COEN 313

Digital System Design II

Lecture Section: U

Digital Design Project

Julian Valencia

ID: 40244362

Presented to: Dr. Otmane Ait Mohamed

Date Due : April 5, 2024

"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"

Julian Valencia

April 4, 2024

Abstract

The following report on the Digital Design Project conducted by the students outside of class hours has a couple of objectives. First of all, we need to design (using the concepts learned in class) a system that can accurately track how many people are located inside of a room. The system has to be able to keep track of the maximum number of people allowed in the room, as well as deny access to the room if it is full. To do so, we need to draw a conceptual design of our circuit, implement it using VHDL, simulate it using Modelsim, and synthesize it using Vivado. Finally, we must comment on our design based on it's resource utilization and speed. The students are expected to be able to fully understand and implement this circuit with the use of VHDL concepts learned in lecture hours.

Table of Contents

List of Figures

Conceptual Design

As was stated in the abstract, we have to design a system that can accurately track the room occupancy using VHDL. The following figure will present my conceptual design based on the task given to us in the pdf explaining the project:

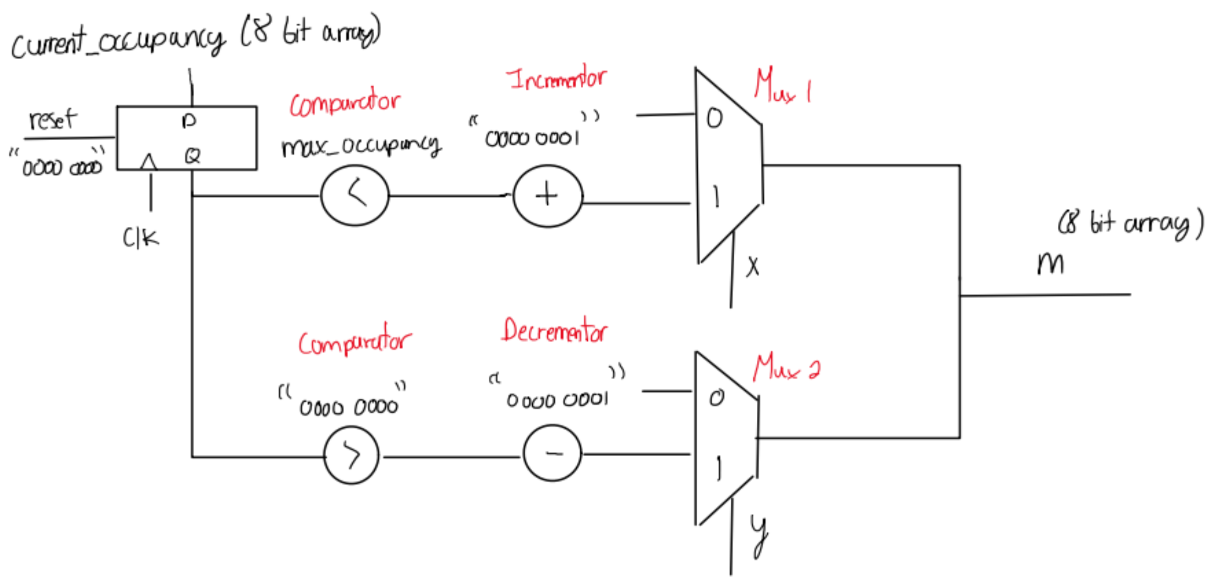


Figure 1: Conceptual design

We start with an 8-bit signal called current_occupancy, which tracks the current number of people that are inside the room. The use of a clocked process is crucial, every time a rising edge occurs (clk goes from low to high), the system updates current_occupancy depending on if a person leaves or enters the room. The reset input is used to reset current_occupancy back to 0, so that we can further test our design.

Since we were not given a specific maximum number of people allowed in the room, I went with the highest number that can be obtained from an 8-bit number (11111111), which is equal to 255 people allowed in the room. The max_occupancy signal is a constant that is always equal to 11111111, or 255.

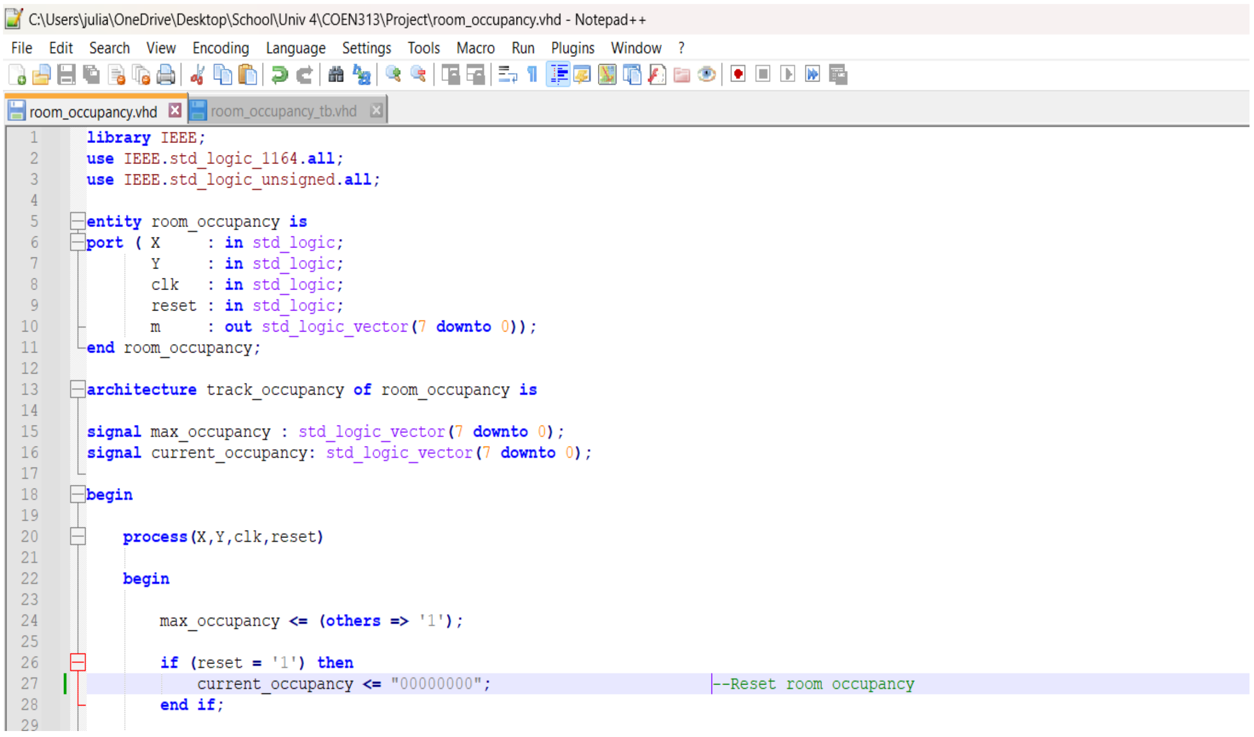
The first bit signal that is defined in my design is X, which equates to someone entering the room. If X has been activated (the photocell sensed someone entering the room) and if current_occupancy is smaller than max_occupancy, this means that there is space in the room, thus enabling current_occupancy to be incremented by 1.

On the other hand, we have another bit signal called Y, which equates to someone leaving the room. If Y has been activated (the second photocell sensed someone leave the room) and if current_occupancy is bigger than 0, this means that there are people in the room that are able to leave, thus decreasing current_occupancy by 1. When X and Y are not activated, nothing happens of importance.

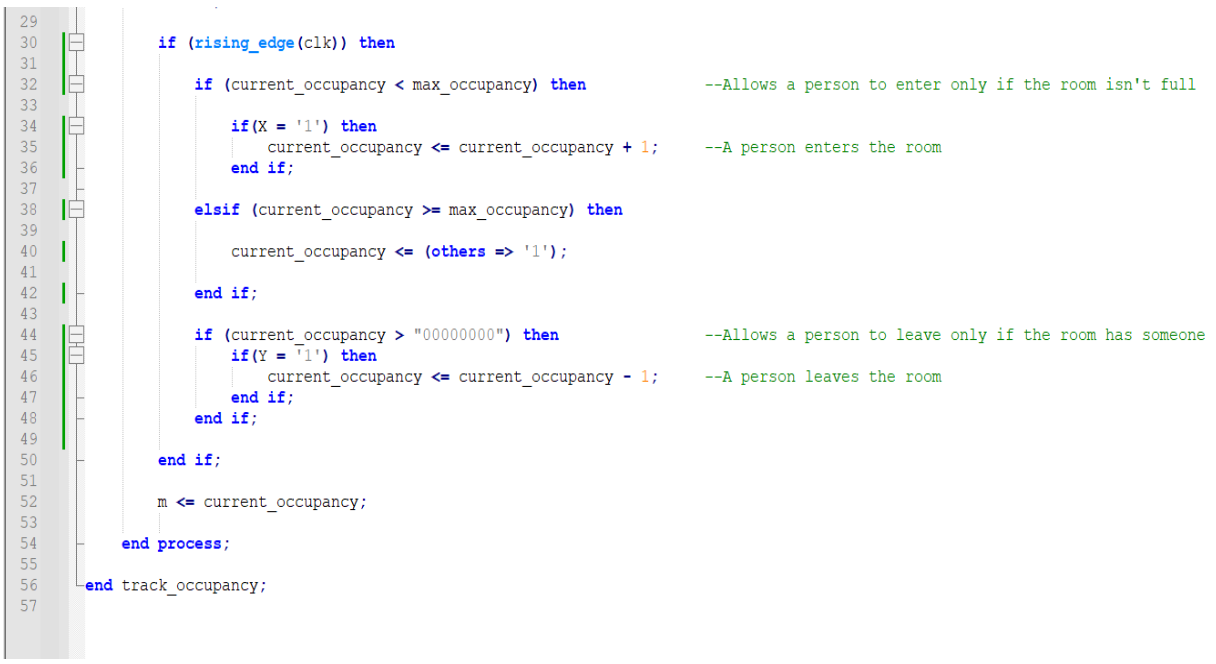
The resulting current_occupancy goes to the final output called m, which is also an 8-bit signal. This is where the final occupancy of the room is stored after people enter and leave the room. The signal is updating every time X or Y is activated (is equal to 1).

VHDL Code Explained

After designing the circuit by hand, I wrote the code that represents my design using VHDL. The following figure will provide the full VHDL code written:


C:\Users\julia\OneDrive\Desktop\School\Univ 4\COEN313\Project\room_occupancy.vhd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
room_occupancy.vhd
room_occupancy_tb.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity room_occupancy is
port ( X     : in std_logic;
       Y     : in std_logic;
       clk   : in std_logic;
       reset : in std_logic;
       m     : out std_logic_vector(7 downto 0));
end room_occupancy;

architecture track_occupancy of room_occupancy is

signal max_occupancy : std_logic_vector(7 downto 0);
signal current_occupancy: std_logic_vector(7 downto 0);

begin

    process(X,Y,clk,reset)

    begin

        max_occupancy <= (others => '1');

        if (reset = '1') then
            current_occupancy <= "00000000";                    --Reset room occupancy
        end if;
```



```vhdl
        if (rising_edge(clk)) then

            if (current_occupancy < max_occupancy) then            --Allows a person to enter only if the room isn't full

                if(X = '1') then
                    current_occupancy <= current_occupancy + 1;     --A person enters the room
                end if;

            elsif (current_occupancy >= max_occupancy) then

                current_occupancy <= (others => '1');

            end if;

            if (current_occupancy > "00000000") then                --Allows a person to leave only if the room has someone
                if(Y = '1') then
                    current_occupancy <= current_occupancy - 1;     --A person leaves the room
                end if;
            end if;

        end if;

        m <= current_occupancy;

    end process;

end track_occupancy;
```

Figure 2: Room occupancy VHDL code

Note that the VHDL file has also been placed inside of the zip file submitted for this project.

We have 4 inputs and 1 output (X, Y, clk, reset and m, respectively), which have all been defined and explained in the Conceptual Design section.

We can see that the VHDL code follows the design provided quite closely. The max_occupancy signal has been set at 11111111, the current_occupancy goes to 00000000 if reset is activated, and when X is activated and current_occupancy is smaller than max_occupancy, current_occupancy is incremented by 1 (someone enters the room).

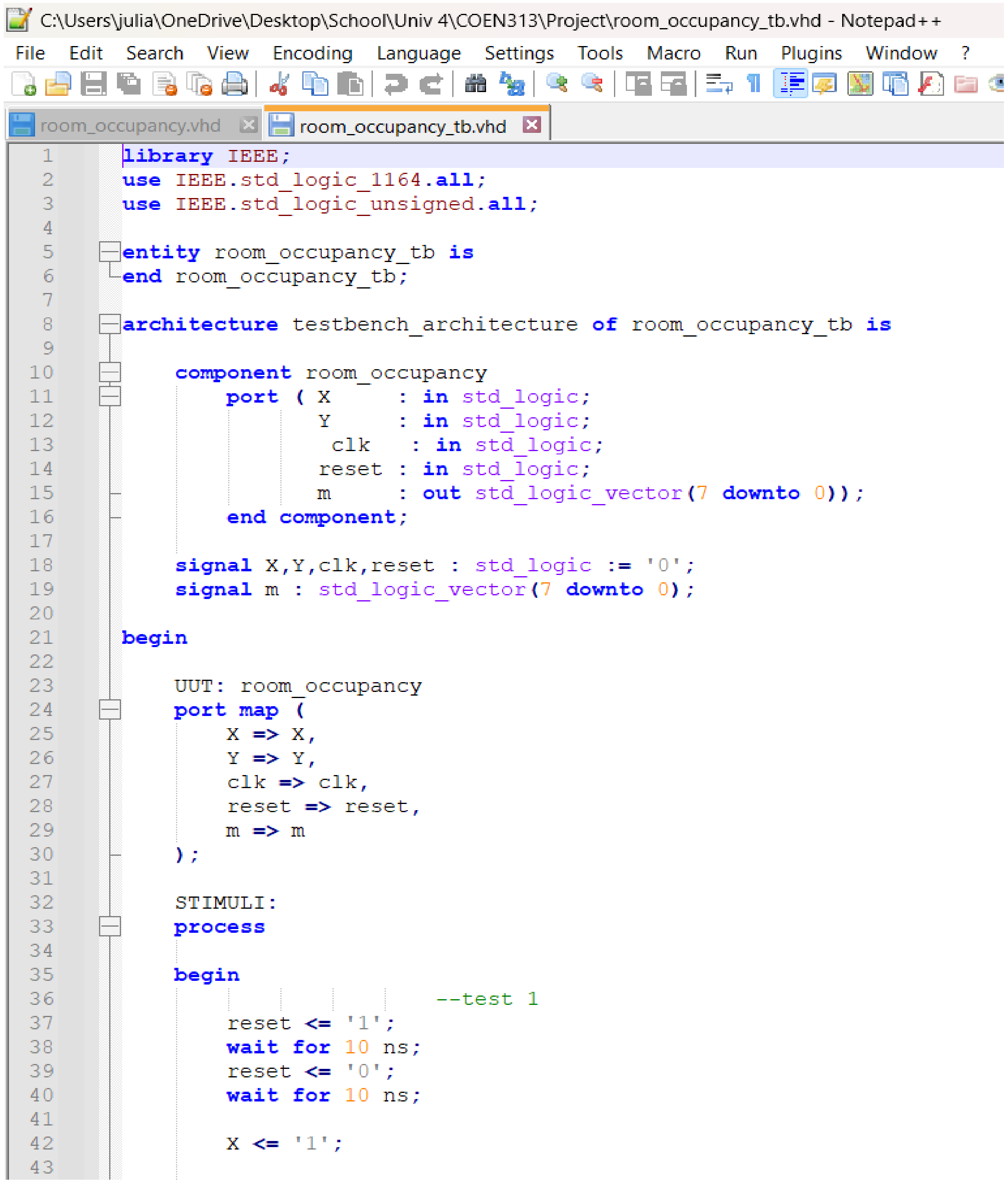
I have added something that was not present in the initial design, if current_occupancy is bigger (impossible because they are both 8-bit) or equal to max_occupancy, then current_occupancy is set at 11111111 indefinitely, this prevents current_occupancy going back to 00000000, as this was a problem when testing the first variation of the VHDL code.

Finally, when Y is activated and current_occupancy is bigger than 0000000 (at least 1 person is in the room), then current_occupancy is decreased by 1 (someone leaves the room).

In short, aside from 1 small modification to the code, the provided VHDL code is an accurate representation of the initial conceptual design.

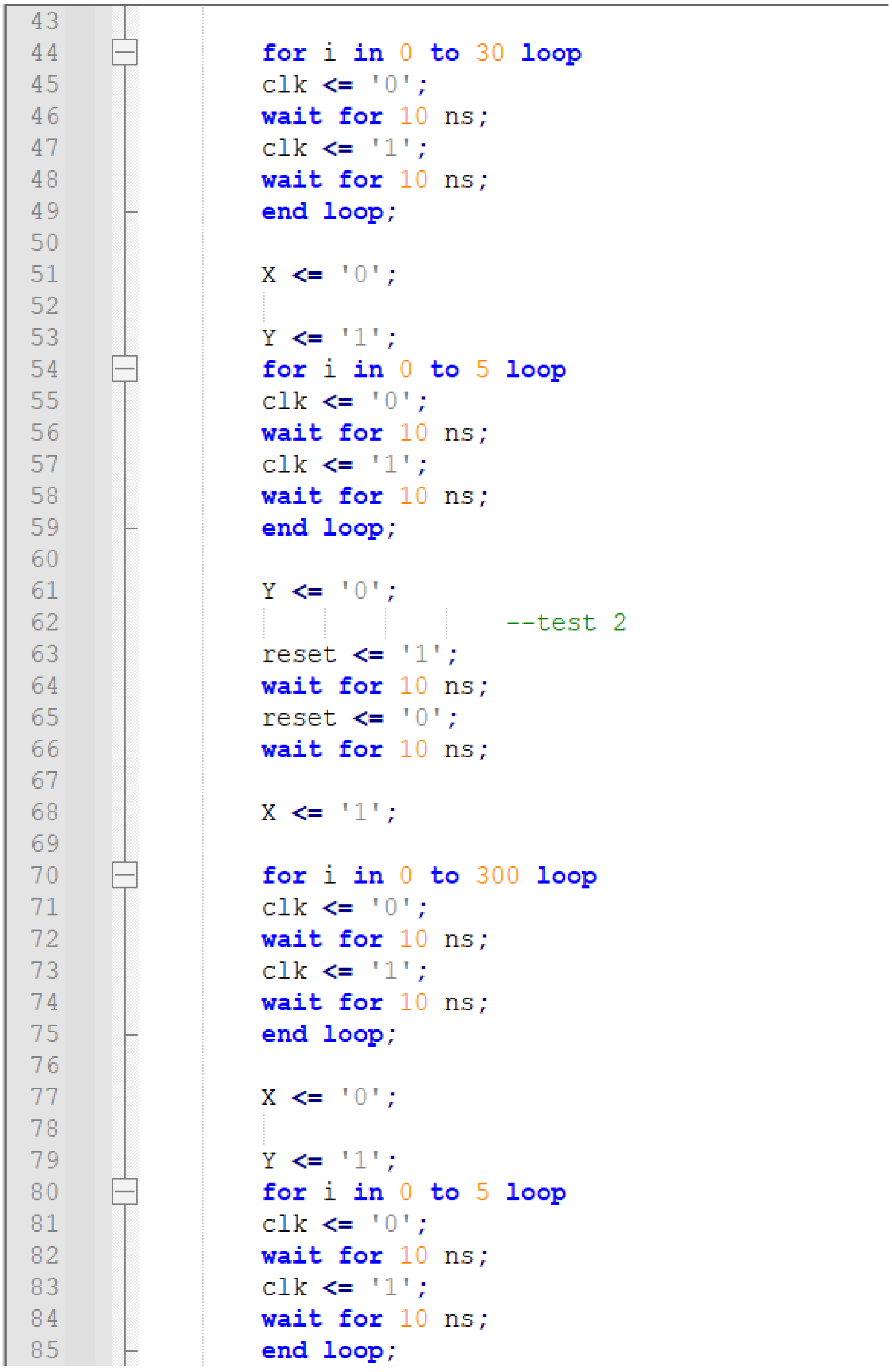
Testbench Simulation Results

After writing the VHDL code that tracks the room occupancy, we need to test it in order to see if the design was implemented correctly. The following figure will provide the testbench VHDL code written to accomplish this task:



C:\Users\julia\OneDrive\Desktop\School\Univ 4\COEN313\Project\room_occupancy_tb.vhd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
room_occupancy.vhd
room_occupancy_tb.vhd

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity room_occupancy_tb is
end room_occupancy_tb;

architecture testbench_architecture of room_occupancy_tb is

    component room_occupancy
        port ( X      : in std_logic;
               Y      : in std_logic;
                clk    : in std_logic;
               reset : in std_logic;
               m      : out std_logic_vector(7 downto 0));
        end component;

    signal X,Y,clk,reset : std_logic := '0';
    signal m : std_logic_vector(7 downto 0);

begin

    UUT: room_occupancy
    port map (
        X => X,
        Y => Y,
        clk => clk,
        reset => reset,
        m => m
    );

    STIMULI:
    process

    begin
                    --test 1
        reset <= '1';
        wait for 10 ns;
        reset <= '0';
        wait for 10 ns;

        X <= '1';

```



```vhdl

for i in 0 to 30 loop
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end loop;

X <= '0';

Y <= '1';
for i in 0 to 5 loop
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end loop;

Y <= '0';
                --test 2
reset <= '1';
wait for 10 ns;
reset <= '0';
wait for 10 ns;

X <= '1';

for i in 0 to 300 loop
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end loop;

X <= '0';

Y <= '1';
for i in 0 to 5 loop
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end loop;
```

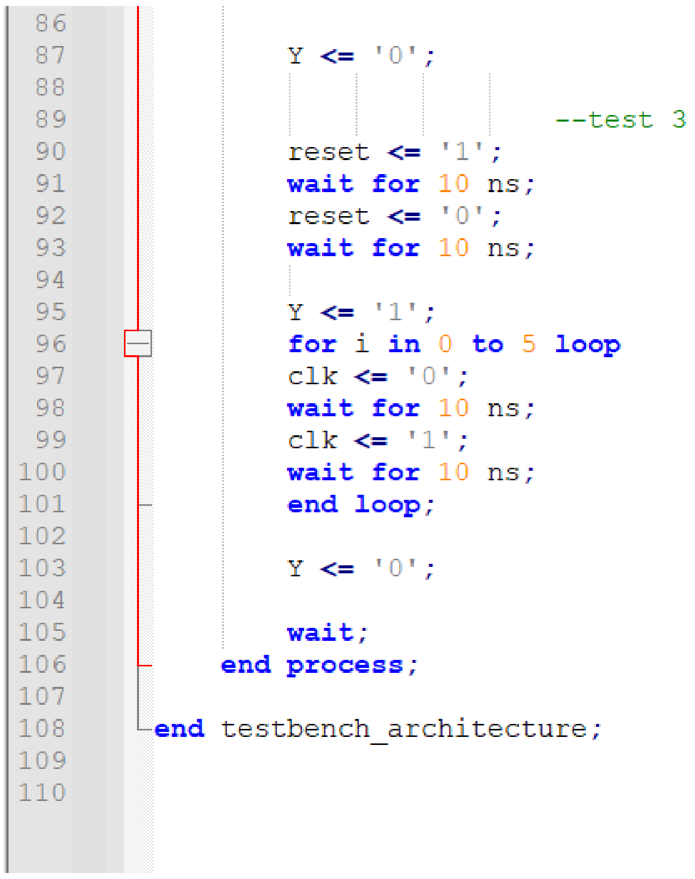

```
        Y <= '0';
                                  --test 3
        reset <= '1';
        wait for 10 ns;
        reset <= '0';
        wait for 10 ns;

        Y <= '1';
        for i in 0 to 5 loop
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
        end loop;

        Y <= '0';

        wait;
    end process;

end testbench_architecture;
```

Figure 3: Testbench VHDL code

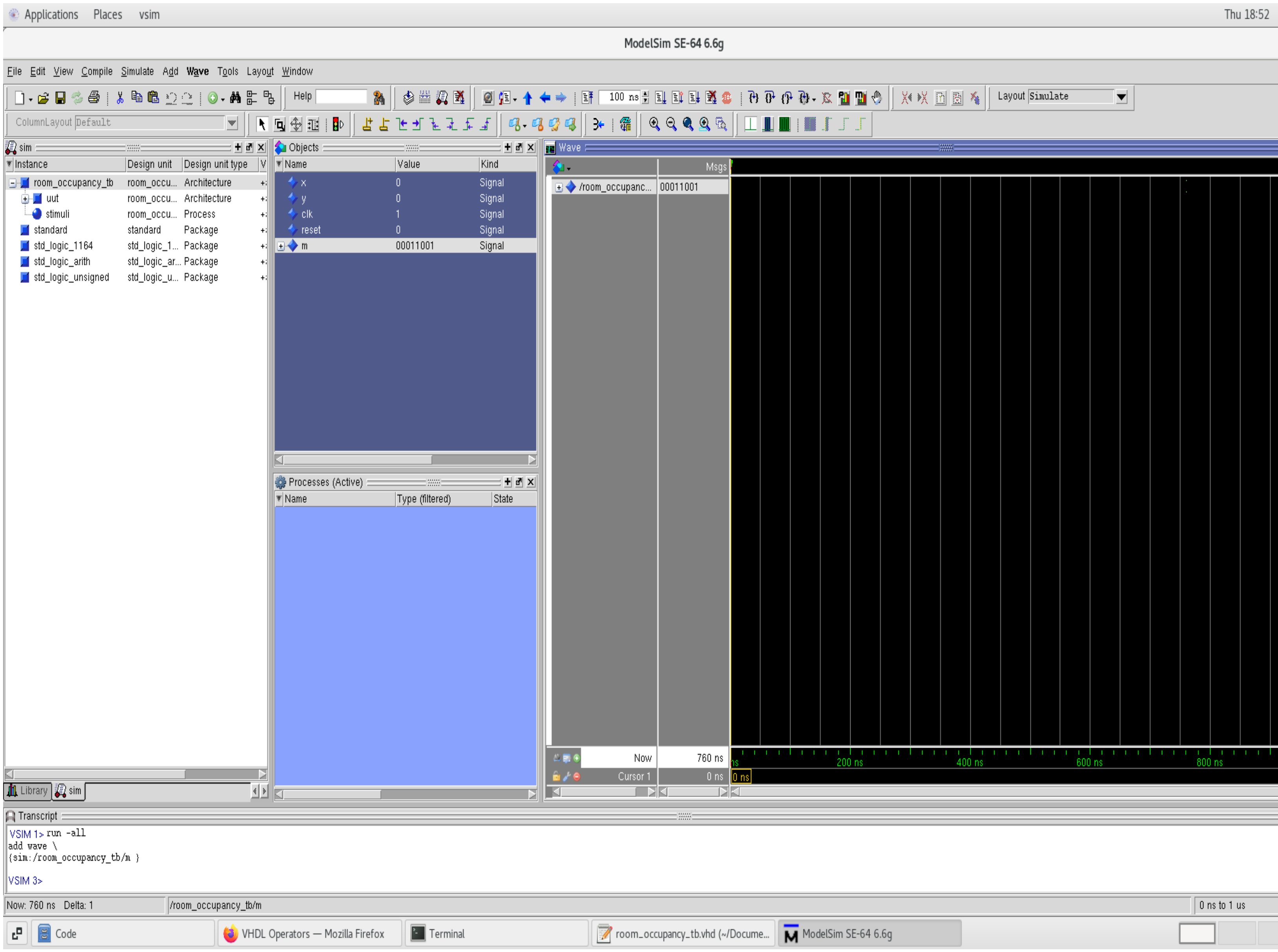
Note that the testbench VHDL code was also provided in the zip file submitted.

There are 3 tests that have been written in order to test our implementation. The first test sets X at 1 for 31 clock cycles, simulating 31 people entering a room. It then sets Y at 1 for 6 clock cycles, simulating 6 people leaving the room. It is expected that the remaining amount of people left in the room is 25.

The second test is used to test the max_occupancy of the circuit. It starts by resetting current_occupancy to 0, then sets X at 1 for 301 clock cycles. Since the room can only occupy 255 persons, current_occupancy should cap out at 11111111, or 255, even if the photosensor detects more people trying to enter. Then, it sets Y at 1 for 6 clock cycles, which equates to 6 people leaving the full room, thus the expected current_occupancy is 249.

The final test is used to test people leaving if there is nobody in the room. The testbench resets current_occupancy, then sets Y at 1 for 6 clock cycles. This should be impossible since there is nobody in the room, thus the expected current_occupancy is 0.

Using these 3 tests, we will be able to fully test the functionality of my VHDL design and test it's boundaries and limitations. The following figure will present the simulation results for test 1:



Instance	Design unit	Design unit type
room_occupancy_tb	room_occu...	Architecture
uut	room_occu...	Architecture
stimuli	room_occu...	Process
standard	standard	Package
std_logic_1164	std_logic_1...	Package
std_logic_arith	std_logic_ar...	Package
std_logic_unsigned	std_logic_u...	Package

Name	Value	Kind
x	0	Signal
y	0	Signal
clk	1	Signal
reset	0	Signal
m	00011001	Signal

	Msgs
/room_occupanc...	00011001
Now	760 ns
Cursor 1	0 ns

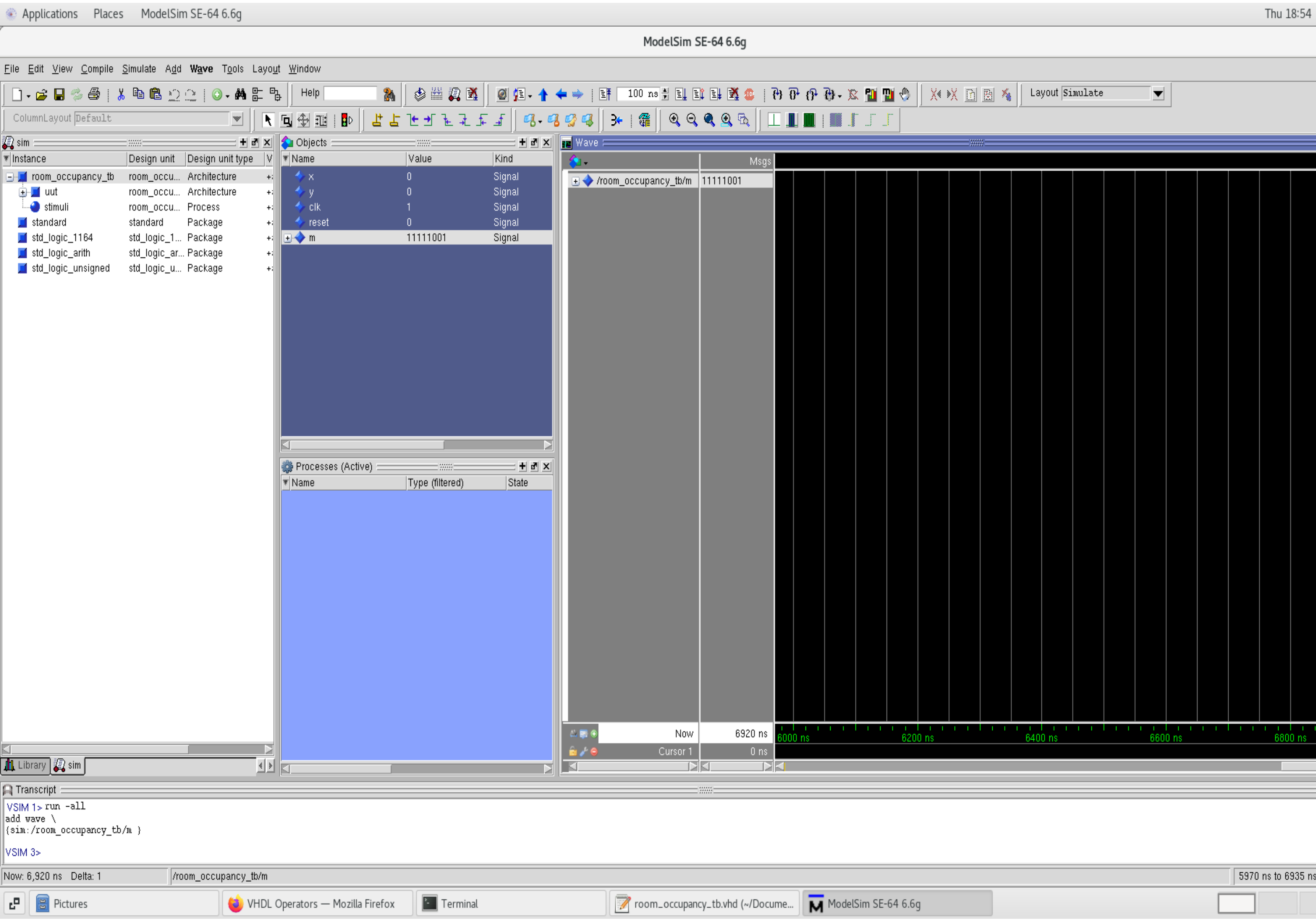
```
VSIM 1> run -all
add wave \
{sim:/room_occupancy_tb/m }

VSIM 3>
```

Now: 760 ns Delta: 1 /room_occupancy_tb/m 0 ns to 1 us

Figure 4: Test 1 simulation results

After running the first test in the testbench (after commenting tests 2-3), we can see that the signal m (equating to current_occupancy) is equal to 00011001, which in binary is equal to 25. This is the current_occupancy expected after 31 people enter the room and 6 people leave the room. We can conclude that the first test was a success.

The following figure will present the simulation results for test 2:



Instance	Design unit	Design unit type
room_occupancy_tb	room_occu...	Architecture
uut	room_occu...	Architecture
stimuli	room_occu...	Process
standard	standard	Package
std_logic_1164	std_logic_1...	Package
std_logic_arith	std_logic_ar...	Package
std_logic_unsigned	std_logic_u...	Package

Name	Value	Kind
x	0	Signal
y	0	Signal
clk	1	Signal
reset	0	Signal
m	11111001	Signal

Wave	Msgs
/room_occupancy_tb/m	11111001
Now	6920 ns
Cursor 1	0 ns

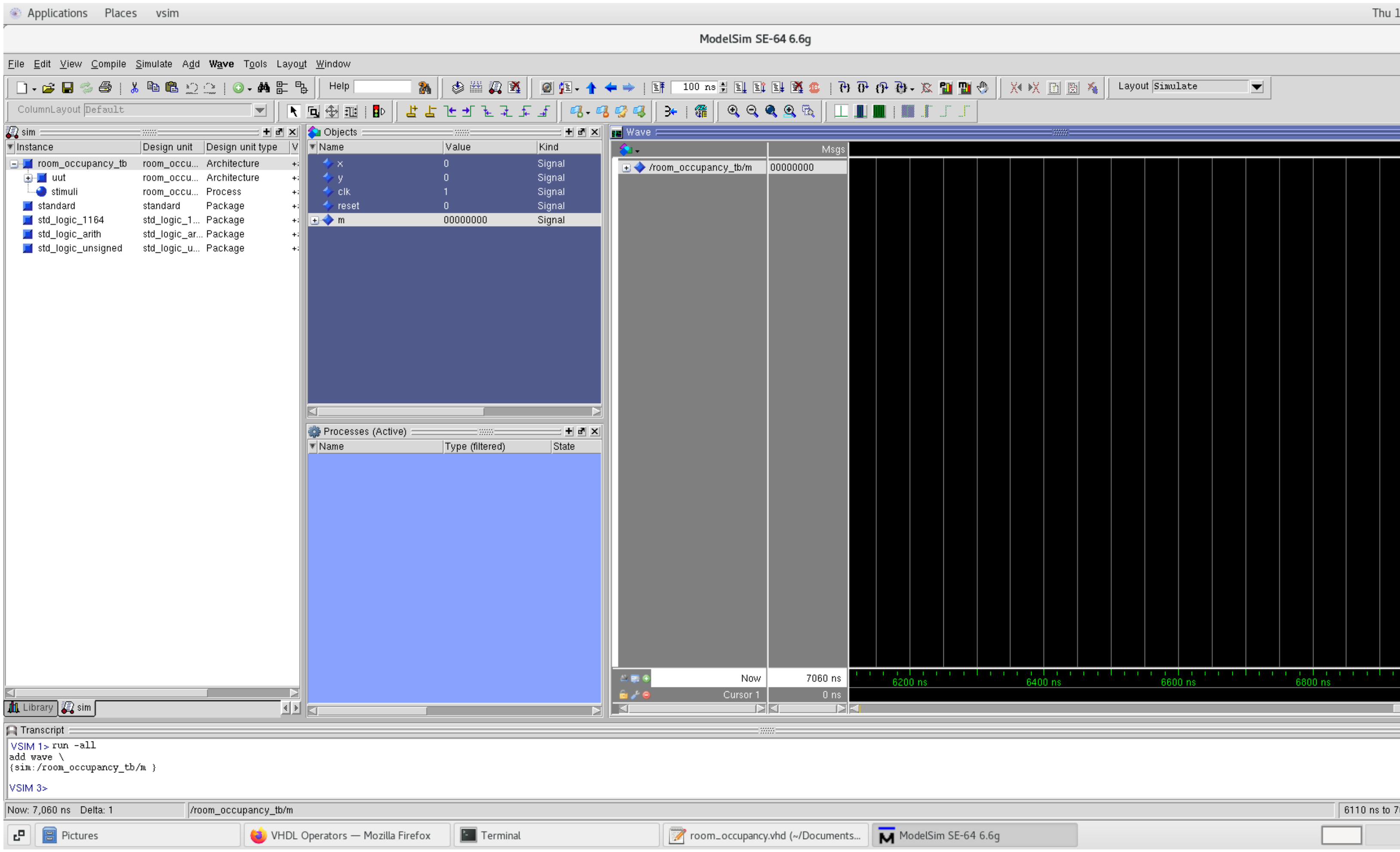
```
VSIM 1> run -all
add wave \
{sim:/room_occupancy_tb/m }

VSIM 3>
```

```
Now: 6,920 ns   Delta: 1      /room_occupancy_tb/m      5970 ns to 6935 ns
```

Figure 5: Test 2 simulation results

After running the second test in the testbench (commented tests 1-3), we can see that the signal m is now equal to 11111001, which in binary is equal to 249. This is exactly what was expected from the second test, we have confirmed that even if X is activated 301 times, the current_occupancy of the room caps out at 255, not letting people enter unless people leave the room. If 6 people left the room, then we would be left with 249 persons left in the room. We can conclude that the second test was a success.

The following figure will present the simulation results for test 3:



Instance	Design unit	Design unit type
room_occupancy_tb	room_occu...	Architecture
uut	room_occu...	Architecture
stimuli	room_occu...	Process
standard	standard	Package
std_logic_1164	std_logic_1...	Package
std_logic_arith	std_logic_ar...	Package
std_logic_unsigned	std_logic_u...	Package

Name	Value	Kind
x	0	Signal
y	0	Signal
clk	1	Signal
reset	0	Signal
m	00000000	Signal

Name	Type (filtered)	State

```
/room_occupancy_tb/m    00000000
Now       7060 ns
Cursor 1  0 ns
6200 ns   6400 ns   6600 ns   6800 ns
```

```
VSIM 1> run -all
add wave \
{sim:/room_occupancy_tb/m }

VSIM 3>
```

```
Now: 7,060 ns   Delta: 1   /room_occupancy_tb/m   6110 ns to 7
```

Figure 6: Test 3 simulation results

After running the third test of the testbench (commented tests 1-2), we can see that the signal m is equal to 00000000. This is what we expected to see, even if the signal detected that 6 people left the room, current_occupancy should still be at 0 since there is nobody present in the room. We can thus confirm that the third and final test of the testbench was a success.

In short, since all 3 of our tests passed, we can confirm that my VHDL design is an accurate representation of the task that was given to us for this project.

Synthesis Results

After confirming that my circuit design worked as expected, the next step is to produce the synthesis results using Vivado.

The following 2 figures will present the Elaborated and Implemented designs synthesised by Vivado:

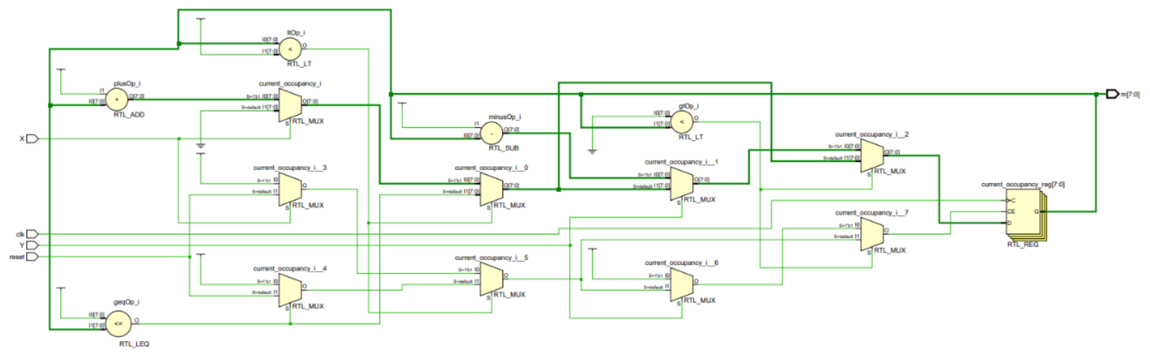


Figure 7: Elaborated schematic

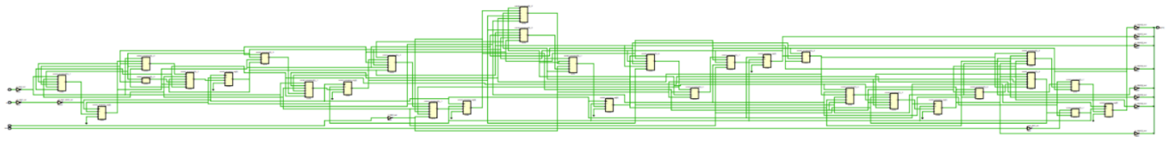
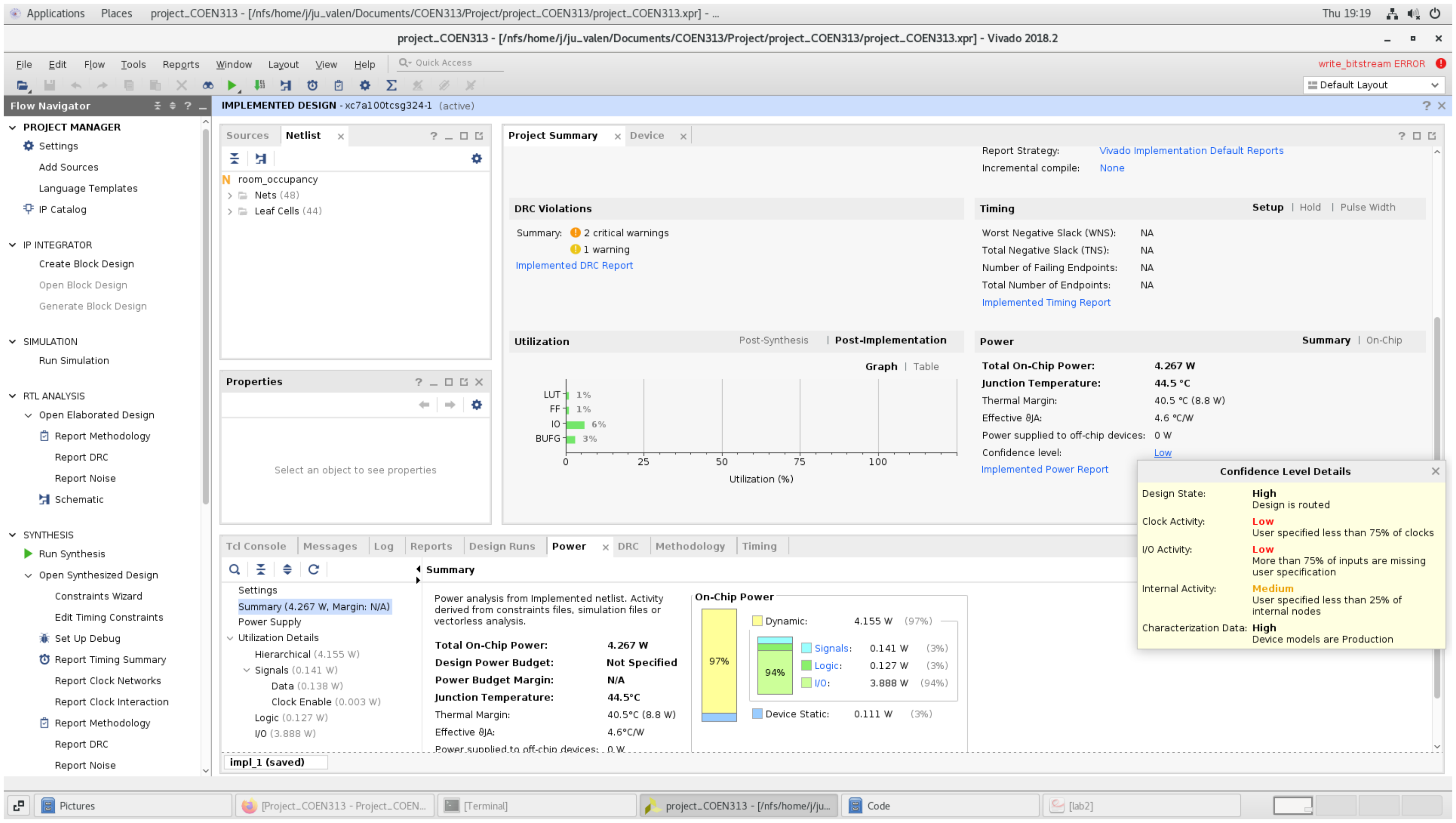


Figure 8: Implemented schematic

Since these schematics are hard to fit in the report, their respective pdf files will be included in the zip file submitted. The Vivado log files for the implementation and synthetization will also be inserted in the zip file, since they are too long to put in the report.

Comments on Quality of the Design

The following figure will show the resource utilization of my design:



Applications Places project_COEN313 - [/nfs/home/j/ju_valen/Documents/COEN313/Project/project_COEN313/project_COEN313.xpr] - ... Thu 19:19

project_COEN313 - [/nfs/home/j/ju_valen/Documents/COEN313/Project/project_COEN313/project_COEN313.xpr] - Vivado 2018.2

File Edit Flow Tools Reports Window Layout View Help Quick Access write_bitstream ERROR

Default Layout

Flow Navigator — IMPLEMENTED DESIGN - xc7a100tcsg324-1 (active)

PROJECT MANAGER
- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR
- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION
- Run Simulation

RTL ANALYSIS
- Open Elaborated Design
  - Report Methodology
  - Report DRC
  - Report Noise
  - Schematic

SYNTHESIS
- Run Synthesis
- Open Synthesized Design
  - Constraints Wizard
  - Edit Timing Constraints
  - Set Up Debug
  - Report Timing Summary
  - Report Clock Networks
  - Report Clock Interaction
  - Report Methodology
  - Report DRC
  - Report Noise

Sources | Netlist

room_occupancy
- Nets (48)
- Leaf Cells (44)

Properties

Select an object to see properties

Project Summary | Device

Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None

DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Timing — Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization — Post-Synthesis | Post-Implementation — Graph | Table

Resource	Utilization (%)
LUT	1%
FF	1%
IO	6%
BUFG	3%

Power — Summary | On-Chip

Total On-Chip Power:	4.267 W
Junction Temperature:	44.5 °C
Thermal Margin:	40.5 °C (8.8 W)
Effective θJA:	4.6 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

Confidence Level Details

Design State:	High Design is routed
Clock Activity:	Low User specified less than 75% of clocks
I/O Activity:	Low More than 75% of inputs are missing user specification
Internal Activity:	Medium User specified less than 25% of internal nodes
Characterization Data:	High Device models are Production

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Methodology | Timing

- Settings
- Summary (4.267 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (4.155 W)
  - Signals (0.141 W)
    - Data (0.138 W)
    - Clock Enable (0.003 W)
  - Logic (0.127 W)
  - I/O (3.888 W)

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	4.267 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	44.5°C
Thermal Margin:	40.5°C (8.8 W)
Effective θJA:	4.6°C/W
Power supplied to off-chip devices:	0 W

On-Chip Power

Dynamic:	4.155 W	(97%)
Signals:	0.141 W	(3%)
Logic:	0.127 W	(3%)
I/O:	3.888 W	(94%)
Device Static:	0.111 W	(3%)

impl_1 (saved)

Pictures | [Project_COEN313 - Project_COEN... | [Terminal] | project_COEN313 - [/nfs/home/j/ju... | Code | [lab2]

Figure 9: Resource utilization

We can observe that the schematics obtained are quite more complicated than the conceptual design drawn by hand. This is represented by the “low” confidence level for the power consumed given by Vivado.

It seems that my initial design was simple enough, however translating it to VHDL meant that more components and conditions needed to be added in order for the circuit to function as expected. 97% of the chip’s power is dynamic, while 3% is static. The overall on-chip power is around 4W, which is not that many watts. The utilization percentages for the LUT, FF, IO and BUFG are also pretty low, the biggest one being 6%. No timing statistics were recorded.

In short, the overall resource utilization of this design is generally pretty low, although there are definitely many ways to optimize my design to use even less resources.

Conclusion

In conclusion, I have completed all of the necessary tasks for this project. For starters, I have drawn a conceptual design of the circuit that needs to be implemented and explained it to make the reader understand my design process. Next, I have provided the VHDL code that was written to accurately represent the initial design. I have then provided a relevant testbench with multiple tests, along with their successful simulation results using Modelsim. The synthesis results (schematics and Vivado log files) were also provided. Finally, I have captured a screenshot of the resource utilization of my design and commented on it, concluding that the design did not use a lot of resources. We can conclude this report by saying that my circuit design can accurately track and monitor the number of occupants in a room.