

Dongyun Lee

ID: 112794190

PreLab11: Multi-Digit Muxed Seven Segment SPI

ESE382-L01

Bench #4

```

1  -----
2  --
3  -- Title       : rx_buff_reg
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\rx_buff_reg.vhd
12 -- Generated   : Mon Apr 29 16:13:14 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25 entity rx_buff_reg is
26     port (
27         rst_bar : in std_logic; -- asynchronous reset
28         clk      : in std_logic; -- system clock
29         load_en  : in std_logic; -- enable shift
30         rx_buff_in : in std_logic_vector(7 downto 0); -- received data in
31         rx_buff_out : out std_logic_vector(7 downto 0) -- received data out
32     );
33 end rx_buff_reg;
34
35 architecture Behavioral of rx_buff_reg is
36 begin
37     double_buffer : process (clk, rst_bar)
38     begin
39         if rst_bar = '0' then
40             rx_buff_out <= (others => '0');
41         elsif rising_edge(clk) then
42             if load_en = '1' then
43                 rx_buff_out <= rx_buff_in;
44             end if;
45         end if;
46     end process;
47 end Behavioral;
48
49

```

```

1  -----
2  --
3  -- Title       : rx_buff_reg_tb
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\rx_buff_reg_tb.vhd
12 -- Generated   : Mon Apr 29 16:33:59 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 ----
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24 use work.all;
25
26 entity rx_buff_reg_tb is
27 end rx_buff_reg_tb;
28
29 architecture rx_buff_reg_tb of rx_buff_reg_tb is
30 signal rst_bar, clk, load_en : std_logic; --input
31 signal rx_buff_in : std_logic_vector(7 downto 0); -- input load
32 signal rx_buff_out : std_logic_vector(7 downto 0); -- output load
33
34 constant clk_period : time := 10ns;
35 begin
36
37 UUT: entity rx_buff_reg
38     port map (
39         rst_bar => rst_bar,
40         clk => clk,
41         load_en => load_en,
42         rx_buff_in => rx_buff_in,
43         rx_buff_out => rx_buff_out
44 );
45
46 clock_tb : process
47 begin
48     while true loop
49         clk <= '0';
50         wait for clk_period/2;
51         clk <= '1';
52         wait for clk_period/2;

```

```
53     end loop;
54 end process;
55
56 tb : process
57 begin
58     -- reset
59     rst_bar <= '0';
60     wait for clk_period * 2;
61     rst_bar <= '1';
62     wait for clk_period * 2;
63
64     -- load buffer
65     rx_buff_in <= "10101010";
66     load_en <= '1';
67     wait for clk_period;
68     load_en <= '0';
69
70     wait for clk_period * 5;
71
72     for i in 0 to 255 loop
73         rx_buff_in <= std_logic_vector(to_unsigned(i, 8));
74         load_en <= '1';
75         wait for clk_period;
76         load_en <= '0';
77         wait for clk_period;
78         assert rx_buff_out = std_logic_vector(to_unsigned(i,8))
79             report "Error at " & integer'image(i)
80             severity error;
81     end loop;
82     std.env.finish;
83 end process;
84
85 end rx_buff_reg_tb;
86
```

```

1  -----
2  --
3  -- Title       : hex_digit_reg
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\hex_digit_reg.vhd
12 -- Generated   : Mon Apr 29 20:18:24 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 --
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25 entity hex_digit_reg is
26     port (
27         rst_bar : in std_logic; -- asynchronous reset
28         clk      : in std_logic; -- system clock
29         load_en1 : in std_logic; -- enable load
30         load_en2 : in std_logic; -- enable load
31         hex_dig_in : in std_logic_vector(3 downto 0); -- received data in
32         hex_dig_out : out std_logic_vector(3 downto 0) -- received data out
33     );
34 end hex_digit_reg;
35
36 architecture behavioral of hex_digit_reg is
37 begin
38     reg : process (clk, rst_bar)
39     begin
40         if rst_bar = '0' then
41             hex_dig_out <= (others => '0');
42         elsif rising_edge(clk) then
43             if (load_en1 = '1') and (load_en2 = '1') then
44                 hex_dig_out <= hex_dig_in;
45             end if;
46         end if;
47     end process;
48 end behavioral;
49

```

```

1  -----
2  --
3  -- Title       : hex_digit_reg_tb
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\hex_digit_reg_tb.vhd
12 -- Generated   : Mon Apr 29 20:28:44 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24 use work.all;
25
26 entity hex_digit_reg_tb is
27 end hex_digit_reg_tb;
28
29
30 architecture hex_digit_reg_tb of hex_digit_reg_tb is
31 signal rst_bar, clk, load_en1, load_en2 : std_logic;
32 signal hex_dig_in : std_logic_vector (3 downto 0);
33 signal hex_dig_out : std_logic_vector (3 downto 0);
34
35 constant clk_period : time := 20ns;
36 begin
37     UUT: entity hex_digit_reg
38         port map(
39             rst_bar => rst_bar,
40             clk => clk,
41             load_en1 => load_en1,
42             load_en2 => load_en2,
43             hex_dig_in => hex_dig_in,
44             hex_dig_out => hex_dig_out
45         );
46
47
48     clk_tb : process
49     begin
50         while true loop
51             clk <= '0';
52             wait for clk_period/2;

```

```

53         clk <= '1';
54         wait for clk_period/2;
55     end loop;
56 end process;
57
58 -- Stimulus process
59 tb : process
60 begin
61     -- reset
62     rst_bar <= '0';
63     wait for clk_period * 2;
64     rst_bar <= '1';
65     wait for clk_period * 2;
66
67     -- Testing different inputs
68     for i in 0 to 15 loop -- only 16 possible values for 4 bits
69         hex_dig_in <= std_logic_vector(to_unsigned(i, 4));
70         load_en1 <= '1';
71         load_en2 <= '1';
72         wait for clk_period;
73         load_en1 <= '0';
74         load_en2 <= '0';
75
76         -- Conditional assertion check
77         if load_en1 = '1' and load_en2 = '1' then
78             assert hex_dig_out = std_logic_vector(to_unsigned(i, 4))
79                 report "Error at : " & integer'image(i)
80                 severity error;
81         end if;
82
83         -- Wait a bit before the next test to see changes clearly in
simulation
84         wait for clk_period * 2;
85     end loop;
86
87     -- Finish test
88     std.env.finish;
89 end process;
90
91
92
93 end hex_digit_reg_tb;
94

```

```

1  -----
2  --
3  -- Title       : decoder_2to4
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File         : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\decoder_2to4.vhd
12 -- Generated    : Mon Apr 29 22:16:17 2024
13 -- From         : interface description file
14 -- By           : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 --
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25 entity decoder_2to4 is
26     port (
27         b : in std_logic; -- most significant address bit
28         a : in std_logic; -- least significant address bit
29         Y : out std_logic_vector(3 downto 0) -- selected output asserted
30         high
31     );
32 end decoder_2to4;
33
34 architecture dataflow of decoder_2to4 is
35 begin
36     Y <= "0001" when (a & b = "00") else
37         "0010" when (a & b = "01") else
38         "0100" when (a & b = "10") else
39         "1000" when (a & b = "11") else
40         (others => '0'); -- default case to handle undefined statesend
41 dataflow;
42 end dataflow;

```

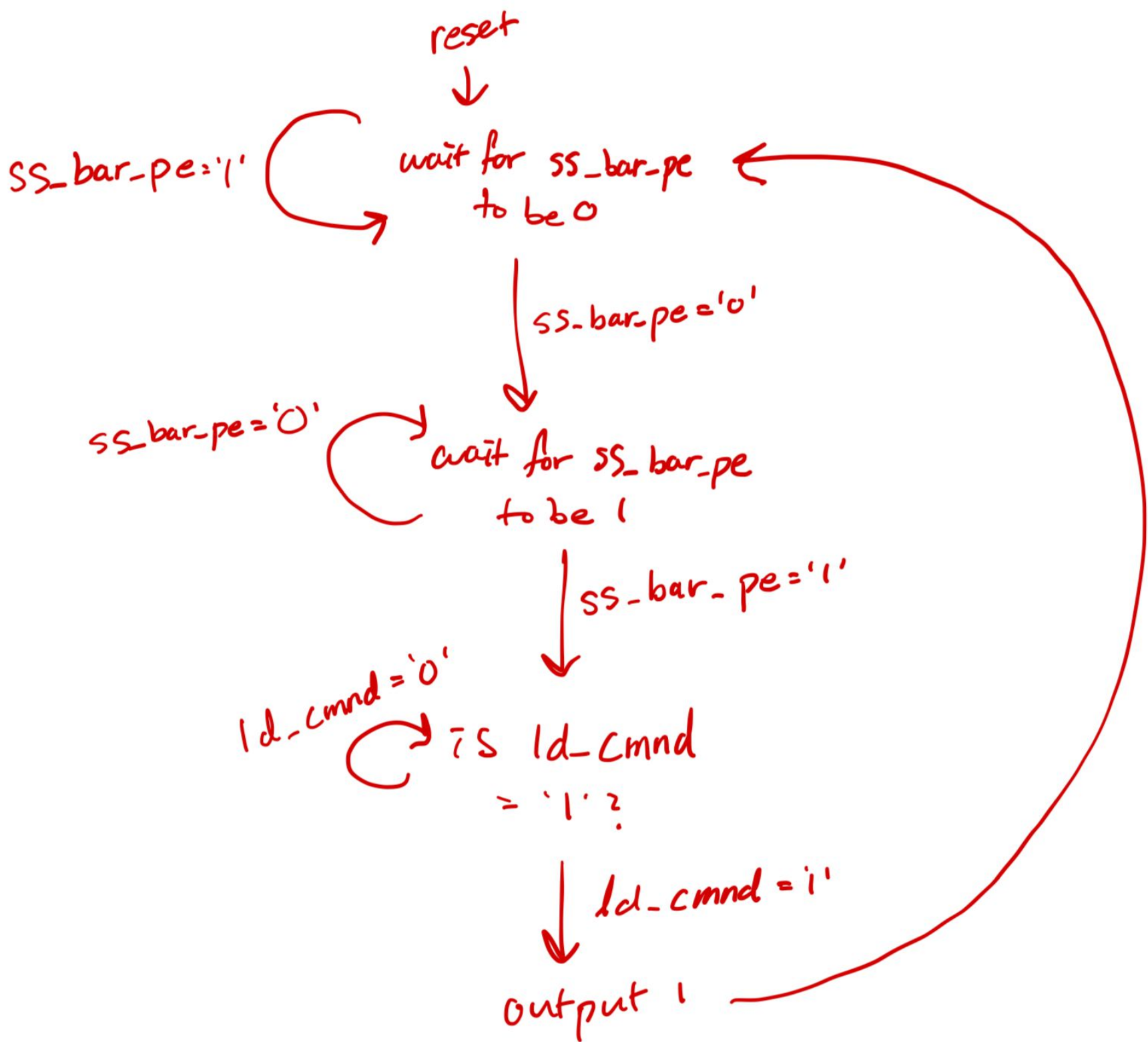


```

1  -----
2  --
3  -- Title       : decoder_2to4_tb
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\decoder_2to4_tb.vhd
12 -- Generated   : Mon Apr 29 22:33:59 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 --
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24 use work.all;
25
26
27 entity decoder_2to4_tb is
28 end decoder_2to4_tb;
29
30 architecture decoder_2to4_tb of decoder_2to4_tb is
31 signal a, b : std_logic;
32 signal Y : std_logic_vector(3 downto 0);
33
34 type test_vector is record
35     a : std_logic;
36     b : std_logic;
37     Y : std_logic_vector(3 downto 0);
38 end record;
39
40 type test_vector_table is array (natural range <>) of test_vector;
41
42 constant LUT : test_vector_table := (
43     --      a      b      Y
44     ('0', '0', "0001"),
45     ('0', '1', "0010"),
46     ('1', '0', "0100"),
47     ('1', '1', "1000")
48 );
49
50
51 begin
52     UUT : entity decoder_2to4

```

```
53     port map (  
54         a => a,  
55         b => b,  
56         Y => Y  
57     );  
58  
59     tb : process  
60     begin  
61         for i in LUT'range loop  
62             a <= LUT(i).a;  
63             b <= LUT(i).b;  
64             wait for 20ns;  
65             assert Y = LUT(i).Y  
66             report "Error at input : " & std_logic'image(a) & std_logic'  
image(b)  
67                 severity error;  
68         end loop;  
69         std.env.finish;  
70     end process;  
71  
72 end decoder_2to4_tb;  
73
```



```

1  -----
2  --
3  -- Title       : load_digit_fsm
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\load_digit_fsm.vhd
12 -- Generated   : Tue Apr 30 00:02:22 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25
26 entity load_digit_fsm is
27     port (
28         rst_bar : in std_logic; -- asynchronous system reset
29         clk      : in std_logic; -- system clock
30         ss_bar_pe : in std_logic; -- positive edge of ss_bar detected
31         ld_cmd    : in std_logic; -- bit 7 is '1' for load command
32         load_dig  : out std_logic -- enable a hex_digit to be loaded
33     );
34 end load_digit_fsm;
35
36 architecture moore_fsm of load_digit_fsm is
37     type state is (wait_for_sb_0, wait_for_sb_1, wait_ldc_1, output_state);
38     signal present_state, next_state : state;
39     begin
40         -- first state : detects rst_bar
41         state_reg : process (clk, rst_bar)
42         begin
43             if rst_bar = '0' then
44                 present_state <= wait_for_sb_0;
45             elsif rising_edge(clk) then
46                 present_state <= next_state;
47             end if;
48         end process;
49
50         -- process where it outputs
51         outputs: process (present_state)
52         begin

```

```
53         case present_state is
54             when output_state => load_dig <= '1';
55             when others => load_dig <= '0';
56         end case;
57     end process;
58
59     nxt_state: process (present_state, ss_bar_pe, ld_cmd)
60     begin
61         case present_state is
62             when wait_for_sb_0 =>
63                 if ss_bar_pe = '0' then
64                     next_state <= wait_for_sb_1;
65                 else
66                     next_state <= wait_for_sb_0;
67                 end if;
68
69             when wait_for_sb_1 =>
70                 if ss_bar_pe = '1' then
71                     next_state <= wait_ldc_1;
72                 else
73                     next_state <= wait_for_sb_1;
74                 end if;
75
76             when wait_ldc_1 =>
77                 if ld_cmd = '1' then
78                     next_state <= output_state;
79                 else
80                     next_state <= wait_ldc_1;
81                 end if;
82
83             when output_state =>
84                 next_state <= wait_for_sb_0;
85
86             when others =>
87                 next_state <= wait_for_sb_0;
88
89         end case;
90     end process;
91
92 end moore_fsm;
93
```

```

1  -----
2  --
3  -- Title       : load_digit_fsm_tb
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\load_digit_fsm_tb.vhd
12 -- Generated   : Tue Apr 30 01:08:38 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 --
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24 use work.all;
25
26 entity load_digit_fsm_tb is
27 end load_digit_fsm_tb;
28
29 architecture load_digit_fsm_tb of load_digit_fsm_tb is
30 -- Signals for the FSM inputs and outputs
31 signal rst_bar    : std_logic;
32 signal clk        : std_logic;
33 signal ss_bar_pe  : std_logic;
34 signal ld_cmd     : std_logic;
35 signal load_dig   : std_logic;
36
37 begin
38 UUT: entity load_digit_fsm
39     port map (
40         rst_bar    => rst_bar,
41         clk        => clk,
42         ss_bar_pe  => ss_bar_pe,
43         ld_cmd     => ld_cmd,
44         load_dig   => load_dig
45     );
46
47 -- Clock process
48 clocking : process
49 begin
50     while true loop
51         clk <= '0';
52         wait for 10 ns; -- Clock low for 10 ns

```

```

53         clk <= '1';
54         wait for 10 ns; -- Clock high for 10 ns
55     end loop;
56 end process;
57 -- Test stimulus process
58 stim_proc : process
59 begin
60     -- Initial Reset
61     rst_bar <= '0';
62     ss_bar_pe <= '0';
63     ld_cmd <= '0';
64     wait for 100 ns; -- Hold reset for a few clock cycles
65     rst_bar <= '1';
66     wait for 100 ns; -- Wait after reset is de-asserted
67
68     -- First event: simulating ss_bar_pe positive edge
69     ss_bar_pe <= '1';
70     wait for 40 ns; -- Wait long enough for FSM to detect ss_bar_pe
71
72     -- Second event: simulating ld_cmd being '1'
73     ld_cmd <= '1';
74     wait for 80 ns; -- Wait long enough for FSM to detect ld_cmd
75
76     -- Both ss_bar_pe and ld_cmd are '1', now the FSM should transition to
output_state
77     -- FSM should now output '1' on load_dig signal
78     -- Wait and observe the load_dig signal in the waveform viewer
79     wait for 40 ns;
80
81     -- De-assert ld_cmd and ss_bar_pe to observe FSM returning to wait
state
82     ld_cmd <= '0';
83     ss_bar_pe <= '0';
84     wait for 40 ns;
85
86     -- Asserting only ss_bar_pe to '1' to ensure FSM does not transition to
output_state incorrectly
87     ss_bar_pe <= '1';
88     wait for 40 ns;
89
90     -- End the simulation
91     wait;
92 end process;
93
94 end load_digit_fsm_tb;
95

```

```

1  -----
2  --
3  -- Title       : hex_dig_mux
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\hex_dig_mux.vhd
12 -- Generated   : Tue Apr 30 12:04:27 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 library ieee;
21 use ieee.std_logic_1164.all;
22 use ieee.numeric_std.all;
23
24 entity hex_dig_mux is
25     port (
26         hex_dig_0 : in std_logic_vector(3 downto 0); -- mux input vectors
27         hex_dig_1 : in std_logic_vector(3 downto 0); -- mux input vectors
28         hex_dig_2 : in std_logic_vector(3 downto 0); -- mux input vectors
29         hex_dig_3 : in std_logic_vector(3 downto 0); -- mux input vectors
30         sel       : in std_logic_vector(1 downto 0); -- multiplexer select inputs
31         hex_dig_out : out std_logic_vector(3 downto 0) -- multiplexer
32         output
33     );
34 end hex_dig_mux;
35
36 architecture behavioral of hex_dig_mux is
37     begin
38         with sel select
39             hex_dig_out <= hex_dig_0 when "00",
40             hex_dig_1 when "01",
41             hex_dig_2 when "10",
42             hex_dig_3 when "11",
43             "----" when others;
44     end behavioral;
45

```



```

1  -----
2  --
3  -- Title       : hex_dig_mux_tb
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File        : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\hex_dig_mux_tb.vhd
12 -- Generated   : Tue Apr 30 13:06:07 2024
13 -- From        : interface description file
14 -- By          : Itf2Vhdl ver. 1.22
15 -----
16 --
17 -- Description :
18 --
19 -----
20 ----
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24 use work.all;
25
26 entity hex_dig_mux_tb is
27 end hex_dig_mux_tb;
28
29 architecture hex_dig_mux_tb of hex_dig_mux_tb is
30 signal hex_dig_0, hex_dig_1, hex_dig_2, hex_dig_3 : std_logic_vector(3
31 downto 0);
32 signal sel : std_logic_vector(1 downto 0);
33 signal hex_dig_out : std_logic_vector(3 downto 0);
34
35 type test_vector is record
36     hex_dig_0 : std_logic_vector(3 downto 0);
37     hex_dig_1 : std_logic_vector(3 downto 0);
38     hex_dig_2 : std_logic_vector(3 downto 0);
39     hex_dig_3 : std_logic_vector(3 downto 0);
40     sel       : std_logic_vector(1 downto 0);
41     hex_dig_out : std_logic_vector(3 downto 0);
42 end record;
43
44 type test_table is array (natural range <>) of test_vector;
45
46 constant LUT : test_table := (
47     ("0001", "0010", "0100", "1000", "00", "0001"),
48     ("0001", "0010", "0100", "1000", "01", "0010"),
49     ("0001", "0010", "0100", "1000", "10", "0100"),
50     ("0001", "0010", "0100", "1000", "11", "1000")
51 );

```

```
52
53 begin
54     UUT : entity hex_dig_mux
55         port map (
56             hex_dig_0 => hex_dig_0,
57             hex_dig_1 => hex_dig_1,
58             hex_dig_2 => hex_dig_2,
59             hex_dig_3 => hex_dig_3,
60             sel       => sel,
61             hex_dig_out => hex_dig_out
62         );
63
64
65     tb : process
66     begin
67         for i in LUT'range loop
68             hex_dig_0 <= LUT(i).hex_dig_0;
69             hex_dig_1 <= LUT(i).hex_dig_1;
70             hex_dig_2 <= LUT(i).hex_dig_2;
71             hex_dig_3 <= LUT(i).hex_dig_3;
72             sel       <= LUT(i).sel;
73             wait for 20 ns;
74             assert hex_dig_out = LUT(i).hex_dig_out
75                 report "Error at select input : " & to_string(sel)
76                 severity error;
77         end loop;
78         std.env.finish;
79     end process;
80
81
82
83 end hex_dig_mux_tb;
84
```

```

1  -----
2  --
3  -- Title       : spi_mux_digit_driver
4  -- Design      : prelab11
5  -- Author      : Dongyun Lee
6  -- Company     : Stony Brook University
7  --
8  -----
9  --
10 -- File         : \\Mac\Home\Documents\SBU 2024 Spring\ESE
11 382\Prelab11\Prelab11\prelab11\src\spi_mux_digit_driver.vhd
12 -- Generated    : Tue Apr 30 15:39:45 2024
13 -- From         : interface description file
14 -- By           : Itf2Vhdl ver. 1.22
15 --
16 -----
17 -- Description :
18 --
19 -----
20
21 -----slv_spi_rx_shifter-----
22 library ieee;
23 use ieee.std_logic_1164.all;
24 use ieee.numeric_std.all;
25
26
27 entity slv_spi_rx_shifter is
28     port(
29         rxd          : in  std_logic;           -- Data received from master
30         rst_bar      : in  std_logic;           -- Asynchronous reset
31         sel_bar       : in  std_logic;           -- Selects shifter for
32         operation
33         clk           : in  std_logic;           -- System clock
34         shift_en      : in  std_logic;           -- Enable shift
35         rx_data_out   : out std_logic_vector(7 downto 0) -- Received data
36     );
37 end entity slv_spi_rx_shifter;
38
39 architecture slv_spi_rx_shifter of slv_spi_rx_shifter is
40 begin
41     shift: process (clk, rst_bar)
42         -- variable memory : unsigned(7 downto 0);
43     begin
44         if rst_bar = '0' then
45             rx_data_out <= (others => '0');
46         elsif rising_edge(clk) then
47             if sel_bar = '0' then
48                 if (shift_en = '1') and (rxd = '1' or rxd = '0') then
49                     rx_data_out <= rx_data_out(6 downto 0) & rxd;
50                 end if;
51             end if;
52         end if;
53     end process;
54 end architecture;

```

```

52
53         end if;
54
55     end process;
56 end slv_spi_rx_shifter;
57
58
59 -----edge_det-----
60 library ieee;
61 use ieee.std_logic_1164.all;
62 use ieee.numeric_std.all;
63
64
65 entity edge_det is
66     port(
67         rst_bar    : in  std_logic; -- Asynchronous system reset
68         clk        : in  std_logic; -- System clock
69         sig        : in  std_logic; -- Input signal
70         pos        : in  std_logic; -- '1' for positive edge, '0' for
        negative
71         sig_edge   : out std_logic  -- High for one system clk after edge
        detected
72     );
73 end entity edge_det;
74
75
76 architecture moore_fsm of edge_det is
77     type state is (state_a, state_b, state_c);
78     signal present_state, next_state : state;
79 begin
80     -- first state : detects rst_bar
81     state_reg: process (clk, rst_bar)
82     begin
83         if rst_bar = '0' then
84             present_state <= state_a;
85         elsif rising_edge(clk) then
86             present_state <= next_state;
87         end if;
88     end process;
89
90     -- process where it outputs
91     outputs: process (present_state)
92     begin
93         case present_state is
94             when state_c => sig_edge <= '1';
95             when others => sig_edge <= '0';
96         end case;
97     end process;
98
99     nxt_state: process (present_state, sig)
100     begin
101         case present_state is
102             when state_a =>
103                 if (pos = '1' and sig = '0') or (pos = '0' and sig = '1') then
104                     next_state <= state_b;
105                 else
106                     next_state <= state_a;

```

```

107         end if;
108
109         when state_b =>
110             if (pos = '1' and sig = '1') or (pos = '0' and sig = '0') then
111                 next_state <= state_c;
112             else
113                 next_state <= state_b;
114             end if;
115
116         when others =>
117             if (pos = '1' and sig = '0') or (pos = '0' and sig = '1') then
118                 next_state <= state_b;
119             else
120                 next_state <= state_a;
121             end if;
122         end case;
123     end process;
124
125 end moore_fsm;
126
127
128 -----rx_buff_reg-----
129 library ieee;
130 use ieee.std_logic_1164.all;
131 use ieee.numeric_std.all;
132
133 entity rx_buff_reg is
134     port (
135         rst_bar : in std_logic; -- asynchronous reset
136         clk : in std_logic; -- system clock
137         load_en : in std_logic; -- enable shift
138         rx_buff_in : in std_logic_vector(7 downto 0); -- received data in
139         rx_buff_out : out std_logic_vector(7 downto 0) -- received data
140     out
141     );
142 end rx_buff_reg;
143
144 architecture Behavioral of rx_buff_reg is
145     begin
146         double_buffer : process (clk, rst_bar)
147         begin
148             if rst_bar = '0' then
149                 rx_buff_out <= (others => '0');
150             elsif rising_edge(clk) then
151                 if load_en = '1' then
152                     rx_buff_out <= rx_buff_in;
153                 end if;
154             end if;
155         end process;
156     end Behavioral;
157
158 -----hex_digit_reg-----
159 library ieee;
160 use ieee.std_logic_1164.all;
161 use ieee.numeric_std.all;
162

```

```

163 entity hex_digit_reg is
164     port (
165         rst_bar : in std_logic; -- asynchronous reset
166         clk : in std_logic; -- system clock
167         load_en1 : in std_logic; -- enable load
168         load_en2 : in std_logic; -- enable load
169         hex_dig_in : in std_logic_vector(3 downto 0); -- received data in
170         hex_dig_out : out std_logic_vector(3 downto 0) -- received data
171     out
172 );
173 end hex_digit_reg;
174 architecture behavioral of hex_digit_reg is
175 begin
176     reg : process (clk, rst_bar)
177     begin
178         if rst_bar = '0' then
179             hex_dig_out <= (others => '0');
180         elsif rising_edge(clk) then
181             if (load_en1 = '1') and (load_en2 = '1') then
182                 hex_dig_out <= hex_dig_in;
183             end if;
184         end if;
185     end process;
186 end behavioral;
187
188 -----decoder_2to4-----
189
190 library ieee;
191 use ieee.std_logic_1164.all;
192 use ieee.numeric_std.all;
193
194 entity decoder_2to4 is
195     port (
196         b : in std_logic; -- most significant address bit
197         a : in std_logic; -- least significant address bit
198         Y : out std_logic_vector(3 downto 0) -- selected output asserted
199     high
200 );
201 end decoder_2to4;
202
203 architecture dataflow of decoder_2to4 is
204 begin
205     Y <= "0001" when (a & b = "00") else
206         "0010" when (a & b = "01") else
207         "0100" when (a & b = "10") else
208         "1000" when (a & b = "11") else
209         (others => '0'); -- default case to handle undefined statesend
210 dataflow;
211 end dataflow;
212
213 -----load_digit_fsm-----
214 library ieee;
215 use ieee.std_logic_1164.all;
216 use ieee.numeric_std.all;

```

```

217 entity load_digit_fsm is
218     port (
219         rst_bar : in std_logic; -- asynchronous system reset
220         clk : in std_logic; -- system clock
221         ss_bar_pe : in std_logic; -- positive edge of ss_bar detected
222         ld_cmd : in std_logic; -- bit 7 is '1' for load command
223         load_dig : out std_logic -- enable a hex_digit to be loaded
224     );
225 end load_digit_fsm;
226
227 architecture moore_fsm of load_digit_fsm is
228     type state is (wait_for_sb_0, wait_for_sb_1, wait_ldc_1, output_state);
229     signal present_state, next_state : state;
230 begin
231     -- first state : detects rst_bar
232     state_reg : process (clk, rst_bar)
233     begin
234         if rst_bar = '0' then
235             present_state <= wait_for_sb_0;
236         elsif rising_edge(clk) then
237             present_state <= next_state;
238         end if;
239     end process;
240
241     -- process where it outputs
242     outputs: process (present_state)
243     begin
244         case present_state is
245             when output_state => load_dig <= '1';
246             when others => load_dig <= '0';
247         end case;
248     end process;
249
250     nxt_state: process (present_state, ss_bar_pe, ld_cmd)
251     begin
252         case present_state is
253             when wait_for_sb_0 =>
254                 if ss_bar_pe = '0' then
255                     next_state <= wait_for_sb_1;
256                 else
257                     next_state <= wait_for_sb_0;
258                 end if;
259
260             when wait_for_sb_1 =>
261                 if ss_bar_pe = '1' then
262                     next_state <= wait_ldc_1;
263                 else
264                     next_state <= wait_for_sb_1;
265                 end if;
266
267             when wait_ldc_1 =>
268                 if ld_cmd = '1' then
269                     next_state <= output_state;
270                 else
271                     next_state <= wait_ldc_1;
272                 end if;
273

```

```

274         when output_state =>
275             next_state <= wait_for_sb_0;
276
277         when others =>
278             next_state <= wait_for_sb_0;
279
280     end case;
281 end process;
282
283 end moore_fsm;
284
285 -----hex_dig_mux -----
286
287 library ieee;
288 use ieee.std_logic_1164.all;
289 use ieee.numeric_std.all;
290
291 entity hex_dig_mux is
292     port (
293         hex_dig_0 : in std_logic_vector(3 downto 0); -- mux input vectors
294         hex_dig_1 : in std_logic_vector(3 downto 0); -- mux input vectors
295         hex_dig_2 : in std_logic_vector(3 downto 0); -- mux input vectors
296         hex_dig_3 : in std_logic_vector(3 downto 0); -- mux input vectors
297         sel : in std_logic_vector(1 downto 0); -- multiplexer select
298
299         inputs
300         hex_dig_out : out std_logic_vector(3 downto 0) -- multiplexer
301         output
302     );
303 end hex_dig_mux;
304
305 architecture behavioral of hex_dig_mux is
306     begin
307         with sel select
308             hex_dig_out <= hex_dig_0 when "00",
309             hex_dig_1 when "01",
310             hex_dig_2 when "10",
311             hex_dig_3 when "11",
312             "----" when others;
313     end behavioral;
314
315 -----hex_seven-----
316
317 library ieee;
318 use ieee.std_logic_1164.all;
319 use ieee.numeric_std.all;
320
321 entity hex_seven is
322     port(
323         hex : in std_logic_vector(3 downto 0); -- hexadecimal input
324         -- segs. a..g right justified
325         seg_drive : out std_logic_vector(7 downto 0)
326     );
327 end hex_seven;
328

```



```

329 architecture behavioral of hex_seven is
330 begin
331     with hex select
332     seg_drive <=
333         "01111110" when "0000", -- 0
334         "00110000" when "0001", -- 1
335         "01101101" when "0010", -- 2
336         "01111001" when "0011", -- 3
337         "00110011" when "0100", -- 4
338         "01011011" when "0101", -- 5
339         "01011111" when "0110", -- 6
340         "01110000" when "0111", -- 7
341         "01111111" when "1000", -- 8
342         "01111011" when "1001", -- 9
343         "01110111" when "1010", -- A
344         "00011111" when "1011", -- b
345         "01001110" when "1100", -- C
346         "00111101" when "1101", -- d
347         "01001111" when "1110", -- E
348         "01000111" when others; -- F
349 end architecture behavioral;
350
351
352 -----top level-----
353 library ieee;
354 use ieee.std_logic_1164.all;
355 use ieee.numeric_std.all;
356 use work.all;
357
358 entity spi_mux_digit_driver is
359 port(
360     rst_bar : in std_logic; -- asynchronous system reset
361     clk : in std_logic; -- system clock
362     mosi : in std_logic; -- master out slave in SPI serial data
363     sck : in std_logic; -- SPI shift clock to slave
364     ss_bar : in std_logic; -- slave select signal
365     sel : in std_logic_vector(1 downto 0);
366     seg_drive : out std_logic_vector(7 downto 0)
367 );
368 end spi_mux_digit_driver;
369
370 architecture spi_mux_digit_driver of spi_mux_digit_driver is
371 signal sig_edge_to_shift_en : std_logic;
372 signal rx_8bits : std_logic_vector(7 downto 0);
373 signal u3_to_u4 : std_logic;
374 signal u4_output_8bits : std_logic_vector(7 downto 0);
375 signal u5_output : std_logic_vector(3 downto 0);
376 signal u12_output : std_logic;
377 signal hex_reg_0, hex_reg_1, hex_reg_2, hex_reg_3 : std_logic_vector(3
downto 0);
378 signal u10_output : std_logic_vector(3 downto 0);
379 begin
380     u1 : entity edge_det port map (clk => clk, rst_bar => rst_bar, sig =>
sck, pos => '1', sig_edge => sig_edge_to_shift_en);
381     u2 : entity slv_spi_rx_shifter
382         port map (
383         shift_en => sig_edge_to_shift_en,

```

```

384         clk => clk,
385         sel_bar => ss_bar,
386         rst_bar => rst_bar,
387         rxd => mosi,
388         rx_data_out => rx_8bits
389     );
390     u3 : entity edge_det port map (clk => clk, rst_bar => rst_bar, sig =>
ss_bar, pos => '1', sig_edge => u3_to_u4);
391     u4 : entity rx_buff_reg
392         port map(
393             rst_bar => rst_bar,
394             clk => clk,
395             load_en => u3_to_u4,
396             rx_buff_in => rx_8bits,
397             rx_buff_out => u4_output_8bits
398         );
399     u5 : entity decoder_2to4
400         port map(
401             b => u4_output_8bits(5),
402             a => u4_output_8bits(4),
403             y => u5_output
404         );
405     u6 : entity hex_digit_reg
406         port map(
407             rst_bar => rst_bar,
408             clk => clk,
409             load_en1 => u5_output(0),
410             load_en2 => u12_output,
411             hex_dig_in => u4_output_8bits(3 downto 0),
412             hex_dig_out => hex_reg_0
413         );
414
415     u7 : entity hex_digit_reg
416         port map(
417             rst_bar => rst_bar,
418             clk => clk,
419             load_en1 => u5_output(1),
420             load_en2 => u12_output,
421             hex_dig_in => u4_output_8bits(3 downto 0),
422             hex_dig_out => hex_reg_1
423         );
424
425     u8 : entity hex_digit_reg
426         port map(
427             rst_bar => rst_bar,
428             clk => clk,
429             load_en1 => u5_output(2),
430             load_en2 => u12_output,
431             hex_dig_in => u4_output_8bits(3 downto 0),
432             hex_dig_out => hex_reg_2
433         );
434
435     u9 : entity hex_digit_reg
436         port map(
437             rst_bar => rst_bar,
438             clk => clk,
439             load_en1 => u5_output(3),

```

```
440         load_en2 => u12_output,
441         hex_dig_in => u4_output_8bits(3 downto 0),
442         hex_dig_out => hex_reg_3
443     );
444
445     u10 : entity hex_dig_mux
446     port map(
447         hex_dig_0 => hex_reg_0,
448         hex_dig_1 => hex_reg_1,
449         hex_dig_2 => hex_reg_2,
450         hex_dig_3 => hex_reg_3,
451         sel => sel,
452         hex_dig_out => u10_output
453     );
454
455     u11 : entity hex_seven
456     port map(
457         hex => u10_output,
458         seg_drive => seg_drive
459     );
460
461
462     u12 : entity load_digit_fsm
463     port map(
464         rst_bar => rst_bar,
465         clk => clk,
466         ss_bar_pe => u3_to_u4,
467         ld_cmd => u4_output_8bits(7),
468         load_dig => u12_output
469     );
470
471
472 end spi_mux_digit_driver;
473
```

```

1  -- Testbench for Laboratory 11 Spring 2024
2
3
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7  use ieee.numeric_std.all;
8  use work.all;
9
10
11  entity spi_mux_digit_driver_tb is
12  end spi_mux_digit_driver_tb;
13
14  architecture TB_ARCHITECTURE of spi_mux_digit_driver_tb is
15
16      -- Stimulus signals - signals mapped to the input and inout ports of
17      -- tested entity
18      signal rst_bar : std_logic;
19      signal clk : std_logic;
20      signal mosi : std_logic;
21      signal sck : std_logic;
22      signal ss_bar : std_logic;
23      signal sel : std_logic_vector(1 downto 0) := "00";
24      -- Observed signals - signals mapped to the output ports of tested
25      -- entity
26      signal seg_drive : std_logic_vector(7 downto 0);
27
28      -- system clock period is being specified relative to shift
29      -- clock period so that effect of changing the system clock
30      -- on system's operation can be observed
31      constant sck_period : time := 4.0 us;
32      constant period : time := sck_period/4.0;
33      signal end_sim : boolean := false;
34
35  begin
36
37      -- Unit Under Test port map
38      UUT : entity spi_mux_digit_driver
39      port map (
40          rst_bar => rst_bar,
41          clk => clk,
42          mosi => mosi,
43          sck => sck,
44          ss_bar => ss_bar,
45          sel => sel,
46          seg_drive => seg_drive
47      );
48
49
50      -- generate system reset
51      rst_bar <= '0', '1' after period;
52
53
54      -- system clock runs until end_sim = false
55      clock_gen : process

```

```

56     begin
57         clk <= '0';
58         loop
59             wait for period/2;
60             clk <= not clk;
61             exit when end_sim = true;
62         end loop;
63         wait;
64     end process;
65
66
67     -- Generate SPI Shift Clock and MOSI data
68     send_spi_byte: process
69         variable data_in : std_logic_vector(7 downto 0);
70         variable addr : unsigned(7 downto 0) := "00000000";
71
72     begin
73         for k in 0 to 15 loop
74             data_in := "10000000" or std_logic_vector(addr) or
std_logic_vector(to_unsigned(k, 8));
75             ss_bar <= '1'; -- select slave
76             sck <= '0';    -- starting shift clock value CPOL = 0
77             mosi <= '0';
78
79             wait for 2 * sck_period;
80             ss_bar <= '0';
81             wait for sck_period;
82             for i in 7 downto 0 loop -- generate 8 data bits
83
84                 mosi <= data_in(i); -- and shift clock pulses
85                 wait for sck_period/2;
86                 sck <= not sck;
87                 wait for sck_period/2;
88                 sck <= not sck;
89             end loop; -- i index
90             wait for sck_period;
91             ss_bar <= '1'; -- deselect slave
92
93             for n in 0 to 3 loop
94                 sel <= std_logic_vector(to_unsigned(n, 2));
95                 wait for 10 * sck_period;
96             end loop; -- n indexed loop
97
98             addr := (addr + "00010000") and "00110000";
99         end loop; -- k indexed loop
100         end_sim <= true; -- stop system clock
101         std.env.finish; -- stop simulation
102     end process;
103 end tb_architecture;
104
105
106
107

```