
VHDL 및 실습

micro processor

과목	VHDL 및 실습
학과	전자공학과
학번	2011144024
이름	유대성 (오전)
제출일	
담당교수	최종성 교수님

1. 주제 및 배경이론

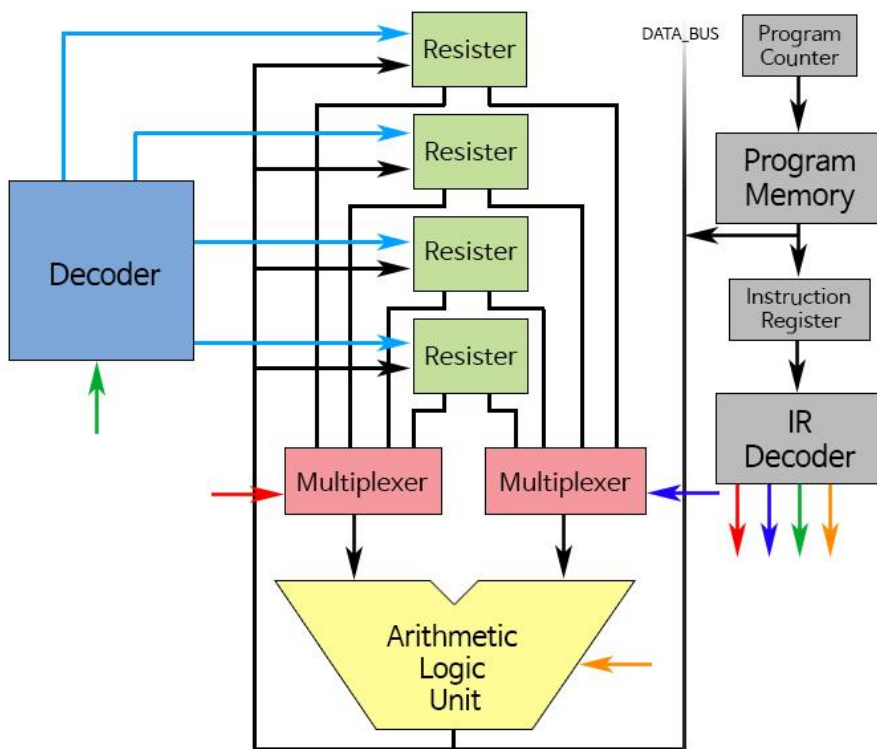
i VHDL의 계층구조를 이해하고 디지털 시계를 설계해보자.

Micro Processor 란?

마이크로프로세서(초소형 연산 처리 장치)는 컴퓨터의 중앙 처리장치 즉, CPU를 말한다. 이 초소형 처리 장치는 기계어 코드를 단계별로 실행하는 데 이 처리단위를 마이크로 단위로 처리하는 논리회로라고 할 수 있다. 마이크로프로세서를 설계한다는 말은 하나의 동작을 적어도 하나의 기계어 코드에 대응시켜 동작하도록 구성한다는 것이다. 이러한 동작을 인간이 알아들을 수 있는 언어로 매칭하여 '컴파일러'를 만들어 제공할 수 있다. 우리가 아는 컴파일러는 인간이 알아들을 수 있는 명령어가 어떠한 테이블에 의해 기계어로 변환되어 cpu로 전달할 수 있게 해주는 프로그램이다.

Micro Processor의 구성

그렇다면 마이크로프로세서는 어떠한 형태로 되어있을까? 복잡한 IC 회로가 아닌 기능상의 단위로 설명하도록 한다.



Program Counter : 명령어 인출을 위한 프로그램 카운터이다. 이 카운터를 통해 메모리 상의 주소에 접근하게 된다.

Program Memory : 명령어를 저장하고 있는 메모리이다.

Instruction Register : 출력된 명령어들을 해당 큐에 넣어 하나씩 출력한다.

Instruction Decoder : 명령어를 받아 해석하고 필요한 제어신호를 생성한다.

Register : 소스 데이터나 연산 결과를 저장하는 데 사용하는 레지스터이다.

Arithmetic and Logic Unit : 연산기

Decoder : Register 을 선택하는 신호를 발생

Multiplexer : Register 의 값을 선택하고 ALU 에 값을 전달한다.

2. Component 소개

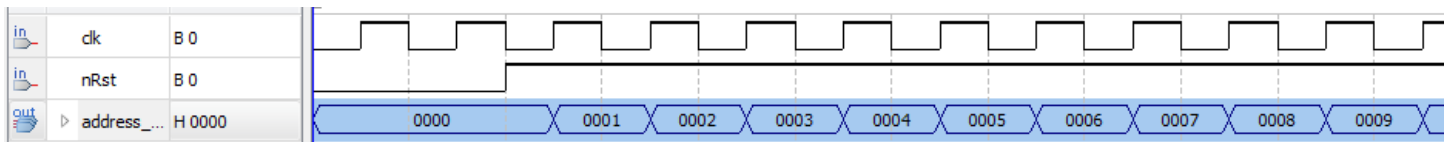
i Top Design 설계를 위한 컴포넌트를 소개한다.

Program Counter

Program ROM의 명령어를 읽어오기 위한 프로그램 카운터 설계

```
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 program_counter is
7   port(
8     nRst : in std_logic;
9     clk : in std_logic;
10    address_bus : out std_logic_vector(15 downto 0)
11  );
12 end program_counter;
13
14 architecture BEH of program_counter is
15   signal address_cnt : std_logic_vector(15 downto 0);
16 begin
17   process(nRst,clk)
18   begin
19     if(nRst='0') then
20       address_cnt <= (others => '0');
21     elsif rising_edge(clk) then
22       if(address_cnt = 50) then
23         address_cnt <= (others => '0');
24       else
25         address_cnt <= address_cnt + 1;
26       end if;
27     end if;
28   end process;
29   address_bus <= address_cnt;
30 end BEH;
```

Colored by Color Scripter CS

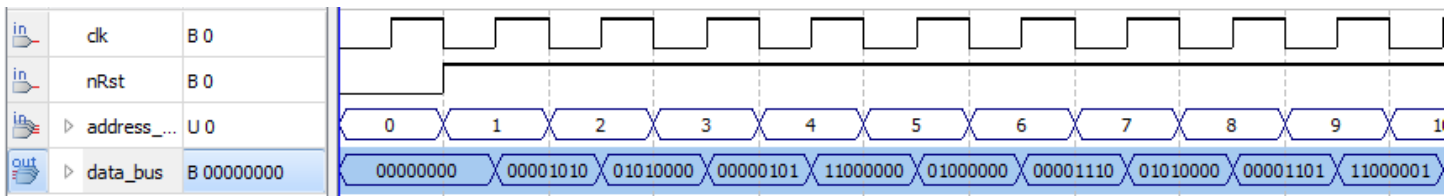


Program Memory

Processor 의 기능확인을 위한 명령어 내장

```
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 program_memory is
7   port(
8     nRst : in std_logic;
9     clk : in std_logic;
10    address_bus : in std_logic_vector(15 downto 0);
11    data_bus : out std_logic_vector(7 downto 0)
12  );
13 end program_memory;
14
15 architecture BEH of program_memory is
16   type mem_array is array(0 to 19) of std_logic_vector (7 downto 0);
17   signal mem : mem_array;
18 begin
19   process(nRst,clk, address_bus)
20     variable index : integer range 0 to 19 := 0;
21   begin
22     if(nRst='0') then
23       mem(0) <= "01000000";
24       mem(1) <= "00001010";
25       mem(2) <= "01010000";
26       mem(3) <= "00000101";
27       mem(4) <= "11000000";
28
29       mem(5) <= "01000000";
30       mem(6) <= "00001110";
31       mem(7) <= "01010000";
32       mem(8) <= "00001101";
33       mem(9) <= "11000001";
34
35       mem(10) <= "01000000";
36       mem(11) <= "00001111";
37       mem(12) <= "01010000";
38       mem(13) <= "00001010";
39       mem(14) <= "11000010";
40
41       mem(15) <= "01000000";
42       mem(16) <= "00001000";
43       mem(17) <= "01010000";
44       mem(18) <= "00000001";
45       mem(19) <= "11000011";
46
47     elsif rising_edge(clk) then
48       index := conv_integer(address_bus(4 downto 0));
49       data_bus <= mem(index);
50     end if;
51   end process;
52 end BEH;
```

Colored by Color ScripterCS

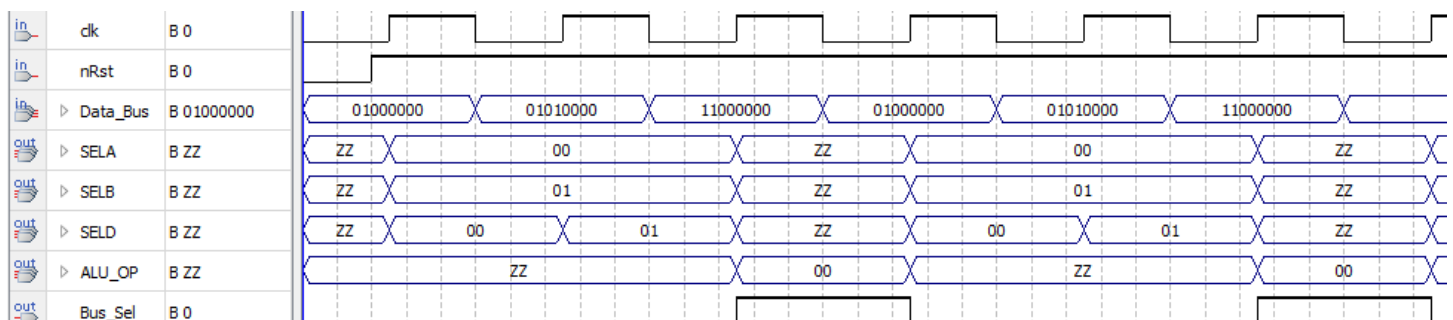


IR Decoder

입력받은 명령어를 해석하고 필요한 제어신호를 발생시킨다.

```
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 IR_Decoder is
7   port (
8     nRst : in std_logic;
9     clk : in std_logic;
10    Data_Bus : in std_logic_vector(7 downto 0);
11    SELA : out std_logic_vector(1 downto 0);
12    SELB : out std_logic_vector(1 downto 0);
13    SELD : out std_logic_vector(1 downto 0);
14    ALU_OP : out std_logic_vector(1 downto 0);
15    Bus_Sel : out std_logic
16  );
17 end IR_Decoder;
18
19 architecture BEH of IR_Decoder is
20 begin
21   process(nRst, clk)
22     variable mode, D, OP : std_logic_vector(1 downto 0);
23   begin
24     mode := Data_Bus(7 downto 6);
25     D := Data_Bus(5 downto 4);
26     OP := Data_Bus(1 downto 0);
27     if(nRst = '0') then
28       SELA <= (others => 'Z');
29       SELB <= (others => 'Z');
30       SELD <= (others => 'Z');
31       ALU_OP <= (others => 'Z');
32       Bus_Sel <= '0';
33     elsif rising_edge(clk) then
34       if(mode = "01") then
35         SELA <= "00";
36         SELB <= "01";
37         SELD <= D;
38         ALU_OP <= (others => 'Z');
39         Bus_Sel <= '0';
40       elsif(mode = "11") then
41         SELA <= "00";
42         SELB <= "01";
43         SELD <= "00";
44         ALU_OP <= OP;
45         Bus_Sel <= '1';
46       else
47         SELA <= (others => 'Z');
48         SELB <= (others => 'Z');
49         SELD <= (others => 'Z');
50         ALU_OP <= (others => 'Z');
51         Bus_Sel <= '0';
52       end if;
53     end if;
54   end process;
55 end BEH;
```

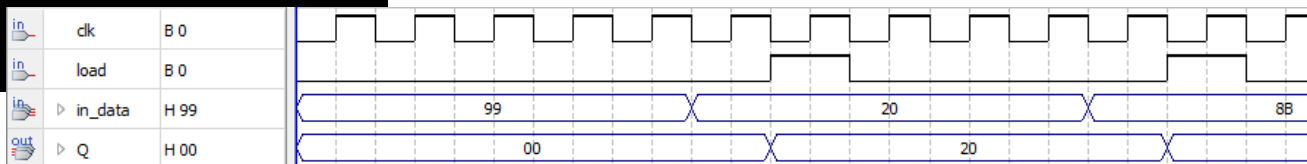
Colored by Color Script CS



Register

입력 Load = '1'인 구간에서 클럭의 하강에지에 in_data 를 출력 Q 로 전달하는 Latch 설계

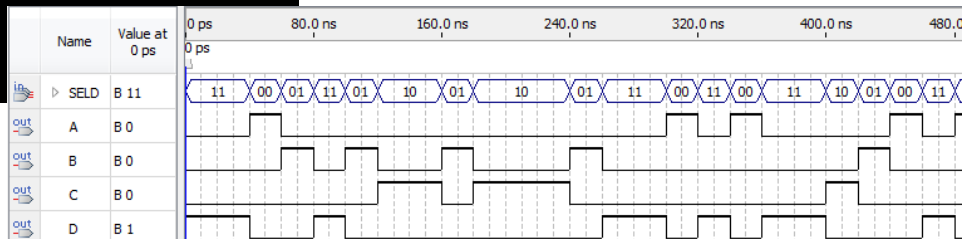
```
1 library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_unsigned.all;
5
6 entity reg_Latch is
7   port (
8     clk : in std_logic;
9     load : in std_logic;
10    in_data : in std_logic_vector(7 downto 0);
11    Q : out std_logic_vector(7 downto 0)
12  );
13 end reg_Latch;
14
15 architecture BEH of reg_Latch is
16 begin
17   process(clk)
18     variable inner_data : std_logic_vector(7 downto 0);
19     begin
20     if falling_edge(clk) then
21       if (load = '1') then
22         inner_data := in_data;
23       else
24         inner_data := inner_data;
25       end if;
26     end if;
27     Q <= inner_data;
28   end process;
29 end BEH;
```



Decoder 2x4

Register 데이터 저장을 제어하기 위한 Decoder 설계

```
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 decoder_2x4 is
7   port(
8     SELD : in std_logic_vector(1 downto 0);
9     A : out std_logic;
10    B : out std_logic;
11    C : out std_logic;
12    D : out std_logic
13  );
14 end decoder_2x4;
15
16 architecture BEH of decoder_2x4 is
17 begin
18   A <= '1' when SELD = 0 else '0'; -- 각 비트가 가질 수 있는 경우의 수는 8 가지 임
19   B <= '1' when SELD = 1 else '0'; -- 예외는 모두 0 으로 처리
20   C <= '1' when SELD = 2 else '0';
21   D <= '1' when SELD = 3 else '0';
22 end BEH;
```



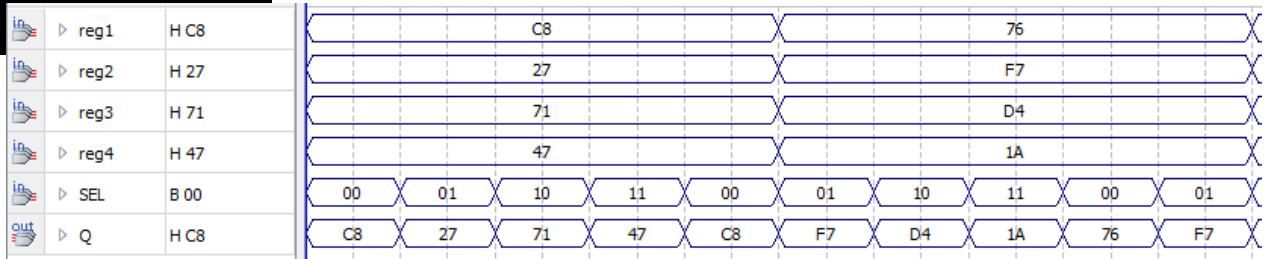
MUX 4x1

Register 1~4로부터 ALU로 데이터를 전달하기 위한 MUX 설계

```

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 MUX_4x1 is
7   port(
8     reg1 : in std_logic_vector (7 downto 0);
9     reg2 : in std_logic_vector (7 downto 0);
10    reg3 : in std_logic_vector (7 downto 0);
11    reg4 : in std_logic_vector (7 downto 0);
12    SEL : in std_logic_vector (1 downto 0);
13    Q : out std_logic_vector (7 downto 0)
14  );
15 end MUX_4x1;
16
17 architecture BEH of MUX_4x1 is
18
19 begin
20   Q <= reg1 when SEL = 0 else
21     reg2 when SEL = 1 else
22     reg3 when SEL = 2 else
23     reg4 when SEL = 3 else
24     (others => 'Z');
25 end BEH;

```



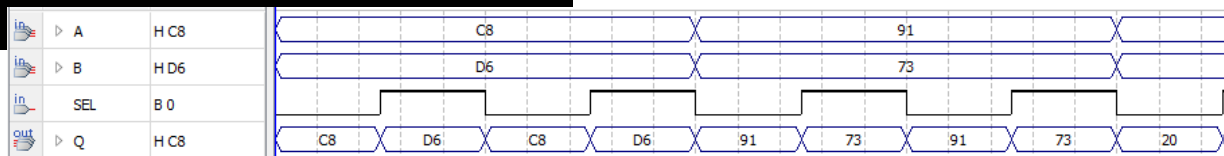
MUX 2x1

PROGRAM Memory와 ALU의 연산결과를 Data Bus에 올려놓기 위한 MUX 설계

```

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 MUX_2x1 is
7   port(
8     A : in std_logic_vector (7 downto 0);
9     B : in std_logic_vector (7 downto 0);
10    sel : in std_logic;
11    Q : out std_logic_vector (7 downto 0)
12  );
13 end MUX_2x1;
14
15 architecture BEH of MUX_2x1 is
16
17 begin
18   Q <= A when sel = '0' else
19     B when sel = '1' else
20     (others => 'Z'); -- 예외 출력 조건을 다음과 같이 Z로 설정, Z는 대문자 사용
21 end BEH;

```







Arithmetic and Logic Unit

실질적인 연산을 처리하는 부분으로 이번 과제에서는 덧셈, 뺄셈, & 연산, | 연산을 사용합니다.


```

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 ALU is
7     port(
8         op : in std_logic_vector (1 downto 0);
9         A : in std_logic_vector (7 downto 0);
10        B : in std_logic_vector (7 downto 0);
11        Q : out std_logic_vector (7 downto 0)
12    );
13 end ALU;
14
15 architecture BEH of ALU is
16     begin
17         Q <= (A+B) when op = 0 else
18             (A-B) when op = 1 else
19             (A and B) when op = 2 else
20             (A or B) when op = 3 else
21             (others => 'Z');
22
23 end BEH;

```

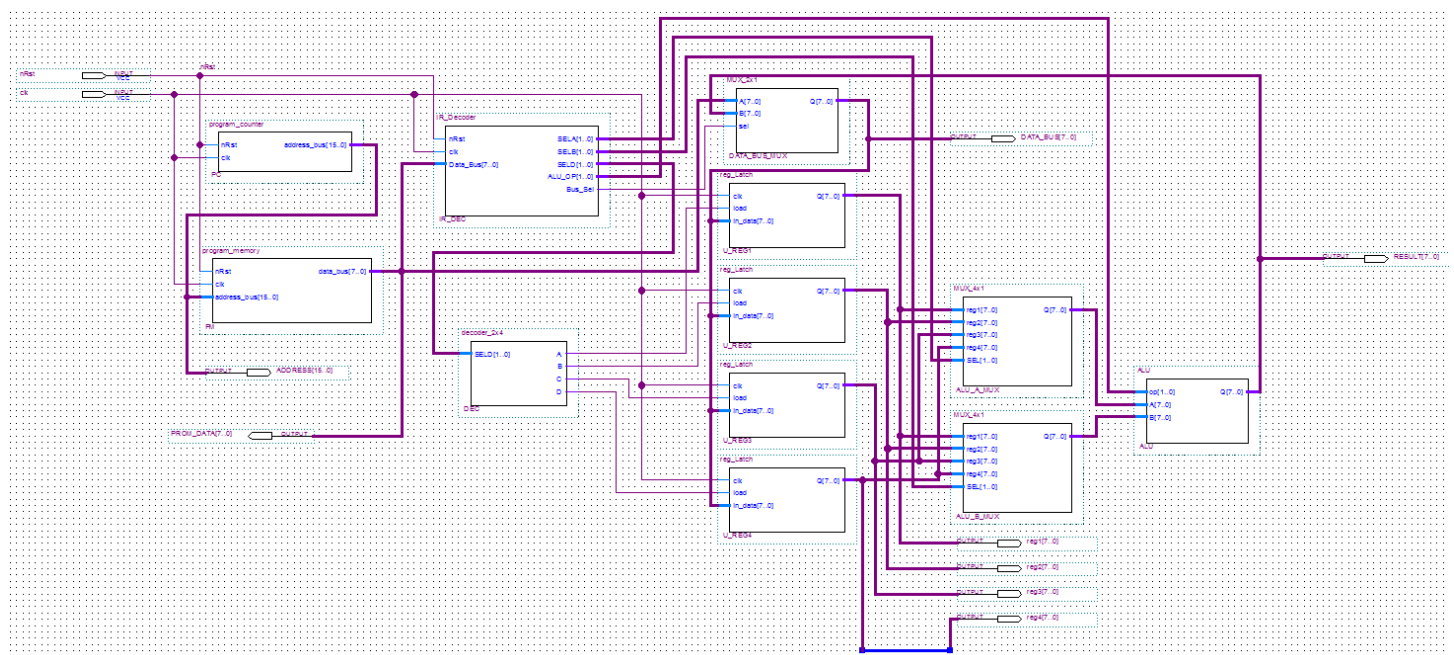
	▷ A	B 00000110				00000110		
	▷ B	B 00000011				00000011		
	▷ op	B 00						
	▷ Q	B 00001001						

3. 소스코드 및 코드 설명(3, 4 단계 함께 기술)

 소스코드를 설명과 함께 기술한다.

TOP Design 설계 (Schematic)

내부 Signal 의 선언이 어려우므로 Schematic 파일을 생성한 후 선 테스트 진행



TOP Design 설계(VHDL)

내부 Signal 을 선언하여 각 Component 를 연결

```
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 top_design is
7     port(
8         nRst : in std_logic;
9         clk : in std_logic;
10        OUT_ADDRESS : out std_logic_vector(15 downto 0);
11        OUT_DATABUS : out std_logic_vector(7 downto 0);
12        OUT_PROMDATA : out std_logic_vector(7 downto 0);
13        OUT_REG1 : out std_logic_vector(7 downto 0);
14        OUT_REG2 : out std_logic_vector(7 downto 0);
15        OUT_REG3 : out std_logic_vector(7 downto 0);
16        OUT_REG4 : out std_logic_vector(7 downto 0)
17    );
18 end top_design;
19
20 architecture BEH of top_design is
21
22     component program_counter
23     port(
24         nRst : in std_logic;
25         clk : in std_logic;
26         address_bus : out std_logic_vector(15 downto 0)
27     );
28 end component;
29
30     component program_memory
31     port(
32         nRst : in std_logic;
33         clk : in std_logic;
34         address_bus : in std_logic_vector(15 downto 0);
35         data_bus : out std_logic_vector(7 downto 0)
36     );
37 end component;
38
39     component IR_Decoder is
40     port (
41         nRst : in std_logic;
42         clk : in std_logic;
43         Data_Bus : in std_logic_vector(7 downto 0);
44         SELA : out std_logic_vector(1 downto 0);
45         SELB : out std_logic_vector(1 downto 0);
46         SELD : out std_logic_vector(1 downto 0);
47         ALU_OP : out std_logic_vector(1 downto 0);
48         Bus_Sel : out std_logic
49     );
50 end component;
51
52     component decoder_2x4
53     port(
54         SELD : in std_logic_vector(1 downto 0);
55         A : out std_logic;
56         B : out std_logic;
57         C : out std_logic;
58         D : out std_logic
59     );
60 end component;
61
62     component MUX_2x1
63     port(
64         A : in std_logic_vector(7 downto 0);
65         B : in std_logic_vector(7 downto 0);
66         sel : in std_logic;
67         Q : out std_logic_vector(7 downto 0)
68     );
69 end component;
70
71     component reg_Latch
72     port (
73         clk : in std_logic;
74
75         load : in std_logic;
76         in_data : in std_logic_vector(7 downto 0);
77         Q : out std_logic_vector(7 downto 0)
78     );
79 end component;
80
81     component MUX_4x1
82     port(
83         reg1 : in std_logic_vector(7 downto 0);
84         reg2 : in std_logic_vector(7 downto 0);
85         reg3 : in std_logic_vector(7 downto 0);
86         reg4 : in std_logic_vector(7 downto 0);
87         SEL : in std_logic_vector(1 downto 0);
88         Q : out std_logic_vector(7 downto 0)
89     );
90 end component;
91
92     component ALU
93     port(
94         op : in std_logic_vector(1 downto 0);
95         A : in std_logic_vector(7 downto 0);
96         B : in std_logic_vector(7 downto 0);
97         Q : out std_logic_vector(7 downto 0)
98     );
99 end component;
100
101     signal ADDRESS : std_logic_vector(15 downto 0);
102     signal DATA_BUS : std_logic_vector(7 downto 0);
103     signal PROM_DATA : std_logic_vector(7 downto 0);
104     signal SELA, SELB, SELD, ALU_OP : std_logic_vector(1 downto 0);
105     signal A, B, C, D : std_logic;
106     signal reg1, reg2, reg3, reg4 : std_logic_vector(7 downto 0);
107     signal QA, QB : std_logic_vector(7 downto 0);
108     signal in_data : std_logic_vector(7 downto 0);
109     signal RESULT : std_logic_vector(7 downto 0);
110     signal Bus_Sel : std_logic;
111
112 begin
113     PC : program_counter
114     port map(
115         nRst => nRst,
116         clk => clk,
117         address_bus => ADDRESS
118     );
119
120     PM : program_memory
121     port map(
122         nRst => nRst,
123         clk => clk,
124         address_bus => ADDRESS,
125         data_bus => PROM_DATA
126     );
127
128     IR_DEC : IR_Decoder
129     port map(
130         nRst => nRst,
131         clk => clk,
132         Data_Bus => PROM_DATA,
133         SELA => SELA,
134         SELB => SELB,
135         SELD => SELD,
136         ALU_OP => ALU_OP,
137         Bus_Sel => Bus_Sel
138     );
139
140     DEC : decoder_2x4
141     port map(
142         SELD => SELD,
143         A => A,
144         B => B,
145         C => C,
146         D => D
147     );
```

```

147
148 DATA_BUS_MUX : MUX_2x1
149 port map(
150     A => PROM_DATA,
151     B => RESULT,
152     sel => Bus_Sel,
153     Q => DATA_BUS
154 );
155
156
157 U_REG01 : reg_Latch
158 port map(
159     clk => clk,
160     load => B,
161     in_data => DATA_BUS,
162     Q => reg1
163 );
164
165 U_REG02 : reg_Latch
166 port map(
167     clk => clk,
168     load => C,
169     in_data => DATA_BUS,
170     Q => reg2
171 );
172
173 U_REG03 : reg_Latch
174 port map(
175     clk => clk,
176     load => A,
177     in_data => DATA_BUS,
178     Q => reg3
179 );
180
181 U_REG04 : reg_Latch
182 port map(
183     clk => clk,
184     load => D,
185     in_data => DATA_BUS,
186     Q => reg4
187
188
189 ALU_A_MUX : MUX_4x1
190 port map(
191     reg1 => reg1,
192     reg2 => reg2,
193     reg3 => reg3,
194     reg4 => reg4,
195     SEL => SELA,
196     Q => QA
197 );
198
199 ALU_B_MUX : MUX_4x1
200 port map(
201     reg1 => reg1,
202     reg2 => reg2,
203     reg3 => reg3,
204     reg4 => reg4,
205     SEL => SELB,
206     Q => QB
207 );
208
209 ALU_UNIT : ALU
210 port map(
211     op => ALU_OP,
212     A => QA,
213     B => QB,
214     Q => RESULT
215 );
216
217 OUT_ADDRESS <= ADDRESS;
218 OUT_DATABUS <= DATA_BUS;
219 OUT_PROMDATA <= PROM_DATA;
220 OUT_REG1 <= reg1;
221 OUT_REG2 <= reg2;
222 OUT_REG3 <= reg3;
223 OUT_REG4 <= reg4;
224
225 end BEH;

```

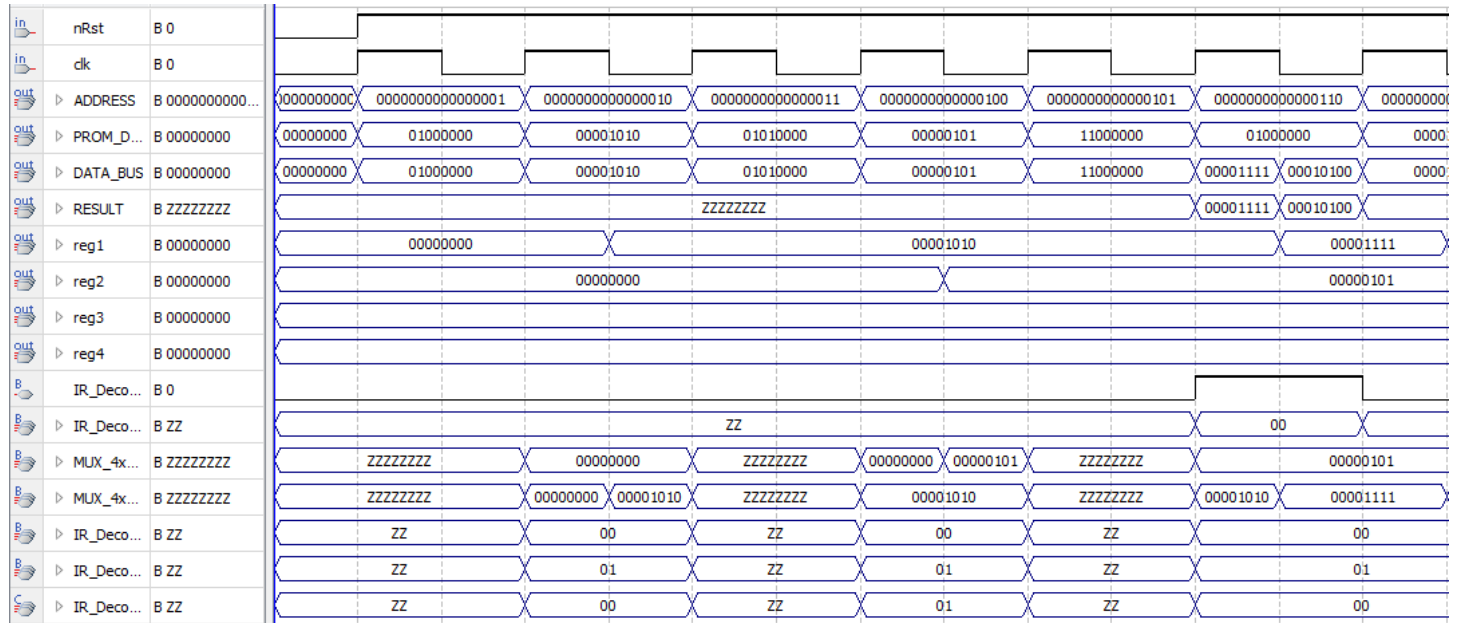
Colored by Color Scripter

4. 시뮬레이션 결과 및 설명(3, 4 단계 함께 기술)

i 코드 시뮬레이션과 그 결과를 기술한다.

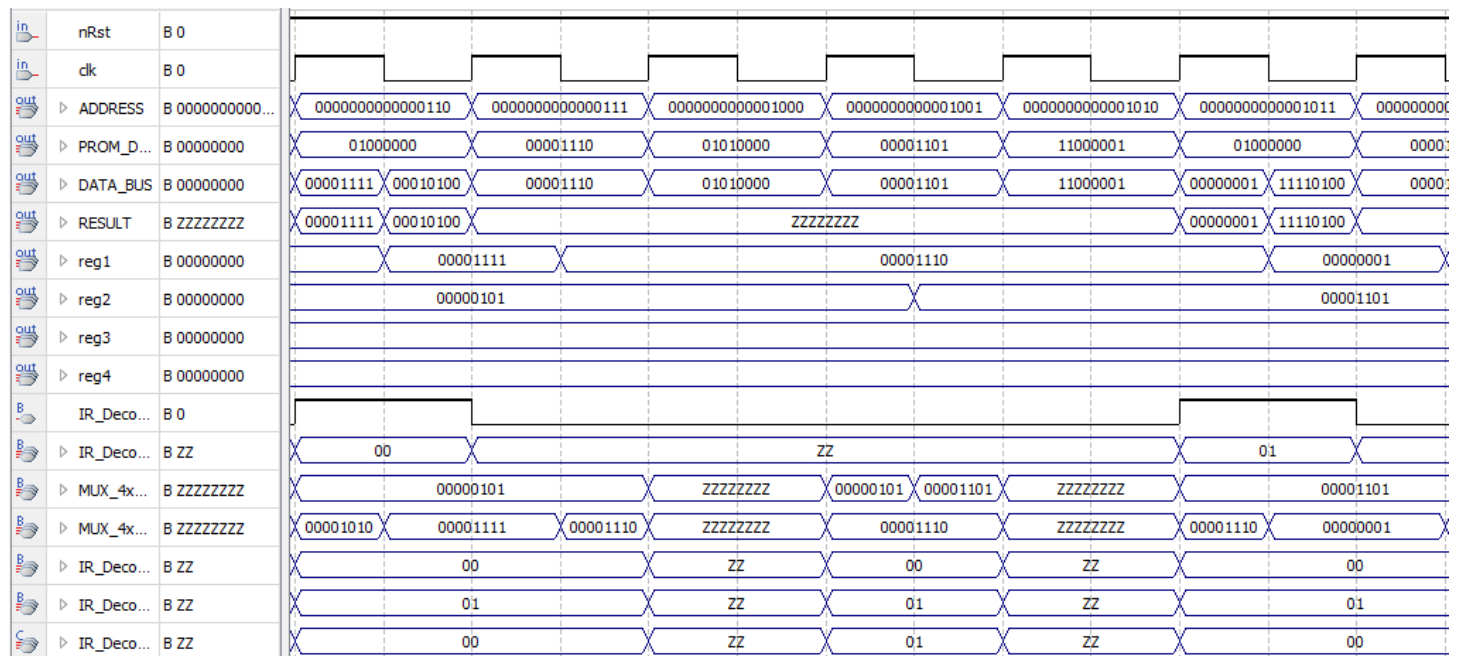
덧셈 연산

$$00001010(10) + 00000101(5) = 00001111(15)$$



뺄셈 연산

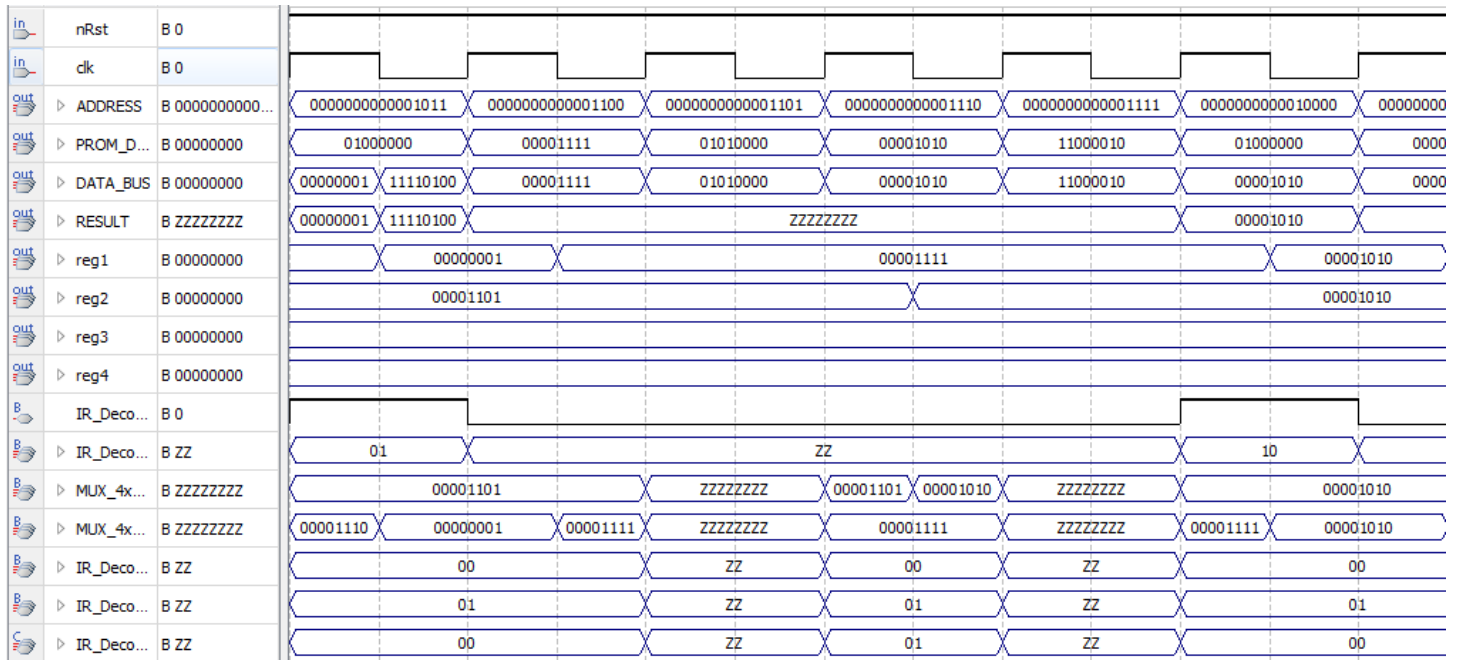
$$00001110(14) - 00001101(13) = 00000001(1)$$



& (AND) 연산

00001111 & 00001010 = 00001010

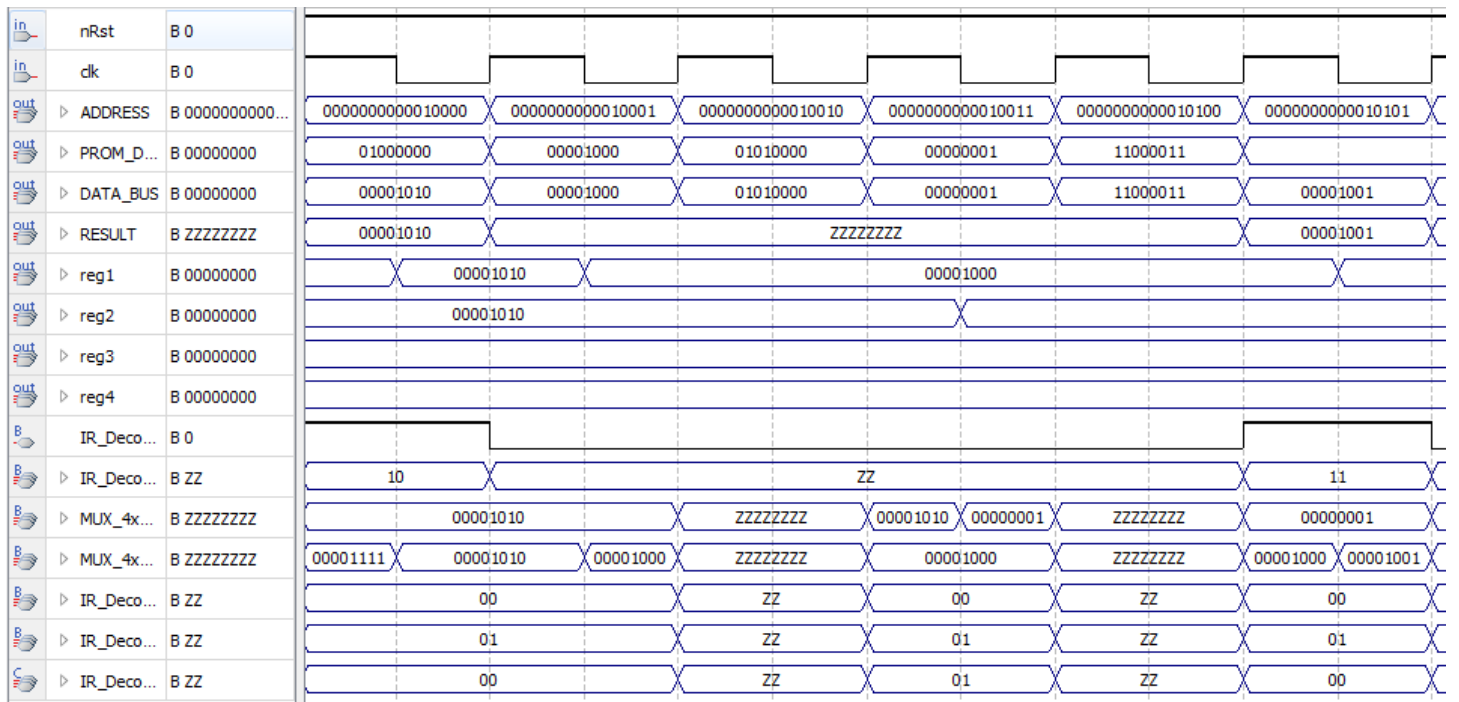
둘 다 1 일 때 1 이 되는 것을 확인할 수 있다.



| (OR) 연산

00001000 | 00000001 = 00001001

캡처가 잘려 잘 보이지 않지만, 둘 중 하나가 1 일 때 1 이 되는 것을 확인할 수 있다.



5. 실습보드 적용 결과

i 소스를 보드에 다운로드하여 예상한 결과에 맞게 동작하는 지 테스트하십시오

보드 테스트는 미처 진행하지 못했습니다. 이번 실습을 바탕으로 추후에 보드테스트 또한 진행할 수 있도록 하겠습니다.

6. 실습소감

i 실습을 통해 경험한 것을 자유롭게 서술하십시오.

이번 실습은 특히나 어려운 실습이었다. VHDL 로 Top Design 을 설계했을 때 Function Simulation 이 계속 U 로 표시되는 이상증상과 원치 않은 출력이 나오는 것을 반복하고 또 이를 수정하는 작업이 계속되었다. 시험기간과 겹치면서 포기할까 생각도 했지만 조금만 더하면 될 거 같은데라는 생각이 이 프로젝트에서 손을 뗄 수 없게 했다. 다행히도 수정과 새로 작성을 반복하던 끝에 원하는 결과가 출력되었다. 이번 실습의 성공을 통해 VHDL 의 모든 실습을 성공적으로 마칠 수 있게 되었다. 아직 모르는 것도 산더미고 작은 프로젝트 하나를 완성시키는 데에 온 힘을 쏟아 넣어야 했지만 포기하지 않고 하다보면 원하는 결과를 얻을 수 있다는 긍정적인 힘이 생긴 것 같아 뿌듯하다.

7. 감사합니다. 최종성 교수님

i 그 동안 고생하셨습니다. 또 감사 합니다.

최종성 교수님 안녕하십니까. 여기까지 읽어주셨다면 아마 제 과제가 미흡하다는 것이겠죠. 과제는 별로일 지 모르겠지만 이번 한 학기 동안 새로운 분야에 도전하여 주어진 과제를 잘 해냈다는 만족감에 매우 뿌듯합니다. 하루에 2~3 시간 주무시는 교수님을 보고 시간이 없다는 것은 핑계라고 생각했고 힘든 몸을 이끌고 강의하시는 모습에서 과연 나도 저럴 수 있을까라는 생각을 해봤습니다. 그래서 게으른 제 자신을 한 번 돌아보게 된 것 같습니다. 이번 학기는 엄청나게 많은 과제들과 함께했지만 교수님의 모습에서 열정을 본받아 잘 헤쳐나간 것 같습니다. 앞으로 남은 3 학기 동안 교수님을 다시 뵈게 될 지 모르나 이번 강의의 모습으로 최종성이라는 멋진 교수님이 있었다는 기억을 계속 간직하게 될 것 같습니다.

VHDL 강의 내용자체로 또, 그 밖의 다른 요소들로 많이 배워가는 한 학기였습니다. 다시 한번 감사 드립니다!!!