Lab 2

Adrian Torres and Angel Ordonez Retamar

Part 1

initial boot

```
Tcl Console  Messages  Serial I/O Links  Serial I/O Scans

connect_hw_server: Time (s): cpu = 00:00:00 ; elapsed = 00:00:06 . Memory (MB): peak = 1817.238 ; gain = 0.000
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210292B0F203A
set_property PROGRAM.FILE {C:/Users/angel/Documents/CPE487/VHDL/LAB2/hex4count/hex4count.runs/impl_1/hexcount.bit} [get_hw_devices xc7a100t_0]
current_hw_device [get_hw_devices xc7a100t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1435] Device xc7a100t (JTAG device index = 0) is not programmed (DONE status = 0).
set_property PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property PROGRAM.FILE {C:/Users/angel/Documents/CPE487/VHDL/LAB2/hex4count/hex4count.runs/impl_1/hexcount.bit} [get_hw_devices xc7a100t_0]
program_hw_devices [get_hw_devices xc7a100t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.

Type a Tcl command here
```

Booting from configuration memory device

```
Tcl Console  Messages  Serial I/O Links  Serial I/O Scans

program_hw_cfgmem -hw_cfgmem [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xc7a100t_0] 0]]
INFO: [Xicom 50-104] The verify operation will only be performed on the address range specified by the Memory Configuration File (MCS).
Mfg ID : 1   Memory Type : 20   Memory Capacity : 18   Device ID 1 : 0   Device ID 2 : 0
Performing Erase Operation...
Erase Operation successful.
Performing Program and Verify Operations...
Program/Verify Operation successful.
INFO: [Labtoolstcl 44-377] Flash programming completed successfully
program_hw_cfgmem: Time (s): cpu = 00:00:00 ; elapsed = 00:01:45 . Memory (MB): peak = 1903.410 ; gain = 0.000
endgroup
boot_hw_device  [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtoolstcl 44-664] Will wait up to 180 seconds for booting to complete.
INFO: [Labtools 27-32] Done pin status: HIGH
boot_hw_device: Time (s): cpu = 00:00:00 ; elapsed = 00:00:12 . Memory (MB): peak = 1903.410 ; gain = 0.000
refresh_hw_device [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.

Type a Tcl command here
```

video of the program running from memory when the project is closed on vivado
the board is turned on and after 10 seconds begins to count as expected
https://youtu.be/05EXtZHwp_8

Part 2

https://youtube.com/shorts/l1vs7eMLQdE?feature=share
the board turns on and counts up to 3 F before flipping and counting down to 2 0 and then continues counting up and down, back and forth between those values

short answer: We initially tried to get the code working with 2 flips from 0 to 1 and found that the counter would constantly count up and never flip. We think that by making it only flip from 1 to 0 once it will make the sequence much more likely to come up from the counter bit and therefore make the flip actually happen.

Original counter.vhd:

```
-- counter.vhd --

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY counter IS
	PORT (
		clk : IN STD_LOGIC;
		count : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
		mpx : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
END counter;

ARCHITECTURE Behavioral OF counter IS
	SIGNAL cnt : STD_LOGIC_VECTOR (38 DOWNTO 0); -- 39-bit counter
BEGIN
	PROCESS (clk)
	BEGIN
		IF clk'EVENT AND clk = '1' THEN -- on rising edge of clock
			cnt <= cnt + 1; -- increment counter
		END IF;
	END PROCESS;
	count <= cnt (38 DOWNTO 23); -- 16 bits
	mpx <= cnt (19 DOWNTO 17); -- 3 bits
END Behavioral;
```

Changed counter.vhd:

```
-- counter.vhd --

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY counter IS
   PORT (
      clk, x : IN STD_LOGIC;
      count : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
      mpx : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
      Y : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
      Z : OUT STD_LOGIC);
END counter;

ARCHITECTURE Behavioral OF counter IS
   SIGNAL cnt : STD_LOGIC_VECTOR (38 DOWNTO 0);  -- 39-bit counter
   SIGNAL direction : STD_LOGIC := '1';
   SIGNAL z_int : STD_LOGIC := '0';
   TYPE state_type IS (A, B, C, D, E);
   SIGNAL PS, NS : state_type;

BEGIN
   -- Clock process: handles counter and state updating
```

```
PROCESS (clk)
BEGIN
   IF clk'EVENT AND clk = '1' THEN
      IF direction = '1' THEN
         cnt <= cnt + 1;
      ELSE
         cnt <= cnt - 1;
      END IF;
      PS <= NS;
   END IF;
END PROCESS;


count <= cnt(38 DOWNTO 23);  -- 16-bit output
mpx <= cnt(19 DOWNTO 17);    -- 3-bit output

-- FSM logic process
stateAndOutputLogic: PROCESS(PS, cnt(28))
BEGIN
   CASE PS IS
      WHEN A =>
         IF (cnt(28) = '1') THEN
            z_int <= '0';
            NS <= B;
         ELSE
            z_int <= '0';
            NS <= A;
         END IF;
      WHEN B =>
         IF (cnt(28) = '1') THEN
            z_int <= '0';
            NS <= C;
         ELSE
            z_int <= '0';
            NS <= A;
         END IF;
      WHEN C =>
         IF (cnt(28) = '1') THEN
            z_int <= '0';
            NS <= D;
         ELSE
            z_int <= '0';
            NS <= A;
         END IF;
      WHEN D =>
         IF (cnt(28) = '1') THEN
            z_int <= '0';
            NS <= D;
         ELSE
            z_int <= '0';
            NS <= E;
         END IF;
      WHEN E =>
         IF (cnt(28) = '1') THEN
            z_int <= '0';
            NS <= B;
```

```
            ELSE
                z_int <= '1';
                NS <= A;
            END IF;
        END CASE;
    END PROCESS stateAndOutputLogic;

    -- Output Y based on the state
    WITH PS SELECT
        Y <= "000" WHEN A,
            "001" WHEN B,
            "010" WHEN C,
            "011" WHEN D,
            "100" WHEN E,
            "000" WHEN OTHERS;


    PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk = '1' THEN
            IF z_int = '1' THEN
                direction <= NOT direction;
            END IF;
        END IF;
    END PROCESS;


    Z <= z_int;
END Behavioral;
```

I made three key modifications to the original code. First, I integrated the FSM logic from Simulation Activity 3, adjusting it to utilize the *cnt(28)* input signal. The second modification involved adding a direction signal that toggles whenever the *z* signal changes to 1. Finally, I implemented a direction counter that increments or decrements based on the value of the *direction* bit.