Up/Down Counter

Tiago Piai / ASU ID: 1209830918

EEE 333 - Hardware Design and Programmable Logic

Arizona State University

October 20, 2015

Abstract

For this lab was developed an up/down counter. The requirements of the project were that the counter goes from 0000 to FFFF on up mode and FFFF to 0000 on down mode, if the up/down switch is changed to down mode, for example, the count should continue from the value that it was before the switch was changed. Also was needed to implement a reset button, that return the count to 0000, and a pause button that pauses the counting.

In this design is possible to see concurrent and sequential logic. It contains, besides the counter, a component to generate slower clocks and a component to convert the binary data to the seven segment display.

1 Introduction

The combinational logic is the logic designed in a way that the output only depends on the current input to operate. On the other side, the sequential logic could depend on the current input but also depends on a previous state.

The sequential statements are only allowed inside PROCESS, FUNCTIONS or PROCEDURES. Some example of sequential statements are: IF, CASE and WAIT. A generic sequential machine could be seen in the Figure 1:

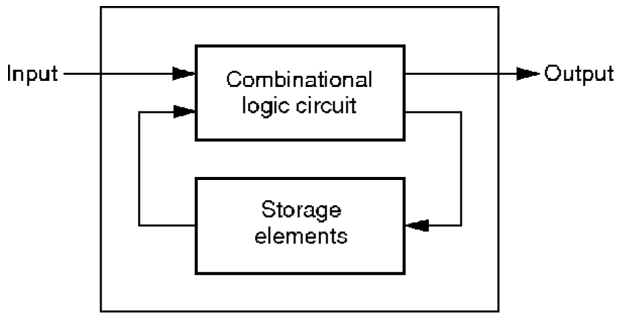

Figure 1: Block Diagram for Sequential Logic

As is possible to see the logic is driven based not only in the input but also in the previous element stored. Sequential circuits could be used on counters and finite state machines for example.

A synchronous logic is a logic dependent on a clock edge to trigger an event. In a sequential synchronous machine the clock is used to trigger the flip-flops all at the same time, at regular intervals. This flip-flops are generally used as the memory elements of the circuit and triggered when needed to check the previous state.

2 Design Explanation

This design contains three components: one responsible for operating the counter, one to divide the clock and another to convert the data of the counter to seven segment form. In the main file, where all the components were called, a logic was implemented to make possible to change the display that are being accessed in the FPGA board.

An up/down counter could easily be implemented in hardware using JK flip-flops with AND, NOT and OR gates. The JK flip-flops characterize the sequential part of the circuit that communicates with the logic gates which are the concurrent part of the implementation. In the following figure is shown one possible design of a simple up/down counter:

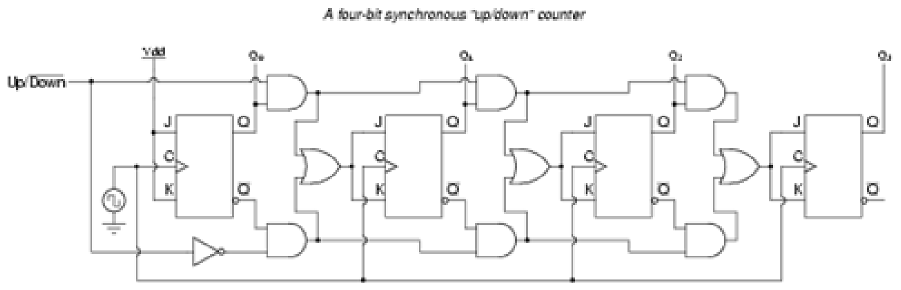

Figure 2: Up/Down Counter Hardware

In VHDL a counter could be implemented in a faster way. By analyzing the inputs, verifying if the reset button and the pause button are not set, the counting can begin by incrementing or decrementing a signal. If any of the previous inputs change the behavior of the output will change, otherwise the counting will continue.

By implementing the counter on the FPGA board there are some requirements that must be match: the clock and the display configuration.

In this board the seven segment displays works based on an input that turns on and off the specific display. This input has four bits, one for each display, which is activated by sending 0 to the desired display. For example, to access the second display the input must receive 1101. A decoder could be placed in order to refresh the states of the display. The inputs of the decoder must be changed in a high frequency, but not too high as the 50MHz provided by the board.

To reduce the board frequency is necessary to design a clock divider. In this specific case the clock was reduced to two different frequencies: one faster for the displays and another slower to the counting. A clock divider could be seen in hardware as a Toggle Flip-Flop and it is implemented in the same way in VHDL. For this project was used a clock of 24414.06 Hz for the displays and a clock of 2.98 Hz for the counting.

In order to have a better understanding of the design of the counter the following diagram was made:

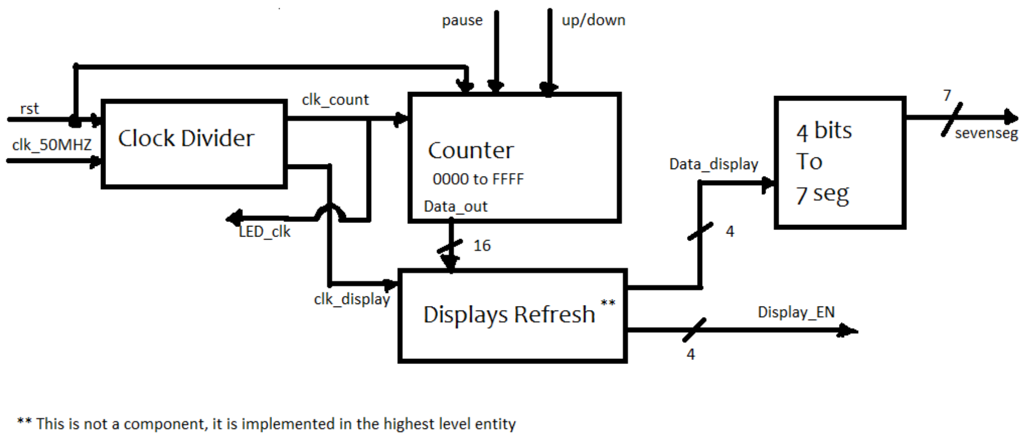

Figure 3: Diagram Block for the Up/Down Counter

3 Implementation

The inputs of the project were: clkin (50 MHz clock), rst (reset switch), pause (pause switch) and up_down (switch - '1' counts up, '0' counts down) all STD_LOGIC. The outputs were: sevenseg (7 bits STD_LOGIC_VECTOR that communicates with the displays), Display_EN (4 bits STD_LOGIC_VECTOR that selects the proper display) and LED_clk (LED that shows the frequency of the counting). All this inputs and outputs were connected to the board hardware and the type of file that make this connections is the UCF file, where is specified which signal is connected to which hardware.

The counter component contains as input signals: clk, up_down, rst and pause. The up_down, rst and pause are the same of the highest entity mentioned above. The clock in this input is the slowest of the clocks specified in the Design Explanation. The output of the counter is an STD_LOGIC_VECTOR with 16 bits called Data_out. The program first checks for the reset switch and then for the pause switch, if both are on zero it will check if the counting should be up or down, if up it increments an INTEGER signal Data_count that goes from 0 to 65535, if down it decrements the current number. The flip-flops used to store the Data_count are the D-type. In the end the current value of Data_count is converted to a 16 bits STD_LOGIC_VECTOR and send to the Data_out.

In the highest level entity was implemented a two bits decoder that increments every time the clock of the display has a rising edge. This clock is fast enough that it can switch all the displays and alter its values in a way that human eyes cannot detect that they are turned off. To alter the displays values was used the Data_out of the counter and for each display a signal called Data_seg receive the specific four bits of the Data_out to display the proper digit.

This Data_seg signal is send to the Hex_Display component, which converts the value of the four bits in the proper value for seven segments and send directly to the display that is being selected in the highest level entity by the Display_EN output. This component outputs the sevseg bits to the proper points described in the UCF file.

4 Results

For this test bench was made two codes, one to show the four bits conversion to the seven segments working and the other that shows the highest level entity outputting the sixteen bits and not coded to the seven segment. For the test bench the clock divider was removed and the clocks were generated manually.

In the Figure 4 is possible to the display conversion four bits input and the seven bits output, meaning that '0' in the sevenseg is ON and '1' is OFF:

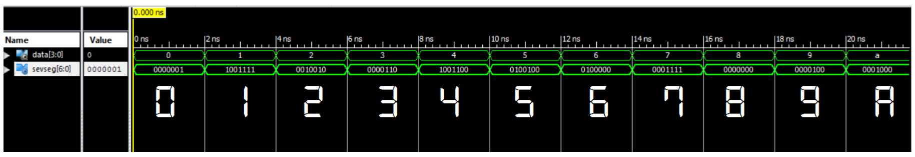

Figure 4: Conversion of a four bit data to seven segment

The following figure shows the display enable changing its value faster than the clock of counting in away that it updates all the displays before the count change the value.

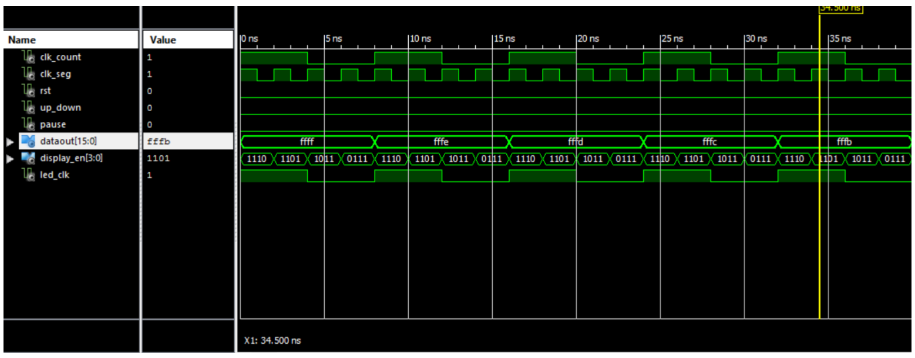

Figure 5: Display Enable Changing

In the Figure 6 is possible to see the down counting four times then it counts up three times and pause. After four clock cycles it returns counting up for two more values and then resets. After the reset return to zero it continues counting up forever.

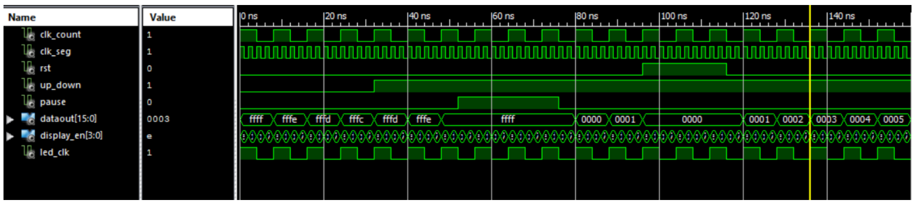

Figure 6: Test of Down/Up counting, Pause and Reset

5 Conclusions

In this lab was implemented in VHDL an up/down counter with reset and pause options. Is possible to see through this project that the implementation of counters are much faster in VHDL than in direct hardware. This project provided a better understanding of sequential machines and some of the FPGA board components. The most difficult part of the design was in the understanding of what is sequential logic and how to develop it in VHDL. It is possible to conclude that this project was important to the student's educational growth.

6 Codes and Test Benches

6.1 Hex_Display.vhd

```
-- File: Hex_Display.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Hex_Display IS
        PORT (data: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                   sevseg: OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
END Hex_Display;

ARCHITECTURE func OF Hex_Display IS

        SIGNAL leds: STD_LOGIC_VECTOR (6 DOWNTO 0);

BEGIN
-- Conversion of a for bit data to the equivalent in seven segment display
        PROCESS (data)
                BEGIN
                        CASE data IS                                    --abcdefg
                                WHEN x"0"=> leds <= "1111110";
                                WHEN x"1"=> leds <= "0110000";
                                WHEN x"2"=> leds <= "1101101";
                                WHEN x"3"=> leds <= "1111001";
                                WHEN x"4"=> leds <= "0110011";
                                WHEN x"5"=> leds <= "1011011";
                                WHEN x"6"=> leds <= "1011111";
                                WHEN x"7"=> leds <= "1110000";
                                WHEN x"8"=> leds <= "1111111";
                                WHEN x"9"=> leds <= "1111011";
                                WHEN x"A" => leds <= "1110111";
                                WHEN x"B" => leds <= "0011111";
                                WHEN x"C" => leds <= "0001101";
                                WHEN x"D" => leds <= "0111101";
                                WHEN x"E" => leds <= "1001111";
                                WHEN OTHERS => leds <= "1000111";
                        END CASE;
                END PROCESS;

        -- Invert the seven segment
        sevseg <= NOT(leds);
end func;
```

6.2 ClkDiv.vhd

```
-- File: ClkDiv.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

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

ENTITY ClkDiv IS
        PORT(Clock_50MHz: IN STD_LOGIC;
                        rst: IN STD_LOGIC;
                        clk_out, clk_seg: OUT STD_LOGIC);
END ClkDiv;

-----------------------------------------
```

```
ARCHITECTURE func OF ClkDiv IS
SIGNAL cnt: STD_LOGIC_VECTOR(23 DOWNTO 0);
SIGNAL cnt_seg: STD_LOGIC_VECTOR(10 DOWNTO 0);
BEGIN

-- Clock divider for the count with reset implemented
clkcount:
PROCESS(rst, Clock_50MHz)
BEGIN
        IF (rst = '1') THEN
                cnt <= x"000000";
        ELSIF (Clock_50MHz'EVENT AND Clock_50MHz = '1') THEN
                cnt <= cnt + 1;
        END IF;
END PROCESS;

-- Clock divider for the seven segment displays without reset
clkseg:
PROCESS(Clock_50MHz)
BEGIN
        IF (Clock_50MHz'EVENT AND Clock_50MHz = '1') THEN
                cnt_seg <= cnt_seg + 1;
        END IF;
END PROCESS;

-- Output statements
clk_seg <= cnt_seg(10); -- refresh hate: 24414.06 Hz
clk_out <= cnt(23); -- count rate: 2.98 Hz

END func;
```

6.3 Counter.vhd

```
-- File: Counter.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;

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

ENTITY Counter IS
PORT(        clk, up_down, rst, pause: IN STD_LOGIC;
                Data_out: OUT STD_LOGIC_VECTOR(15 DOWNTO 0));
END Counter;

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

ARCHITECTURE func OF Counter IS
SIGNAL Data_count: INTEGER RANGE 0 TO 65535;
BEGIN

-- Process that increments or decrements the count with reset
-- and pause implemented
PROCESS(clk, rst, pause)
BEGIN
-- check for reset
IF (rst = '1') THEN
Data_count <= 0;

-- check for pause
ELSIF (pause = '0') THEN
```

```
        IF (clk'EVENT AND clk = '1') THEN
        -- up counting
                IF (up_down = '1') THEN
                        IF (Data_count < 65535) THEN
                                Data_count <= Data_count + 1;
                        ELSE
                                Data_count <= 0; -- restart counting
                        END IF;
        -- down counting
                ELSIF (up_down = '0') THEN
                        IF (Data_count > 0) THEN
                                Data_count <= Data_count - 1;
                        ELSE
                                Data_count <= 65535; -- restart counting
                        END IF;
                END IF;
        END IF;
END IF;
END PROCESS;

-- Output statement
Data_out <= CONV_STD_LOGIC_VECTOR(Data_count, 16);
-- Convert integer to 16 bit std_logic_vector

END func;
```

6.4 Counter_updown.vhd

```
-- Main File: Counter_updown.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
-----------------------------------------

ENTITY Counter_updown IS
PORT(        clkin, rst, up_down, pause: IN STD_LOGIC;
                sevseg: OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
                Display_EN: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
                LED_clk: OUT STD_LOGIC);
END Counter_updown;

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

ARCHITECTURE func OF Counter_updown IS

COMPONENT Counter
PORT(        clk, up_down, rst, pause: IN STD_LOGIC;
                Data_out: OUT STD_LOGIC_VECTOR(15 DOWNTO 0));
END COMPONENT;

COMPONENT ClkDiv
        PORT(Clock_50MHz: IN STD_LOGIC;
                        rst: IN STD_LOGIC;
                        clk_out, clk_seg: OUT STD_LOGIC);
END COMPONENT;

COMPONENT Hex_Display
        PORT (data: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                   sevseg: OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
END COMPONENT;

SIGNAL clkout, clk_seg: STD_LOGIC;
SIGNAL Data_count: STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL Data_seg: STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN
U1: ClkDiv PORT MAP (clkin, rst, clkout, clk_seg);
U2: Counter PORT MAP (clkout, up_down, rst, pause, Data_count);

D0: Hex_Display PORT MAP (Data_seg, sevseg);

LED_clk <= clkin;


-- Process to refresh the values in the displays in a
-- faster clock (clk_seg).

PROCESS(clk_seg)
VARIABLE refresh: INTEGER RANGE 0 TO 3 := 0;
BEGIN
IF (clk_seg'EVENT AND clk_seg = '1') THEN
        CASE refresh IS
                WHEN 0 =>
                        Data_seg <= Data_count(3 DOWNTO 0);
                        Display_EN <= "1110";
                        refresh := refresh + 1;
                WHEN 1 =>
                        Data_seg <= Data_count(7 DOWNTO 4);
                        Display_EN <= "1101";
                        refresh := refresh + 1;
                WHEN 2 =>
                        Data_seg <= Data_count(11 DOWNTO 8);
                        Display_EN <= "1011";
                        refresh := refresh + 1;
                WHEN 3 =>
```

```
                        Data_seg <= Data_count(15 DOWNTO 12);
                        Display_EN <= "0111";
                        refresh := 0;
        END CASE;
END IF;
END PROCESS;

END func;
```

6.5 Test Bench - Hex_Display_tb.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Hex_Display_tb IS
END Hex_Display_tb;

ARCHITECTURE behavior OF Hex_Display_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Hex_Display
    PORT(
         data : IN  std_logic_vector(3 downto 0);
         sevseg : OUT  std_logic_vector(6 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal data : std_logic_vector(3 downto 0) := (others => '0');
```

```
         --Outputs
   signal sevseg : std_logic_vector(6 downto 0);



BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: Hex_Display PORT MAP (
          data => data,
          sevseg => sevseg
        );

PROCESS
        BEGIN

        data <= x"0";
        WAIT FOR 2 NS;

        data <= x"1";
        WAIT FOR 2 NS;

        data <= x"2";
        WAIT FOR 2 NS;

        data <= x"3";
        WAIT FOR 2 NS;

        data <= x"4";
        WAIT FOR 2 NS;

        data <= x"5";
        WAIT FOR 2 NS;
        data <= x"6";
        WAIT FOR 2 NS;

        data <= x"7";
        WAIT FOR 2 NS;

        data <= x"8";
        WAIT FOR 2 NS;

        data <= x"9";
        WAIT FOR 2 NS;

        data <= x"A";
        WAIT FOR 2 NS;

        data <= x"B";
        WAIT FOR 2 NS;

        data <= x"C";
        WAIT FOR 2 NS;

        data <= x"D";
        WAIT FOR 2 NS;

        data <= x"E";
        WAIT FOR 2 NS;

        data <= x"F";

        WAIT FOR 2 NS;
        data <= x"0";
        WAIT;
END PROCESS;



END;
```

6.6 Test Bench - Counter_updown_tb.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Counter_updown_tb IS
END Counter_updown_tb;

ARCHITECTURE behavior OF Counter_updown_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Counter_updown
    PORT(
         clk_count : IN  std_logic;
         clk_seg : IN  std_logic;
         rst : IN  std_logic;
         up_down : IN  std_logic;
         pause : IN  std_logic;
         DataOut : OUT  std_logic_vector(15 downto 0);
         sevseg : OUT  std_logic_vector(6 downto 0);
         Display_EN : OUT  std_logic_vector(3 downto 0);
         LED_clk : OUT  std_logic
        );
    END COMPONENT;
```

```
   --Inputs
   signal clk_count : std_logic := '0';
   signal clk_seg : std_logic := '0';
   signal rst : std_logic := '0';
   signal up_down : std_logic := '0';
   signal pause : std_logic := '0';

         --Outputs
   signal DataOut : std_logic_vector(15 downto 0);
   signal sevseg : std_logic_vector(6 downto 0);
   signal Display_EN : std_logic_vector(3 downto 0);
   signal LED_clk : std_logic;


BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: Counter_updown PORT MAP (
          clk_count => clk_count,
          clk_seg => clk_seg,
          rst => rst,
          up_down => up_down,
          pause => pause,
          DataOut => DataOut,
          sevseg => sevseg,
          Display_EN => Display_EN,
          LED_clk => LED_clk
        );

-- Process for the clock of the counter
```

```
        PROCESS
        BEGIN
                clk_count <= '1';
                WAIT FOR 4 NS;
                clk_count <= '0';
                WAIT FOR 4 NS;
        END PROCESS;

-- Process for the clock of the seven segment display
-- 1/4 of clk_count
        PROCESS
        BEGIN
                clk_seg <= '1';
                WAIT FOR 1 NS;
                clk_seg <= '0';
                WAIT FOR 1 NS;
        END PROCESS;

-- Process to test reset, pause and up/down counting
        PROCESS
        BEGIN
        -- Down couting
                up_down <= '0';
                rst <= '0';
                pause <= '0';
                WAIT FOR 32 NS;

        -- Up counting
                up_down <= '1';
                WAIT FOR 20 NS;

        -- Pause test
```

```
                pause <= '1';
                WAIT FOR 24 NS;
                pause <= '0';
                WAIT FOR 20 NS;

        -- Reset test
                rst <= '1';
                WAIT FOR 20 NS;
                rst <= '0';
                WAIT;
        END PROCESS;
END;
```

6.7 UCF File - Counter_updown.ucf

```
NET "clkin" LOC = "B8";

NET "up_down" LOC = "P11";
NET "pause" LOC = "L3";
NET "rst" LOC = "K3";

NET "LED_clk" LOC = "M5";

NET "Display_EN(0)" LOC = "F12";
NET "Display_EN(1)" LOC = "J12";
NET "Display_EN(2)" LOC = "M13";
NET "Display_EN(3)" LOC = "K14";

NET "sevseg(6)" LOC = "L14";
NET "sevseg(5)" LOC = "H12";
NET "sevseg(4)" LOC = "N14";
NET "sevseg(3)" LOC = "N11";
NET "sevseg(2)" LOC = "P12";

NET "sevseg(1)" LOC = "L13";

NET "sevseg(0)" LOC = "M12";
```