```vhdl

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY LogicalStep_Lab4_top IS
   PORT
   (
   clkin_50    : in  std_logic;
   rst_n       : in  std_logic;
   pb          : in  std_logic_vector(3 downto 0);
   sw          : in  std_logic_vector(7 downto 0); -- The switch inputs
   leds        : out std_logic_vector(7 downto 0); -- for displaying the switch content
   seg7_data   : out std_logic_vector(6 downto 0); -- 7-bit outputs to a 7-segment
   seg7_char1  : out std_logic;                    -- seg7 digi selectors
   seg7_char2  : out std_logic                     -- seg7 digi selectors
   );
END LogicalStep_Lab4_top;

ARCHITECTURE SimpleCircuit OF LogicalStep_Lab4_top IS


   -- Signals used are declared here
--------------------------------------------------------------------------------------------------
   CONSTANT SIM                     :  boolean := FALSE;    -- set to TRUE for simulation runs otherwise keep at 0.
   CONSTANT CLK_DIV_SIZE            :  INTEGER := 26;    -- size of vectors for the counters

   SIGNAL   Main_CLK                :  STD_LOGIC;        -- main clock to drive sequencing of State Machine

   SIGNAL   bin_counter             :  UNSIGNED(CLK_DIV_SIZE-1 downto 0); -- := to_unsigned(0,CLK_DIV_SIZE); -- reset binary counter to zero

   signal seg7_A     : std_logic_vector(6 downto 0);
   signal seg7_B     : std_logic_vector(6 downto 0);
   signal Digit1     : std_logic_vector(6 downto 0);
   signal Digit2     : std_logic_vector(6 downto 0);

   signal shift_enable        : std_logic;
   signal shift_dirout        : std_logic;
   signal Extender_leds       : std_logic_vector(3 downto 0);
   signal Grapple_en          : std_logic;
   signal Extend_out          : std_logic;
   signal EXT_EN              : std_logic;
   signal Extend              : std_logic;
   signal Grapple             : std_logic;
   signal G_led               : std_logic;
   signal Grapple_tog         : std_logic;
   signal Extender_tog        : std_logic;

   signal X_TARGET            : std_logic_vector(3 downto 0);
   signal X_CURRENT           : std_logic_vector(3 downto 0) := "0000";
   signal X_DRIVE             : std_logic;
   signal X_OUT               : std_logic_vector(3 downto 0);
   signal X_Up1_Down0         : std_logic;
   signal X_gt                : std_logic;
   signal X_eq                : std_logic;
   signal X_lt                : std_logic;
   signal X_count             : std_logic;
   signal X_ERROR_led         : std_logic;

   signal Y_TARGET            : std_logic_vector(3 downto 0);
   signal Y_CURRENT           : std_logic_vector(3 downto 0) := "0000";
   signal Y_OUT               : std_logic_vector(3 downto 0);
   signal Y_DRIVE             : std_logic;
   signal Y_Up1_Down0         : std_logic;
   signal Y_gt                : std_logic;
   signal Y_eq                : std_logic;
   signal Y_lt                : std_logic;
   signal Y_count             : std_logic;
   signal Y_ERROR_led         : std_logic;

   signal ERR_SIG             : std_logic;
-------------------------------------------------------------------------------------------
```

```vhdl
--------
-- Components Used are declared here:

component Bidir_shift_reg port
   (
         CLK               : in std_logic := '0';
         RESET_n           : 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 component Bidir_shift_reg;

component U_D_Bin_Counter4bit port
   (
         CLK               : in std_logic := '0';
         RESET_n           : 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 component U_D_Bin_Counter4bit;

component Compx4 port
   (
         inp_A             : in std_logic_vector(3 downto 0);
         inp_B             : in std_logic_vector(3 downto 0);
         AGTB              : out std_logic;
         AEQB              : out std_logic;
         ALTB              : out std_logic
   );
   end component Compx4;

component SevenSegment port
   (
         hex               :  in  std_logic_vector(3 downto 0);
         sevenseg          :  out std_logic_vector(6 downto 0)
   );
   end component SevenSegment;

component segment7_mux port
   (
         clk         : in  std_logic := '0';
         DIN2        : in  std_logic_vector(6 downto 0);
         DIN1        : in  std_logic_vector(6 downto 0);
         DOUT        : out  std_logic_vector(6 downto 0);
         DIG2        : out  std_logic;
         DIG1        : out  std_logic
    );
   end component segment7_mux;

component MEALY_SM PORT (
         clk_input,X_MOTION, Y_MOTION, Extender_Out,X_EQ,X_GT,X_LT,Y_EQ,Y_GT,Y_LT  : in std_logic;
         rst_n                                                                      : in std_logic := '0';
         clk_en_X,clk_en_Y, Xcount,Ycount,ERROR_led,Extender_en                     : out std_logic
          );
end component MEALY_SM;

component MOORE_SM Port
(
         clk_input, rst_n, Extender_en, Extend_tog                 : in std_logic;
         leds                                                      : in std_logic_vector(3 downto 0);
         shift_reg_en, shift_reg_dir, Extender_out, Grappler_en    : out std_logic
 );
end component MOORE_SM;

component MOORE_SM2 Port
(
         clk_input, rst_n, button, enable       : in std_logic;
         led                                    : out std_logic
 );
 end component MOORE_SM2;
```

```

component Error_SM port (
         clk_input      : in  std_logic;
         rst_n          : in  std_logic;
         INPUT1         : in  std_logic_vector(6 downto 0); --when button 3 pressed
         INPUT2         : in  std_logic_vector(6 downto 0);
         Selector       : in  std_logic;
         OUTPUT         : out std_logic_vector(6 downto 0)
 );
end component Error_SM;

BEGIN

-- CLOCKING GENERATOR WHICH DIVIDES THE INPUT CLOCK DOWN TO A LOWER FREQUENCY

BinCLK: PROCESS(clkin_50, rst_n) is
   BEGIN
      IF (rising_edge(clkin_50)) THEN -- binary counter increments on rising clock edge
         bin_counter <= bin_counter + 1;
      END IF;
   END PROCESS;

Clock_Source:
            Main_clk <=
            clkin_50 when sim = TRUE else          -- for simulations only
            std_logic(bin_counter(23));                        -- for real FPGA operation

----------------------------------------------------------------------------------------------------

-- Switch Inputs for Target --
   Y_TARGET <= sw(3 downto 0);
   X_TARGET <= sw(7 downto 4);

-- Inputs from the push buttons
   X_DRIVE <= pb(3);      -- X-Motion enable
   Y_DRIVE <= pb(2);      -- Y-Motion enable
   Extender_tog <= pb(1);-- Extender toggle
   Grapple_tog <= pb(0); -- Grapple toggle

   leds(3) <= G_led;   -- led(3) for displaying whether grappler is activated or not

   leds(0) <= ERR_SIG; -- led(0) for giving the error signal

   -- leds, used to show how much the extender is extended
   leds(7) <= Extender_leds(3);
   leds(6) <= Extender_leds(2);
   leds(5) <= Extender_leds(1);
   leds(4) <= Extender_leds(0);


   -- Depending on whether the push button is pressed or not, it determined the output to didplay on the digits
   with X_DRIVE select
      X_OUT <= X_CURRENT when '0',
               X_TARGET  when '1';
   with Y_DRIVE select
      Y_OUT <= Y_CURRENT when '0',
               Y_TARGET  when '1';

   INST1: U_D_Bin_Counter4bit port map (Main_CLK,rst_n,X_count,X_Up1_Down0,X_CURRENT);  --Up/Down Binary Counter for deciding whether to go up or down for X coordinates
   INST2: Compx4 port map (X_CURRENT(3 downto 0), sw(7 downto 4), X_gt,X_eq,X_lt); --this 4bit comparator compares the current X coordinates with the targeted one

   INST3: U_D_Bin_Counter4bit port map (Main_CLK,rst_n,Y_count,Y_Up1_Down0,Y_CURRENT);  --Up/Down Binary Counter for deciding whether to go up or down for X coordinates
   INST4: Compx4 port map (Y_CURRENT(3 downto 0),sw(3 downto 0),Y_gt,Y_eq,Y_lt); --this 4bit comparator compares the current Y coordinates with the targeted one

   -- Mealy state machine for changing the X-coordinates and Y coordinates on the digits using comparator and Up/Down Binary Counter
   INST5: MEALY_SM port map (Main_CLK,X_DRIVE,Y_DRIVE,Extend_out,X_eq,X_gt,X_lt,Y_eq,Y_gt,Y_lt,rst_n,X_count,Y_count,X_Up1_Down0,Y_Up1_Down0,ERR_SIG,EXT_EN);

   INST6: MOORE_SM port map(Main_CLK,rst_n,EXT_EN,Extender_tog,Extender_leds,shift_enable,
```

```vhdl
shift_dirout,Extend_out,Grapple_en);  -- Moore state machine for extender
   INST7: Bidir_shift_reg port map(Main_CLK,rst_n,shift_enable,shift_dirout,Extender_leds);  -- Bidirectional shift register used for the extender to shift the leds

   INST8: MOORE_SM2 port map (Main_CLK, rst_n, Grapple_tog, Grapple_en, G_led);   -- Moore state machine for grappler

   INST9: Error_SM port map (Main_CLK, rst_n, "0000000", seg7_A, ERR_SIG, Digit1);   -- Error State Machine for dispaying error on the digits
   INST10: Error_SM port map (Main_CLK, rst_n, "0000000", seg7_B, ERR_SIG, Digit2);

   INST11: SevenSegment port map(X_OUT, seg7_A);   -- 7 Segment Display
   INST12: SevenSegment port map(Y_OUT, seg7_B);

   INST13: segment7_mux port map(clkin_50, Digit1, Digit2, seg7_data, seg7_char1, seg7_char2);  -- For displaying the current or target coordinates on the digits 1 and 2

END SimpleCircuit;
```