Lab 4 Report

ECE 124

Group 3 Session 201

Nandita Lohani

Puneet Bhullar

Logical Step Top

```
1  --Author: Group 3, Nandita Lohani, Puneet Bhullar
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4  USE ieee.numeric_std.ALL;
5
6  ENTITY LogicalStep_Lab4_top IS
7     PORT
8     (
9     Clk         : in  std_logic;
10    pb_n        : in  std_logic_vector(3 downto 0);
11    sw          : in  std_logic_vector(7 downto 0);
12    leds        : out std_logic_vector(7 downto 0);
13
14 ------------------------------------------------------------------
15    xreg, yreg  : out std_logic_vector(3 downto 0);-- (for SIMULATION only)
16    xPOS, yPOS  : out std_logic_vector(3 downto 0);-- (for SIMULATION only)
17 ------------------------------------------------------------------
18    seg7_data   : out std_logic_vector(6 downto 0); -- 7-bit outputs to a 7-segment display (for LogicalStep only)
19    seg7_char1  : out std_logic                     -- seg7 digit1 selector (for LogicalStep only)
20    seg7_char2  : out std_logic                     -- seg7 digit2 selector (for LogicalStep only)
21
22    );
23 END LogicalStep_Lab4_top;
24
25 ARCHITECTURE Circuit OF LogicalStep_Lab4_top IS
26 ------------------------------------------------------------------
27 -- Project Components Used
28 ------------------------------------------------------------------
29 COMPONENT Clock_Source  port (SIM_FLAG: in boolean;clk_input: in std_logic;clock_out: out std_logic);
30 END COMPONENT;
31 ------------------------------------------------------------------
32 component SevenSegment
33   port
34    (
35       hex      :  in  std_logic_vector(3 downto 0);    -- The 4 bit data to be displayed
36       sevenseg :  out std_logic_vector(6 downto 0)     -- 7-bit outputs to a 7-segment
37    );
38 end component SevenSegment;
39 ------------------------------------------------------------------
```

Logical Step Top Cont.

```
39  |-------------------------------------------------------------------
40  component segment7_mux
41  |  port
42  |   (
43  |      clk         : in  std_logic := '0';
44  |      DIN2        : in  std_logic_vector(6 downto 0);
45  |      DIN1        : in  std_logic_vector(6 downto 0);
46  |      DOUT        : out std_logic_vector(6 downto 0);
47  |      DIG2        : out std_logic;
48  |      DIG1        : out std_logic
49  |   );
50  end component segment7_mux;
51  |-------------------------------------------------------------------
52  component Bidir_shift_reg
53  |   port
54  |   (
55  |      clk            : in std_logic := '0';
56  |      reset          : in std_logic := '0';
57  |      clk_en         : in std_logic := '0';
58  |      left0_right1   : in std_logic := '0';
59  |      reg_bits       : out std_logic_vector( 3 downto 0)
60  |
61  |   );
62  end component Bidir_shift_reg;
63  |-------------------------------------------------------------------
64  component Compx4
65  |   port
66  |   (
67  |         A, B : in std_logic_vector(3 downto 0);
68  |         AGTB_out, AEQB_out, ALTB_out : out std_logic
69  |   );
70  end component Compx4;
71  |-------------------------------------------------------------------
72  component XY_Motion
73  |   port
74  |   (
75  |         clk_input, reset                                                    : in std_logic;
76  |         X_LT, X_EQ, X_GT, motion, Y_LT, Y_EQ, Y_GT, extender_out            : in std_logic;
77  |         clk_en_X, up_down_X, error_led, capture_XY,clk_en_Y, up_down_Y, extender_en  : out std_logic
78  |
79  |   );
80  end component XY_Motion;
81  |-------------------------------------------------------------------
```

Logical Step Top Cont.

```
81  ----------------------------------------------------------------
82  component Extender
83      port
84      (
85            clk_input, reset, extender, extender_en              : in std_logic;
86            leds                                                 : in std_logic_vector (3 downto 0);
87            clk_en, shift_leftright, grappler_en, extender_out   : out std_logic
88      );
89  end component Extender;
90  ----------------------------------------------------------------
91  component Grappler
92      port
93      (
94            clk_input, reset, grappler_push, grappler_en         : in std_logic;
95            led                                                  : out std_logic
96      );
97  end component Grappler;
98  ----------------------------------------------------------------
99  component U_D_Bin_Counter4bit
100     port
101     (
102               clk            : in std_logic := '0';
103               reset          : in std_logic := '0';
104               clk_en         : in std_logic := '0';
105               up1_down0      : in std_logic := '0';
106               counter_bits   : out std_logic_vector( 3 downto 0)
107
108     );
109 end component U_D_Bin_Counter4bit;
110 ----------------------------------------------------------------
111 component registerr
112     port(
113     clk : in std_logic := '0';
114     reset : in std_logic := '0';
115     target : in std_logic_vector (3 downto 0);
116     capture_XY : in std_logic := '0';
117     position : out std_logic_vector (3 downto 0)
118 );
119
120 end component registerr;
121 ----------------------------------------------------------------
122 component Inverter
123     port
124     (
125           grappler, extender, motion, RESET                      : in std_logic;
126           grapplerout, extenderout, motionout, RESETout          : out std_logic
127     );
128
129 end component Inverter;
```

Logical Step Top Cont.

```
130 ------------------------------------------------------------------------
131 constant SIM_FLAG : boolean := TRUE; -- set to FALSE when compiling for FPGA download to LogicalStep board
132 ------------------------------------------------------------------------
133 ------------------------------------------------------------------------
134 -- All Signals
135 signal clk_in, clock : std_logic;
136 signal RESET : std_logic;
137
138
139 signal clk_en_fromext : std_logic;
140 signal left_right     : std_logic;
141 signal ext_pos        : std_logic_vector (3 downto 0);
142
143 signal X_pos : std_logic_vector (3 downto 0);
144 signal Y_pos : std_logic_vector (3 downto 0);
145
146 signal decoderout0 : std_logic_vector (6 downto 0);
147 signal decoderout1 : std_logic_vector (6 downto 0);
148
149 signal X_reg : std_logic_vector ( 3 downto 0);
150 signal Y_reg : std_logic_vector ( 3 downto 0);
151
152 signal X_GT : std_logic;
153 signal X_EQ : std_logic;
154 signal X_LT : std_logic;
155
156 signal Y_GT : std_logic;
157 signal Y_EQ : std_logic;
158 signal Y_LT : std_logic;
159
160 signal motion : std_logic;
161 signal extender_out : std_logic;
162 signal extender_en : std_logic;
163
164 signal clk_en_x : std_logic;
165 signal up_down_x: std_logic;
166
167 signal clk_en_y : std_logic;
168 signal up_down_y : std_logic;
169
170 signal error : std_logic;
171 signal Capture_XY : std_logic;
172
173 signal extender_push : std_logic;
174 signal grappler_push : std_logic;
175
176 signal grappler_en :std_logic;
177 signal grappler_on :std_logic;
```

Logical Step Top Cont.

```
179  signal X_target : std_logic_vector (3 downto 0);
180  signal Y_target :std_logic_vector (3 downto 0);
181
182  begin
183  clk_in <= clk;
184  leds(5 downto 2) <= ext_pos;
185  leds(1) <= grappler_on;
186  leds(0) <= error;
187
188
189  X_target <= sw(7 downto 4);
190  Y_target <= sw(3 downto 0);
191  ---comment for FPGA
192  xPOS <= X_pos;
193  yPOS <= Y_pos;
194  --
195  xreg <= X_reg;
196  yreg <= Y_reg;
197  --
198  ------------------------------------------------------------
199  -----clock instance-------------------------------------------
200  Clock_Selector: Clock_source port map(SIM_FLAG, clk_in, clock);
201
202  ----decoder---------------------------------------------------
203  decoder0: SevenSegment port map (X_pos, decoderout0);
204  decoder1: SevenSegment port map (Y_pos, decoderout1);
205
206  --make changes to mux cause of the 2 before 1 thing
207  mux: segment7_mux port map (clk_in, decoderout1, decoderout0, seg7_data, seg7_char2, seg7_char1);
208
209  ---bidirectional shift----------------------------------------
210  bidi_shift: Bidir_shift_reg port map (clock, RESET, clk_en_fromext, left_right, ext_pos);
211
212  ---4bitcomparator---------------------------------------------
213  fourbit0: Compx4 port map (X_pos, X_reg, X_GT, X_EQ, X_LT);
214  fourbit1: Compx4 port map (Y_pos, Y_reg, Y_GT, Y_EQ,  Y_LT);
215
216  ---------XY_motion---------------------------------------------
217  xy: XY_Motion port map (clock, RESET, X_LT, X_EQ, X_GT, motion, Y_LT, Y_EQ, Y_GT, extender_out,
218                          clk_en_x, up_down_x, error, Capture_XY, clk_en_y, up_down_y, extender_en);
219
220  ----------Extender-----------------------------------------------
221  ext: Extender port map (clock, RESET, extender_push, extender_en, ext_pos, clk_en_fromext, left_right, grappler_en, extender_out);
222
223  --------Grappler-----------------------------------------------
224  grap: Grappler port map (clock, RESET, grappler_push, grappler_en, grappler_on);
225
```

Logical Step Top Cont.

```
223   --------Grappler-----------------------------------------------------
224   grap: Grappler port map (clock, RESET, grappler_push, grappler_en, grappler_on);
225
226   --------up/down shift-----------------------------------------------
227   updown0: U_D_Bin_Counter4bit port map (clock, RESET, clk_en_x, up_down_x, X_pos);
228   updown1: U_D_Bin_Counter4bit port map (clock, RESET, clk_en_y, up_down_y, Y_pos);
229
230   ---------Registers-----------------------------------------------------
231   reg0: registerr port map(clock, RESET, X_target, Capture_XY, X_reg);
232   reg1: registerr port map(clock, RESET, Y_target, Capture_XY, Y_reg);
233
234   ---------Inverter------------------------------------------------------
235   inv: Inverter port map (pb_n(0), pb_n(1), pb_n(2), '1', grappler_push, extender_push, motion, RESET);
236
237
238   end Circuit;
239
```

XY_Motion VHDL

```vhdl
----XY_Motion

--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------------

entity XY_Motion is
      port (
               clk_input, reset                                                    : in std_logic;
               X_LT, X_EQ, X_GT, motion, Y_LT, Y_EQ, Y_GT, extender_out            : in std_logic;
               clk_en_X, up_down_X, error_led, capture_XY,clk_en_Y, up_down_Y, extender_en   : out std_logic

        );
end entity;

architecture sm of XY_Motion is


 type state_names is (initial, button_down, moving, error);   -- list all the STATE_NAMES values


 signal current_state, next_state   :  state_names;      -- signals of type STATE_NAMES


----------------------------------------------------------------------
--State Machine
----------------------------------------------------------------------
begin
-- Register_Logic Process:

Register_Section: process (clk_input, reset, next_state)  -- this process synchronizes the activity to a clock
begin
   if (reset = '1') then
      current_state <= initial;
   elsif(rising_edge(clk_input)) then
      current_state <= next_State;
   end if;
end process;


```

XY_Motion VHDL Cont.

```
42
43  --Transition_Logic Process:
44  Transtion_Section : process (current_state, X_LT, X_EQ, X_GT, motion, Y_LT, Y_EQ, Y_GT, extender_out )
45  begin
46     case current_state is
47
48
49           --nothing is happening at the start
50           when initial =>
51                              if (motion = '1' and extender_out = '1') then
52                                 next_state <= error; --trying to move while the extender is out resutls in error
53
54                              elsif( motion = '1') then
55                                 next_state <= button_down;
56
57                              else
58                               next_state <= initial;
59
60                              end if;
61
62           when error =>
63                              if (extender_out = '0') then
64                                 next_state <= initial; --releasing the extender will get out of error state
65                              else
66                                 next_state <= error;
67                              end if;
68
69
70           when button_down =>
71                              if (motion = '0') then
72                                 next_state <= moving; --releasing the motion will move the bits
73                              else
74                                 next_state <= button_down;
75                              end if;
76
77
78           when moving =>
79                              if (X_EQ = '1' and Y_EQ = '1') then
80                                 next_state <= initial; --stop moving and go back to start if the values are at target values
81                              else
82                                 next_state <= moving;
83                              end if;
84
85
```

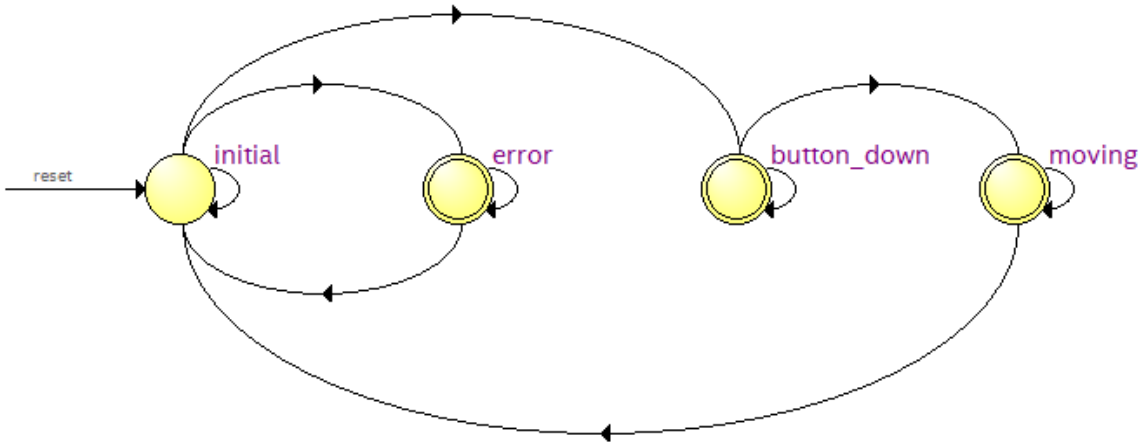
XY_Motion VHDL Cont.

```
92  --Decoder_Logic Process:
93  Decoder_Section: process (current_state, X_LT, X_EQ, X_GT, motion, Y_LT, Y_EQ, Y_GT, extender_out, clk_input)
94  begin
95     case current_state is
96
97                          when initial =>
98                                          extender_en <= X_EQ and Y_EQ; --can only move the extender when the values are at target
99                                          clk_en_x <= '0';
100                                         clk_en_y <= '0';
101                                         capture_XY <= '0';
102                                         error_led <= '0';
103
104                         when error =>
105                                         error_led <= clk_input;     --flashing purposes
106
107                         when button_down =>
108
109                                         capture_XY <= '1'; --capture x and y values when the button is pressed
110
111                         when moving =>
112                                         clk_en_x <= X_LT or X_GT; --counters only activate if the values need to move to target
113                                         clk_en_y <= Y_LT or Y_GT;
114                                         extender_en <= '0';
115                                         capture_XY <= '0';
116
117                                     ----for the x motion
118                                        if (X_LT = '1') then
119                                           up_down_x <= '1'; ---x coordinate is less than target so must bring it up
120
121                                        elsif (X_EQ = '1') then
122                                           clk_en_x <= '0'; -- x is t target so we can disable the counter
123
124                                        elsif (X_GT = '1') then
125                                           up_down_x <= '0'; -- x is higher than target, so we must bring it down
126
127                                        end if;
```

XY_Motion VHDL Cont.

```
129
130                                 --for the y motion, bring it down if the y coordinate is too high and up if too low and disbale counter if at target
131                                 if (Y_LT = '1') then
132                                    up_down_y <= '1';
133
134                                 elsif (Y_EQ = '1') then
135                                    clk_en_y <= '0';
136
137                                 elsif (Y_GT = '1') then
138                                    up_down_y <= '0';
139
140                                 end if;
141
142    end case;
143 end process;
144
145 end architecture sm;
146
```

XY_Motion State Machine

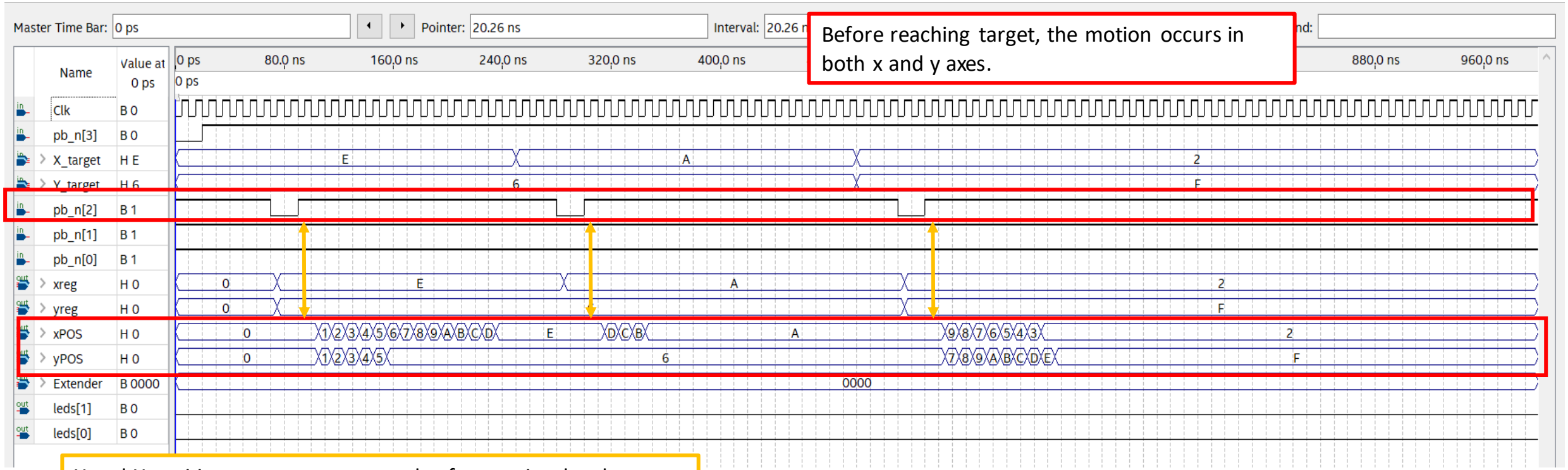


State Table

	Source State	Destination State	Condition
1	button_down	button_down	(motion)
2	button_down	moving	(!motion)
3	error	error	(extender_out)
4	error	initial	(!extender_out)
5	initial	button_down	(motion).(!Transtion_Section)

Transitions | Encoding

XY_Motion Simulation (Sim 1)



Before reaching target, the motion occurs in both x and y axes.

X and Y positions move to target only after motion has been pressed and released. X and Y both reach target values.

Extender VHDL File

```vhdl
--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------------

entity Extender is
      port (
                clk_input, reset, extender, extender_en              : in std_logic;
                leds                                                 : in std_logic_vector (3 downto 0);
                clk_en, shift_leftright, grappler_en, extender_out   : out std_logic

         );
end entity;

architecture statemachine of Extender is


 type state_names is (extending, fully_extended, extender_pause, retracter_pause, retracting, fully_retracted);   -- list all the STATE_NAMES values


 signal current_state, next_state   :  state_names;      -- signals of type STATE_NAMES

begin
----------------------------------------------------------------------
--State Machine
----------------------------------------------------------------------

-- Register_Logic Process:

Register_Section: process (clk_input, reset, next_state)  -- this process synchronizes the activity to a clock
begin
   if (reset = '1') then
      current_state <= fullY_retracted;
   elsif(rising_edge(clk_input)) then
      current_state <= next_State;
   end if;
end process;
```

Extender VHDL File Cont.

```
42  -- Transition_Logic Process:
43  Transition_Section: process (extender, extender_en, leds, current_state)
44
45  begin
46     case current_state is
47
48
49           when fully_retracted =>   -- extender button and extender_en enabled allows for extender to pause else retract
50                            if ( extender = '1' and extender_en = '1') then
51                               next_state <= extender_pause;
52                            else
53                               next_state <= fully_retracted;
54                            end if;
55
56           when extender_pause =>   -- when the button is released, extender is disabled and the extender extends otherwise it remains in same position
57                             if(extender = '0') then
58                               next_state <= extending;
59                            else
60                               next_state <= extender_pause;
61                            end if;
62
63           when extending =>  -- when fully extended, move to fully_extended state else continue extending
64                            if (leds = "1111") then
65                               next_state <= fully_extended;
66                            else
67                               next_state <= extending;
68                            end if;
69
70
71           when fully_extended =>  -- extender button and extender_en enabled allows for arm to remain still otherwise fully extend
72                            if (extender = '1' and extender_en = '1') then
73                               next_state <= retracter_pause;
74                            else
75                               next_state <= fully_extended;
76                            end if;
77
78           when retracter_pause => -- extender button disabled allows for extender to retract otherwise extender remains still
79                             if(extender = '0') then
80                               next_state <= retracting;
81                            else
82                               next_state <= retracter_pause;
83                            end if;
```

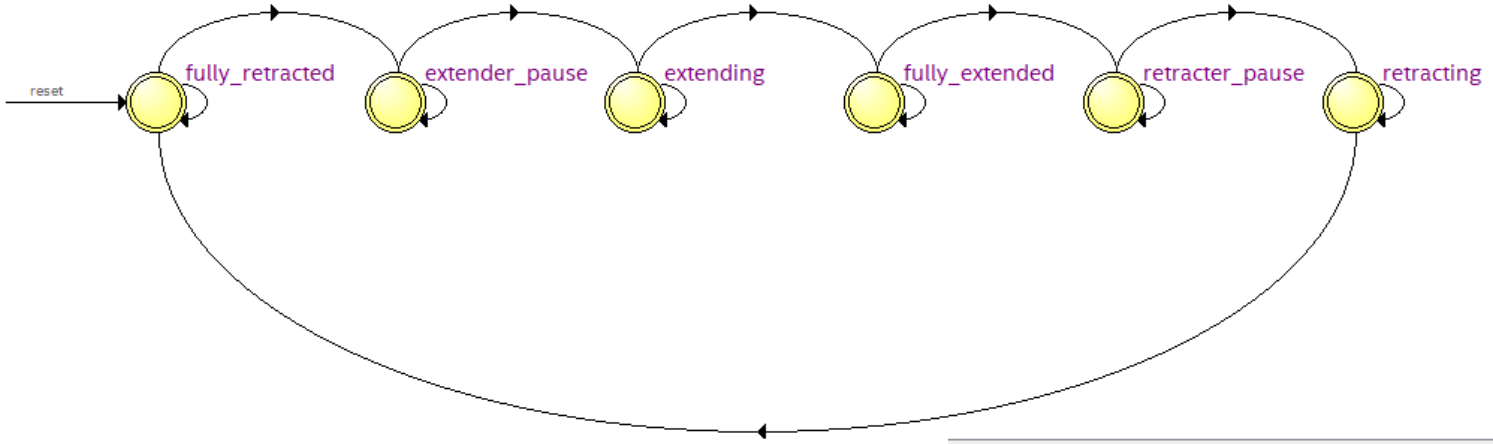
Extender VHDL File Cont.

```
83                              end if;
84
85              when retracting => -- when fully retracted, move to fully_retracted state else continue retracting
86                              if (leds = "0000") then
87                                 next_state <= fully_retracted;
88                              else
89                                 next_state <= retracting;
90                              end if;
91
92
93
94          end case;
95      end process;
96
97
98      -- Decoder_Logic Process:
99      Decoder_Section: process (current_state)
100
101     begin
102         case current_state is
103
104             when fully_retracted => --when fully retracted, no action can occur
105                                     clk_en <= '0';
106                                     shift_leftright <= '0';
107                                     grappler_en <= '0';
108                                     extender_out <= '0';
109
110             when extender_pause =>  --when paused, no action can occur
111                                     clk_en <= '0';
112                                     shift_leftright <= '0';
113                                     grappler_en <= '0';
114                                     extender_out <= '0';
115
116
117             when extending =>       --when extending, everything except the grappler is enabled
118                                     clk_en <= '1';
119                                     shift_leftright <= '1';
120                                     grappler_en <= '0';
121                                     extender_out <= '1';
122
123
```

Extender VHDL File Cont.

```
124
125             when fully_extended =>  --when fully retracted, grappler and extender will be fully extended
126                                     clk_en <= '0';
127                                     shift_leftright <= '0';
128                                     grappler_en <= '1';
129                                     extender_out <= '1';
130
131
132             when retracter_pause => --when fully extended, the extender will be out and grappler will be enabled
133                                     clk_en <= '0';
134                                     shift_leftright <= '0';
135                                     grappler_en <= '1';
136                                     extender_out <= '1';
137
138             when retracting =>      --when retracting, grappler, left and right shift will be off
139                                     clk_en <= '1';
140                                     shift_leftright <= '0';
141                                     grappler_en <= '0';
142                                     extender_out <= '1';
143             when others =>
144                                     grappler_en <= '0';
145
146
147
148
149         end case;
150     end process;
151
152     end statemachine;
153
```

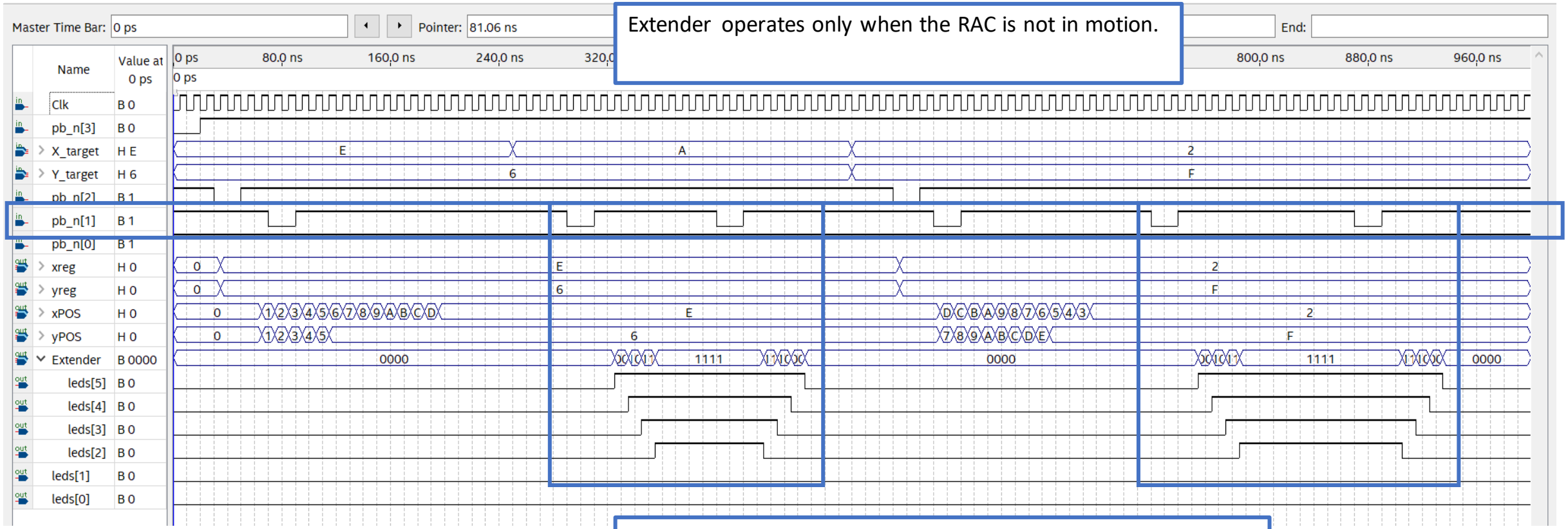
Extender State Machine



	Source State	Destination State	Condition
1	extender_pause	extender_pause	(extender)
2	extender_pause	extending	(!extender)
3	extending	extending	(!leds[0]) + (leds[0]).(!leds[1]) + (leds[0]).(leds[1]).(!leds[2]) + (leds[0]).(leds[1]).(leds[2]).(!leds[3])
4	extending	fully_extended	(leds[0]).(leds[1]).(leds[2]).(leds[3])
5	fully_extended	fully_extended	(!Transition_Section)

Transitions | Encoding

Extender Simulation (Sim 2)



Extender operates only when the RAC is not in motion.

Extender completes expansion and retraction on its own after extender has been pressed and released

Grappler VHDL File

```
1   --Author: Group 3, Nandita Lohani, Puneet Bhullar
2   library ieee;
3   use ieee.std_logic_1164.all;
4   use ieee.numeric_std.all;
5   ----------------------------------------------------------------------
6
7   entity Grappler is
8         port (
9                   clk_input, reset, grappler_push, grappler_en            : in std_logic;
10                  led                                                     : out std_logic
11
12          );
13  end entity;
14
15
16  architecture sm of Grappler is
17
18
19   type state_names is (closed, openn, opening, closing);        -- list all the STATE_NAMES values
20   signal current_state, next_state   :  state_names;      -- signals of type STATE_NAMES
21
22  begin
23  ----------------------------------------------------------------------
24  --State Machine
25  ----------------------------------------------------------------------
26
27  -- Register_Logic Process:
28
29  Register_Section: process (clk_input, reset, next_state)  -- this process synchronizes the activity to a clock
30  begin
31     if (reset = '1') then
32        current_state <= closed;
33     elsif(rising_edge(clk_input)) then
34        current_state <= next_State;
35     end if;
36  end process;
```

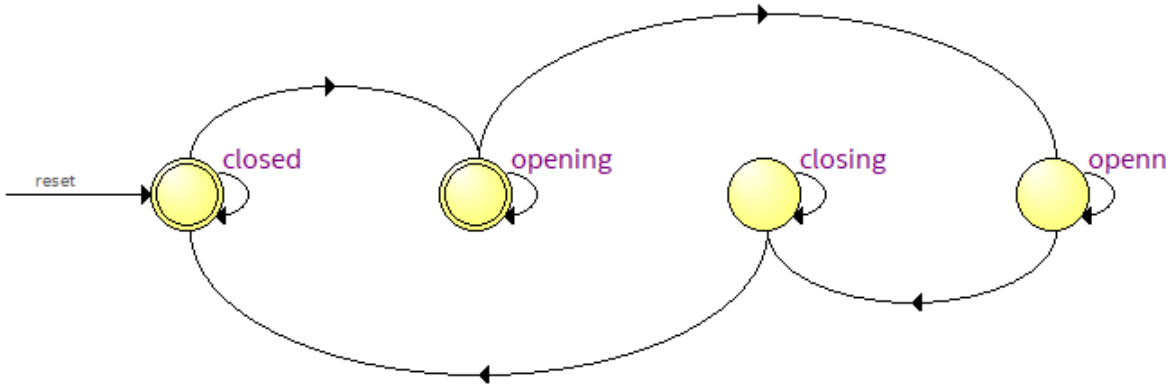
Grappler VHDL File Cont.

```
38  --Transition_Logic Process:
39  Transtion_Section : process (grappler_push, grappler_en, current_state)
40  begin
41     case current_state is
42
43           when closed =>     -- when grappler push-button and grappler_en enabled, the grapler can begin opening otherwise remain closed
44                              if ( grappler_push = '1' and grappler_en = '1') then
45                                 next_state <= opening;
46                              else
47                                 next_state <= closed;
48                              end if;
49
50           when opening =>    -- when grappler push-button is disabled, the grapler is fully open otherwise it is in the process of opening
51                              if (grappler_push = '0') then
52                                 next_state <= openn;
53                              else
54                                 next_state <= opening;
55                              end if;
56
57           when openn =>     -- when grappler push-button and grappler_en enabled, the grapler is in the process of closing otherwise it is fully open
58                              if ( grappler_push = '1' and grappler_en = '1') then
59                                 next_state <= closing;
60                              else
61                                 next_state <= openn;
62                              end if;
63
64           when closing =>    -- when grappler push-button is disabled, the grapler is fully closed otherwise it is in the process of closing
65                              if (grappler_push = '0') then
66                                 next_state <= closed;
67                              else
68                                 next_state <= closing;
69                              end if;
70
71
72     end case;
73  end process;
```

Grappler VHDL File Cont.

```
75  --Decoder_Logic Process:
76  Decoder_Section: process (current_state)
77  begin
78     case current_state is
79           when closed =>     --when grappler closed, led is off
80                              led <= '0';
81           when opening =>    --when grappler is in the process of opening, led is off
82                              led <= '0';
83           when openn =>      --when grappler fully open, led is on
84                              led <= '1';
85           when closing =>    --when grappler is in the process of closing, led is on
86                              led <= '1';
87     end case;
88  end process;
89
90  end architecture sm;
91
```

Grappler State Machine

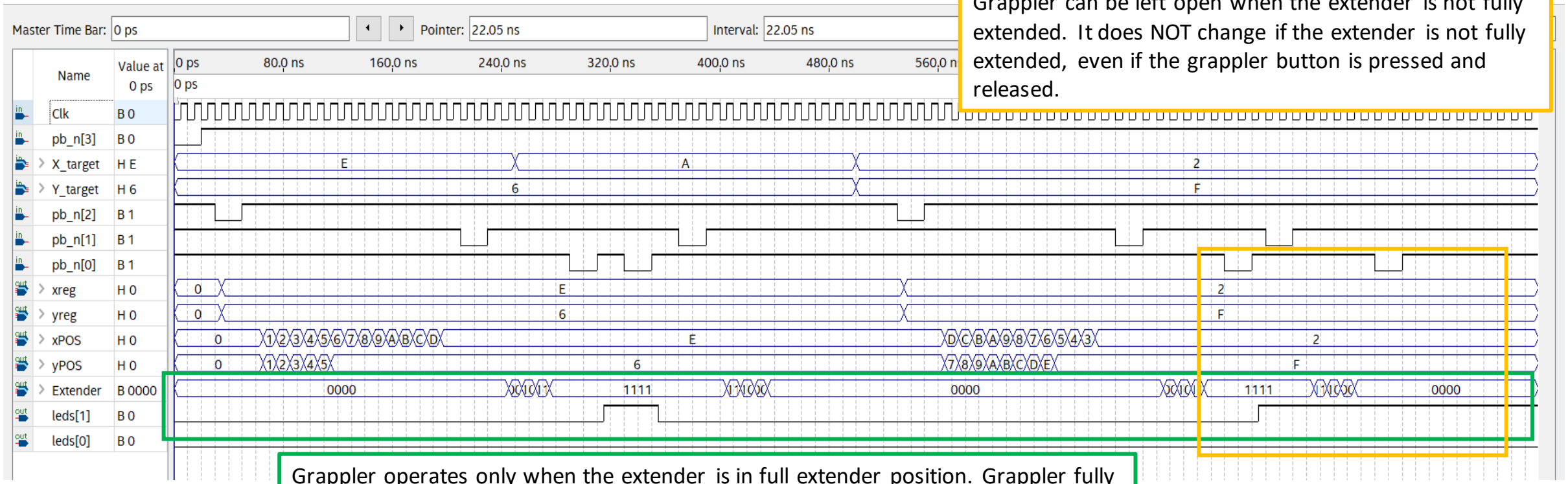


State Table

	Source State	Destination State	Condition
1	closed	closed	(!Transtion_Section)
2	closed	opening	(Transtion_Section)
3	closing	closed	(!grappler_push)
4	closing	closing	(grappler_push)
5	opening	openn	(!grappler_push)

Transitions | Encoding

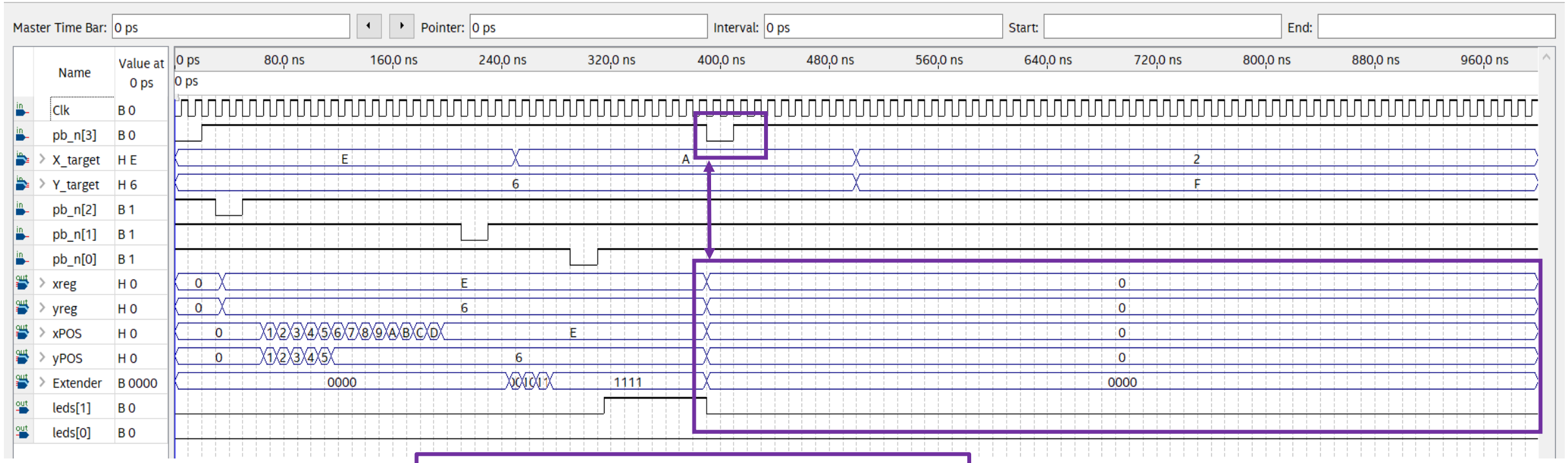
Grappler Simulation (Sim 3)



Grappler can be left open when the extender is not fully extended. It does NOT change if the extender is not fully extended, even if the grappler button is pressed and released.

Grappler operates only when the extender is in full extender position. Grappler fully opens and closes after a single press and release of the grappler button

Reset Simulation (Sim 4)



All registers are reset when the reset button (pb_n(3)) is active

Bidirectional Shift VHDL File

```vhdl
--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------
entity Bidir_shift_reg is
      port (
                clk             : in std_logic := '0';
                reset           : in std_logic := '0';
                clk_en          : in std_logic := '0';
                left0_right1    : in std_logic := '0';
                reg_bits        : out std_logic_vector( 3 downto 0)

        );
end entity;

architecture one of Bidir_shift_reg is

signal sreg : std_logic_vector(3 downto 0);

begin

process(clk, reset, clk_en, left0_right1) is
begin
   if (reset = '1') then
            sreg <= "0000";

   elsif (rising_edge(clk) and (clk_en = '1')) then

         if (left0_right1 = '1') then --if right shift is on

            sreg (3 downto 0) <= '1' & sreg (3 downto 1);  -- right bit shifts

         elsif (left0_right1 = '0') then

            sreg (3 downto 0) <= sreg(2 downto 0) & '0'; -- left bit shifts

         end if;

   end if;
   reg_bits <= sreg;

end process;

end architecture one;
```

U_D_Bin_Counter4bit VHDL File

```vhdl
--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------
entity U_D_Bin_Counter4bit is
      port (
                clk             : in std_logic := '0';
                reset           : in std_logic := '0';
                clk_en          : in std_logic := '0';
                up1_down0       : in std_logic := '0';
                counter_bits    : out std_logic_vector( 3 downto 0)

         );
end entity;

architecture one of U_D_Bin_Counter4bit is

signal ud_bin_counter : unsigned(3 downto 0);

begin

process(clk, reset, clk_en, up1_down0) is
begin
   if (reset = '1') then
            ud_bin_counter <= "0000";

   elsif (rising_edge(clk)) then
```

U_D_Bin_Counter4bit VHDL File Cont.

```
29
30          if ((up1_down0 = '1') and (clk_en = '1')) then
31             ud_bin_counter <= (ud_bin_counter + 1);
32
33          elsif ((up1_down0 = '0') and (clk_en = '1')) then
34             ud_bin_counter <= (ud_bin_counter - 1);
35
36          end if;
37
38       end if;
39
40    end process;
41
42
43    counter_bits <= std_logic_vector(ud_bin_counter);
44
45    end one;
```

Register VHDL File

```vhdl
--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------------
entity registerr is port(
   clk : in std_logic := '0';
   reset : in std_logic := '0';
   target : in std_logic_vector (3 downto 0);
   capture_XY : in std_logic := '0';
   position : out std_logic_vector (3 downto 0)
);

end entity;

architecture arch of registerr is
begin

registerr1:
process (clk, reset, capture_XY) is
begin
      if (capture_XY = '1') then -- holding target value in position
         position <= target;

      elsif (reset = '1') then -- resseting position values
         position <= "0000";

      end if;

end process;

end architecture arch;
```

Inverter VHDL File

```vhdl
--Author: Group 3, Nandita Lohani, Puneet Bhullar
library ieee;
use ieee.std_logic_1164.all;
library work;
----------------------------------------------------------------------------
entity Inverter is
   port (
         grappler, extender, motion, RESET                    : in std_logic;
         grapplerout, extenderout, motionout, RESETout        : out std_logic
         );

end entity Inverter;
----------------------------------------------------------------------------
architecture inverter_logic of Inverter is

begin

--- Ouputs with correct boolean equations using input operands

grapplerout <= not grappler;
extenderout <= not extender;
motionout <= not motion;
RESETout <= not RESET;

end architecture inverter_logic;
```

Compx4 File

```
1  --Author: Group 3, Nandita Lohani, Puneet Bhullar
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.numeric_std.all;
5  ----------------------------------------------------------------------------
6  entity Compx4 is
7     port (
8               A, B : in std_logic_vector(3 downto 0);
9               AGTB_out, AEQB_out, ALTB_out : out std_logic
10         );
11 end Compx4;
12
13 -- ***************************************************************************
14 architecture Four_Bit_Comparator of Compx4 is
15
16 component Compx1
17    port (
18          A, B : in std_logic;
19          AgreatB, AequalB, AlessB : out std_logic
20          );
21 end component;
22
23 ----------------------------------------------------------------
24 signal AGTB, AEQB, ALTB : std_logic_vector (3 downto 0);
25 begin
26
27 -- Outputs expressed as boolean equations of operand inputs
28
29 AGTB_out <= ((AGTB(3)) or
30              (AEQB(3) and AGTB(2)) or
31              (AEQB(3) and AEQB(2) and AGTB(1)) or
32              (AEQB(3) and AEQB(2) and AEQB(1) and AGTB(0))
33          );
```

Compx4 VHDL File Cont.

```
34
35  AEQB_out <= (AEQB(3) and AEQB(2) and AEQB(1) and AEQB(0));
36
37
38  ALTB_out <= ((ALTB(3)) or
39                (AEQB(3) and ALTB(2)) or
40                (AEQB(3) and AEQB(2) and ALTB(1)) or
41                (AEQB(3) and AEQB(2) and AEQB(1) and ALTB(0))
42           );
43  ------instantiation of four single bit comparators----------------------------
44
45  inst1: Compx1 port map (A(3), B(3), AGTB(3), AEQB(3), ALTB(3));
46  inst2: Compx1 port map (A(2), B(2), AGTB(2), AEQB(2), ALTB(2));
47  inst3: Compx1 port map (A(1), B(1), AGTB(1), AEQB(1), ALTB(1));
48  inst4: Compx1 port map (A(0), B(0), AGTB(0), AEQB(0), ALTB(0));
49
50  end architecture Four_Bit_Comparator;
51  ----------------------------------------------------------------
52
53
```