جامعة نيويورك أبوظبي

NYU ABU DHABI

Advanced Digital Logic

ENGR – UH 2310

Lab 2

Group 1

Spring, 2025

This report is entirely our own work, and we have kept a copy for our own records. We are aware of the University's policies on cheating, plagiarism, and the resulting consequences of their breach.

Submitted by:

Name	**Net ID**
Damiane Kapanadze	dk4770
Seoyoon Jung	sj4260
Sipan Hovsepian	sh7437

Contents

1 Snapshot of 8 ALU operations

Click to watch video demonstration

1.1 AND bitwise

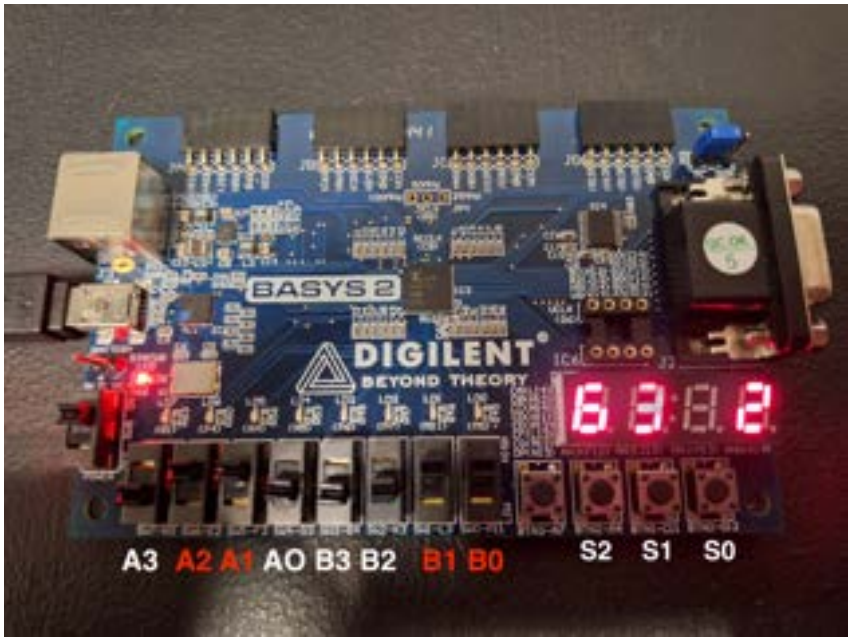

Figure 1: 0110 AND 0011 = 0010

1.2 OR bitwise

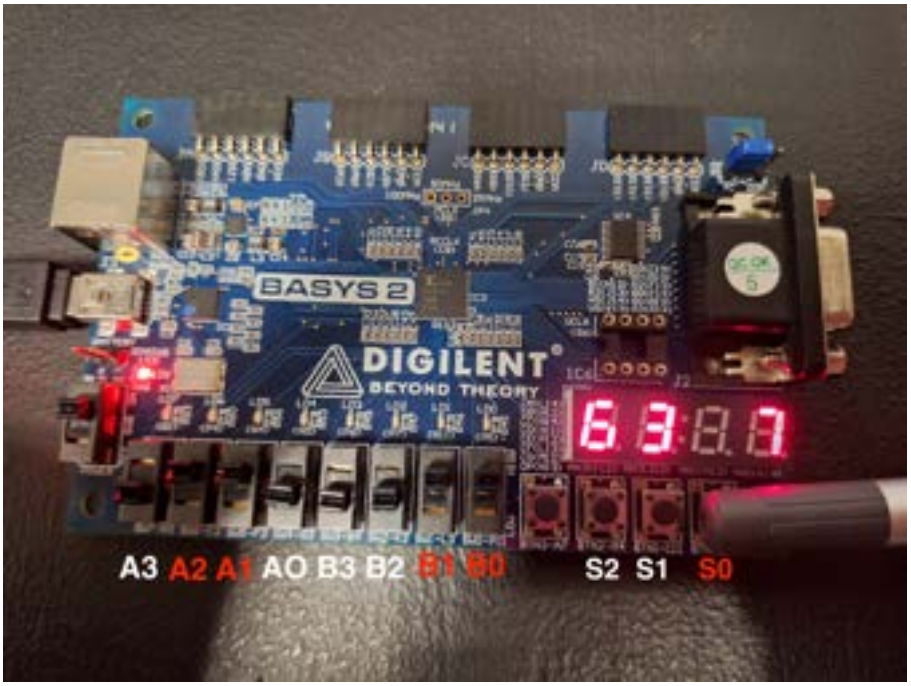

Figure 2: 0110 OR 0011 = 0111

1.3 ADD on signed numbers

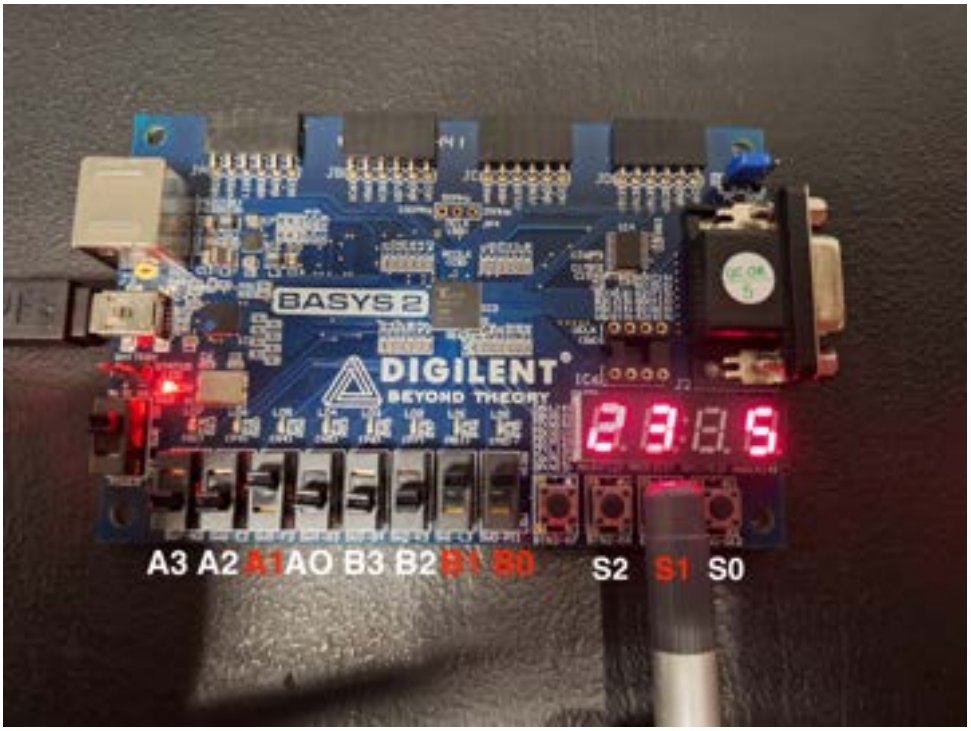

Figure 3: 0010 (2) + 0011 (3) = 5

1.4 SUB on signed numbers

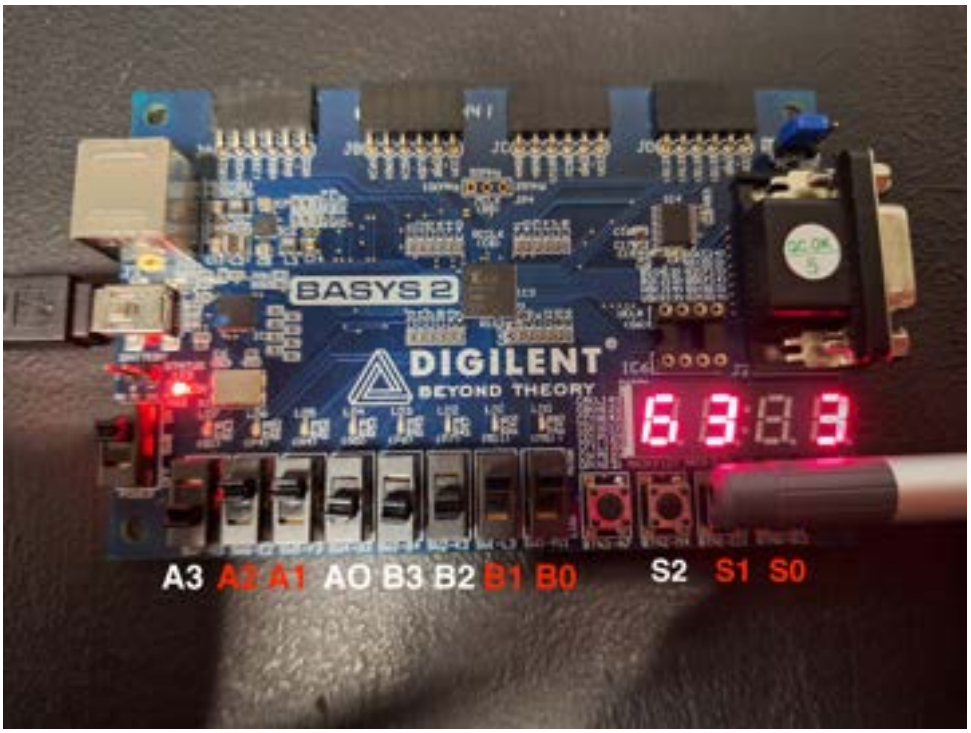

Figure 4: 0110 (6)- 0011 (3) = 3

1.5 ROR Rotate right

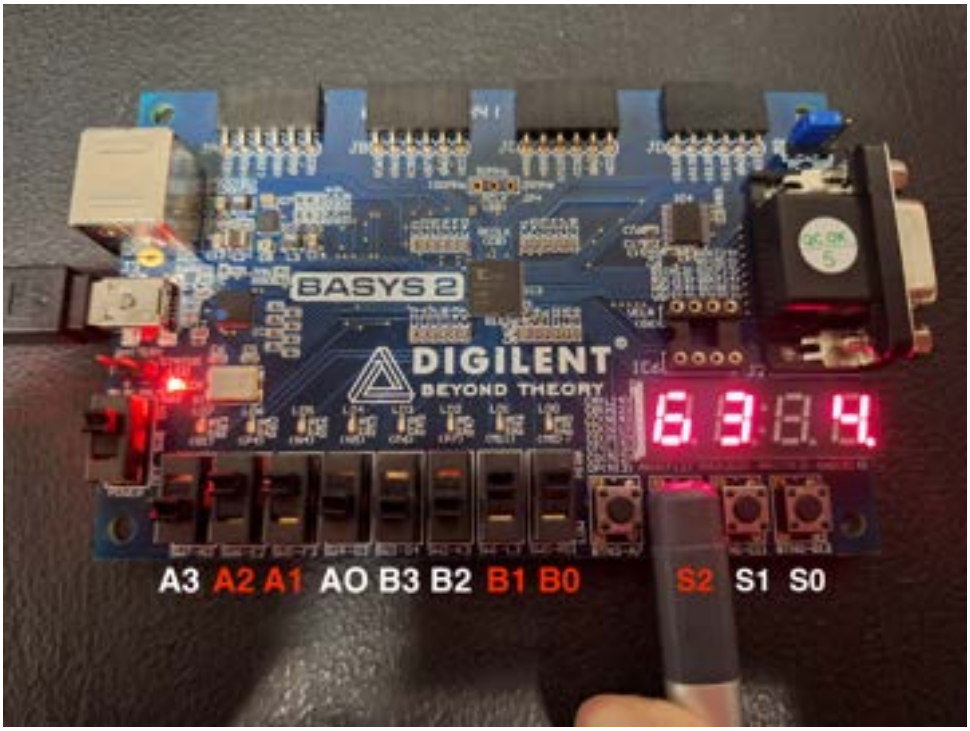

Figure 5: 6 (0110) ROR by 3 = 1100 (-4)

1.6 SLL Shift Left Logical

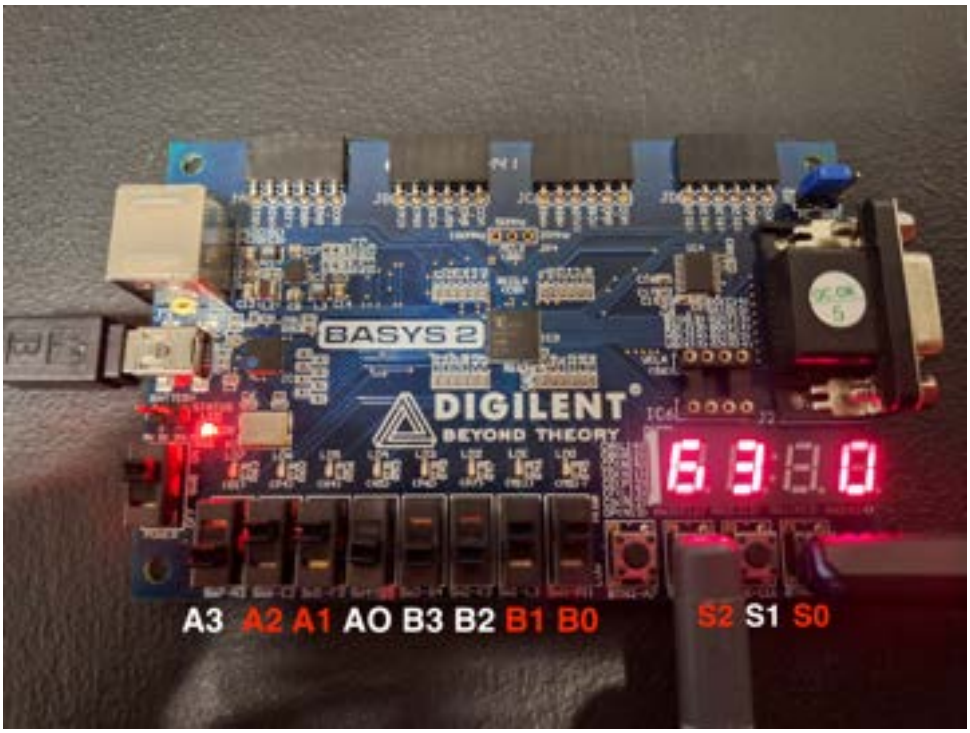

Figure 6: 6 (0110) SLL by 3 = 0000 (0)

1.7 SRL Shift Right Logical

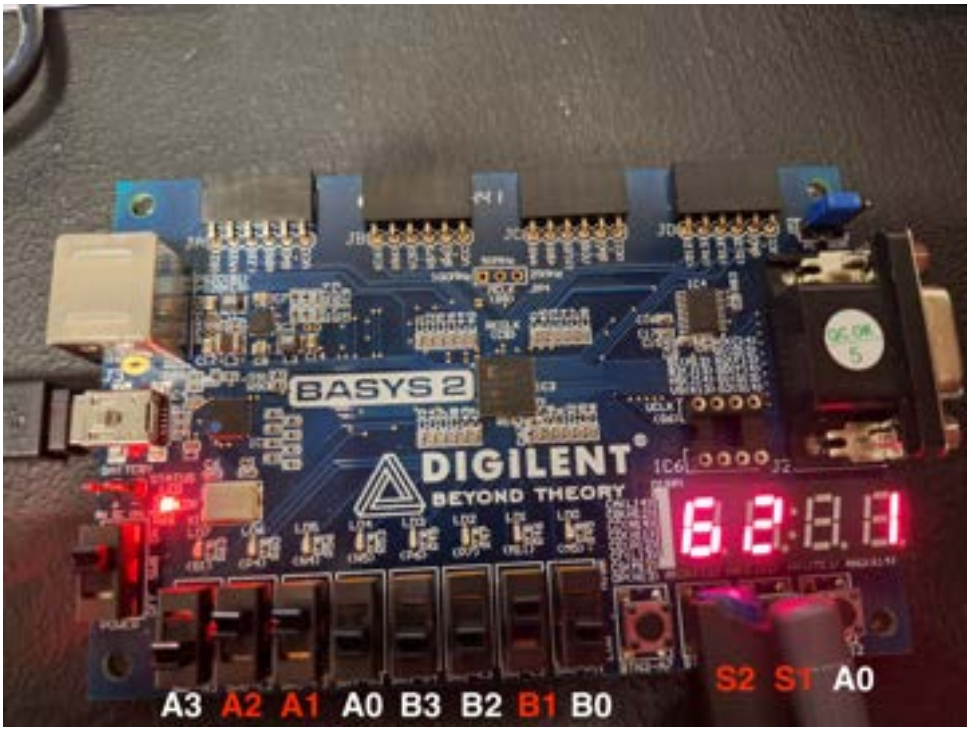

Figure 7: 6 (0110) SRL by 3 = 0000 (0)

1.8 SRA Shift Right Arithmetic

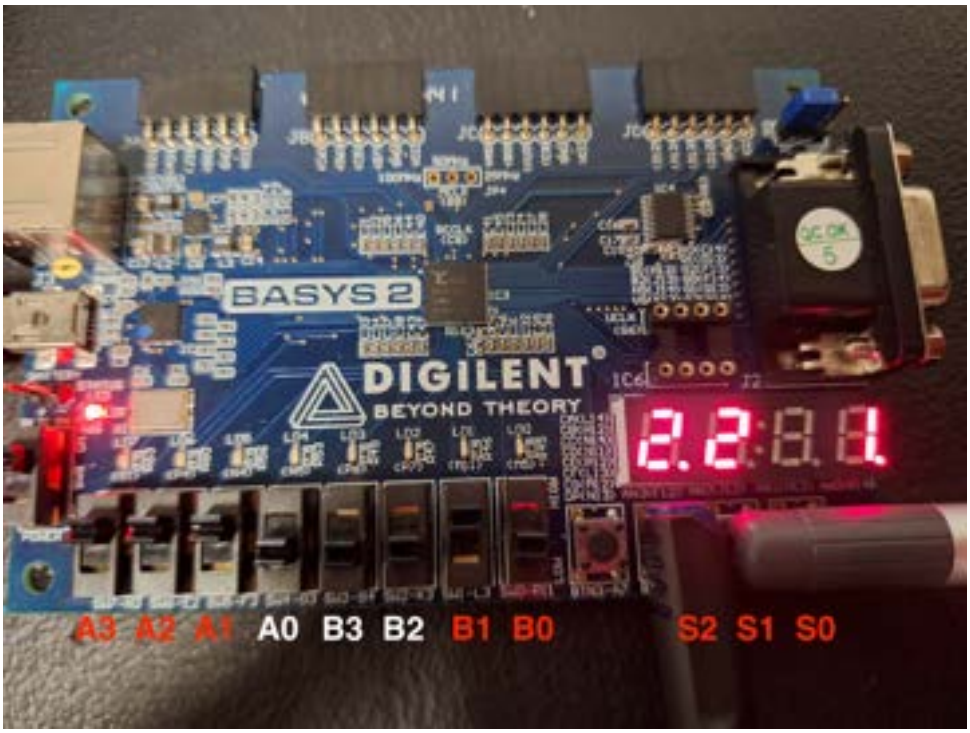

Figure 8: -1 (1110) SRA 3

1.9 ADD on signed numbers with overflow

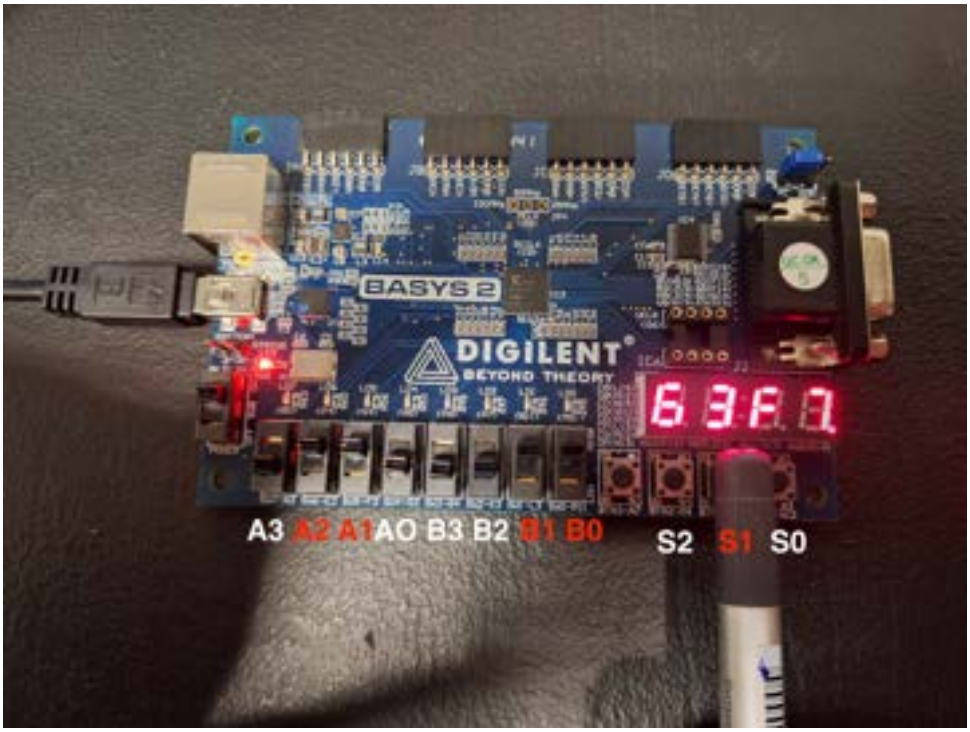

Figure 9: 0110 (6) + 0011 (3) = 1001 (9)out of range, overflow

1.10 SUB on signed numbers with overflow

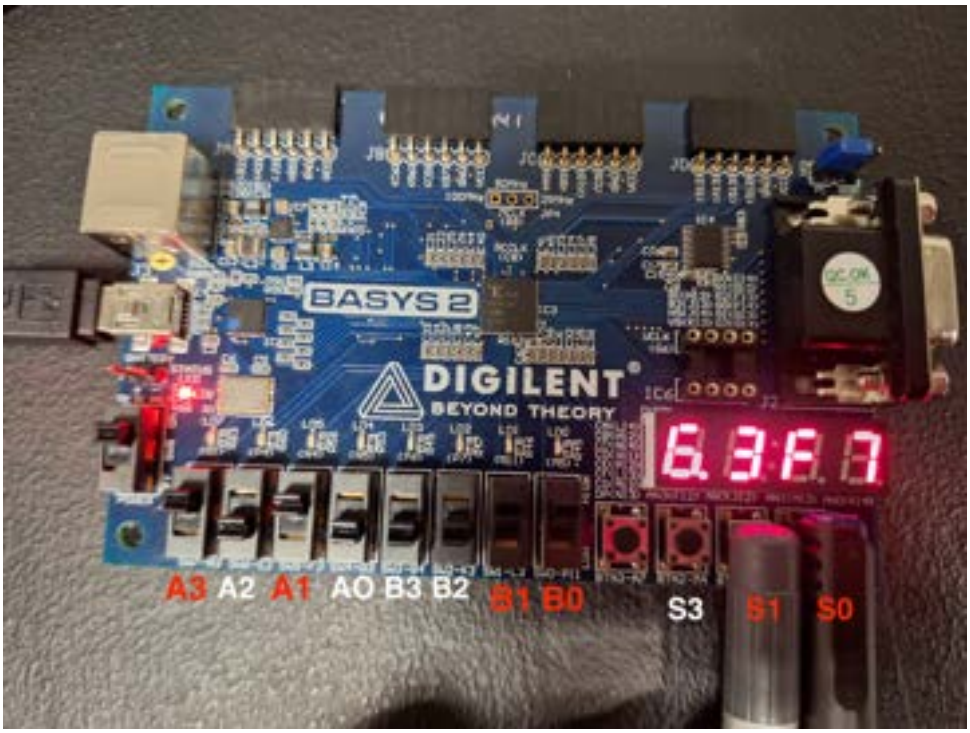

Figure 10: 1010 (-6) - 0011(3) = -9 out of range, overflow

2 VHDL code of all modules

2.1 ALU module

```
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 ALU is
    Port (  a : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit input number
            b : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit input number
            sel : in  STD_LOGIC_VECTOR (2 downto 0); -- 3-bit selection line
            overflow: out STD_LOGIC;
            result : out  STD_LOGIC_VECTOR (3 downto 0));
end ALU;

architecture Behavioral of ALU is

-- Internal signals
signal temp_sum, temp_diff : STD_LOGIC_VECTOR (3 downto 0); -- to check overflow during sum/sub

begin
-- Overflow detection for ADD, SUB
temp_sum <= std_logic_vector(signed(a)+signed(b));
temp_diff <= std_logic_vector(signed(a)-signed(b));

-- Process to check overflow
overflow_checker: process (a,b,sel, temp_sum, temp_diff)
begin
if sel = "010"  then   --if addition is selected
    if ((a(3)='0' and b(3)='0' and temp_sum(3)='1') or ((a(3)='1' and b(3)='1' and temp_sum(3)='0'))) then
        overflow <= '1'; --overflow condition
    else
        overflow <= '0';
    end if;       --close inner if
elsif sel = "011" then     --if subtraction is selected
    if ((a(3)='0' and b(3)='1' and temp_diff(3)='0') or ((a(3)='1' and b(3)='0' and temp_diff(3)='0'))) then
```

```
42          overflow <= '1'; --overflow condition
43      else
44          overflow <= '0';
45      end if;      -- close inner if
46  else
47      overflow <= '0';  -- to avoid latches
48  end if; --close outer if
49  end process;
50
51  -- ALU operations
52  with sel select result <=
53          std_logic_vector(a and b) when "000",  --AND (bitwise)
54          std_logic_vector(a or b) when "001",   --OR (bitwise)
55          temp_sum when "010", --ADD (on signed numbers)
56          temp_diff when "011", --SUB (on signed numbers)
57          std_logic_vector(rotate_right(signed(a),to_integer(signed
               (b)))) when "100", --ROR(rotate right)
58          std_logic_vector(shift_left(unsigned(a),to_integer(signed
               (b)))) when "101", --SLL(Shift left logical)
59          std_logic_vector(shift_right(unsigned(a),to_integer(
               signed(b)))) when "110", --SRL(Shift right logical)
60          std_logic_vector(shift_right(signed(a), to_integer(signed
               (b)))) when "111", --SRA(Shift right arithmetic)

61          (others => 'X') when others;
62
63  end Behavioral;
64
65
66  --000   AND
67  --001   OR
68  --010   ADD
69  --011   SUB
70  --100   ROR(rotate right)
71  --101   SLL(Shift left logical) : multiplying by 2^b, unsigned
72  --110   SRL(Shift right logical) : dividing by 2^b, unsigned
73  --111   SRA(Shift right arithmetic) : dividing by 2^b, preserve
       sign
```

Listing 1: VHDL code for "ALU" module

Note: SRL vs. SRA
SRL places 0 on the MSB of shifted number, while SRA places the original MSB to the shifted MSB to preserve the sign. Thus, SRA performs arithmetic integer division.

2.2 Single number module

Listing 2: VHDL code for "single number" module

2.3 ALU display module

```vhdl
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 ALU_display is
    Port (  clk : in STD_LOGIC; -- clock
            PI_a : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit input
            PI_b : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit input
            PI_sel : in  STD_LOGIC_VECTOR (2 downto 0); -- 3-bit selection
            PO_seg : out  STD_LOGIC_VECTOR (0 to 7); -- number to display, by using "single_number"
            PO_an : out  STD_LOGIC_VECTOR (3 downto 0)); -- anode selection, choice of which 7seg display unit to use
end ALU_display;

architecture Behavioral of ALU_display is

-- TODO component declarations, along with signals as needed

-- Internal signals
signal overflow_flag: STD_LOGIC; -- overflow detection
signal internal_result: STD_LOGIC_VECTOR (3 downto 0); -- 4-bit ALU computation results
signal number_display: STD_LOGIC_VECTOR (3 downto 0); -- 4-bit representation of the number to be displayed
signal segment: STD_LOGIC_VECTOR (7 downto 0); -- 8-bit representation of the number to be displayed

-- Use ALU as a component
component ALU is
    Port (  a : in  STD_LOGIC_VECTOR (3 downto 0);
            b : in  STD_LOGIC_VECTOR (3 downto 0);
            sel : in  STD_LOGIC_VECTOR (2 downto 0);
            overflow : out STD_LOGIC;
            result : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

```

```
41 --Use single_number as a decoder
42 component single_number is
43     Port (  number : in  STD_LOGIC_VECTOR (3 downto 0);
44             seg : out  STD_LOGIC_VECTOR (0 to 7));
45 end component;
46
47
48 -- clock divider signals
49 constant cnt_max : integer := 1e5;
50 signal clk_cnt : integer range 0 to cnt_max;
51 signal seg_mode, seg_mode_new : integer range 0 to 3;
52
53 begin
54 -- TODO component instances
55 ALU_instance: ALU -- (port => signal)
56     port map (  a => PI_a,
57                 b => PI_b,
58                 sel => PI_sel,
59                 result => internal_result,
60                 overflow => overflow_flag );
61
62 single_number_instance: single_number -- (port => signal)
63         port map(  number => number_display,
64                    seg => segment);
65
66 -- process to iterate through seg_mode
67 seg_mode_switch : process (clk)
68 begin
69         if rising_edge(clk) then
70             if (clk_cnt = cnt_max) then
71                         seg_mode <= seg_mode_new;
72                         clk_cnt <= 0;
73             else
74                         clk_cnt <= clk_cnt + 1;
75                 end if;
76         end if;
77 end process;
78
79 -- process to generate output on the four displays
80 -- TODO fill remaining parts
81 display : process (seg_mode, segment, PI_a, PI_b, internal_result
    , overflow_flag)
82     begin
83         if (seg_mode = 3) then
84             number_display <= PI_a; -- leftmost displays the
                first input number
85             PO_an <= "0111"; -- 0 is on
86             PO_seg <= segment;
87             seg_mode_new <= 2;
88         -- TODO fill missing cases
89         elsif (seg_mode = 2) then
```

```
90              number_display <= PI_b; -- second-leftmost displays
                   the second input number
91              PO_an <= "1011"; -- 0 is on
92              PO_seg <= segment;
93              seg_mode_new <= 1;
94          elsif (seg_mode = 1) then -- shows "F" when overflow
95              PO_an <= "1101"; -- 0 is on
96              number_display <= "0000"; -- to avoid latches
97              if(overflow_flag = '1') then
98                  PO_seg <= "01110001";  -- Pattern for "F"
99              else
100                 PO_seg <= "11111111";
101             end if;
102             seg_mode_new <= 0;
103         elsif (seg_mode = 0) then
104             number_display <= internal_result; -- ALU computation
                    result
105             PO_an <= "1110"; -- 0 is on
106             PO_seg <= segment;
107             seg_mode_new <= 3;
108         else
109             number_display <= "0000"; -- to avoid latches
110         end if;
111     end process;
112 end Behavioral;
```

Listing 3: VHDL code for "ALU display" module

2.4 Constraints file

```
1  NET "PO_seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
2  NET "PO_seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
3  NET "PO_seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
4  NET "PO_seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
5  NET "PO_seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
6  NET "PO_seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
7  NET "PO_seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
8  NET "PO_seg<7>" LOC = "N13"; # Bank = 1, Signal name = DP
9
10 NET "PO_an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3
11 NET "PO_an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
12 NET "PO_an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
13 NET "PO_an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0
14
15 #add missing inputs for a,b and op selection.
16
17 NET "clk" LOC = "B8";
18
19 NET "PI_a<3>" LOC = "N3";
20 NET "PI_a<2>" LOC = "E2";
21 NET "PI_a<1>" LOC = "F3";
```

```
22 NET "PI_a<0>" LOC = "G3";
23 NET "PI_b<3>" LOC = "B4";
24 NET "PI_b<2>" LOC = "K3";
25 NET "PI_b<1>" LOC = "L3";
26 NET "PI_b<0>" LOC = "P11";
27 NET "PI_sel<2>" LOC = "M4";
28 NET "PI_sel<1>" LOC = "C11";
29 NET "PI_sel<0>" LOC = "G12";
```

Listing 4: Constraints file for ALU display

3 RTL schmatic

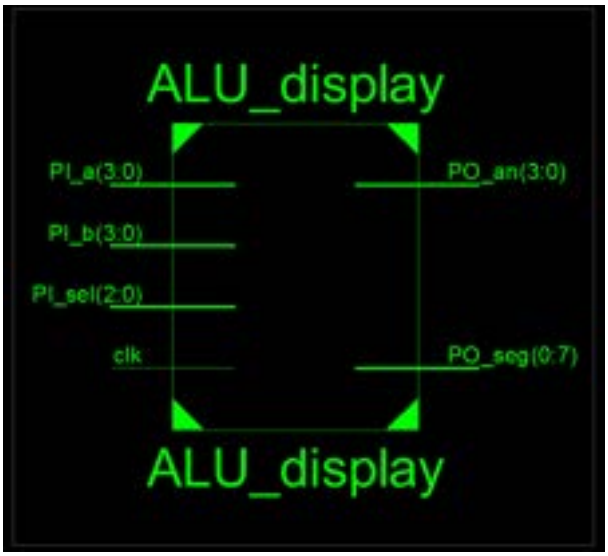

Figure 11: RTL schematic for ALU Display: Output=0