Faculty of Engineering and Computer Science
Expectations of Originality

This form sets out the requirements for originality for work submitted by students in the Faculty of Engineering and Computer Science. Submissions such as assignments, lab reports, project reports, computer programs and take-home exams must conform to the requirements stated on this form and to the Academic Code of Conduct. The course outline may stipulate additional requirements for the course.

1. Your submissions must be your own original work. Group submissions must be the original work of the students in the group.
2. Direct quotations must not exceed 5% of the content of a report, must be enclosed in quotation marks, and must be attributed to the source by a numerical reference citation[1]. Note that engineering reports rarely contain direct quotations.
3. Material paraphrased or taken from a source must be attributed to the source by a numerical reference citation.
4. Text that is inserted from a web site must be enclosed in quotation marks and attributed to the web site by numerical reference citation.
5. Drawings, diagrams, photos, maps or other visual material taken from a source must be attributed to that source by a numerical reference citation.
6. No part of any assignment, lab report or project report submitted for this course can be submitted for any other course.
7. In preparing your submissions, the work of other past or present students cannot be consulted, used, copied, paraphrased or relied upon in any manner whatsoever.
8. Your submissions must consist entirely of your own or your group's ideas, observations, calculations, information and conclusions, except for statements attributed to sources by numerical citation.
9. Your submissions cannot be edited or revised by any other student.
10. For lab reports, the data must be obtained from your own or your lab group's experimental work.
11. For software, the code must be composed by you or by the group submitting the work, except for code that is attributed to its sources by numerical reference.

You must write one of the following statements on each piece of work that you submit:
For individual work: **"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"**, with your signature, I.D. #, and the date.
For group work: **"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"**, with the signatures and I.D. #s of all the team members and the date.

A signed copy of this form must be submitted to the instructor at the beginning of the semester in each course.

I certify that I have read the requirements set out on this form, and that I am aware of these requirements. I certify that all the work I will submit for this course will comply with these requirements and with additional requirements stated in the course outline.

Course Number:	COEN 313	Instructor:	OTMANE AIT MOHAMED
Name:	JOYAL BIJU KULANGARA	I.D. #	40237314
Signature:	[illegible]	Date:	06th April 2024

[1] Rules for reference citation can be found in "Form and Style" by Patrich MacDonagh and Jack Bordan, fourth edition, May, 2000, available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf.
Approved by the ENCS Faculty Council February 10, 2012

1. Abstract

The goal of this project was to create a design of a digital system for real-time occupancy monitoring in a room using VHDL for the needs of implementation and simulation. By employing photocells at both the entrance and exit to detect entries and exits into the room, and updating the occupancy count accordingly, the system has a programmable threshold that allows for setting the maximum occupancy upon reaching the warning signal. This warning signal is activated to prevent further entries, which is meant to enhance the safety of the regulations linked with capacity regulations in the pre-defined room.

With proper usage of VHDL implantation, the counter architecture can track occupancy, as well as have a rest feature that enables the system to revert to an initial state to accurately track from any period. The design's functionality was properly tested using a testbench that explored various scenarios with random maximum occupancy values, simultaneous entries, and exits, as well as the system's response in resetting signals in different conditions that were defined.

Along with the use of an XDC file, the design was tested for suitability in real-world applications. This research project underlines the effectiveness of digital systems in monitoring occupancy levels, by giving a pathway for scalability and applications for other real-life scenarios in various other environments.

Although not extended to physical hardware, the project included the preparation of an XDC file for potential deployment on a Xilinx Nexys A7 FPGA, demonstrating the design's suitability for real-world applications. This concise study underscores the effectiveness of digital systems in occupancy monitoring, highlighting the potential for scalability and application in various environments.

2. Introduction

In our modern era, efficiently managing space and resources has become a key element of our daily lives. With this concept in mind, developing an advanced digital system that tracks and monitors room occupancy levels is a beacon of innovation. This project report outlines the conceptualization, design, and implementation of a new cutting-edge digital system design, which aims to ensure optimal room occupancy levels. Using VHDL, this digital system emphasizes the use of technology by including highly responsive photocells and programmable signals which are tasked to pave the way for real-time monitoring and control of room occupancy.

The system's design surrounds two primary photocells that are positioned at the entrance and exit points of a room. The role of these photocells is to detect the presence of individuals entering or exiting, which translates into physical movements into binary signals, X and Y respectively. The system's main goal is to maintain a high level of responsiveness and accuracy of these photocells to maintain an accurate count of occupants within the room. A key feature of this system is its programmable maximum occupancy threshold, adjustable through an 8-bit signal. This gives control over the room capacity, ensuring efficient use without compromising safety. Another key feature is the max_capacity signal to prevent further entries upon reaching maximum occupancy, which enables effectively managing space utilization in the room. Also, with the help of a reset function, occupancy tracking was enhanced to revert the system to its initial state enhancing the accurate tracking of occupancy to commence anew, whilst making sure the reliability and utility in managing access to spaces are respected.

Overall, the project objectives outlined in this report include providing a conceptual diagram of the digital system, modeling the circuit using VHDL, as well as creating a relevant testbench to verify the design through the simulation and synthesis results. Additionally, insights into the design's speed and FPGA resource utilization will be discussed based on the obtained results.

3. Results and Discussions

2.1. Conceptual Diagram

The diagrams below are what the VHDL code that I made was based from. These designs provide a large-scale image of how my design works in this system.

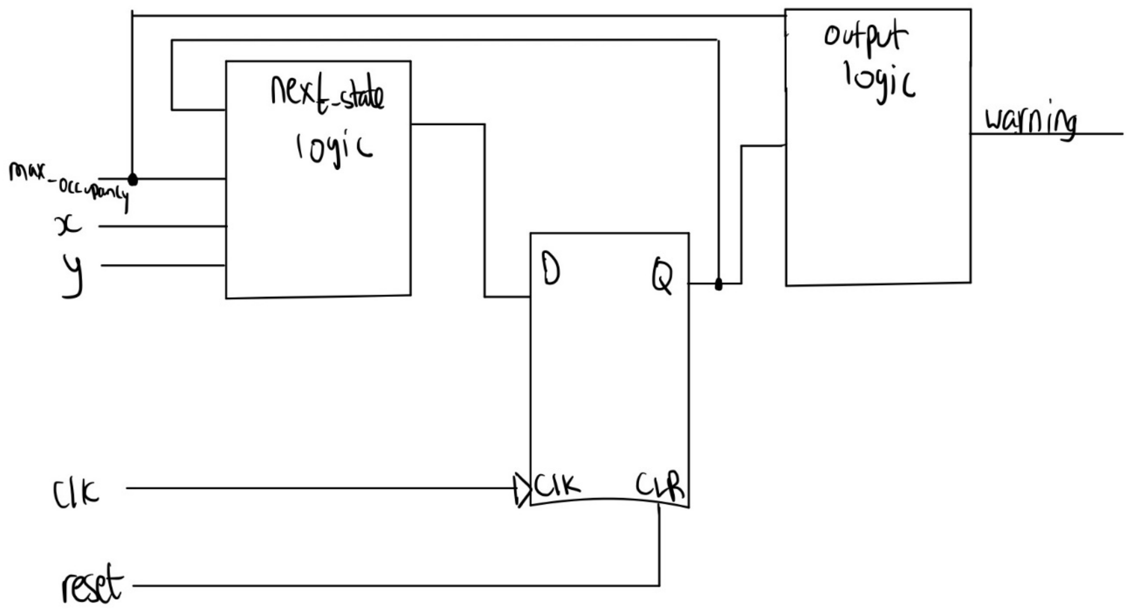


Figure 1. Conceptual Diagram of System

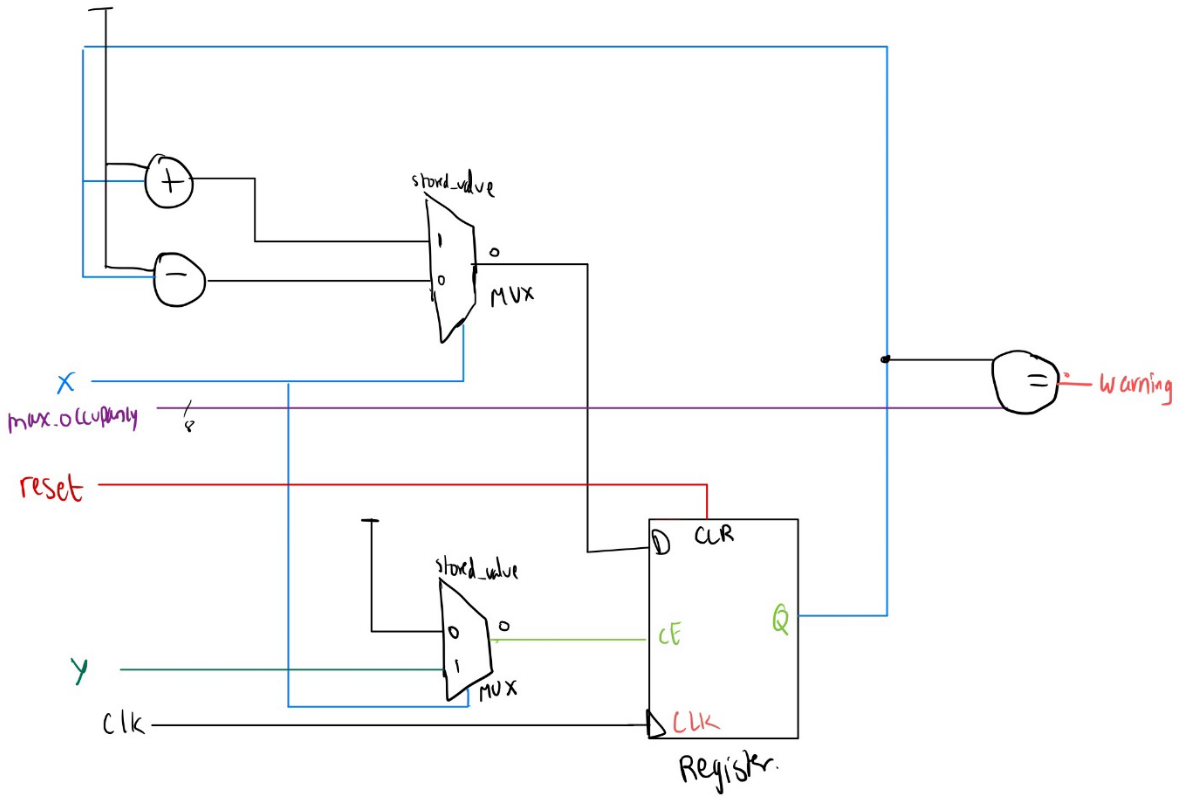


Figure 2. Next-State and Output Logic of Conceptual Diagram

2.2. VHDL Code

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
  port(
    x             : in std_logic;            -- Photocell signal for entrance
    y             : in std_logic;            -- Photocell signal for exit
    clk           : in std_logic;            -- Clock signal
    reset         : in std_logic;            -- Reset signal
    max_occupancy : in std_logic_vector(7 downto 0);  -- Maximum occupancy threshold (8-bit
signal)
    warning       : out std_logic            -- Signal indicating maximum occupancy reached
  );
end counter;

architecture counter_sample of counter is
  signal stored_value : unsigned(7 downto 0) := (others => '0');  -- Counter for tracking
occupancy

begin
  -- Process for updating occupancy counter
  process(clk, reset)
  begin
    if reset = '1' then
      stored_value <= (others => '0');  -- Reset counter to initial state
    elsif rising_edge(clk) then
      if x = '1' then
        stored_value <= stored_value + 1;  -- Increment counter when someone enters the
room
      elsif y = '1' then
        stored_value <= stored_value - 1;  -- Decrement counter when someone exits the room
      end if;
    end if;
  end process;

  -- Process for checking maximum occupancy threshold
  process(stored_value, max_occupancy)
  begin
    if stored_value = unsigned(max_occupancy) then
      warning <= '1';  -- Assert warning signal when occupancy reaches maximum
    else
      warning <= '0';  -- Clear warning signal when occupancy is below maximum
    end if;
  end process;

end architecture counter_sample;
```

Based on the provided code, it can be noted that it represents the simple digital system that is designed to track and monitor room occupancy based on inputs from photocell signals for entrance (x) and exit (y). The system can be broken down into the following specifications based on its key specifications.

1. **Entity Declaration:** The entity named "counter" describes the input and output ports of the design. It includes the inputs for the photocell signals represented by (x and y), the clock signal (clk), the reset signal (reset), and the maximum occupancy threshold (max_occupancy).
2. **The Architecture:** The architecture is named "counter_sample" which defines the behaviour of the digital system.
3. **Counter Initialization:** Inside the architecture, there is a signal declaration for "stored_value" which will act as a counter to track occupancy in the room. This will be initialized initially to be 0.
4. **Clock and Reset Process:** The clock and reset process is a key part of the design. The process block with sensitivity to the clock (clk) and reset signals, which is responsible for updating the occupancy counter. To note, when the reset signal is asserted (reset = "1"), the counter is reset to 0. While on each rising edge of the clock, if the entrance signal x asserts to "1", the counter is incremented by one, it will indicate that someone has entered the room. On the other hand, when the exit signal y asserts to "1", the counter is decremented by one, which shows that someone has exited the room.
5. **Maximum Occupancy(m):** This process block monitors the occupancy counter and compares it with the maximum occupancy threshold provided as the "max_occupancy" signal. If the counter value equals the maximum occupancy threshold, the warning signal is asserted to "1", which indicates that the room has reached its maximum capacity. However, if it does not work, the warning signal is cleared to "0".
6. **Warning:** The warning signal (warning) serves as the output of this program. It is used to let the user know that the maximum occupancy threshold has been reached. It is asserted ("1") when the occupancy counter equals the maximum occupancy threshold, and if not, it will be cleared by ("0"). This signal is used to trigger alerts when the room reaches the maximum capacity, helping the user manage the occupancy level in the room effectively.

2.3. VHDL Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_tb is
end counter_tb;

architecture tb_arch of counter_tb is
  constant CLOCK_PERIOD : time := 10 ns;
  signal x, y, clk, reset, warning : std_logic := '0';
  signal max_occupancy : std_logic_vector(7 downto 0) := (others => '0');
begin
  UUT: entity work.counter
    port map (
      x             => x,
      y             => y,
      clk           => clk,
      reset         => reset,
      max_occupancy => max_occupancy,
      warning       => warning
    );

Clock generation
  clk_process: process
  begin
      clk <= '0';
      while now < 2000 ns loop
          clk <= not clk; wait for CLOCK_PERIOD / 2;
      end loop;
      wait;
  end process;

Stimulus process
  stimulus: process
  begin
      Initial setup
      reset <= '1'; max_occupancy <= "00000100"; -- Testing max occupancy of
4
      wait for CLOCK_PERIOD * 2; reset <= '0';

      Trial 1: Increment the counter within limits
      x <= '1'; wait for CLOCK_PERIOD; x <= '0'; wait for CLOCK_PERIOD;

      Trial 2: Decrement the counter
      y <= '1'; wait for CLOCK_PERIOD; y <= '0'; wait for CLOCK_PERIOD;
      Trial 3: Attempt to exceed the maximum occupancy
      for i in 1 to 5 loop -- 5 attempts to enter
        x <= '1'; wait for CLOCK_PERIOD; x <= '0'; wait for CLOCK_PERIOD;
      end loop;

      Trial 4: Reduce occupancy and attempt re-entry
      y <= '1'; wait for CLOCK_PERIOD; y <= '0'; wait for CLOCK_PERIOD; --
One exit
      x <= '1'; wait for CLOCK_PERIOD; x <= '0'; wait for CLOCK_PERIOD; --
Attempt re-entry

      Trial 5: Rapid entries and exits
      for i in 1 to 3 loop -- Simulate rapid movements
        x <= '1'; wait for CLOCK_PERIOD / 4; x <= '0'; y <= '1'; wait for
CLOCK_PERIOD / 4; y <= '0';
        wait for CLOCK_PERIOD; -- Ensure clock ticks
      end loop;

      Trial 6: Reset and rapid fill to max
      reset <= '1'; wait for CLOCK_PERIOD; reset <= '0'; wait for
CLOCK_PERIOD;
      for i in 1 to 4 loop -- Fill to max quickly
        x <= '1'; wait for CLOCK_PERIOD; x <= '0'; wait for CLOCK_PERIOD;
      end loop;

      Trial 7: Test with reset during operation
      x <= '1'; wait for CLOCK_PERIOD / 2; reset <= '1'; wait for
CLOCK_PERIOD / 2;
      reset <= '0'; x <= '0';

      Trial 8: Trial for max capacity of 2
      reset <= '1'; wait for CLOCK_PERIOD * 2; reset <= '0'; -- Reset the
system
      max_occupancy <= "00000010"; -- Set max capacity to 2
      wait for CLOCK_PERIOD * 2;

      Trial 9: Attempt to fill to max capacity
      x <= '1'; wait for CLOCK_PERIOD; x <= '0'; -- Entry 1
      wait for CLOCK_PERIOD;
      x <= '1'; wait for CLOCK_PERIOD; x <= '0'; -- Entry 2
      wait for CLOCK_PERIOD;

      Trial 10: Attempt to exceed max capacity
      x <= '1'; wait for CLOCK_PERIOD; x <= '0';
      wait for CLOCK_PERIOD;

      Trial 11: Reset for next trial
      reset <= '1'; wait for CLOCK_PERIOD * 2; reset <= '0';
      wait for CLOCK_PERIOD * 2;

      Trial 12: Trial for max capacity of 8
      max_occupancy <= "00001000"; -- Set max capacity to 8
      wait for CLOCK_PERIOD * 2;

      Trial 13: Attempt to fill to max capacity
      for i in 1 to 8 loop
        x <= '1'; wait for CLOCK_PERIOD; x <= '0'; wait for CLOCK_PERIOD;
      end loop;

      Trial 14: Attempt to exceed max capacity
      x <= '1'; wait for CLOCK_PERIOD; x <= '0';
      wait for CLOCK_PERIOD;

      Wait and finish
      wait for 100 ns;
      wait;
  end process;

end tb_arch;
```

Based on the testbench file above, I was able to load the file into Modelsim to get a wave that reflects the different scenarios that I have tried to test through my system. This wave file can be found at **wave_test.bmp** in the code folder, or **Appendix A**.

The testbench's different trial setups included:

- Initial Setup: Where the system is reset, and the maximum occupancy threshold is set to 4.
- Trial 1: Where an entry to increment the counter within limits is simulated.
- Trial 2: Where an exit to decrease the counter is simulated.
- Trial 3: Where an attempt to exceed the maximum occupancy threshold by making multiple entries is simulated.
- Trial 4: Where the reduced occupancy levels and attempted re-entry is simulated and tested.
- Trial 5: Where rapid entries and exits are simulated.
- Trial 6: Where the system resets and quickly fills it to the maximum occupancy.
- Trial 7: When the system tries to test the system behavior with a reset during the operation.
- Trial 8: When the system resets and sets the maximum occupancy threshold to 2.
- Trial 9: When the system attempts to fill to the maximum capacity.
- Trial 10: When the system Attempts to exceed the maximum capacity.
- Trial 11: When the system is reset to test the next trial.
- Trial 12: When the system sets the maximum occupancy threshold to 8.
- Trial 13: When the system tries to fill to the maximum capacity.
- Trial 14: When the system attempts to exceed the maximum capacity.

This testbench provides some comprehensive stimulus scenarios to verify the functionality and robustness of the counter design, by handling various occupancy levels and edge cases. This allows for thorough testing of the project, and validation of the digital system's behavior before it is implemented.

2.4. Simulation and Synthesis Results

For this project, simulation and synthesis were conducted using Xilinx Vivado targeting the Nexys A7 FPGA development board. Without an actual FPGA implementation, I was able to perform both a simulation and synthesis of my design to test the accuracy of my project design. Using an XDC file that was set as the constraints file to mimic the FPGA implementation needed for Xilinx Vivado, I was able to carry out the synthesis and implantation on Vivado. This file can be found in the project folder under the name **counter.xdc.**

Included in the report folder are the Vivado log files containing the synthesis and implementation results, which provide detailed information about the synthesis process, resource utilization, and timing constraints analysis that were carried out by Vivado. These files can be found in the folder under the names **runme_synthesis.log** and **runme_implentation.log.**

A schematic of the implemented design process from Vivado shows the processes and signals used in the VHDL system design and has been attached to **Appendix Part B.**

These results serve as valuable indicators of the design's feasibility and performance characteristics for potential FPGA implementation in future stages of implementing the design in real life.

2.5. Quality of Design

Based on the synthesis results obtained from Vivado, the quality of the design can be evaluated in terms of both speed and FPGA resource utilization.

The speed of the design can be assessed based on the maximum achievable clock frequency, which can be known as the clock period of the system test. In this system, lower clock periods indicate higher operating frequencies and faster performance. If the design does meet the timing constraints and achieves a high clock frequency, it would mean that the design is properly optimized and capable of operating within the FPGA device to be simulated. Also, the design's speed based on the implementation and synthesis shows that it was working at a high efficiency, which is a good sign about this project design. Also, my design is optimized for speed, due to the usage of two MUXs.

Based on Resource Utilization, the FPGA uses hardware resources connected by pins that are shown through the design including the flipflops, Look-Up Tables (LUTs), memory blocks, and other logic elements. Efficient resource utilization can be portrayed with the compact design of the system by showing that the design minimizes scalability and wastage of resources. Inefficiencies and limitations can still be addressed when the design is optimized for larger FPGA devices that need more resources, if the utilization of exceeds the available resources on the FPGA device.

Overall, based on the design that I implemented, I can say that based on my synthesis and implementation results, along with the trials in Modelsim, the resource has satisfactory speed performance, whilst meeting time constraints at a high clock frequency whilst at the same time minimizing FPGA resource consumption.

4. Conclusion

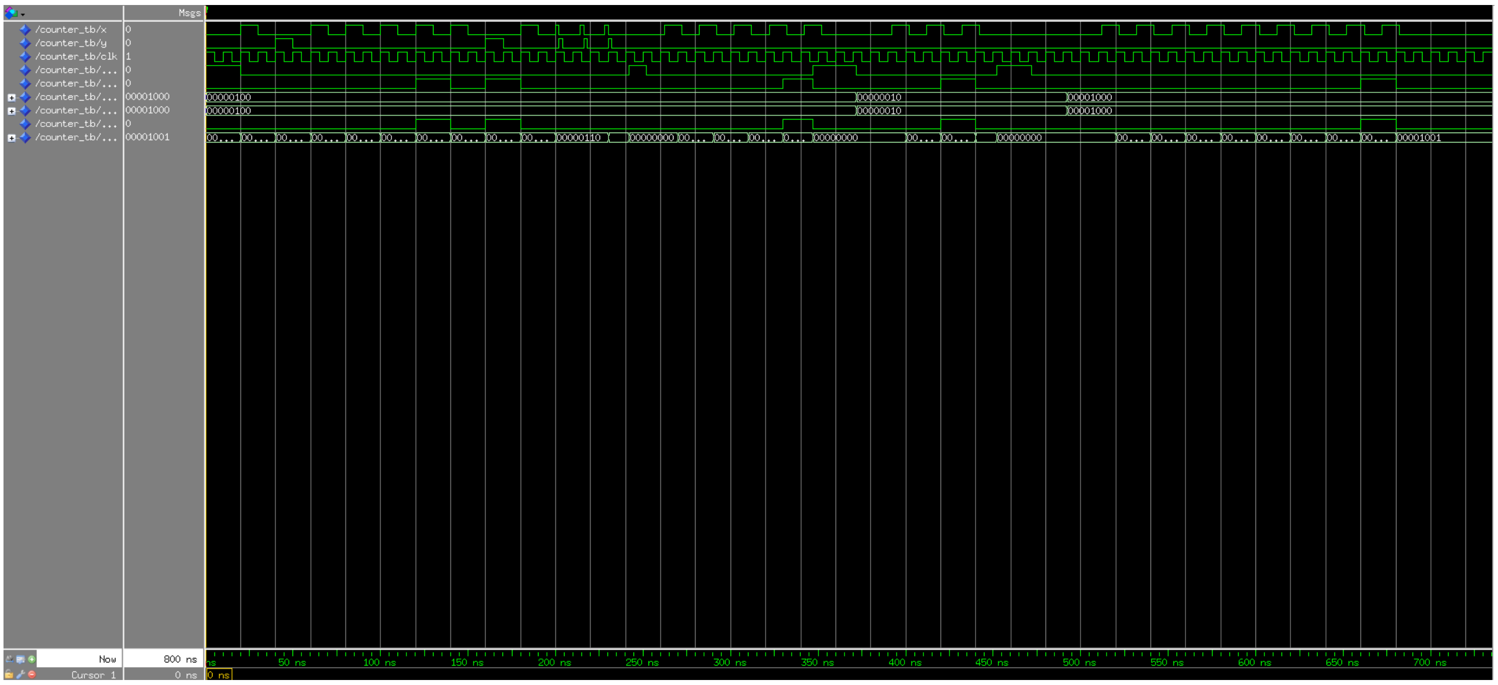
This report successfully outlines the design, VHDL implementation and the testing of a digital system for room occupancy tracking. The use of photocells at entry and exit points shows how the system accurately monitors room occupancy and asserts the warning when maximum capacity is reached in the room. The VHDL code models this behavior and uses a reset feature to reset the system and reinitialize it.

The use of the testbench verifies the system in different scenarios that were outlined in the report including random maximum occupancy levels, simultaneous entries, and exits for three different trials including when there are 2 people, 4 people, and 8 people entering a room. Although, without the FPGA implementation, the XDC file helped in developing synthesis and implementation results on Xilinx Nexys A7 FPGA using Vivado, indicating the design efficiency for real-world applications.

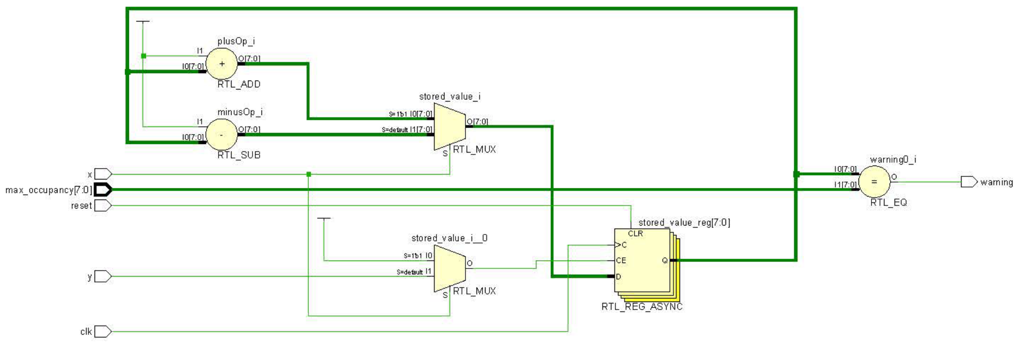
In conclusion, the design fully demonstrates the accuracy of occupancy tracking, portraying the potential for diverse applications to use this programmable occupancy threshold in real life, with proper validation through simulation. This report outlines the validity of the design's functionality and performance, as well as highlighting the potential of this program in terms of adaptation and use in this application beyond the initial scope.

5. Appendix

A. Modelsim Wave



B. Vivado Implementation schematic


plusOp_i
I1
I0[7:0]
O[7:0]
RTL_ADD
minusOp_i
RTL_SUB
stored_value_i
S=1'b1 I0[7:0]
S=default I1[7:0]
O[7:0]
S
RTL_MUX
x
max_occupancy[7:0]
reset
y
clk
stored_value_i__0
S=1'b1 I0
S=default I1
O
RTL_MUX
stored_value_reg[7:0]
CLR
C
CE
D
Q
RTL_REG_ASYNC
warning0_i
I0[7:0]
I1[7:0]
=
O
RTL_EQ
warning