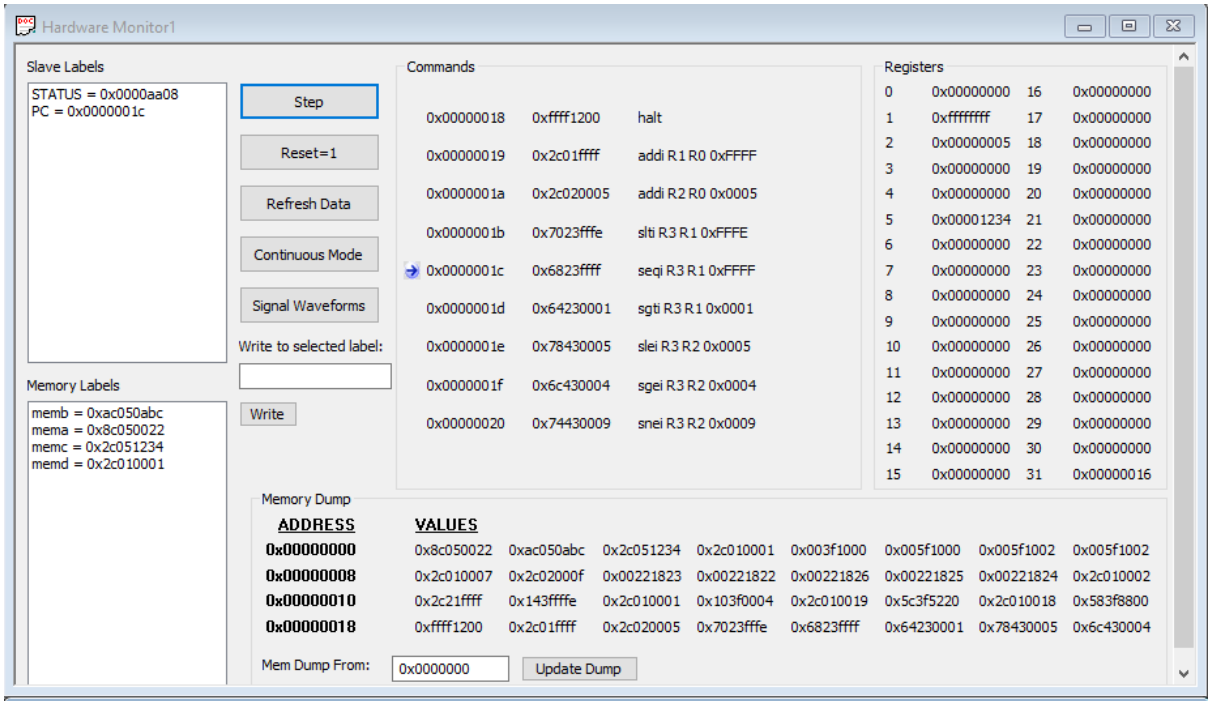

ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x11 0x12 0x0
0 1 2 3 4 5 6 7

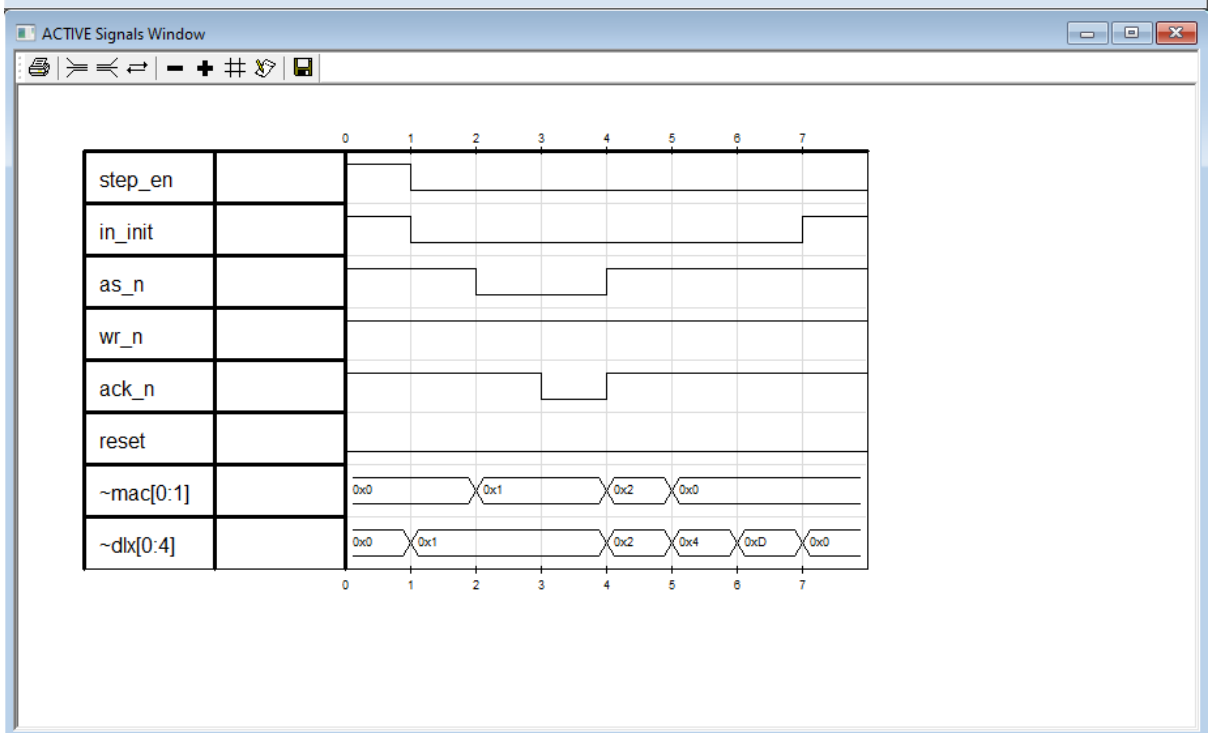

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001a
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x0000000f 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

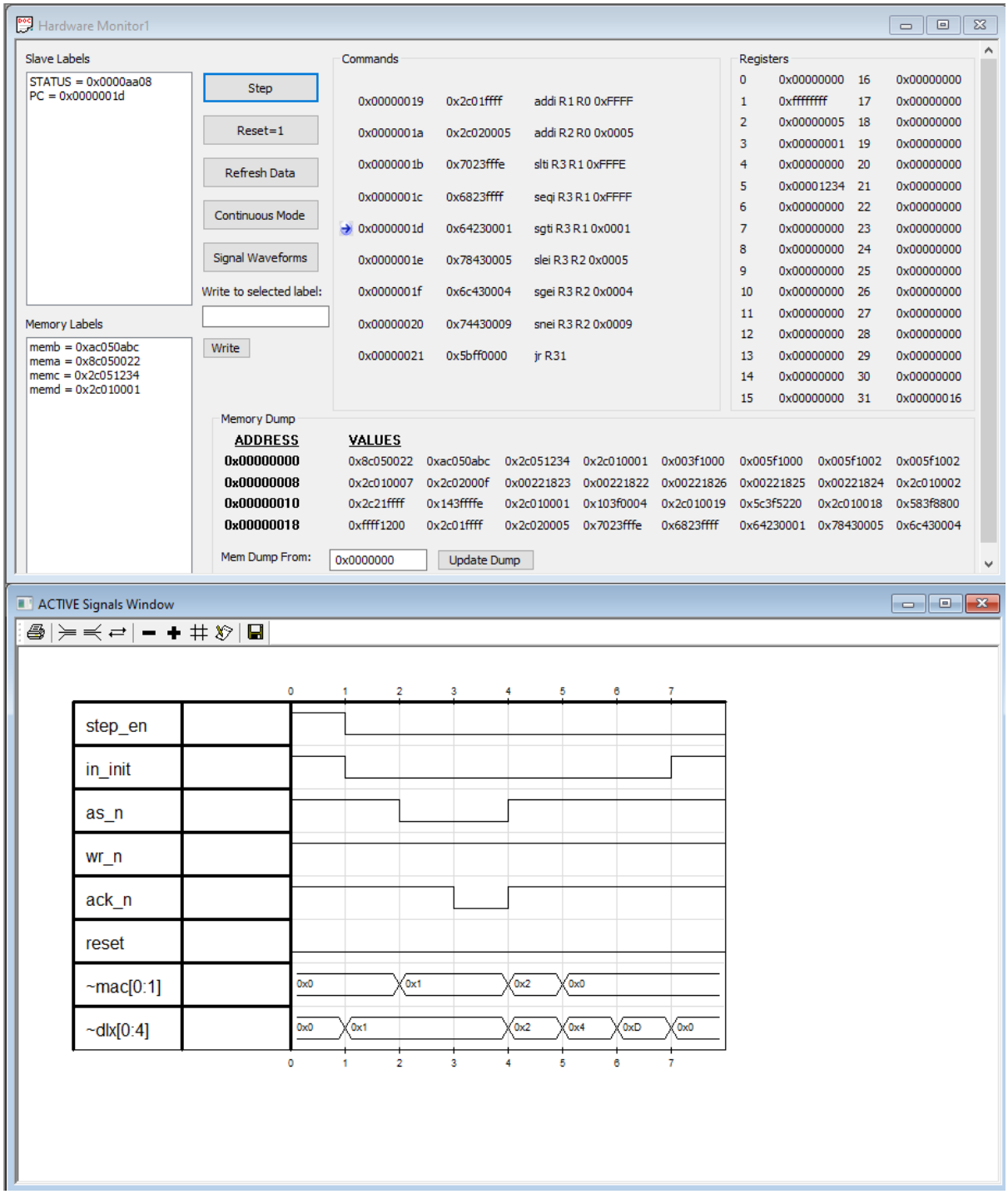

ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0
0x1
0x2
0x0
0x0
0x1
0x2
0x5
0xD
0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001b
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000008 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

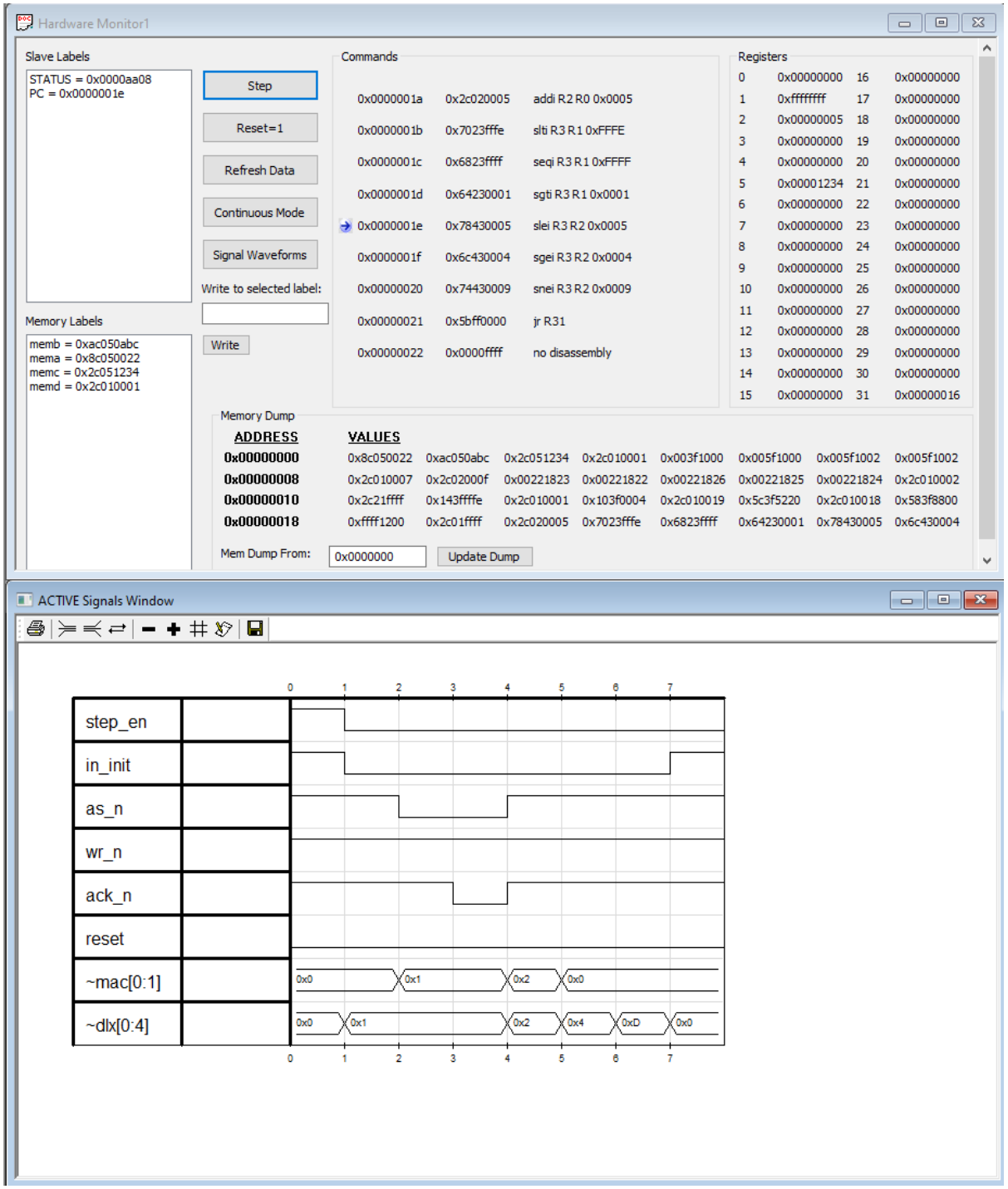

ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x5 0xD 0x0

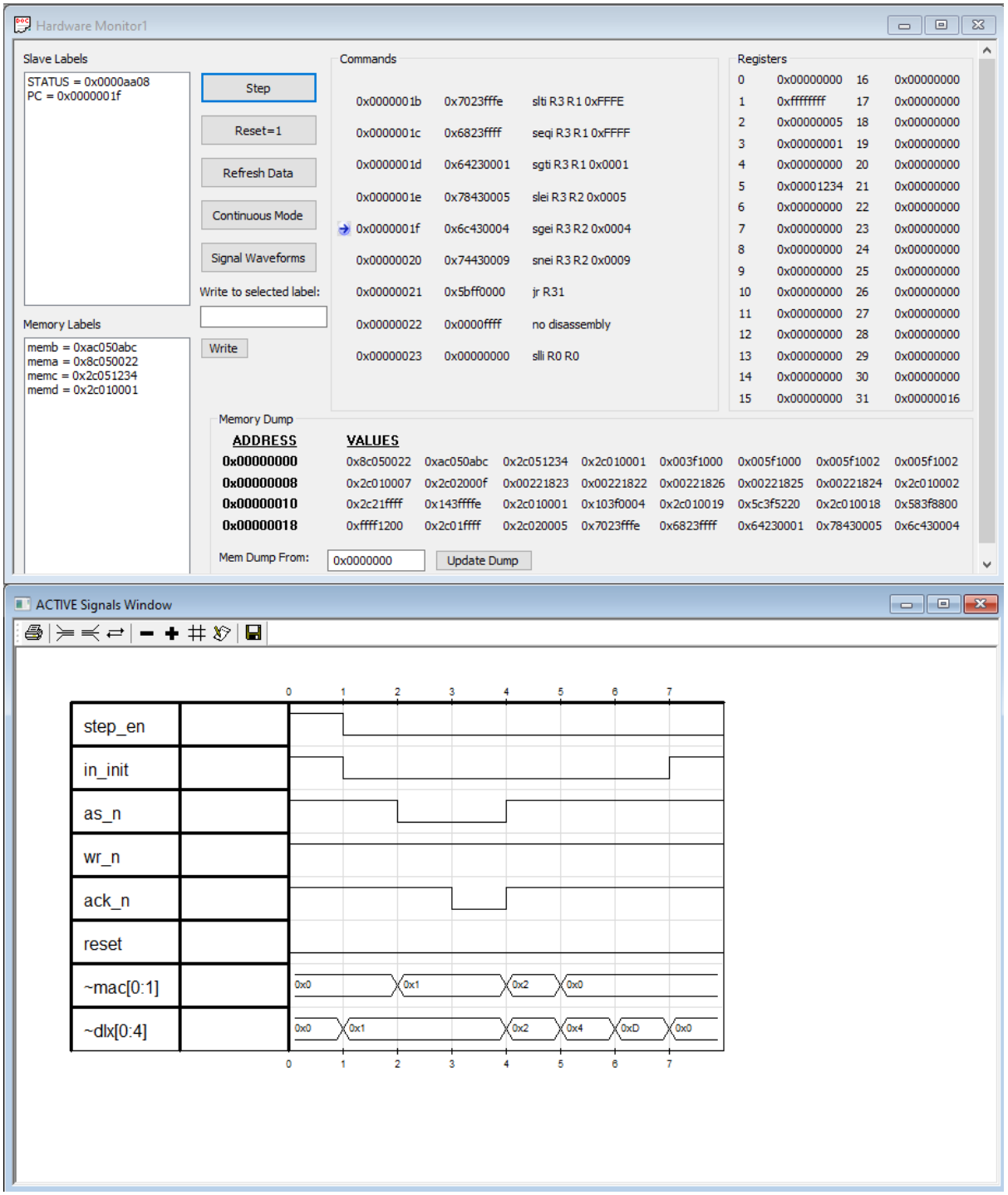

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001c
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

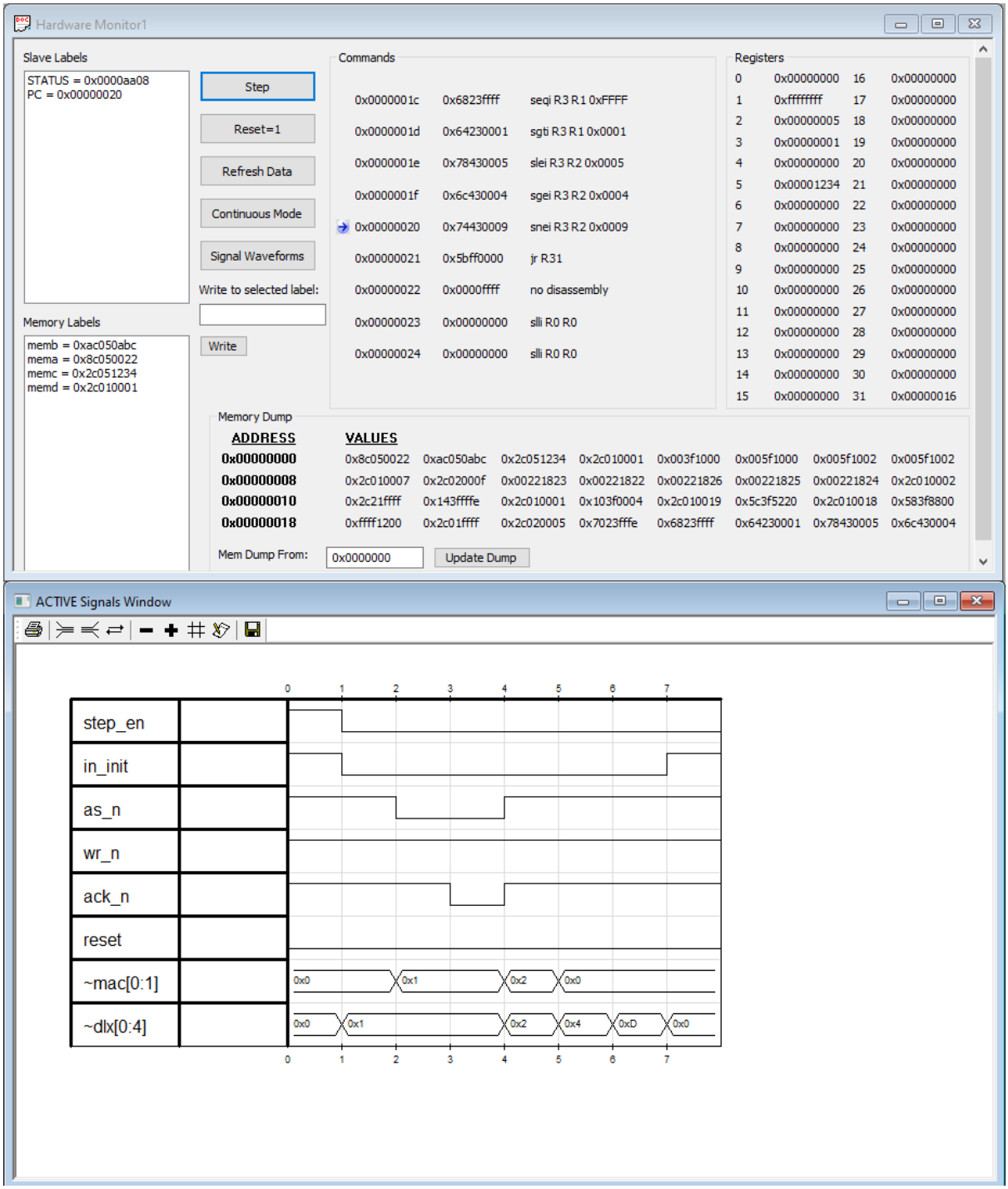

ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x4 0xD 0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001d
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
0x00000021 0x5bff0000 jr R31
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

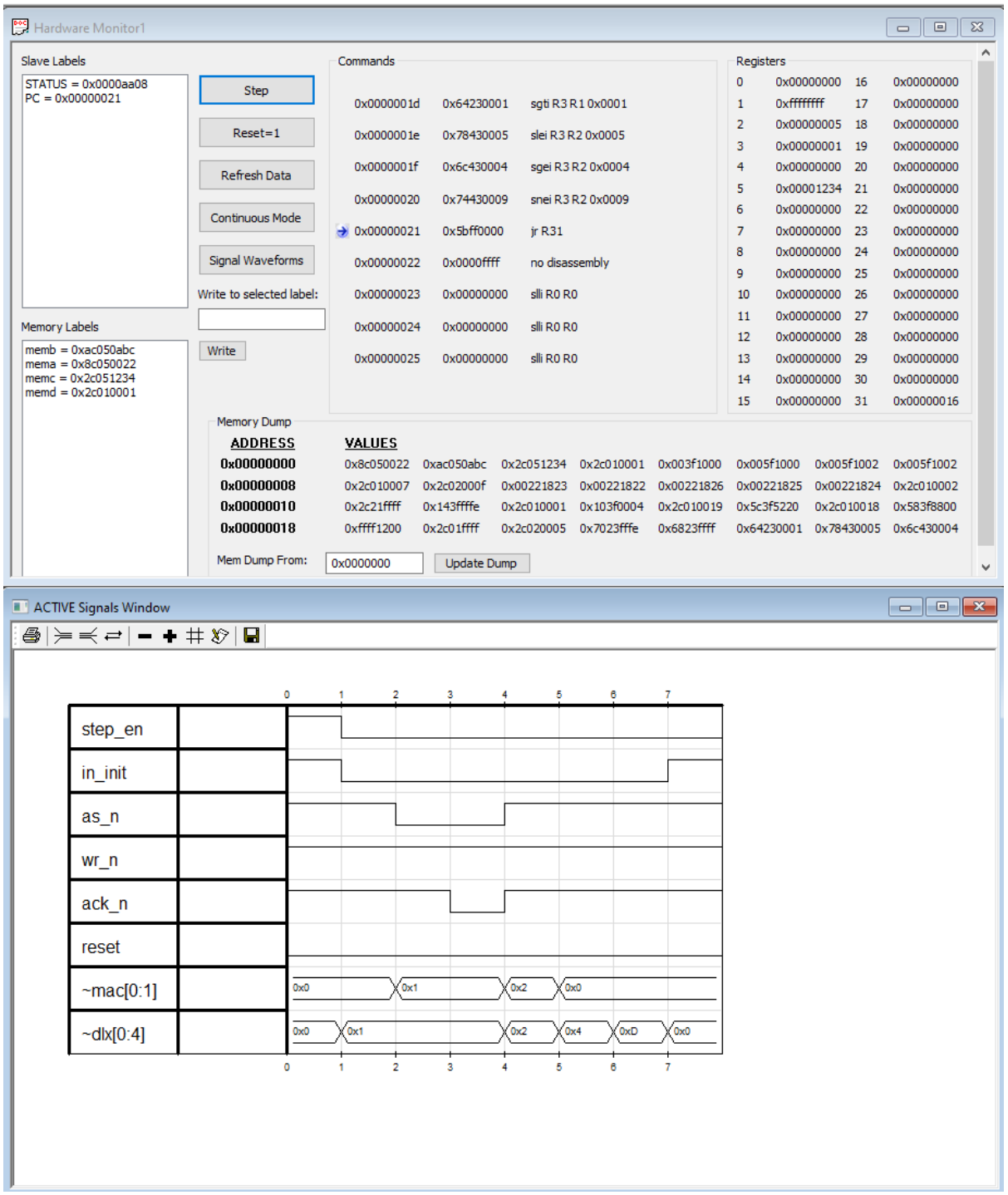
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x4 0xD 0x0
0 1 2 3 4 5 6 7

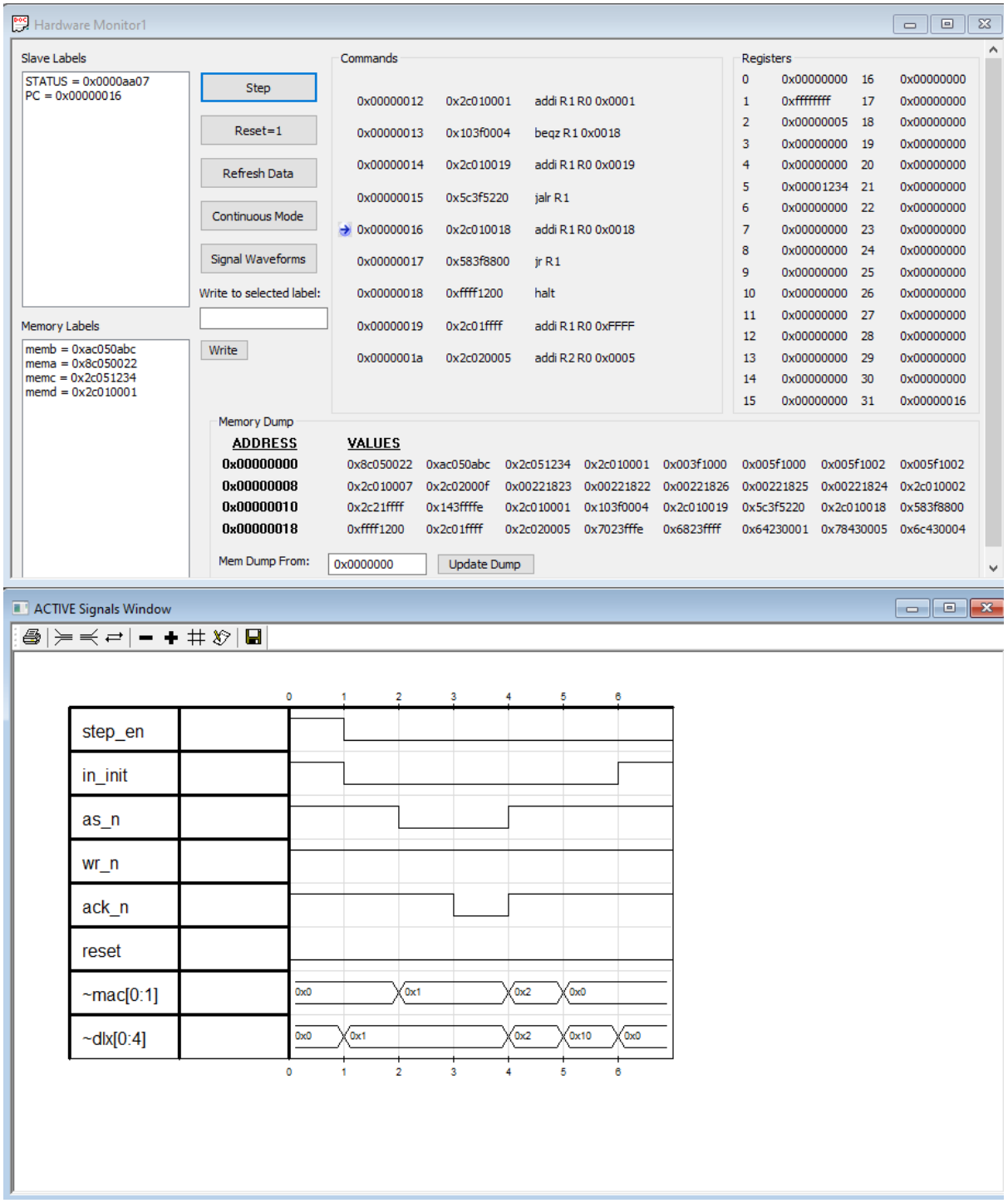

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001e
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
0x00000021 0x5bff0000 jr R31
0x00000022 0x0000ffff no disassembly
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x4 0xD 0x0

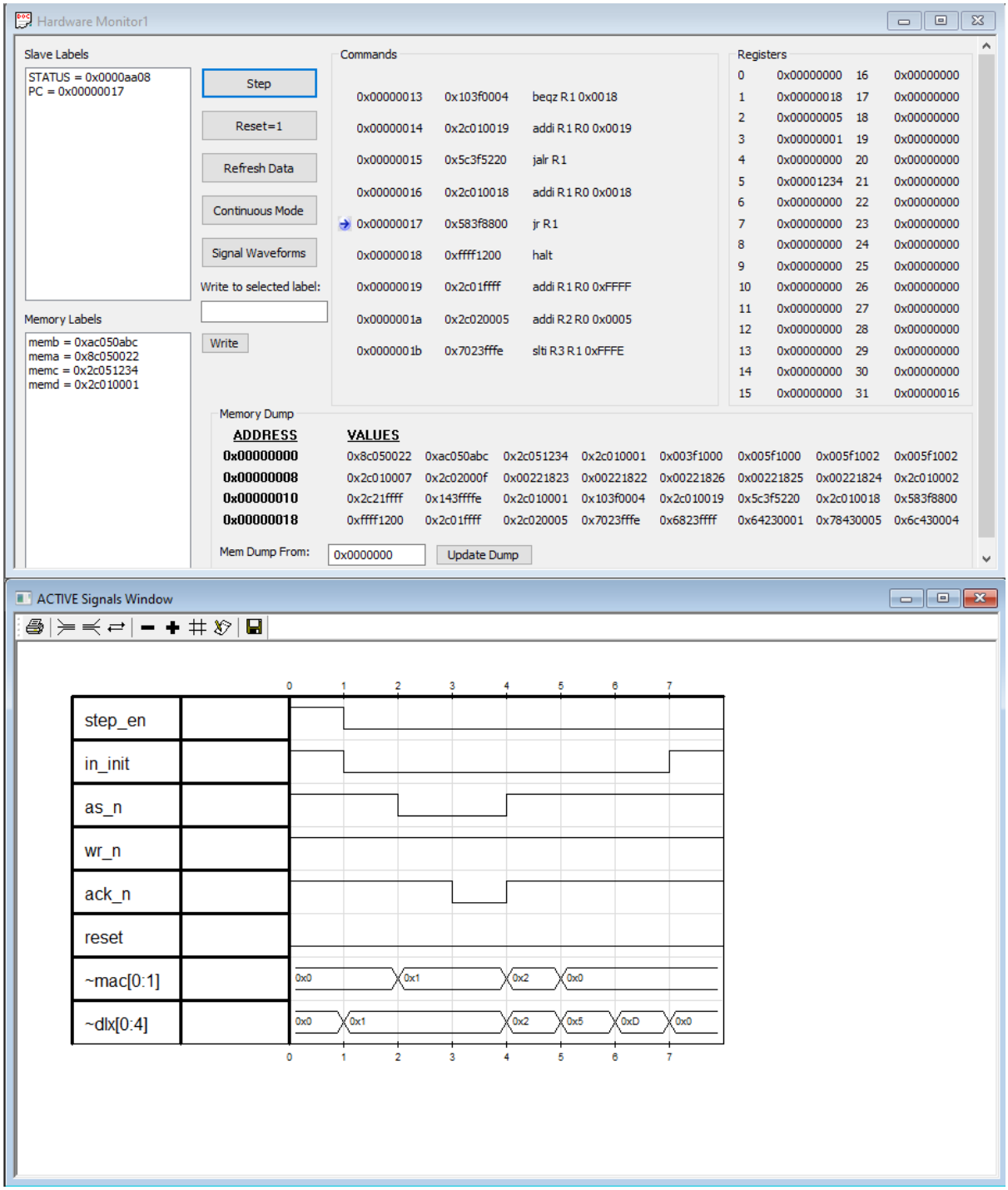

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x0000001f
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
0x00000021 0x5bff0000 jr R31
0x00000022 0x0000ffff no disassembly
0x00000023 0x00000000 slli R0 R0
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0 1 2 3 4 5 6 7
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x4 0xD 0x0

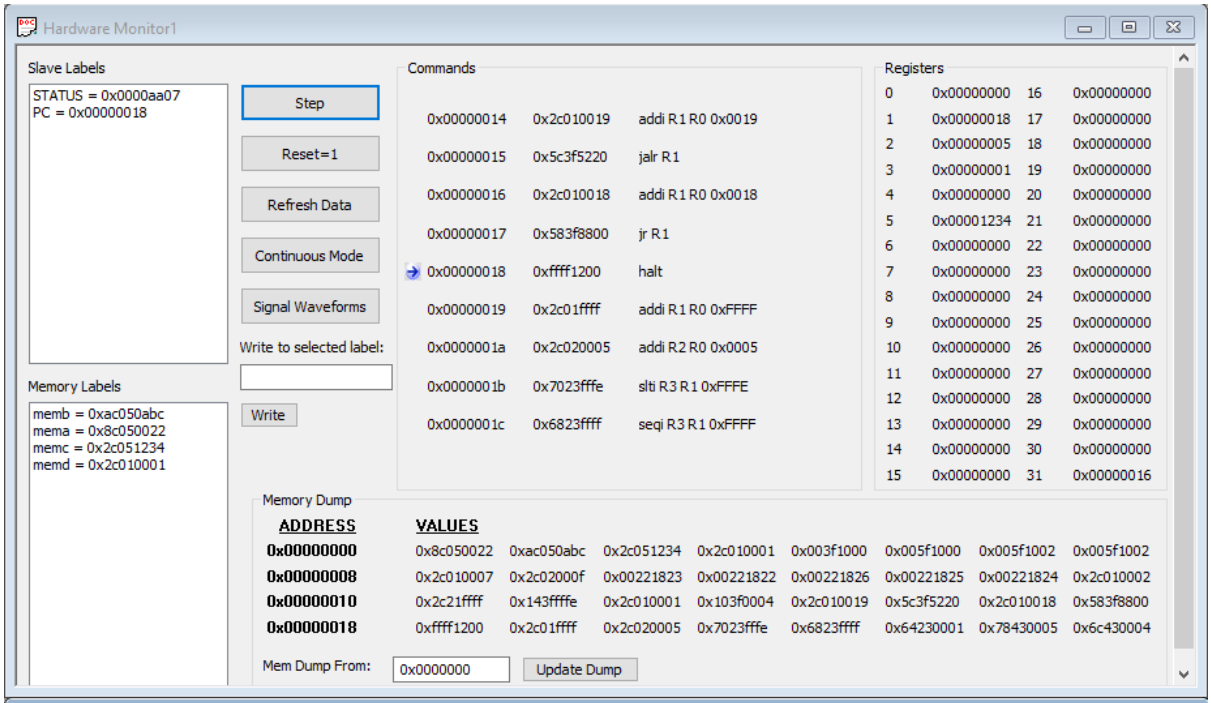

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000020
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
0x00000021 0x5bff0000 jr R31
0x00000022 0x0000ffff no disassembly
0x00000023 0x00000000 slli R0 R0
0x00000024 0x00000000 slli R0 R0
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump

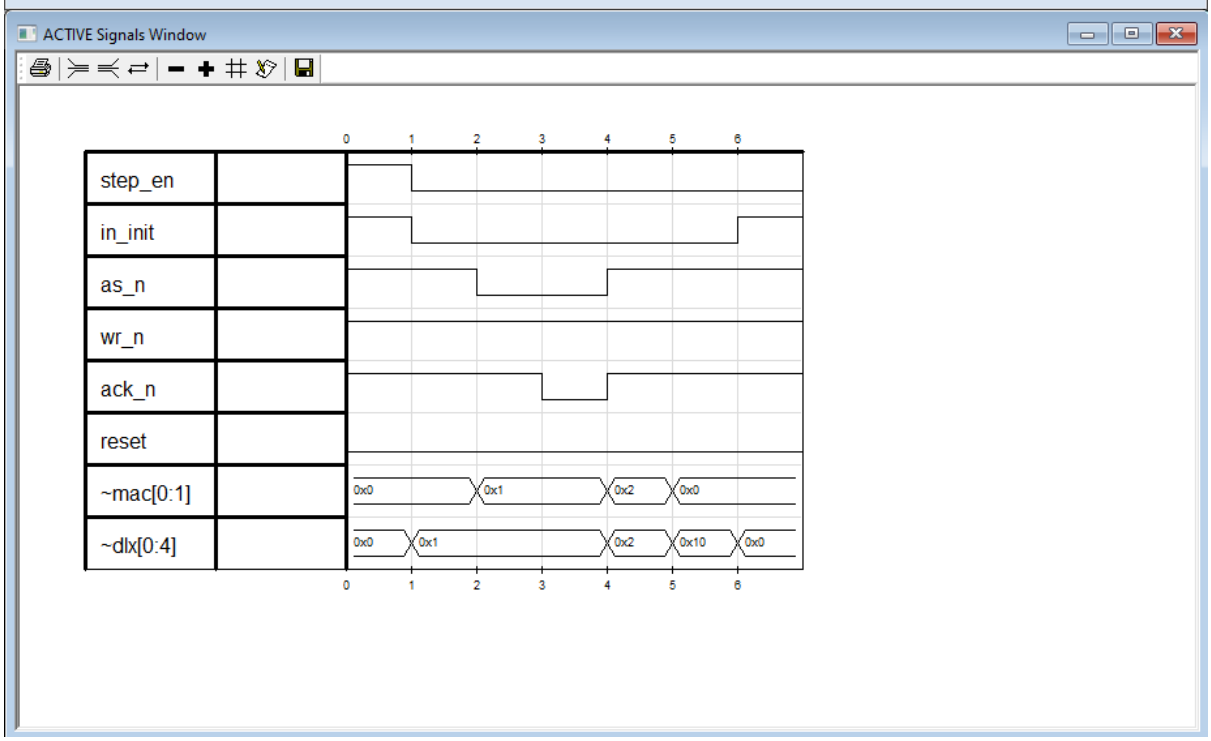
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x4 0xD 0x0
0 1 2 3 4 5 6 7

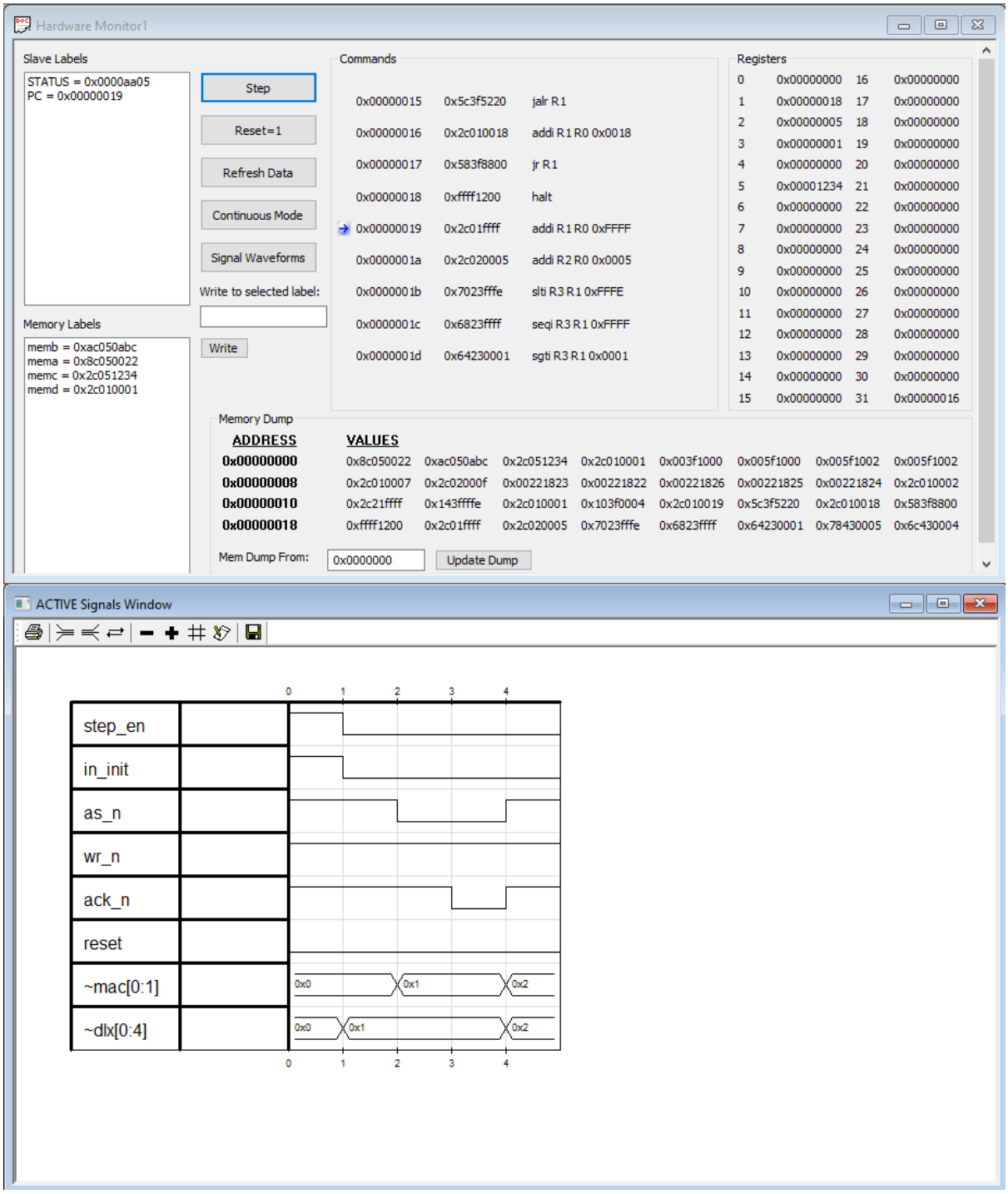

Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000021
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x0000001d 0x64230001 sgti R3 R1 0x0001
0x0000001e 0x78430005 slei R3 R2 0x0005
0x0000001f 0x6c430004 sgei R3 R2 0x0004
0x00000020 0x74430009 snei R3 R2 0x0009
0x00000021 0x5bff0000 jr R31
0x00000022 0x0000ffff no disassembly
0x00000023 0x00000000 slli R0 R0
0x00000024 0x00000000 slli R0 R0
0x00000025 0x00000000 slli R0 R0
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From:
0x0000000
Update Dump
ACTIVE Signals Window
0 1 2 3 4 5 6 7
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x4 0xD 0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa07
PC = 0x00000016
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000012 0x2c010001 addi R1 R0 0x0001
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
Registers
0 0x00000000 16 0x00000000
1 0xffffffff 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000000 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
0 1 2 3 4 5 6
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0 0x1 0x2 0x0
0x0 0x1 0x2 0x10 0x0
0 1 2 3 4 5 6


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa08
PC = 0x00000017
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000013 0x103f0004 beqz R1 0x0018
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
Registers
0 0x00000000 16 0x00000000
1 0x00000018 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0
0x1
0x2
0x0
0x0
0x1
0x2
0x5
0xD
0x0
0 1 2 3 4 5 6 7


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa07
PC = 0x00000018
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000014 0x2c010019 addi R1 R0 0x0019
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
Registers
0 0x00000000 16 0x00000000
1 0x00000018 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143fffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump


ACTIVE Signals Window
0 1 2 3 4 5 6
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
0x0 0x1 0x2 0x0
~dlx[0:4]
0x0 0x1 0x2 0x10 0x0
0 1 2 3 4 5 6


Hardware Monitor1
Slave Labels
STATUS = 0x0000aa05
PC = 0x00000019
Step
Reset=1
Refresh Data
Continuous Mode
Signal Waveforms
Write to selected label:
Write
Memory Labels
memb = 0xac050abc
mema = 0x8c050022
memc = 0x2c051234
memd = 0x2c010001
Commands
0x00000015 0x5c3f5220 jalr R1
0x00000016 0x2c010018 addi R1 R0 0x0018
0x00000017 0x583f8800 jr R1
0x00000018 0xffff1200 halt
0x00000019 0x2c01ffff addi R1 R0 0xFFFF
0x0000001a 0x2c020005 addi R2 R0 0x0005
0x0000001b 0x7023fffe slti R3 R1 0xFFFE
0x0000001c 0x6823ffff seqi R3 R1 0xFFFF
0x0000001d 0x64230001 sgti R3 R1 0x0001
Registers
0 0x00000000 16 0x00000000
1 0x00000018 17 0x00000000
2 0x00000005 18 0x00000000
3 0x00000001 19 0x00000000
4 0x00000000 20 0x00000000
5 0x00001234 21 0x00000000
6 0x00000000 22 0x00000000
7 0x00000000 23 0x00000000
8 0x00000000 24 0x00000000
9 0x00000000 25 0x00000000
10 0x00000000 26 0x00000000
11 0x00000000 27 0x00000000
12 0x00000000 28 0x00000000
13 0x00000000 29 0x00000000
14 0x00000000 30 0x00000000
15 0x00000000 31 0x00000016
Memory Dump
ADDRESS VALUES
0x00000000 0x8c050022 0xac050abc 0x2c051234 0x2c010001 0x003f1000 0x005f1000 0x005f1002 0x005f1002
0x00000008 0x2c010007 0x2c02000f 0x00221823 0x00221822 0x00221826 0x00221825 0x00221824 0x2c010002
0x00000010 0x2c21ffff 0x143ffffe 0x2c010001 0x103f0004 0x2c010019 0x5c3f5220 0x2c010018 0x583f8800
0x00000018 0xffff1200 0x2c01ffff 0x2c020005 0x7023fffe 0x6823ffff 0x64230001 0x78430005 0x6c430004
Mem Dump From: 0x0000000
Update Dump
ACTIVE Signals Window
0 1 2 3 4
step_en
in_init
as_n
wr_n
ack_n
reset
~mac[0:1]
~dlx[0:4]
0x0 0x1 0x2
0x0 0x1 0x2
0 1 2 3 4

And here we have marko's approval that the simplified DLX is working as desired.

Marko Markov

to me

Hi,

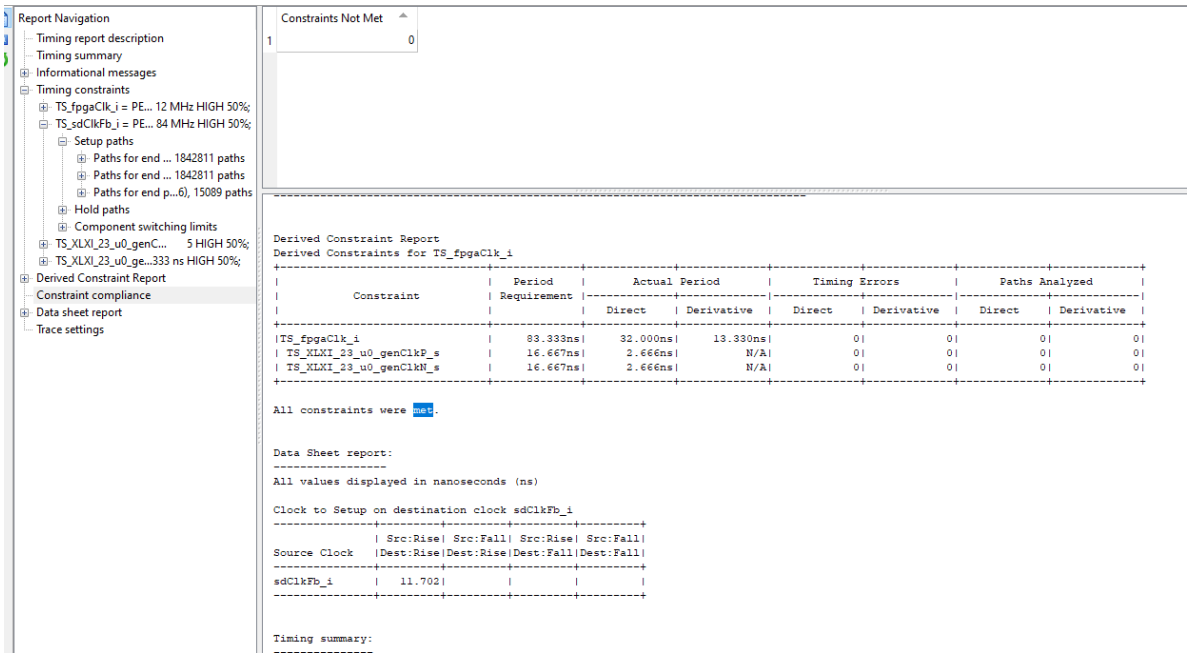
The test of 7.2 was run successfully at remote at 20h30

Regards
Marko

Sent from Mail for Windows

7.3) Timing optimization of your design

Initial Clock speed

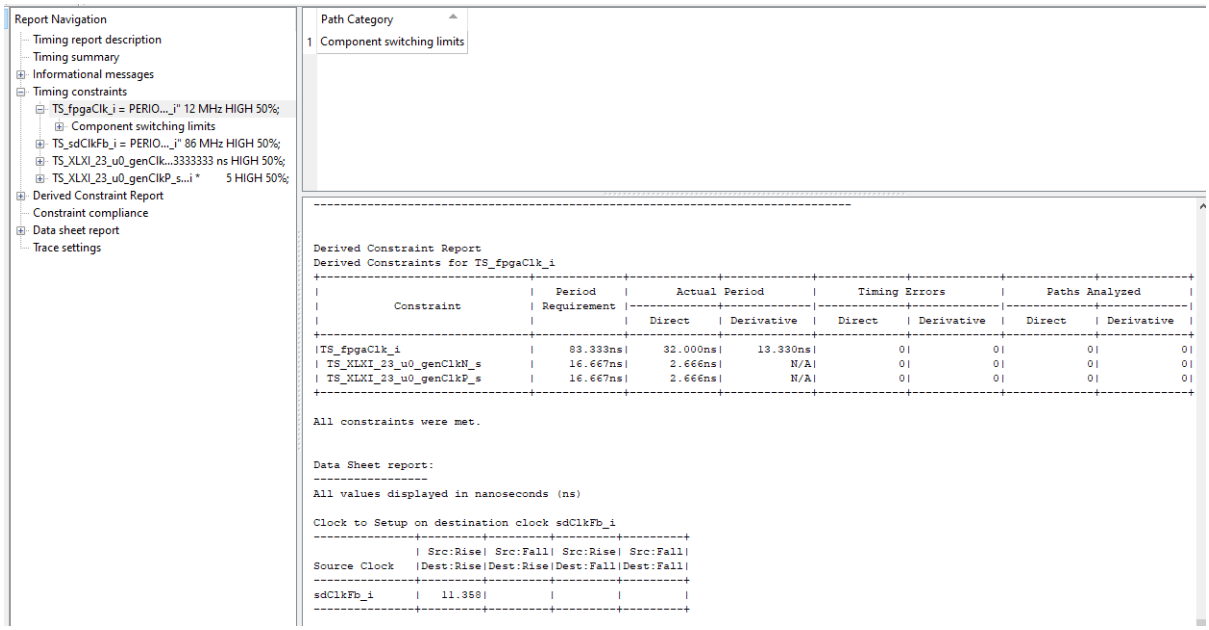


Initial Path delay

	Slack	Source	Destination	Path Delay	Requirement	Logic Levels
1	0.202	XLXI_55/XLXI_2/XLXI_1/state_FSM_FFd2	XLXI_55/XLXI_1/XLXI_16/XLXI_14/DOUT_0	11.666	11.904	15
2	0.202	XLXI_55/XLXI_2/XLXI_1/state_FSM_FFd2	XLXI_55/XLXI_1/XLXI_16/XLXI_14/DOUT_0	11.666	11.904	15
3	0.202	XLXI_55/XLXI_2/XLXI_1/state_FSM_FFd2	XLXI_55/XLXI_1/XLXI_16/XLXI_14/DOUT_0	11.666	11.904	15

This describes the path from DLX_STATE_MACHINE -> the Datapath registers/Program counter.

Final clock speed



Full Grey encoding

Our attempt to add a sixth bit and have a full grey encoding (full grey tree) ended up in a reduction in performance.

Shown here:

```
-- states
constant DECODE3 : std_logic_vector(5 downto 0)    := "100100";

constant HALT : std_logic_vector(5 downto 0) := "100010";
constant JR : std_logic_vector(5 downto 0) := "100011";
------------------------------------------------------------
constant DECODE1 : std_logic_vector(5 downto 0)    := "110000";

constant ADRCOMP : std_logic_vector(5 downto 0)   := "111000";
constant COPYGPR2MDR : std_logic_vector(5 downto 0) := "111001";
constant STORE : std_logic_vector(5 downto 0)     := "111011";
constant LOAD : std_logic_vector(5 downto 0)      := "111100";
constant COPYMDR2C : std_logic_vector(5 downto 0) := "111101";

constant ALUI : std_logic_vector(5 downto 0)       := "110100";
constant TESTI : std_logic_vector(5 downto 0)      := "110001";
constant WBI : std_logic_vector(5 downto 0) := "110101";

constant BRANCH : std_logic_vector(5 downto 0) := "110010";
constant B_TAKEN  : std_logic_vector(5 downto 0) := "110011";
------------------------------------------------------------
constant DECODE2 : std_logic_vector(5 downto 0)    := "101000";

constant ALU : std_logic_vector(5 downto 0)      := "101001";
constant SHIFT : std_logic_vector(5 downto 0)      := "101100";
constant WBR : std_logic_vector(5 downto 0) := "101101";

constant SAVE_PC : std_logic_vector(5 downto 0) := "101010";
constant JALR : std_logic_vector(5 downto 0) := "101011";
------------------------------------------------------------
constant INIT : std_logic_vector(5 downto 0)     := "000000";
constant FETCH : std_logic_vector(5 downto 0)     := "000001";
```

Unfortunately, the above did not work. The performance was faster with normal binary encoding. This is most likely do

Partial Grey Encoding

Another attempt to add grey encoding partially (turning all binary numbers into grey code) ended in a small boost in performance of ~0.5MHz.

Shown here:

INIT: 00000 -> 00000
FETCH: 00001 -> 00001
DECODE: 00010 -> 00011
ALU: 00011 -> 00010
TESTI: 00100 -> 00101
ALUI: 00101 -> 00111
SHIFT: 00110 -> 00110
ADRCOMP: 00111 -> 00100
LOAD: 01000 -> 01000
STORE: 01001 -> 01001
COPYMDR2C: 01010 -> 01101
COPYGPR2MDR: 01011 -> 01100
WBR: 01100 -> 01110
WBI: 01101 -> 01111
BRANCH: 01110 -> 01011
B_TAKEN: 01111 -> 01010
JR: 10000 -> 10110
SAVE_PC: 10001 -> 10111
JALR: 10010 -> 10101
HALT: 10011 -> 10100

7.4 Software program

Programming assignment + Bonus

First, we started with C code:

c_code.c (~\code\pictures_dlx_lab7) - VIM

```
unsigned int reverse(unsigned int a)
{
        unsigned int rev = 0;
        int n = 31;
        while(n>0)
        {
                if(a&1==1)
                        rev^=1;
                a>>=1;
                rev<=1;
                n--;
        }
        return rev;
}
int mod15(unsigned int n)
{
        int q16=num>>4;
        int mod16 = n&0xf;
        int sumModQuo16 = mod16+q16;
        int q = q16 + (mod16>=15);

        mul_q16 = q<<4;
        mul_q15 = mul_q16 - q;

        result = num - mul_q15;

        return result;
}
```

Reverse bits

Where rev is our output, a is the input, and n is the #bits -1.

Through every iteration of while (where there is 32 total iterations), we examine the LSB, and add it to our result. Our result is shifted to the left, and our input shifted the right.

For the input, shifting the right allows the next iteration to access the next LSB.

For the output, shifting to the left will build on our reverse effect – until at the end of the while loop, we will have a reversed number.

Modulo 15

Where result is the output, n is the input.

The modulo of 16 is known as the algorithm for it is very well known (n&0xf).
We have devised an algorithm, utilizing the mod16 of n, and the quotient of n (divided by 16) – we can attain the modulo 15 by the following common formula:

$$mod16 = n\ mod\ 16 = n - \left\lfloor \frac{n}{16} \right\rfloor * 16$$

$$mod15 = n\ mod\ 15 = n - \left\lfloor \frac{n}{15} \right\rfloor * 15$$

Where $q16 \triangleq \left\lfloor \frac{n}{16} \right\rfloor, q15 \triangleq \left\lfloor \frac{n}{15} \right\rfloor$

And: $mul\_q16 \triangleq q16 * 16\ , mul\_q15 \triangleq q15 * 15$

As it is easy to get q16 (by right shifting 4), and just as easy to multiply by 16 (left shifting by 4) we treat these as given. Now we must extract q15:

We have noticed the following relation (which can be proven by induction):

$$q15 = \begin{cases} q16 : mod16 + q16 < 15 \\ q16 + 1 : mod16 + q16 \geq 15 \end{cases}$$

Thus, we can establish mul_q15 for $mod15$ calculation:

$$mul\_q15 = 15 * q15 = 16 * q15 - q15$$

i.e. $mul\_q15 = (q15 \ll 4) - q15$.

Done.

Note: we created this method despite the fact there were other, easier alternatives that used branching, but we opted for this more efficient solution.

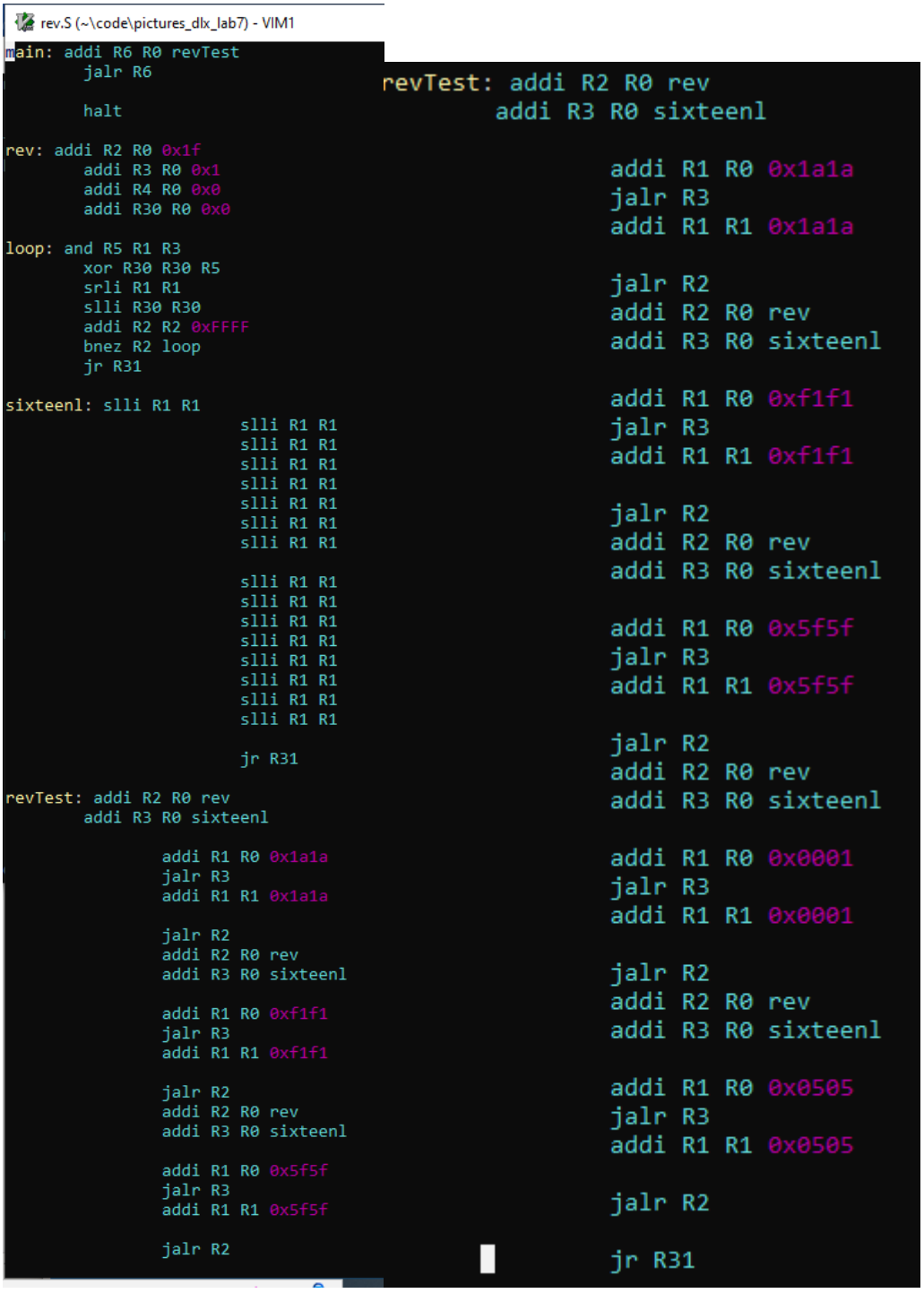
Reverse bits - MIPS

```
rev: addi R2 R0 0x1f
             addi R3 R0 0x1
             addi R4 R0 0x0
             addi R30 R0 0x0

loop: and R5 R1 R3
             xor R30 R30 R5
             srli R1 R1
             slli R30 R30
             addi R2 R2 0xFFFF
             bnez R2 loop
             jr R31
```

Modulo 15- MIPS

```
mod: add R28 R1 R0
             addi R3 R0 fourr
             jalr R3
             addi R2 R0 0xf
             and R29 R1 R2
             add R4 R29 R28
             sgei R4 R4 0xf
             add R28 R28 R4
             addi R27 R28 0x0
             addi R3 R0 fourl
             jalr R3
             sub R15 R27 R28
             sub R30 R1 R15
             jr R31

fourr: srli R28 R28
             srli R28 R28
             srli R28 R28
             srli R28 R28
             jr R31

fourl: slli R27 R27
             slli R27 R27
             slli R27 R27
             slli R27 R27
             jr R31
```

Reverse bits - Testing (cleanRev.c)



```
rev.S (~\code\pictures_dlx_lab7) - VIM1

main: addi R6 R0 revTest
        jalr R6

        halt

rev: addi R2 R0 0x1f
        addi R3 R0 0x1
        addi R4 R0 0x0
        addi R30 R0 0x0

loop: and R5 R1 R3
        xor R30 R30 R5
        srli R1 R1
        slli R30 R30
        addi R2 R2 0xFFFF
        bnez R2 loop
        jr R31

sixteenl: slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1

                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1
                        slli R1 R1

                        jr R31

revTest: addi R2 R0 rev
        addi R3 R0 sixteenl

                addi R1 R0 0x1a1a
                jalr R3
                addi R1 R1 0x1a1a

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0xf1f1
                jalr R3
                addi R1 R1 0xf1f1

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0x5f5f
                jalr R3
                addi R1 R1 0x5f5f

                jalr R2
```

```
revTest: addi R2 R0 rev
         addi R3 R0 sixteenl

                addi R1 R0 0x1a1a
                jalr R3
                addi R1 R1 0x1a1a

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0xf1f1
                jalr R3
                addi R1 R1 0xf1f1

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0x5f5f
                jalr R3
                addi R1 R1 0x5f5f

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0x0001
                jalr R3
                addi R1 R1 0x0001

                jalr R2
                addi R2 R0 rev
                addi R3 R0 sixteenl

                addi R1 R0 0x0505
                jalr R3
                addi R1 R1 0x0505

                jalr R2

                jr R31
```

Note: this test is not perfect for 0xf1f1 – its result was largely ignored.
Where output is R30, input is R1.
sixteenl is used to insert 32 bit data into R1.
revTest inputs 0x1a1a1a1a, 0xf1f1f1f1, 0x5f5f5f5f, 0x00010001, 0x05050505 into the rev function.
Checking R30, it seems to us that the rev function is working perfectly.

Modulo 15 – Testing (cleanMod.c)

mod.S (~\code\pictures_dlx_lab7) - VIM

```
main: addi R6 R0 modTest
        jalr R6
        halt
modTest: addi R1 R0 0x65
loopone: addi R1 R1 0xFFFF
        addi R2 R0 mod
        jalr R2
        bnez R1 loopone
        jr R31
mod: add R28 R1 R0
        addi R3 R0 fourr
        jalr R3
        addi R2 R0 0xf
        and R29 R1 R2
        add R4 R29 R28
        sgei R4 R4 0xf
        add R28 R28 R4
        addi R27 R28 0x0
        addi R3 R0 fourl
        jalr R3
        sub R15 R27 R28
        sub R30 R1 R15
        jr R31
fourr: srli R28 R28
        srli R28 R28
        srli R28 R28
        srli R28 R28
        jr R31
fourl: slli R27 R27
        slli R27 R27
        slli R27 R27
        slli R27 R27
        jr R31
~
~
```

Where output is R30, input is R1.

This test goes through the numbers 64->0, and tests the mod15 of them all.
Functions: fourr – right shift R28 by 4. Fourl – left shift R27 by 4.