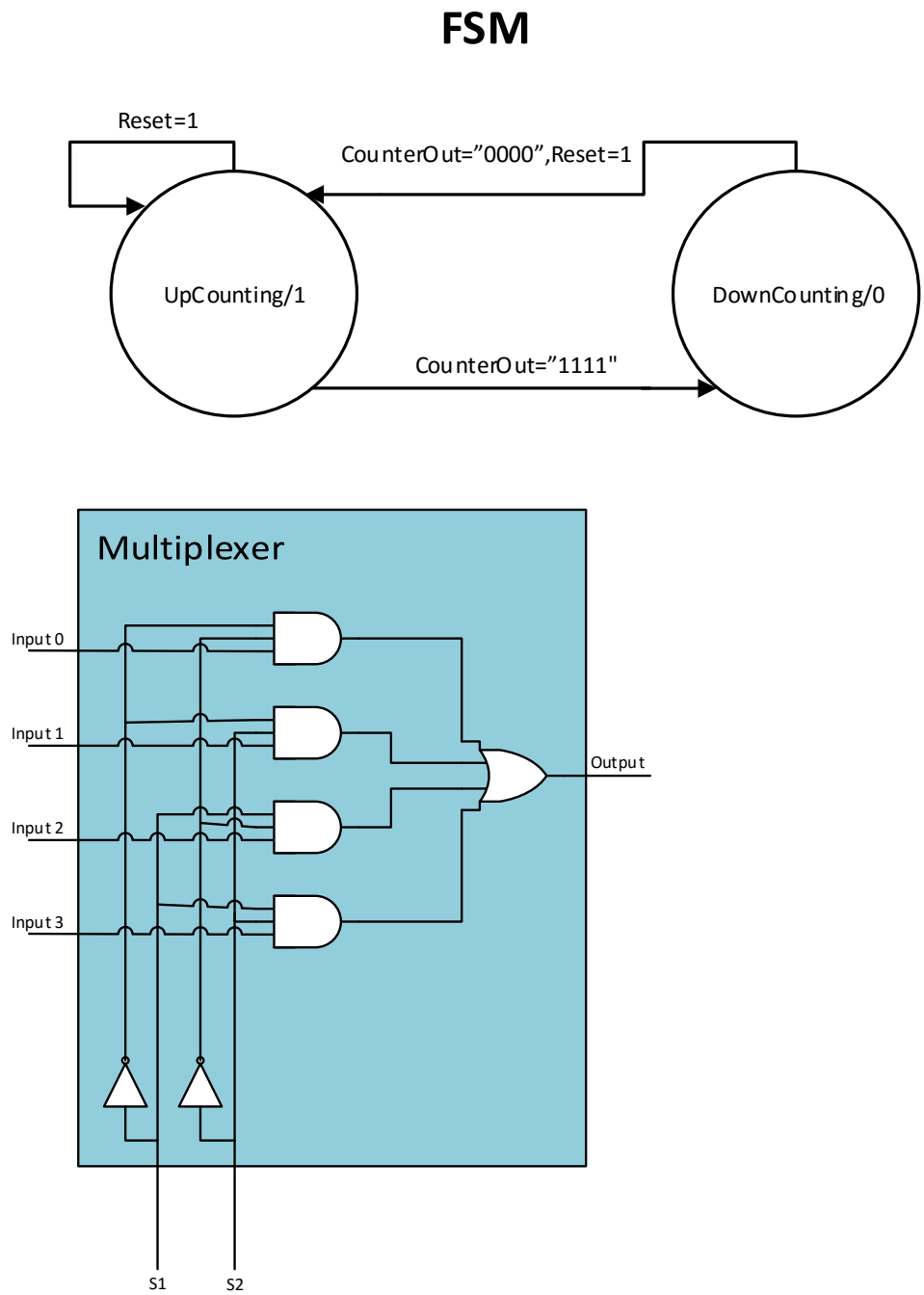
Report of Lab 3
Advanced Logic Design

Team: Κριθαράκης Εμμανουήλ, Φωτάκης Τζανής

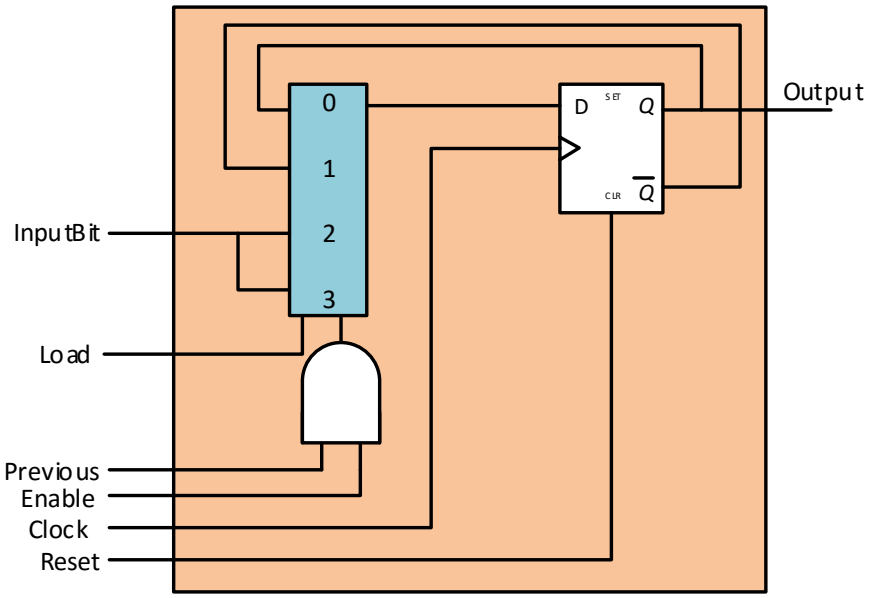
Preparation

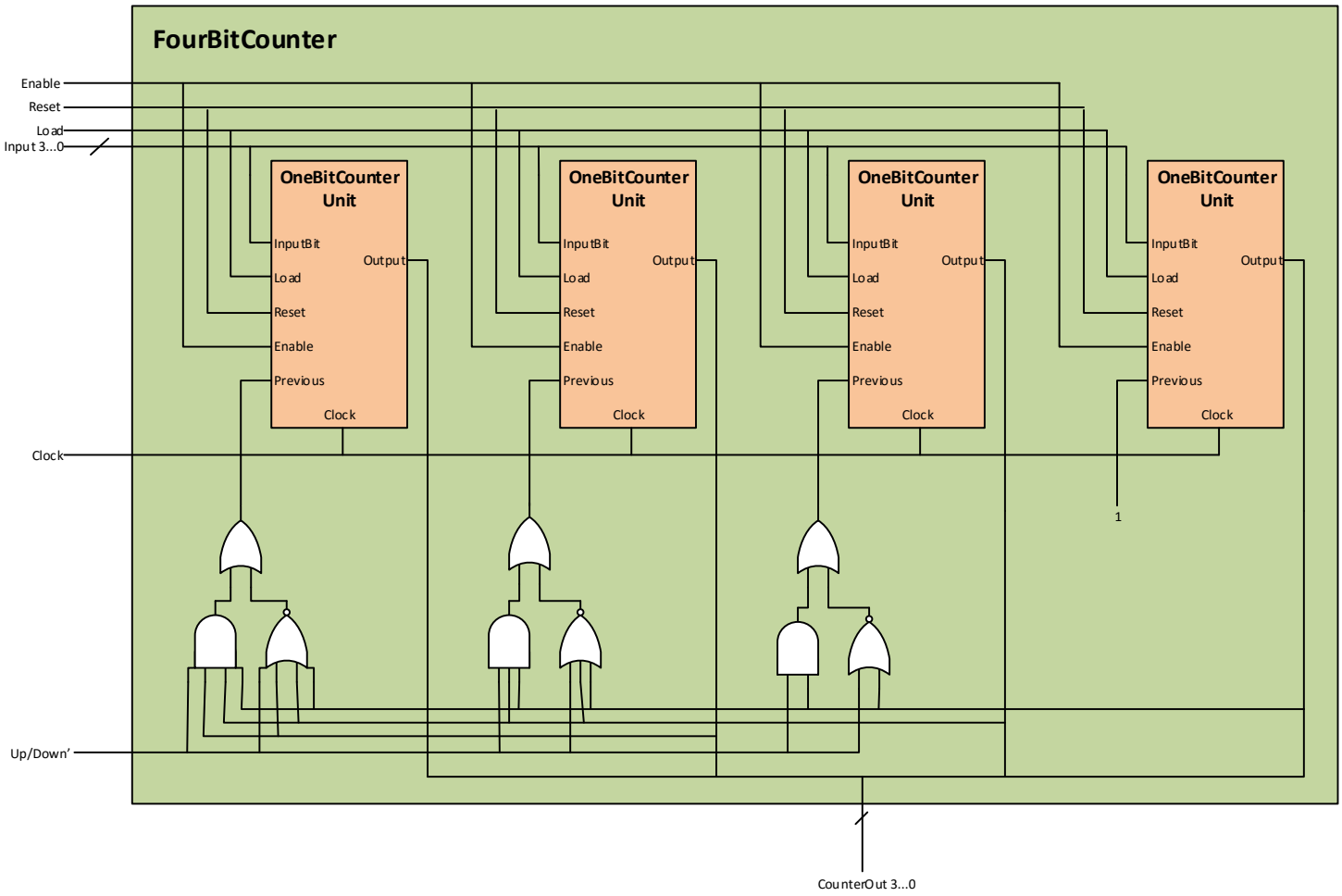
To begin with, the lab was about a counter who was connected with a finite state machine. All we have to do as a preparation was the block diagram of a counter, in which inputs, outputs and internal connections should be as specific as possible. Furthermore, we ought to have design a fsm state diagram and how each and every part interacts with the others. Last but not least was the whole program, which has to have passed the stage of "Implementation". All the above diagrams are displayed:

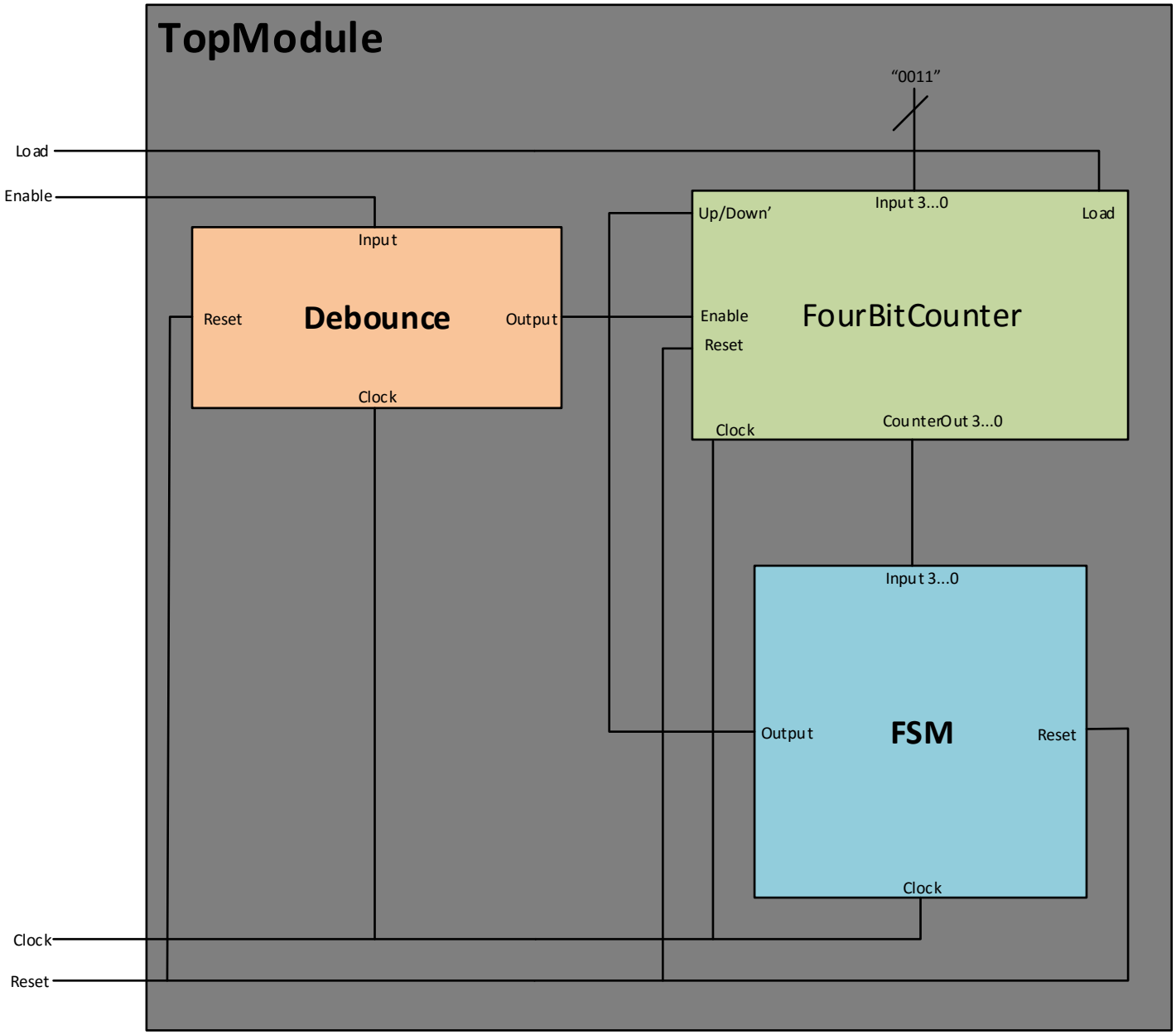
Circuits' Schematic Representation



OneBitCounterUnit







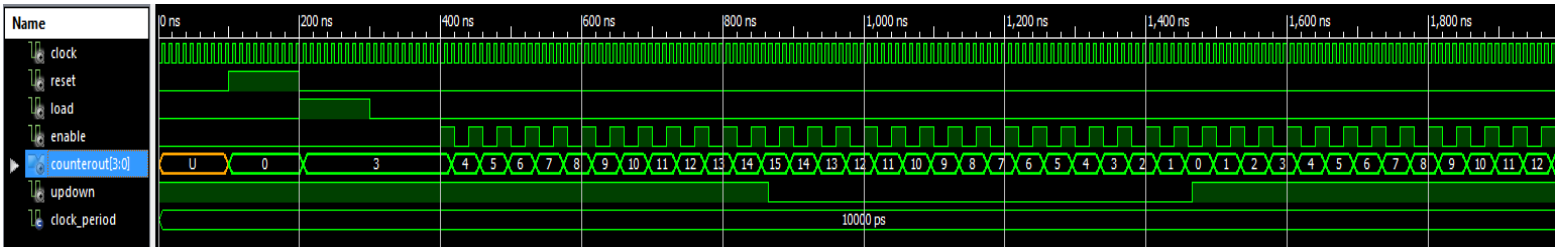
Overview

Having realized through lab1 and lab2 the "structural thought", designing a 4-bit counter, who connects with a fsm was quite simple. First things first, we started decomposing each and every part of a one bit counter and after we composed in a top unit these 3 parts (flip-flop, mux and logic gates). Next, we created a top unit with 4 components of a one-bit-counter. This module has on the one hand inputs enable, reset, upDown, load and clock of course, on the other hand outputs 4 bits (Out 3…0). Then, we constructed the Moore fsm, which was responsible to check if it essential to change the upDown signal of counter. This module has on the one hand inputs clock, reset, output of counter (Out3…0) and on the other hand the upDown signal, which was directly connected to 4 bit-counter. At the end, these two modules were inserted to the top module, where it was carefully connected through internal signals. This top module has as inputs load, reset, enable and clock and as outputs 4 bits output of the counter. Of course we should comment that the enable signal in top module was inserted in a debounce module, which has also inserted in the top one and the output of it was the input of every component which has enable as input (only counter). To be clear, every component through the right port maps lead to design the 4-bit counter.

Waveforms –Simulation

Having done the vhd files for final 4-bit counter all we have to do was to create test bench to check if the circuit work as we want. In order to do that, we should take into consideration some of the possible inputs and see if the outputs are the correct ones. The test benches help us as they produce simulations of how the circuit is going to work with specific inputs. Of course, we consider that the enable button is driving through the debounce circuit so we had to manually set it not only to 1 in order to produce the next number of the counter but also to 0 in order to approach in real time (when we don't push it) and see that it works perfectly.

Here are the simulations of the first and second circuit:



Conclusion

This third lab exercise help us a lot to understand better the vhdl language and the fpga. First of all, we realize one more time the importance of debounce button in the real time execution and how we should insert it in a vhdl file. Furthermore, it was the first time to create a fsm, which is connected to another circuit not only be its input (as its input is the output of counter) but also through its output (as its output is upDown signal of counter).

Code

OneBitCounterUnit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OneBitCounterUnit is
   Port ( InputBit : in  STD_LOGIC;
          Load : in  STD_LOGIC;
          UpDown : in  STD_LOGIC;
          Enable : in  STD_LOGIC;
          Previous : in  STD_LOGIC;
                                    Reset : in  STD_LOGIC;
                                    Clock : in  STD_LOGIC;
          Output : out  STD_LOGIC);
end OneBitCounterUnit;
architecture Behavioral of OneBitCounterUnit is
            component DFlipFlop
                        Port ( D : in  STD_LOGIC;
                                                Reset : in  STD_LOGIC;
                                                Clock : in  STD_LOGIC;
                                                Qp : inout  STD_LOGIC;
                                                Qn : inout  STD_LOGIC);
            end component;
            component mux
                        Port ( input : in  STD_LOGIC_VECTOR (3 downto 0);
                                                output : out  STD_LOGIC;
                                                s : in  STD_LOGIC_VECTOR (1 downto 0));
            end component;
            signal s0: std_logic;
            signal s1: std_logic;
            signal QpInternal: std_logic;
            signal QnInternal: std_logic;
            signal MuxOutput: std_logic;
begin
            s0<=Previous and Enable;
            s1<=Load;
            multiplexer: mux port map(input(0)=>QpInternal, input(1)=>QnInternal, input(2)=>InputBit,
input(3)=>InputBit,output=>MuxOutput,s(0)=>s0,s(1)=>s1);
            FlipFlop: DFlipFlop port map (D=>MuxOutput,Reset=>Reset,Clock=>Clock,Qp=>QpInternal,Qn=>QnInternal);
            Output<=QpInternal;
end Behavioral;
```

FourBitCounterUnit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FourBitCounter is
   Port ( Input : in  STD_LOGIC_VECTOR (3 downto 0);
          Enable : in  STD_LOGIC;
          Load : in  STD_LOGIC;
          UpDown : in  STD_LOGIC;
          Reset : in  STD_LOGIC;
          Clock : in  STD_LOGIC;
          Output : out  STD_LOGIC_VECTOR (3 downto 0));
end FourBitCounter;
architecture Behavioral of FourBitCounter is

            component OneBitCounterUnit
                  Port ( InputBit : in  STD_LOGIC;
                                    Load : in  STD_LOGIC;
                                    UpDown : in  STD_LOGIC;
                                    Enable : in  STD_LOGIC;
                                    Previous : in  STD_LOGIC;
                                    Reset : in  STD_LOGIC;
                                    Clock : in  STD_LOGIC;
                                    Output : out  STD_LOGIC);
            end component;
            signal PreviousSignal: STD_LOGIC_VECTOR (3 downto 0);
            signal CounterOutputInternal: STD_LOGIC_VECTOR (3 downto 0);
begin
            Counter0: OneBitCounterUnit port
map(InputBit=>Input(0),Load=>Load,UpDown=>UpDown,Enable=>Enable,Previous=>PreviousSignal(0),Reset=>Reset,Clock=>Clock,Output=>CounterOut
putInternal(0));
            Counter1: OneBitCounterUnit port
map(InputBit=>Input(1),Load=>Load,UpDown=>UpDown,Enable=>Enable,Previous=>PreviousSignal(1),Reset=>Reset,Clock=>Clock,Output=>CounterOut
putInternal(1));
            Counter2: OneBitCounterUnit port
map(InputBit=>Input(2),Load=>Load,UpDown=>UpDown,Enable=>Enable,Previous=>PreviousSignal(2),Reset=>Reset,Clock=>Clock,Output=>CounterOut
putInternal(2));
            Counter3: OneBitCounterUnit port
map(InputBit=>Input(3),Load=>Load,UpDown=>UpDown,Enable=>Enable,Previous=>PreviousSignal(3),Reset=>Reset,Clock=>Clock,Output=>CounterOut
putInternal(3));
            Output(0)<=CounterOutputInternal(0);
            Output(1)<=CounterOutputInternal(1);
            Output(2)<=CounterOutputInternal(2);
            Output(3)<=CounterOutputInternal(3);
            PreviousSignal(0)<='1';
            PreviousSignal(1)<=(CounterOutputInternal(0) and UpDown)or((CounterOutputInternal(0) nor UpDown));
            PreviousSignal(2)<=((CounterOutputInternal(0) and CounterOutputInternal(1)) and UpDown)or(not((CounterOutputInternal(0) or
CounterOutputInternal(1)) or UpDown));
            PreviousSignal(3)<=(((CounterOutputInternal(0) and CounterOutputInternal(1)) and CounterOutputInternal(2)) and
UpDown)or(not(((CounterOutputInternal(0) or CounterOutputInternal(1)) or CounterOutputInternal(2)) or UpDown));
end Behavioral;
```

TopUnit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TopUnit is
   Port ( Clock : in  STD_LOGIC;
          Reset : in  STD_LOGIC;
          Load : in  STD_LOGIC;
          Enable : in  STD_LOGIC;
          CounterOut : out  STD_LOGIC_VECTOR (3 downto 0);
          UpDown : out  STD_LOGIC);
end TopUnit;
architecture Behavioral of TopUnit is

            component FourBitCounter
   Port ( Input : in  STD_LOGIC_VECTOR (3 downto 0);
          Enable : in  STD_LOGIC;
          Load : in  STD_LOGIC;
          UpDown : in  STD_LOGIC;
          Reset : in  STD_LOGIC;
          Clock : in  STD_LOGIC;
```

```
        Output : out  STD_LOGIC_VECTOR (3 downto 0));
            end component;

            component FSM
                    Port ( input : in  STD_LOGIC_VECTOR (3 downto 0);
                                        output : out  STD_LOGIC;
                                        reset : in  STD_LOGIC;
                                        clock : in  STD_LOGIC);
            end component;

            component debouncebutton
                    Port ( clk                  : in std_logic;
                                        rst                     : in std_logic;
                                        input                   : in std_logic;
                                        output      : out std_logic);
            end component;

            signal DebounceOutput: std_logic;
            signal FSMOutput: std_logic;
            signal CounterOutput: std_logic_vector (3 downto 0);

begin

            TheCounter: FourBitCounter
                    Port map(Input          =>"0011",
                                                Enable=>DebounceOutput,
                                                Load        =>Load,
                                                UpDown=>FSMOutput,
                                                Reset       =>Reset,
                                                Clock       =>Clock,
                                                Output=>CounterOutput);

            CounterOut<=CounterOutput;

            TheFSM: FSM
                    Port map(input          =>CounterOutput,
                                                output=>FSMOutput,
                                                reset       =>Reset,
                                                clock       =>Clock);

            UpDown<=FSMOutput;

            TheDebounce: debouncebutton
                    Port map(clk            =>Clock,
                                                rst         =>Reset,
                                                input       =>Enable,
                                                output=>DebounceOutput);
end Behavioral;
```