Lab 4b Tutorial

Data-Path Design

OVERVIEW

In this lab we will build the and test the data-path for the 32-bit processor. Please copy the following VHDL files from the previous labs into your Lab4b folder, then add them to your project in the Add Files section of the New Project Wizard:

- From Lab 2:
  - "register32.vhd"
  - "add.vhd"
  - "mux2to1.vhd"
  - "pc.vhd"
- From Lab 3a:
  - "fulladd.vhd"
  - "adder4.vhd"
  - "adder16.vhd"
  - "adder32.vhd"
  - "alu.vhd"
- From Lab 4a:
  - "data_mem.vhd"

We also need to write the following VHDL files:

- “LZE.vhd”
- “UZE.vhd”
- “RED.vhd”
- “mux4to1.vhd”
- Data_Path.vhd”

PROCEDURE

“register32.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6   entity register32 is
7   port(
8      d     :  in std_logic_vector(31 downto 0);
9      ld    :  in std_logic;
10     clr      :  in std_logic;
11     clk      :  in std_logic;
12     Q     :  out   std_logic_vector(31 downto 0)
13     );
14  end register32;
15
16  architecture Behavior of register32 is
17  begin
18     process (ld, clr, clk)
19     begin
20        if clr = '1' then
21           Q <= (others => '0');
22        elsif ((clk'event and clk = '1') and (ld = '1')) then
23           Q <= d;
24        end if;
25     end process;
26  end Behavior;
```

“add.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6   entity add is
7   port (A  :  in    std_logic_vector(31 downto 0);
8         B  :  out   std_logic_vector(31 downto 0)
9         );
10  end add;
11
12  architecture Behavior of add is
13  begin
14  B <= A + 4;
15  end Behavior;
16
```

“mux2to1.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity mux2to1 is
5      port (   s     :  in std_logic;
6            w0, w1   :  in std_logic_vector(31 downto 0);
7            f     :  out   std_logic_vector(31 downto 0));
8   end mux2to1;
9
10  architecture Behavior of mux2to1 is
11  begin
12     with s select
13        f <= w0 when '0',
14             w1   when others;
15  end Behavior;
```

"pc.vhd"

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6   entity pc is
7   port(
8      clr   :  in    std_logic;
9      clk   :  in    std_logic;
10     ld :  in    std_logic;
11     inc   :  in    std_logic;
12     d  :  in    std_logic_vector(31 downto 0);
13     q  :  out   std_logic_vector(31 downto 0)
14     );
15  end pc;
16
17  architecture Behavior of pc is
18     component add
19        port (
20           A  :  in    std_logic_vector(31 downto 0);
21           B  :  out   std_logic_vector(31 downto 0)
22        );
23     end component;
24     component mux2to1
25        port (
26           s     :  in std_logic;
27           w0, w1   :  in std_logic_vector(31 downto 0);
28           f     :  out   std_logic_vector(31 downto 0)
29           );
30     end component;
31     component register32
32        port(
33           d     :  in std_logic_vector(31 downto 0);
34           ld    :  in std_logic;
35           clr      :  in std_logic;
36           clk      :  in std_logic;
37           Q     :  out   std_logic_vector(31 downto 0)
38        );
39     end component;
40     signal add_out : std_logic_vector(31 downto 0);
41     signal mux_out : std_logic_vector(31 downto 0);
42     signal q_out   : std_logic_vector(31 downto 0);
43     --signal ld    : std_logic := '1';
44  begin
45     add0: add port map(q_out, add_out);
46     mux0: mux2to1 port map(inc, d, add_out, mux_out);
47     reg0: register32 port map (mux_out, ld, clr, clk, q_out);
48     q <= q_out;
49  end Behavior;
50
```

“fulladd.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity fulladd is
6   port(
7      Cin, x, y : in  std_logic;
8      s, Cout   : out std_logic
9      );
10  end fulladd;
11
12  architecture Behavior of fulladd is
13  begin
14     s <= x xor y xor Cin;
15     Cout <= (x and y) or (Cin and x) or (Cin and y);
16  end Behavior;
17
```

"adder4.vhd"

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity adder4 is
6   port(
7      Cin      :  in std_logic;
8      X,Y      :  in  std_logic_vector(3 downto 0);
9      S     :  out   std_logic_vector(3 downto 0);
10     Cout  :  out   std_logic
11     );
12  end adder4;
13
14  architecture Behavior of adder4 is
15     component fulladd
16     port(
17     Cin, x, y : in  std_logic;
18     s, Cout   : out std_logic
19     );
20     end component;
21
22     signal C : std_logic_vector (1 to 3);
23  begin
24     stage0:  fulladd port map (Cin,  X(0), Y(0), S(0), C(1));
25     stage1:  fulladd port map (C(1), X(1), Y(1), S(1), C(2));
26     stage2:  fulladd port map (C(2), X(2), Y(2), S(2), C(3));
27     stage3:  fulladd port map (C(3), X(3), Y(3), S(3), Cout);
28  end Behavior;
29
```

"adder16.vhd"

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity adder16 is
6   port(
7      Cin      :  in std_logic;
8      X,Y      :  in  std_logic_vector(15 downto 0);
9      S     :  out   std_logic_vector(15 downto 0);
10     Cout  :  out   std_logic
11     );
12  end adder16;
13
14  architecture Behavior of adder16 is
15     component adder4
16     port(
17     Cin      :  in std_logic;
18     X,Y      :  in  std_logic_vector(3 downto 0);
19     S     :  out   std_logic_vector(3 downto 0);
20     Cout  :  out   std_logic
21     );
22     end component;
23
24     signal C : std_logic_vector (1 to 3);
25  begin
26     stage0:  adder4 port map (Cin,   X(3  downto  0), Y(3 downto   0), S(3 downto   0), C(1
    ));
27     stage1:  adder4 port map (C(1),  X(7  downto  4), Y(7 downto   4), S(7 downto   4), C(2
    ));
28     stage2:  adder4 port map (C(2),  X(11 downto  8), Y(11 downto  8), S(11 downto  8), C(3
    ));
29     stage3:  adder4 port map (C(3),  X(15 downto 12), Y(15 downto 12), S(15 downto 12),
    Cout);
30  end Behavior;
31
```

“adder32.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity adder32 is
6   port(
7      Cin      :  in std_logic;
8      X,Y      :  in  std_logic_vector(31 downto 0);
9      S     :  out   std_logic_vector(31 downto 0);
10     Cout  :  out   std_logic
11     );
12  end adder32;
13
14  architecture Behavior of adder32 is
15     component adder16
16     port(
17     Cin      :  in std_logic;
18     X,Y      :  in  std_logic_vector(15 downto 0);
19     S     :  out   std_logic_vector(15 downto 0);
20     Cout  :  out   std_logic
21     );
22     end component;
23
24     signal C : std_logic;
25  begin
26     stage0:  adder16 port map (Cin,   X(15  downto  0), Y(15 downto   0), S(15 downto   0),
       C);
27     stage1:  adder16 port map (C, X(31 downto 16), Y(31 downto 16), S(31 downto 16), Cout);
28  end Behavior;
29
```

"alu.vhd"

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5   use ieee.numeric_std.all;
6
7   entity alu is
8   port(
9      a     :  in  std_logic_vector(31 downto 0);
10     b     :  in  std_logic_vector(31 downto 0);
11     op    :  in  std_logic_vector(2 downto 0);
12     result   :  out   std_logic_vector(31 downto 0);
13     zero  :  out std_logic;
14     cout  :  out std_logic
15     );
16  end alu;
17
18  architecture Behavior of alu is
19     component adder32
20           port(
21              Cin      :  in std_logic;
22              X,Y      :  in  std_logic_vector(31 downto 0);
23              S     :  out   std_logic_vector(31 downto 0);
24              Cout  :  out   std_logic
25           );
26     end component;
27
28     signal result_s: std_logic_vector(31 downto 0):= (others => '0');
29     signal result_add: std_logic_vector(31 downto 0):= (others => '0');
30     signal result_sub:  std_logic_vector(31 downto 0):= (others => '0');
31     signal cout_s : std_logic := '0';
32     signal cout_add : std_logic := '0';
33     signal cout_sub : std_logic := '0';
34     signal zero_s : std_logic;
35
36  begin
37        add0 : adder32 port map (op(2), a, b, result_add, cout_add);
38        sub0 : adder32 port map (op(2), a, not b, result_sub, cout_sub);
39
40     process (a, b, op)
41     begin
42        case (op) is
43           when "000" =>           -- "000" a and b
44              result_s<= a and b;
45              cout_s <= '0';
46           when "001" =>           -- "001" a or b
47              result_s <= a or b;
48              cout_s <= '0';
49           when "010" =>           -- "010" a + b
50              result_s <= result_add;
51              cout_s <= cout_add;
52           when "011" =>           -- "011" b
53              result_s <= b;
54              cout_s <= '0';
55           when "110" =>           -- "110" a - b
56              result_s <= result_sub;
57              cout_s <= cout_sub;
58           when "100" =>           -- a sll 1
59              result_s <= a(30 downto 0) & '0';
60              cout_s <= a(31);
61           when "101" =>           -- a srl 1
```

```
62              result_s <= '0' & a(31 downto 1);
63              cout_s <= '0';
64           when others =>
65              result_s <= a;
66              cout_s <= '0';
67        end case;
68
69        case (result_s) is
70           when (others => '0') =>
71              zero_s <= '1';
72           when others =>
73              zero_s <= '0';
74        end case;
75     end process;
76
77     result <= result_s;
78     cout <= cout_s;
79     zero <= zero_s;
80  end Behavior;
81
82
```

“data_mem.vhd”

```
 1   library ieee;
 2   use ieee.std_logic_1164.all;
 3   use ieee.numeric_std.all;
 4
 5   entity data_mem is
 6   port(
 7      clk         :  in std_logic;
 8      addr     :  in unsigned(7 downto 0);
 9      data_in     :  in    std_logic_vector(31 downto 0);
10      wen         :  in    std_logic;
11      en       :  in std_logic;
12      data_out :  out   std_logic_vector(31 downto 0)
13      );
14   end data_mem;
15
16   architecture Behavior of data_mem is
17      type RAM is array (0 to 255) of std_logic_vector(31 downto 0);
18      signal DATAMEM : RAM;
19   begin
20      process(clk, en, wen)
21      begin
22         if(clk'event and clk='0') then
23            if (en = '0') then
24               data_out <= (others => '0');
25            else
26               if (wen = '0') then
27                  data_out <= DATAMEM(to_integer(addr));
28               end if;
29               if (wen = '1') then
30                  DATAMEM(to_integer(addr)) <= data_in;
31                  data_out <= (others => '0');
32               end if;
33            end if;
34         end if;
35      end process;
36   end Behavior;
```

“LZE.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity LZE is
6   port( LZE_in      :     in    std_logic_vector(31 downto 0);
7         LZE_out   :     out     std_logic_vector(31 downto 0)
8      );
9   end entity;
10
11  architecture Behavior of LZE is
12  signal  zeros: std_logic_vector(15 downto 0) := (others => '0');
13  begin
14     LZE_out <= zeros & LZE_in(15 downto 0);
15  end Behavior;
```

“UZE.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity UZE is
6   port(
7      UZE_in   :  in std_logic_vector(31 downto 0);
8      UZE_out  :  out   std_logic_vector(31 downto 0)
9      );
10  end entity;
11
12  architecture Behavior of UZE is
13     signal zeros : std_logic_vector(15 downto 0) := (others => '0');
14  begin
15     UZE_out <= UZE_in(15 downto 0) & zeros;
16  end Behavior;
17
```

“RED.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity RED is
6   port(
7      RED_in   :  in std_logic_vector(31 downto 0);
8      RED_out  :  out   unsigned(7 downto 0)
9      );
10  end entity;
11
12  architecture Behavior of RED is
13  begin
14     RED_out <= unsigned (RED_in(7 downto 0));
15  end Behavior;
```

“mux4to1.vhd”

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity mux4to1 is
5      port (   s           :  in    std_logic_vector (1 downto 0);
6            X1, X2, X3, X4 :  in    std_logic_vector (31 downto 0);
7            f            :  out     std_logic_vector (31 downto 0));
8   end mux4to1;
9
10  architecture Behavior of mux4to1 is
11  begin
12     with s select
13        f <=  X1 when "00",
14              X2 when "01",
15              X3 when "10",
16              X4 when "11";
17  end Behavior;
```

"Data_Path.vhd"

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6   entity data_path is
7      port(
8         --Clock Signal
9         Clk, mClk   :  in std_logic;
10
11        --Memory Signals
12        WEN, EN     :  in std_logic;
13
14        --Register Control Signals (CLR and LD)
15        Clr_A, Ld_A     :  in std_logic;
16        Clr_B, Ld_B     :  in std_logic;
17        Clr_C, Ld_C     :  in std_logic;
18        Clr_Z, Ld_Z     :  in std_logic;
19        ClrPC, Ld_PC    :  in std_logic;
20        ClrIR, Ld_IR    :  in std_logic;
21
22
23        --Register Outputs
24        Out_A       :  out   std_logic_vector(31 downto 0);
25        Out_B       :  out   std_logic_vector(31 downto 0);
26        Out_C       :  out   std_logic;
27        Out_Z       :  out   std_logic;
28        Out_PC          :  out   std_logic_vector(31 downto 0);
29        Out_IR          :  out   std_logic_vector(31 downto 0);
30
31        --Special Inputs to PC
32        Inc_PC          :  in std_logic;
33
34        --Address and Data Bus signals for debugging
35        ADDR_OUT    :  out   std_logic_vector(31 downto 0);
36        DATA_IN         :  in std_logic_vector(31 downto 0);
37        DATA_BUS,
38        MEM_OUT,
39        MEM_IN          :  out   std_logic_vector(31 downto 0);
40        MEM_ADDR    :  out   unsigned(7 downto 0);
41
42        --Various MUX controls
43        DATA_MUX    :  in std_logic_vector(1 downto 0);
44        REG_MUX         :  in std_logic;
45        A_MUX,
46        B_MUX       :  in std_logic;
47        IM_MUX1         :  in std_logic;
48        IM_MUX2         :  in std_logic_vector(1 downto 0);
49
50        --ALU Operations
51        ALU_Op          :  in std_logic_vector(2 downto 0)
52        );
53  end entity;
54
55  architecture Behavior of Data_Path is
56     -- Component Instantiations
57     --Data Memory Module
58     component data_mem is
59        port(
60           clk          :  in std_logic;
61           addr     :  in unsigned(7 downto 0);
```

```
62            data_in     :  in    std_logic_vector(31 downto 0);
63            wen         :  in    std_logic;
64            en       :  in std_logic;
65            data_out :  out   std_logic_vector(31 downto 0)
66            );
67         end component;
68
69
70      --Register32
71      component register32 is
72         port(
73            d     :  in std_logic_vector(31 downto 0);
74            ld    :  in std_logic;
75            clr      :  in std_logic;
76            clk      :  in std_logic;
77            Q     :  out   std_logic_vector(31 downto 0)
78            );
79      end component;
80
81      --Program Counter
82      component pc is
83         port(
84            clr   :  in    std_logic;
85            clk   :  in    std_logic;
86            ld    :  in    std_logic;
87            inc   :  in    std_logic;
88            d  :  in    std_logic_vector(31 downto 0);
89            q  :  out   std_logic_vector(31 downto 0)
90            );
91         end component;
92
93      --LZE
94      component LZE is
95         port( LZE_in      :     in    std_logic_vector(31 downto 0);
96               LZE_out  :     out      std_logic_vector(31 downto 0)
97            );
98      end component;
99
100     --UZE
101     component UZE is
102        port(
103           UZE_in   :  in std_logic_vector(31 downto 0);
104           UZE_out  :  out   std_logic_vector(31 downto 0)
105           );
106     end component;
107
108     --RED
109     component RED is
110        port(
111           RED_in   :  in std_logic_vector(31 downto 0);
112           RED_out  :  out   unsigned(7 downto 0)
113           );
114     end component;
115
116     --Mux2to1
117     component mux2to1 is
118        port(
119           s     :  in std_logic;
120           w0, w1   :  in std_logic_vector(31 downto 0);
121           f     :  out   std_logic_vector(31 downto 0)
122           );
```

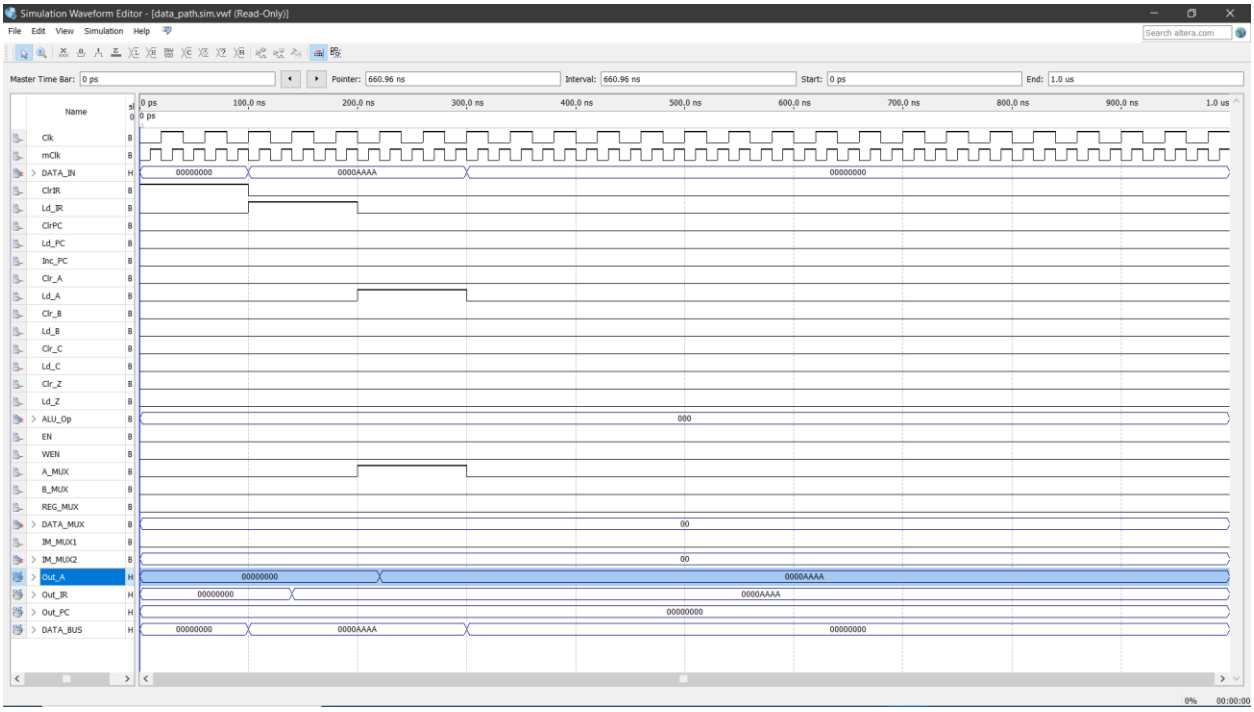
```
123     end component;
124
125     --Mux4to1
126     component mux4to1 is
127        port(
128           s           :  in    std_logic_vector (1 downto 0);
129           X1, X2, X3, X4 :  in    std_logic_vector (31 downto 0);
130           f           :  out     std_logic_vector (31 downto 0)
131           );
132     end component;
133
134     --ALU
135     component alu is
136        port(
137           a     :  in  std_logic_vector(31 downto 0);
138           b     :  in  std_logic_vector(31 downto 0);
139           op    :  in  std_logic_vector(2 downto 0);
140           result   :  out   std_logic_vector(31 downto 0);
141           zero  :  out std_logic;
142           cout  :  out std_logic
143           );
144        end component;
145
146
147     --Signal Instantiations
148     signal   IR_OUT            :  std_logic_vector(31 downto 0);
149     signal   data_bus_s        :  std_logic_vector(31 downto 0);
150     signal   LZE_out_PC  :        std_logic_vector(31 downto 0);
151     signal   LZE_out_A_Mux     :  std_logic_vector(31 downto 0);
152     signal   LZE_out_B_Mux     :  std_logic_vector(31 downto 0);
153     signal   RED_out_Data_Mem  :  unsigned(7 downto 0);
154     signal   A_Mux_out         :  std_logic_vector(31 downto 0);
155     signal   B_Mux_out         :  std_logic_vector(31 downto 0);
156     signal   reg_A_out         :  std_logic_vector(31 downto 0);
157     signal   reg_B_out         :  std_logic_vector(31 downto 0);
158     signal   reg_Mux_out       :  std_logic_vector(31 downto 0);
159     signal   data_mem_out      :  std_logic_vector(31 downto 0);
160     signal   UZE_IM_MUX1_out      :  std_logic_vector(31 downto 0);
161     signal   IM_MUX1_out       :  std_logic_vector(31 downto 0);
162     signal   LZE_IM_MUX2_out      :  std_logic_vector(31 downto 0);
163     signal   IM_MUX2_out       :  std_logic_vector(31 downto 0);
164     signal   ALU_out           :  std_logic_vector(31 downto 0);
165     signal   zero_flag         :  std_logic;
166     signal   carry_flag        :  std_logic;
167     signal   temp           :  std_logic_vector(30 downto 0) := (others => '0');
168     signal   out_pc_sig     :  std_logic_vector(31 downto 0);
169
170  begin
171     IR:   register32 port map(
172           data_bus_s,
173           Ld_IR,
174           ClrIR,
175           Clk   ,
176           IR_OUT
177           );
178
179     LZE_PC:  LZE   port map (
180           IR_OUT,
181           LZE_out_PC
182           );
183
```

```
184     PC0:  PC port map (
185           CLRPC,
186           Clk,
187           ld_PC,
188           INC_PC,
189           LZE_out_PC,
190           --ADDR_OUT
191           out_Pc_sig
192           );
193
194     LZE_A_Mux:  LZE   port map (
195           IR_OUT,
196           LZE_out_A_Mux
197           );
198
199     A_Mux0:  mux2to1 port map (
200           A_MUX,
201           data_bus_s, LZE_out_A_Mux,
202           A_Mux_out
203           );
204
205     Reg_A:   register32 port map (
206           A_Mux_out,
207           Ld_A,
208           Clr_A,
209           Clk   ,
210           reg_A_out
211           );
212
213     LZE_B_Mux:  LZE   port map (
214           IR_OUT,
215           LZE_out_B_Mux
216           );
217
218     B_Mux0:  mux2to1 port map (
219           B_MUX,
220           data_bus_s, LZE_out_B_Mux,
221           B_Mux_out
222           );
223
224     Reg_B:   register32 port map (
225           B_Mux_out,
226           Ld_B,
227           Clr_B,
228           Clk   ,
229           reg_B_out
230           );
231
232     Reg_Mux0:   mux2to1 port map (
233           REG_MUX,
234           Reg_A_out, Reg_B_out,
235           Reg_Mux_out
236           );
237
238
239     RED_Data_Mem: RED port map (
240           IR_OUT,
241           RED_out_data_mem
242           );
243
244     Data_Mem0:  data_mem port map (
```

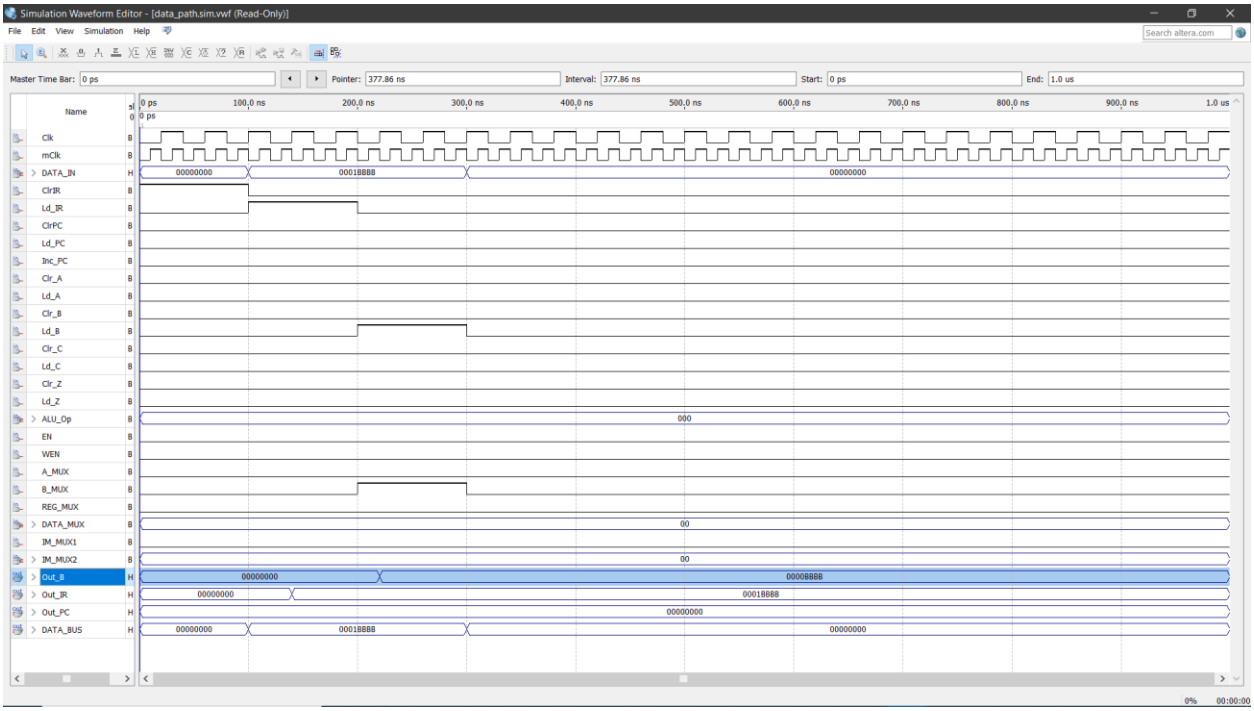
```
245           mClk,
246           RED_out_data_mem,
247           Reg_Mux_out,
248           WEN,
249           EN,
250           data_mem_out
251           );
252
253     UZE_IM_MUX1: UZE port map (
254           IR_OUT,
255           UZE_IM_MUX1_out
256           );
257
258     IM_MUX1a:   mux2to1 port map (
259           IM_MUX1,
260           reg_A_out, UZE_IM_MUX1_out,
261           IM_MUX1_out
262           );
263
264     LZE_IM_MUX2: LZE port map (
265           IR_OUT,
266           LZE_IM_MUX2_out
267           );
268
269     IM_MUX2a:   mux4to1 port map (
270           IM_MUX2,
271           reg_B_out, LZE_IM_MUX2_out,  (temp &'1'), (others => '0'),
272           IM_MUX2_out
273           );
274
275     ALU0: ALU port map (
276           IM_MUX1_out,
277           IM_MUX2_out,
278           ALU_OP,
279           ALU_out,
280           zero_flag,
281           carry_flag
282           );
283
284     DATA_MUX0:  mux4to1 port map (
285           DATA_MUX,
286           DATA_IN, data_mem_out, ALU_out, (others => '0'),
287           data_bus_s
288           );
289
290     DATA_BUS <= data_bus_s;
291     OUT_A <= reg_A_out;
292     OUT_B <= reg_B_out;
293     OUT_IR <= IR_OUT;
294     ADDR_OUT <= out_pc_sig;
295     OUT_PC <= out_pc_sig;
296
297     MEM_ADDR <= RED_out_Data_Mem;
298     MEM_IN <= Reg_Mux_out;
299     MEM_OUT <= data_mem_out;
300
301
302
303   end Behavior;
304
```

DATA PATH FUNCTIONAL SIMULATION WAVEFORMS

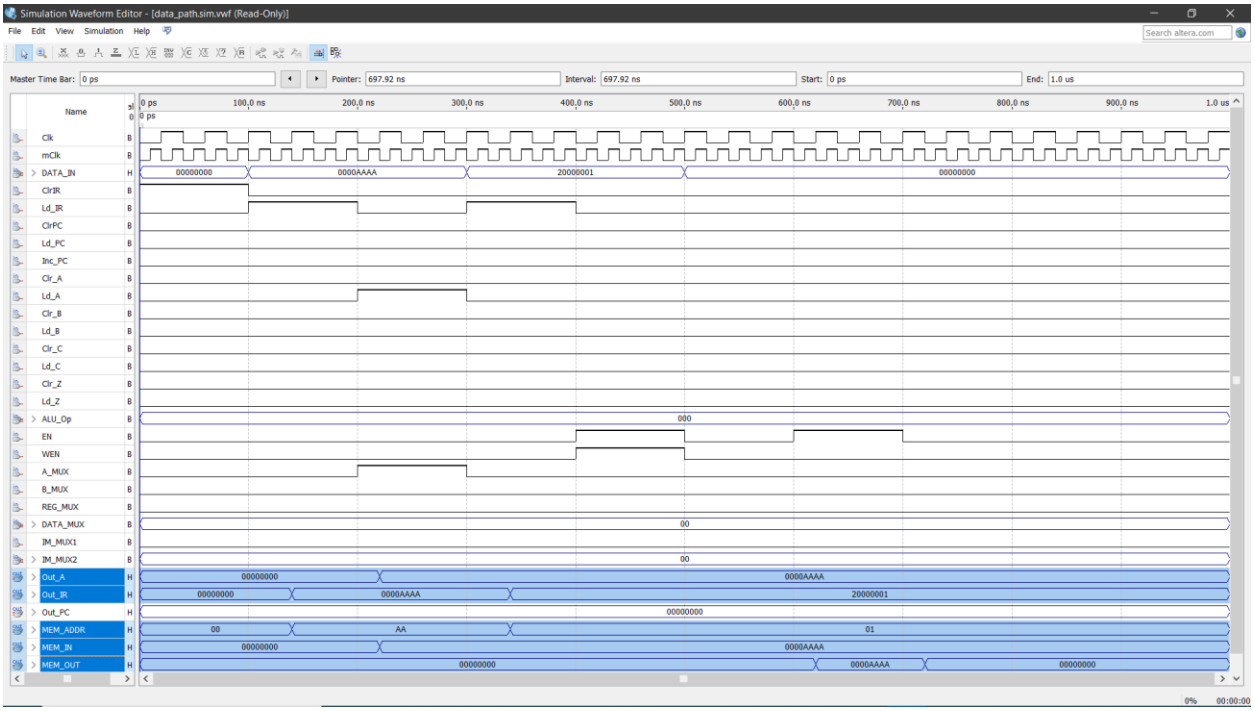
1. LDAI



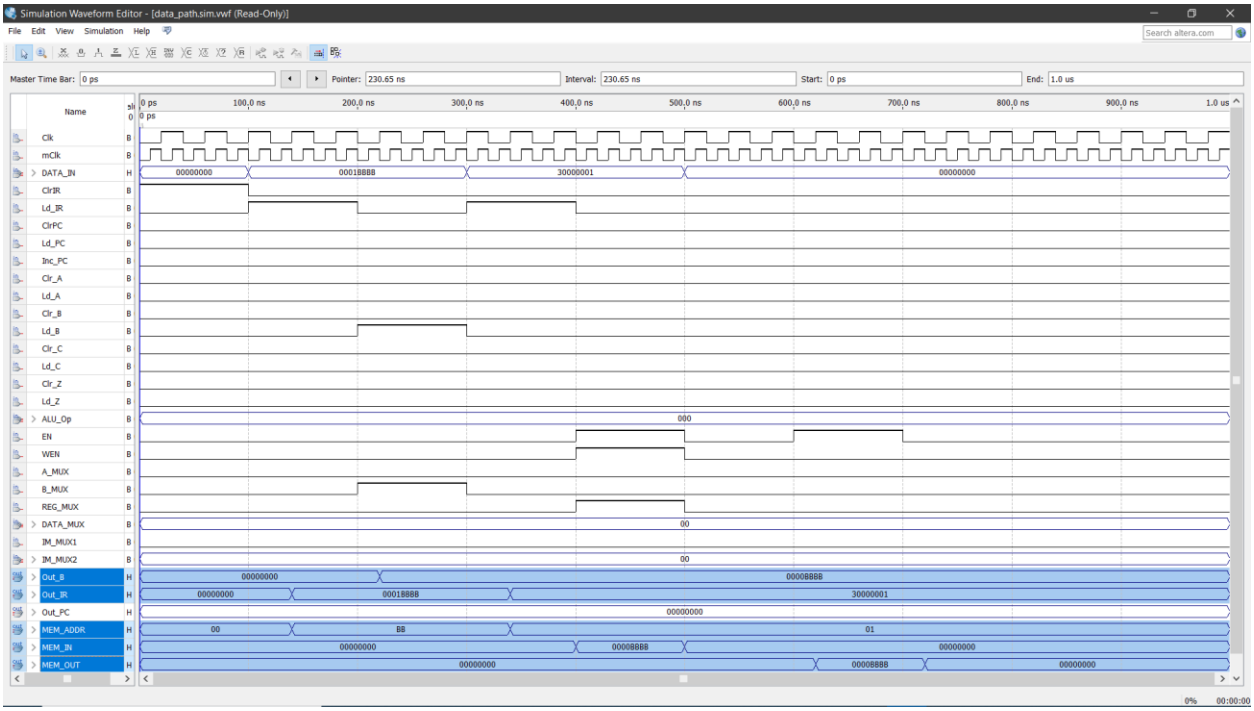
2. LDBI


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 377.86 ns
Interval: 377.86 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_B
Out_IR
Out_PC
DATA_BUS
00000000
0001BBBB
000
00
0000BBBB

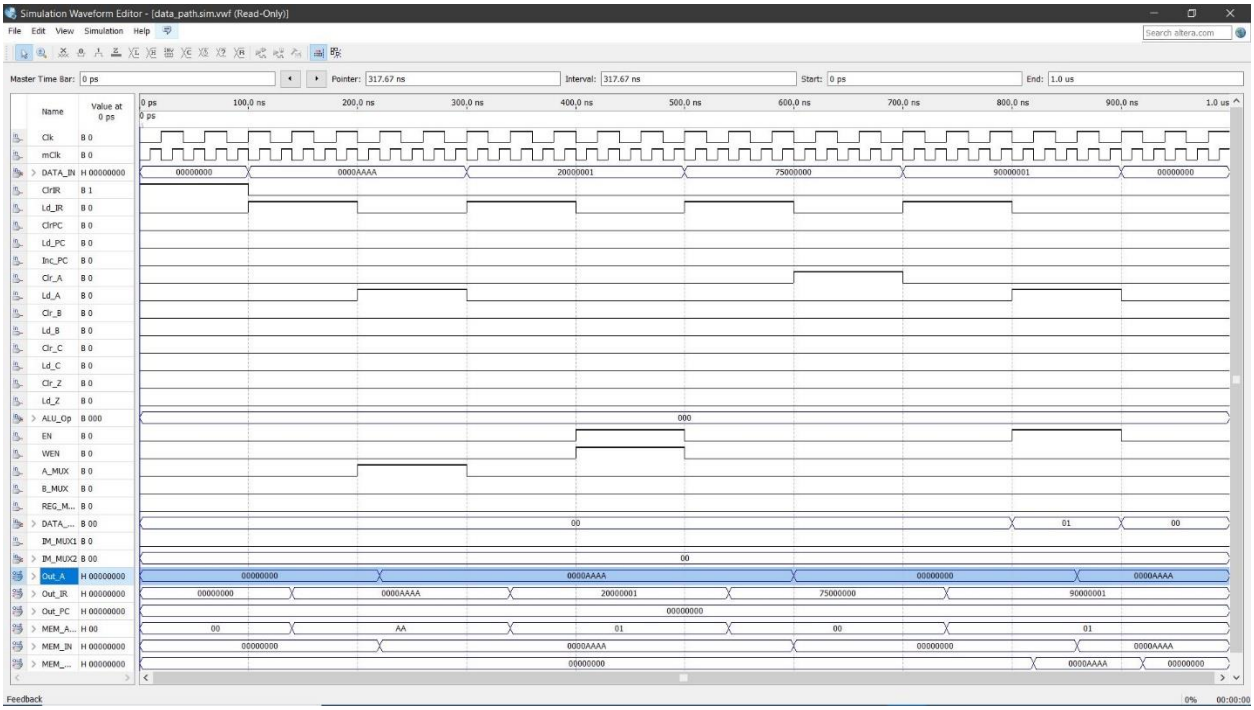
3. STA


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 697.92 ns
Interval: 697.92 ns
Start: 0 ps
End: 1.0 us
Name
0 ps
100,0 ns
200,0 ns
300,0 ns
400,0 ns
500,0 ns
600,0 ns
700,0 ns
800,0 ns
900,0 ns
1.0 us
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
MEM_ADDR
MEM_IN
MEM_OUT
00000000
0000AAAA
20000001
00000000
000
00
00
00000000
0000AAAA
00000000
0000AAAA
20000001
00000000
00
AA
01
00000000
0000AAAA
00000000
0000AAAA
00000000
0%
00:00:00

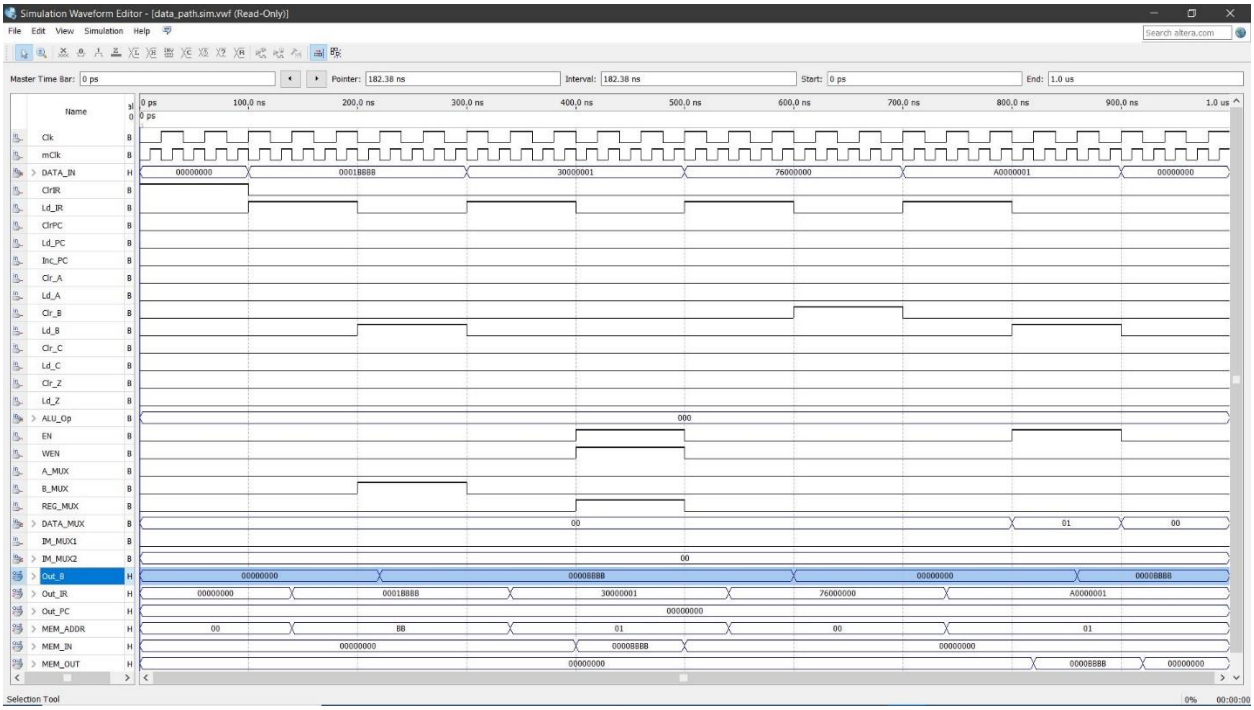
4. STB


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 230.65 ns
Interval: 230.65 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_B
Out_IR
Out_PC
MEM_ADDR
MEM_IN
MEM_OUT

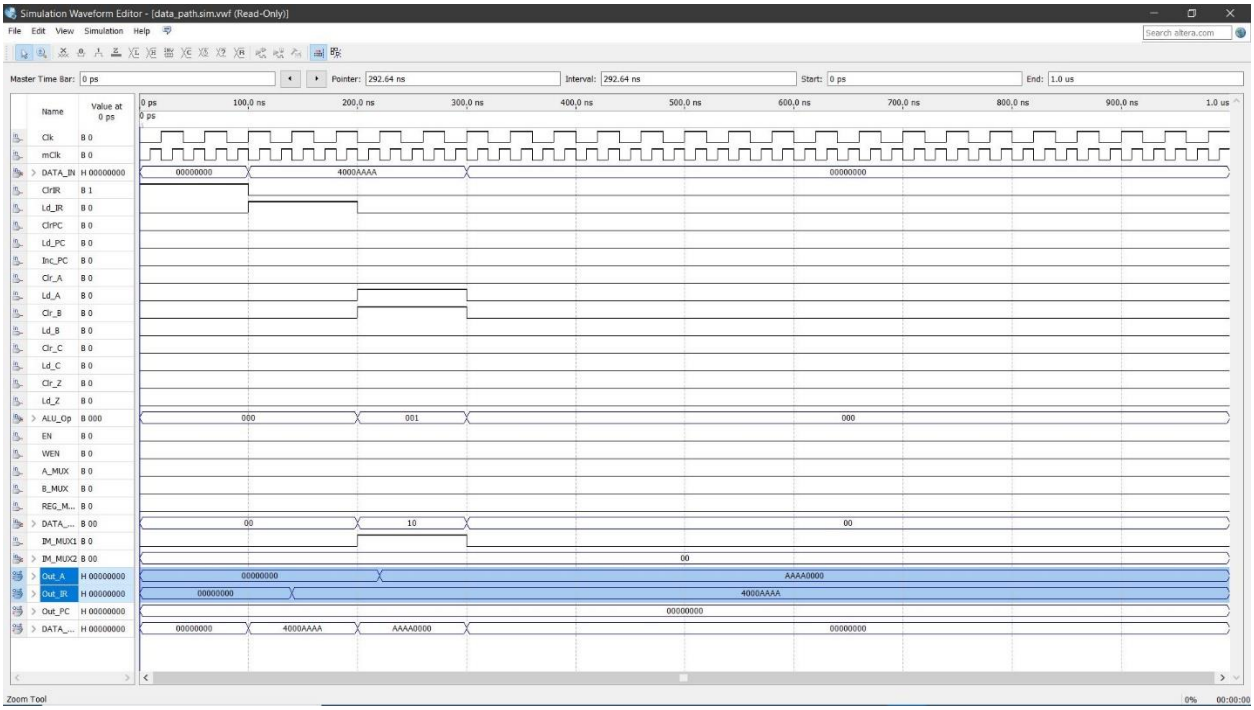
5. LDA


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 317.67 ns
Interval: 317.67 ns
Start: 0 ps
End: 1.0 us
Name
Value at 0 ps
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_M...
DATA_...
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
MEM_A...
MEM_IN
MEM_...
Feedback

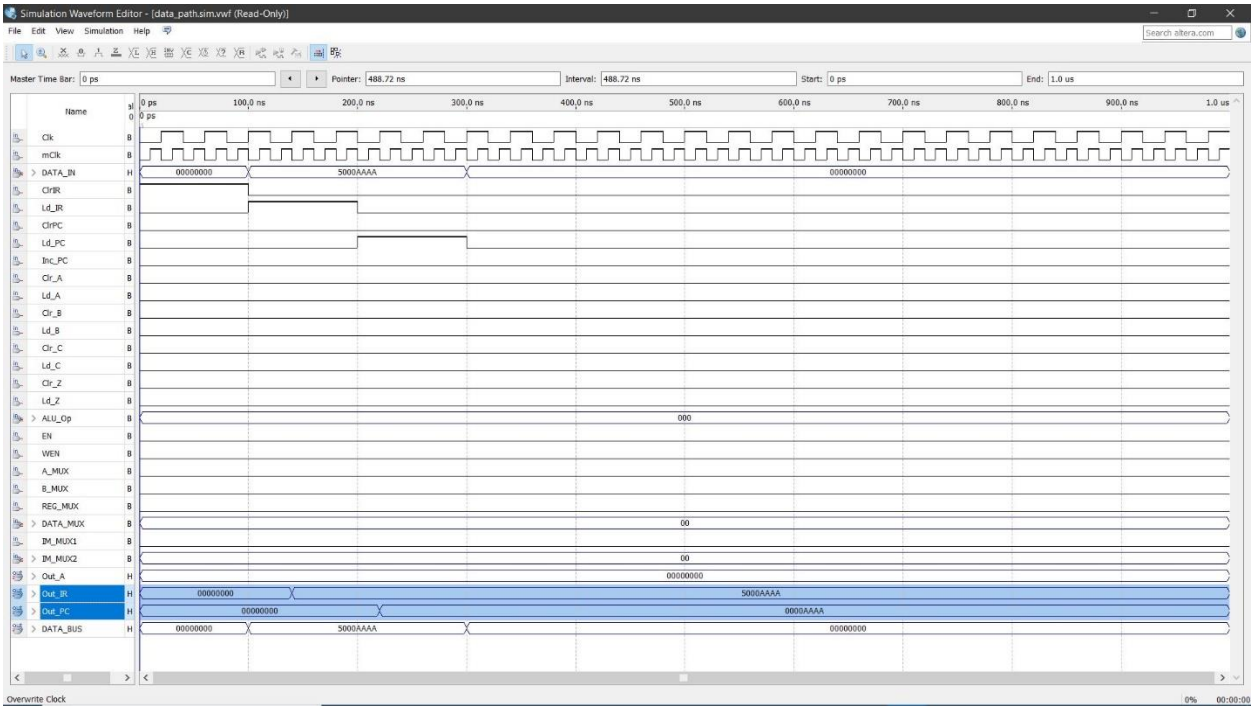
6. LDB


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 182.38 ns
Interval: 182.38 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_B
Out_IR
Out_PC
MEM_ADDR
MEM_IN
MEM_OUT
Selection Tool

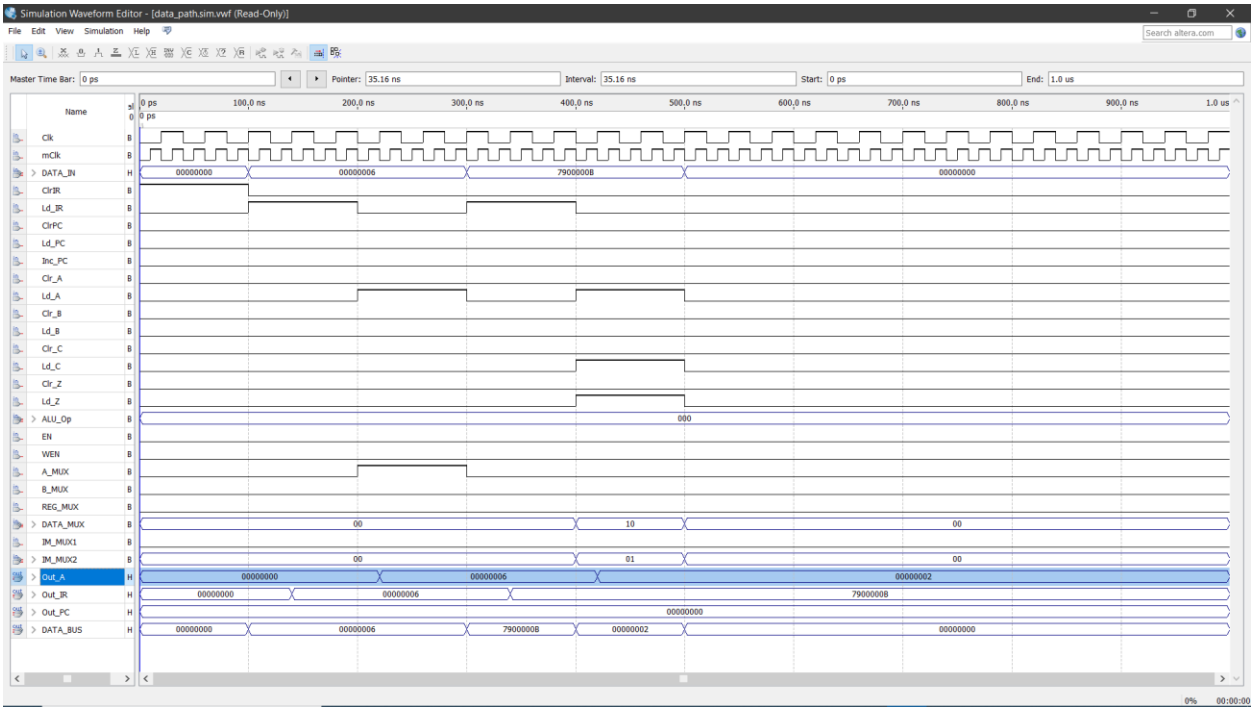
7. LUI


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 292.64 ns
Interval: 292.64 ns
Start: 0 ps
End: 1.0 us
Name
Value at 0 ps
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_M...
DATA_...
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_...
00000000
4000AAAA
AAAA0000
Zoom Tool

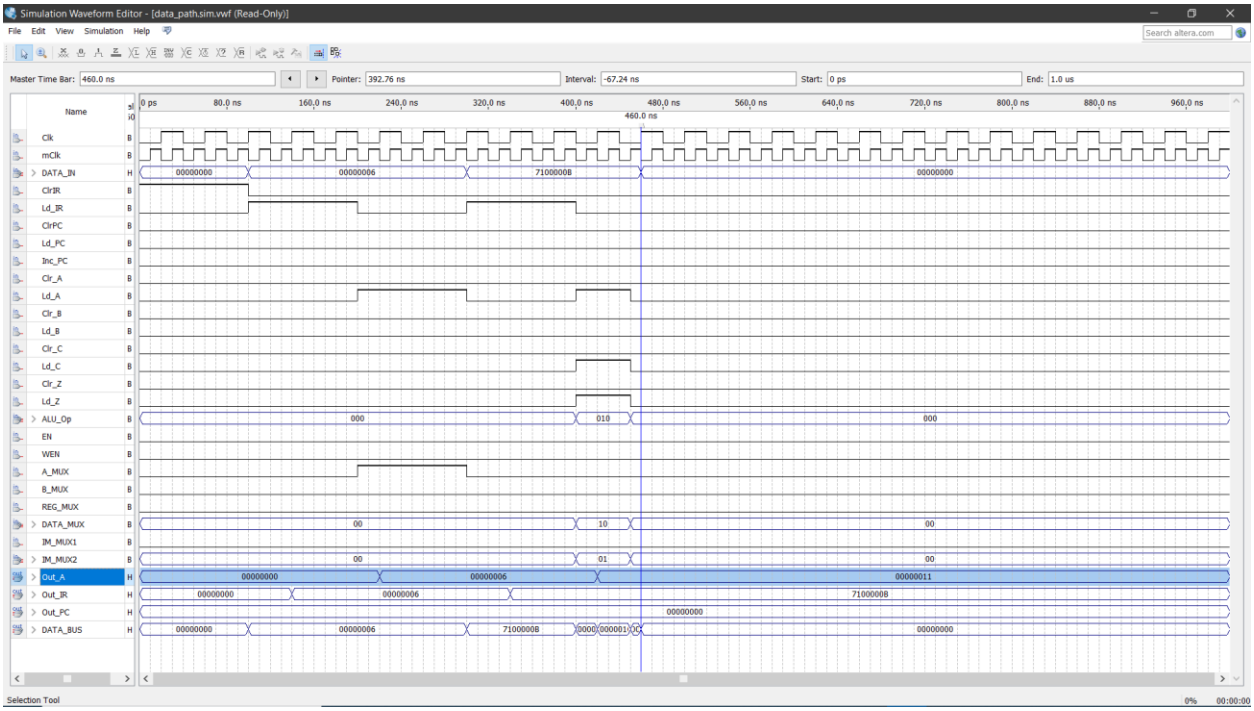
8. JMP


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 488.72 ns
Interval: 488.72 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
5000AAAA
000
00
0000AAAA
Overwrite Clock

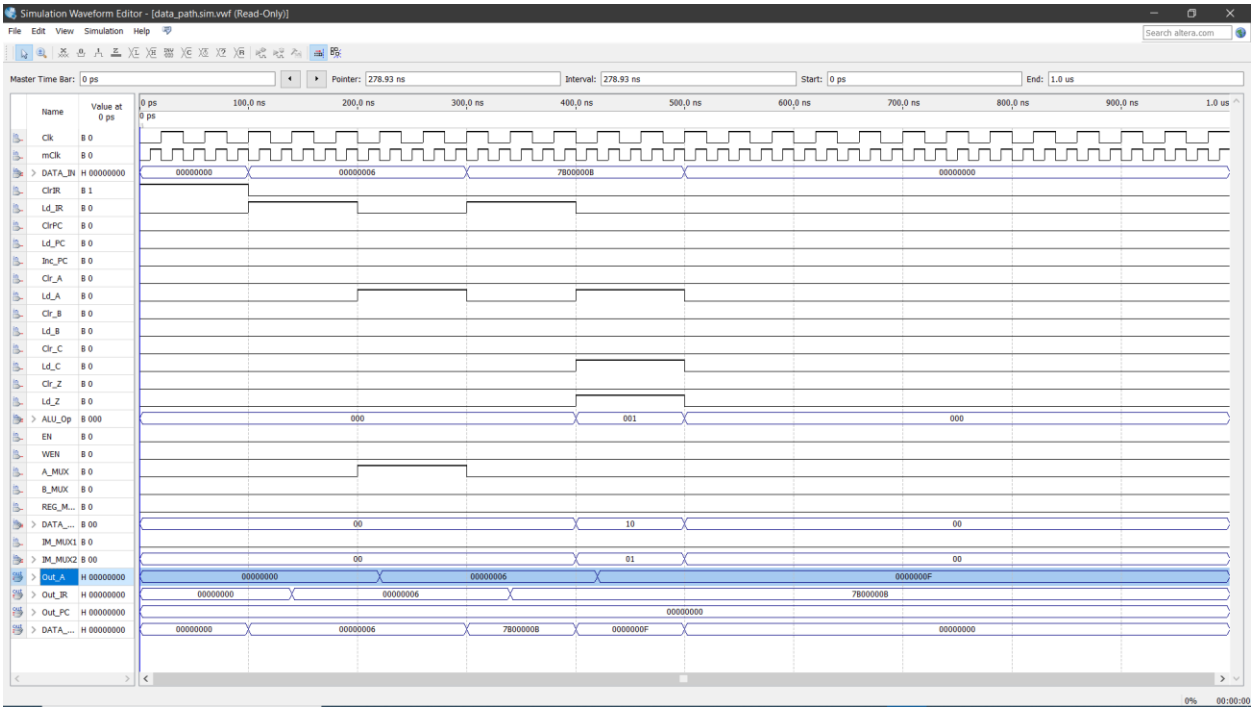
9. ANDI


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 35.16 ns
Interval: 35.16 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
00000006
7900000B
00000002
000
00
10
01

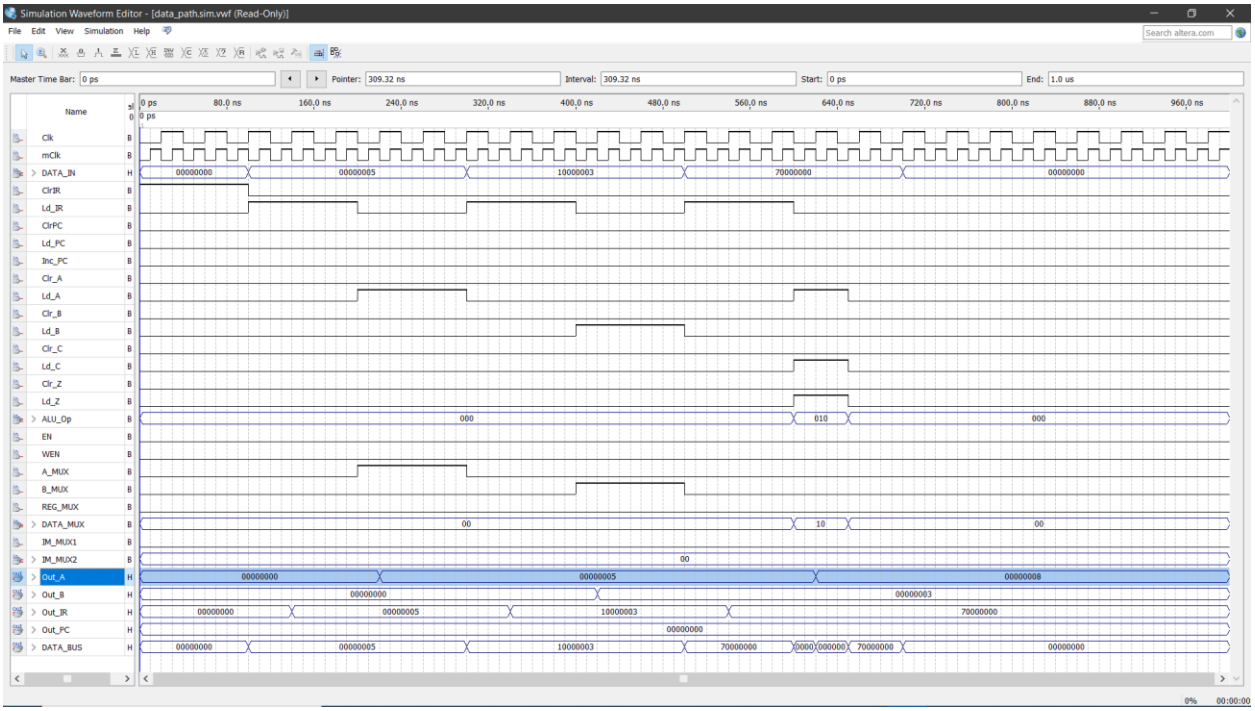
10. ADDI


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 460.0 ns
Pointer: 392.76 ns
Interval: -67.24 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
Selection Tool

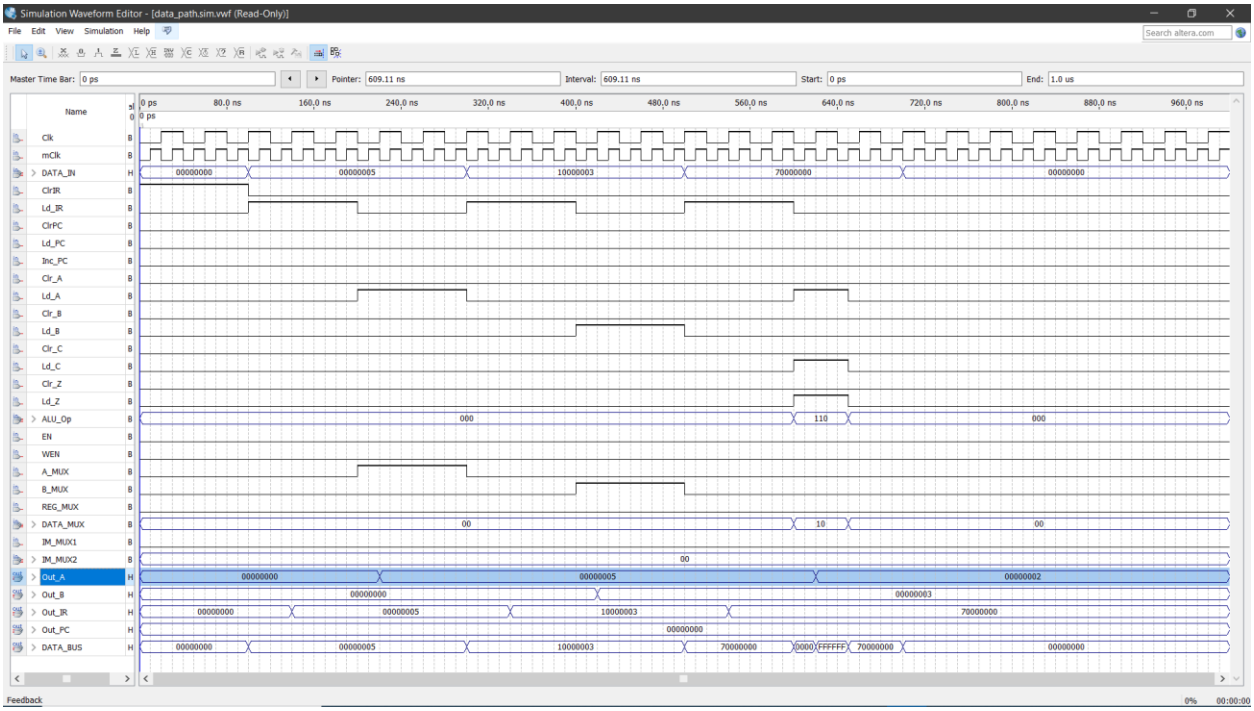
11. ORI


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 278.93 ns
Interval: 278.93 ns
Start: 0 ps
End: 1.0 us
Name
Value at 0 ps
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_M...
DATA_...
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_...
00000000
00000006
7B00000B
00000000
000
001
000
00
10
00
00
01
00
00000000
00000006
0000000F
00000000
00000006
7B00000B
00000000
00000000
00000006
7B00000B
0000000F
00000000

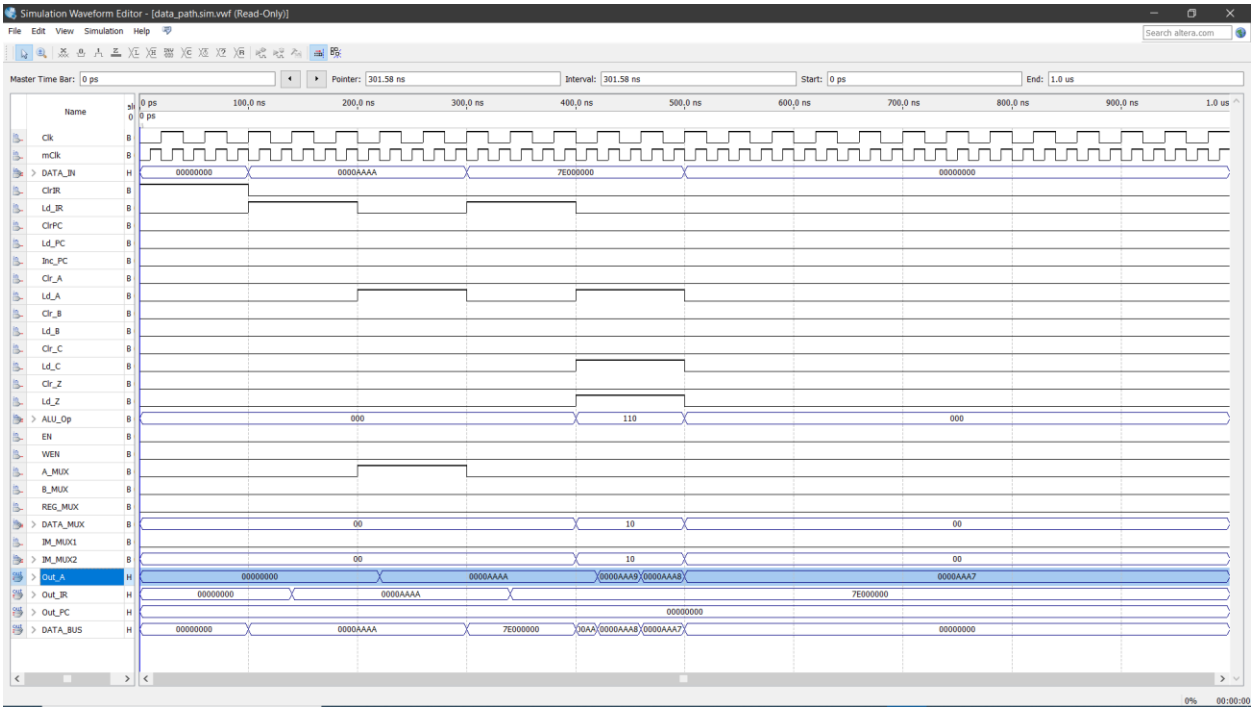
12. ADD


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 309.32 ns
Interval: 309.32 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_B
Out_IR
Out_PC
DATA_BUS

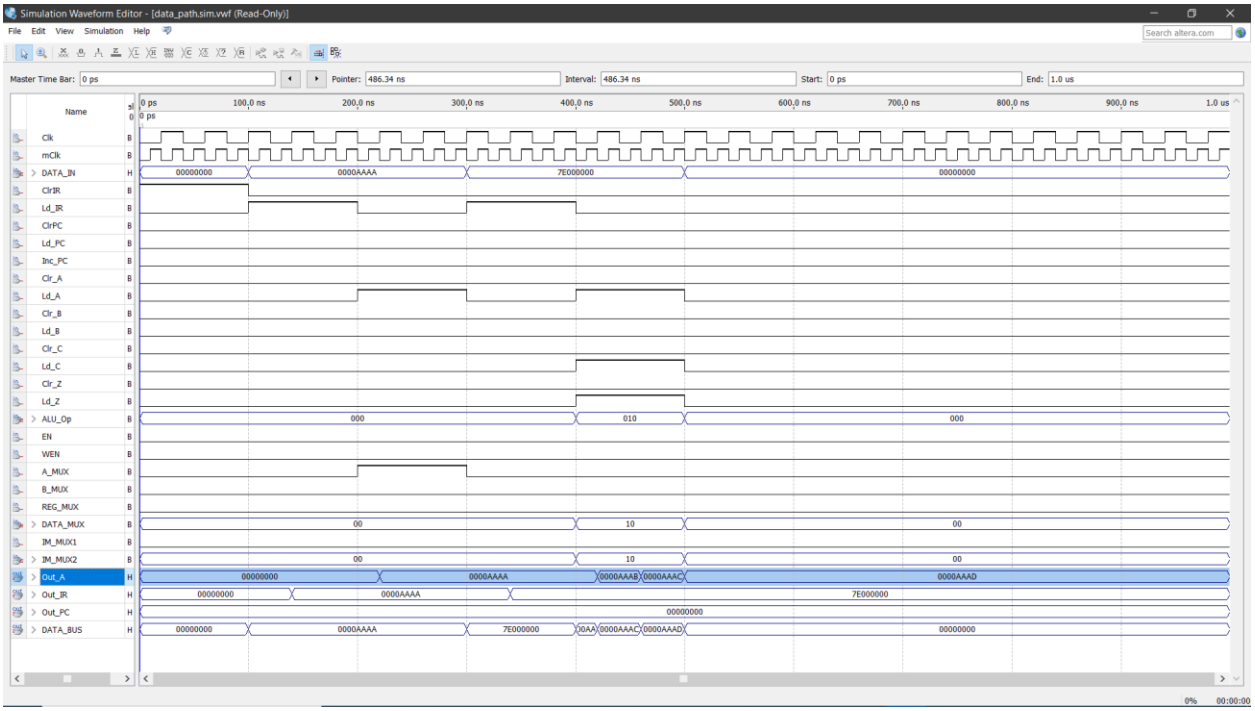
13. SUB


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 609.11 ns
Interval: 609.11 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_B
Out_IR
Out_PC
DATA_BUS
Feedback

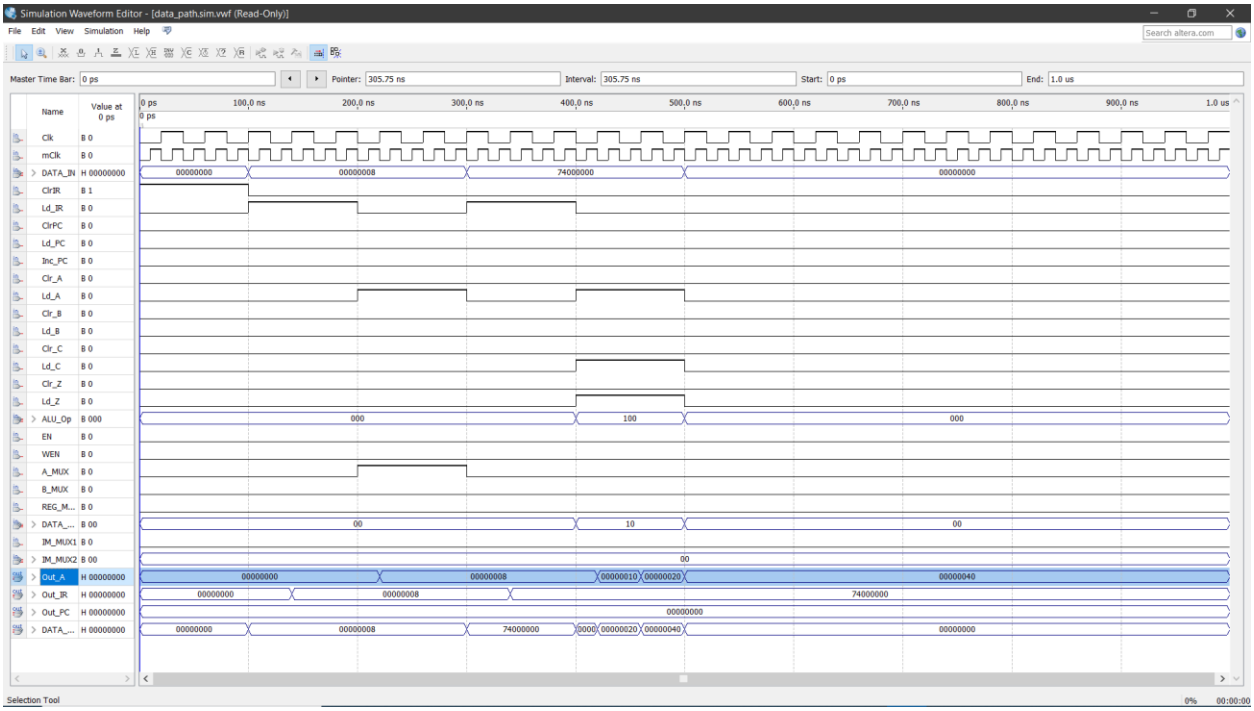
14. DECA


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps Pointer: 301.58 ns Interval: 301.58 ns Start: 0 ps End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
0000AAAA
7E000000
00000000
000
110
000
00
10
00
00
10
00
00000000
0000AAAA
0000AAA9
0000AAAB
0000AAA7
00000000
0000AAAA
7E000000
00000000
00000000
0000AAAA
7E000000
00AA
0000AAAB
0000AAA7
00000000

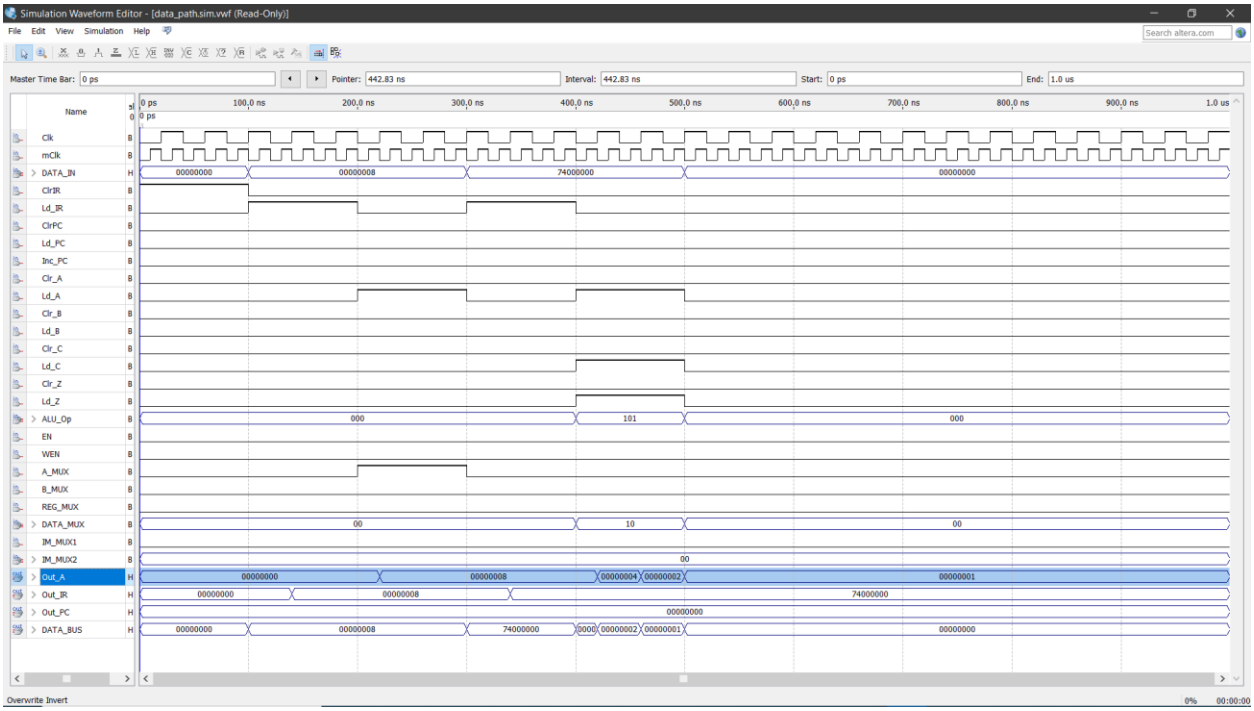
15. INCA


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 486.34 ns
Interval: 486.34 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
0000AAAA
7E000000
00000000
000
010
000
00
10
00
00
10
00
00000000
0000AAAA
0000AAAB
0000AAAC
0000AAAD
00000000
0000AAAA
7E000000
00000000
00000000
0000AAAA
7E000000
00AA
0000AAAC
0000AAAD
00000000

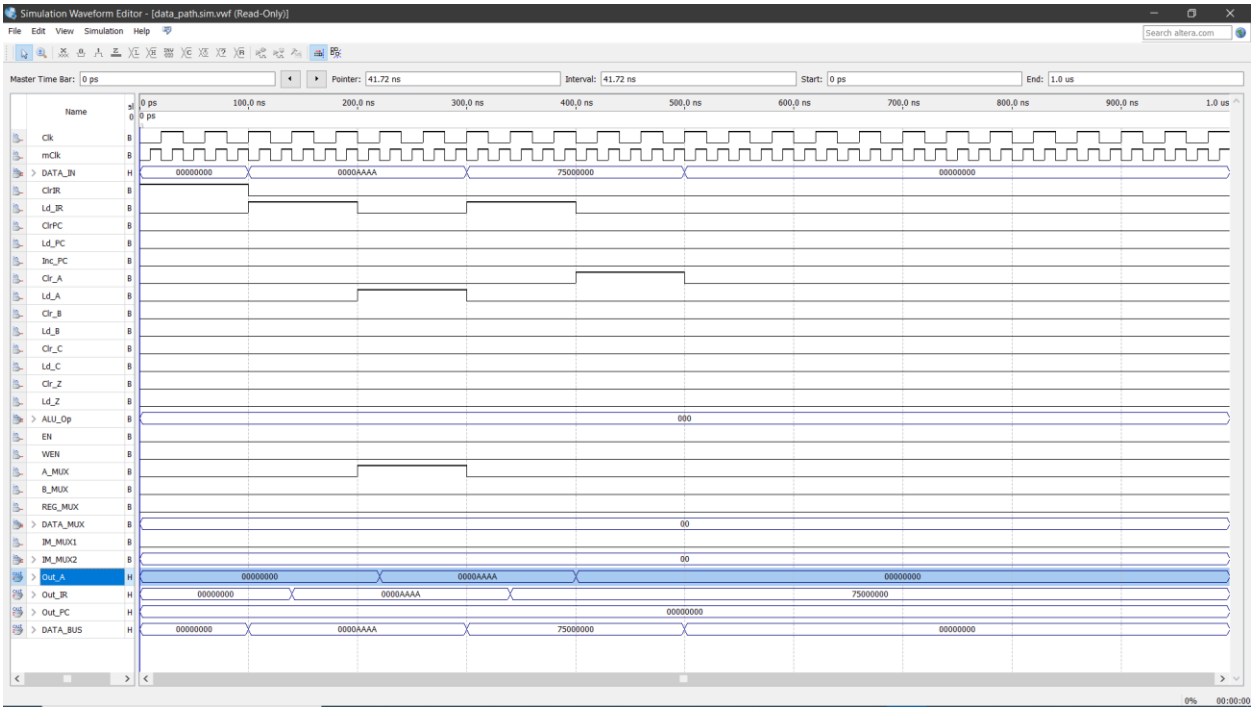
16. ROL


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 305.75 ns
Interval: 305.75 ns
Start: 0 ps
End: 1.0 us
Name
Value at 0 ps
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_M...
DATA_...
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_...
00000000
00000008
74000000
00000000
000
100
000
00
10
00
00
00000000
00000008
00000010
00000020
00000040
00000000
00000008
74000000
00000000
00000000
00000008
74000000
0000
00000020
00000040
00000000
Selection Tool
0%
00:00:00

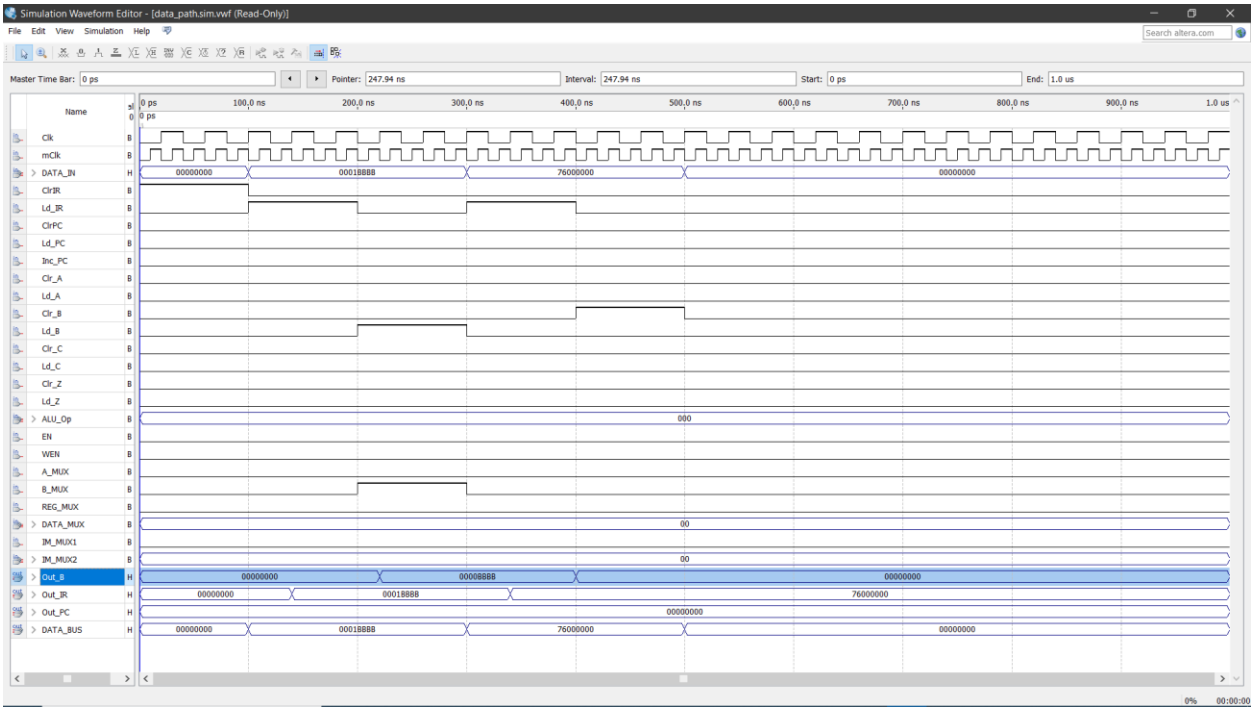
17. ROR


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 442.83 ns
Interval: 442.83 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
00000008
74000000
00000000
000
101
000
00
10
00
00
00000000
00000008
00000004
00000002
00000001
00000000
00000008
74000000
00000000
00000000
00000008
74000000
0000
00000002
00000001
00000000
Overwrite Invert
0%
00:00:00

18. CLRA


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 41.72 ns
Interval: 41.72 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_A
Out_IR
Out_PC
DATA_BUS
00000000
0000AAAA
75000000
00000000
000
00
00
00000000
0000AAAA
00000000
00000000
75000000
00000000
00000000
0000AAAA
75000000
00000000

19. CLRB


Simulation Waveform Editor - [data_path.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 247.94 ns
Interval: 247.94 ns
Start: 0 ps
End: 1.0 us
Name
Clk
mClk
DATA_IN
ClrIR
Ld_IR
ClrPC
Ld_PC
Inc_PC
Clr_A
Ld_A
Clr_B
Ld_B
Clr_C
Ld_C
Clr_Z
Ld_Z
ALU_Op
EN
WEN
A_MUX
B_MUX
REG_MUX
DATA_MUX
IM_MUX1
IM_MUX2
Out_B
Out_IR
Out_PC
DATA_BUS
00000000
0001BBBB
76000000
00000000
000
00
00
00000000
0000BBBB
00000000
00000000
00000000
00000000
0001BBBB
76000000
00000000