**Report**

**introduction**:

In the realm of digital electronics, the design and implementation of circuits for counting and displaying numerical values hold significant importance across various applications. From basic timekeeping devices to complex control systems, counters play a pivotal role in monitoring and managing numerical data. In this project, we delve into the creation of a digital circuit aimed at counting and displaying values on a 7-segment display, while incorporating functionalities for both up and down counting mechanisms.

**Project Overview:**

The primary objective of this project is to develop a versatile digital counter capable of incrementing and decrementing numerical values based on user inputs. The design incorporates a finite state machine (FSM) to control the behavior of the counter, allowing seamless transitions between different states to facilitate up and down counting. Additionally, the circuit includes a clock divider module to generate a slower clock signal, essential for synchronizing the counting process and ensuring accurate display updates.

A crucial aspect of the project is the integration of a 7-segment display decoder, responsible for translating binary count values into signals compatible with the display hardware. This enables the visualization of numerical counts in a human-readable format, enhancing the usability and applicability of the circuit in real-world scenarios.In the subsequent sections, we delve deeper into the individual components and operational details of the circuit, elucidating the design choices and implementation strategies employed to achieve the desired functionality. Through rigorous analysis and testing, we aim to validate the performance and reliability of the circuit, ensuring its suitability for a wide range of digital counting applications.

|  |  |  |
| --- | --- | --- |
| **Input/Output** | **Description** | **Possible Values** |
| up | Signal indicating if the back-end queue is clear | Logic '0' (interrupted) or '1' (clear) |
| down | Signal indicating if the front-end queue is clear | Logic '0' (interrupted) or '1' (clear) |
| Counter(pcount) | Number of people standing in the queue | Integer, incremented or decremented by 1 |
| Tcount | Number of tellers currently in service | Integer, 1, 2, or 3 |
| Reset | Signal to reset the system | Logic '1' (reset) or '0' (normal) |
| Empty flag | Flag indicating if the queue is empty | Logic '1' (empty) or '0' (not empty) |
| Full flag | Flag indicating if the queue is full | Logic '1' (full) or '0' (not full) |
| Wtime | Expected waiting time in the queue before being served | Integer, calculated based on Pcount and Tcount |

**1.Clock Divider (clock\_divider1):**

This module divides the clock frequency to generate a slower clock (slow\_clk) signal.

It uses a counter (count) to count the number of clock cycles and toggles slow\_clk every 25,000,000 clock cycles.

**2.Flip-Flop (FF):**

This is a simple D flip-flop module. It stores the value of input d on the rising edge of the clock and outputs it as q.

**3.This module implements a 3-bit up-down counter.**

It increments or decrements the counter based on the up and down inputs and resets the counter when necessary.

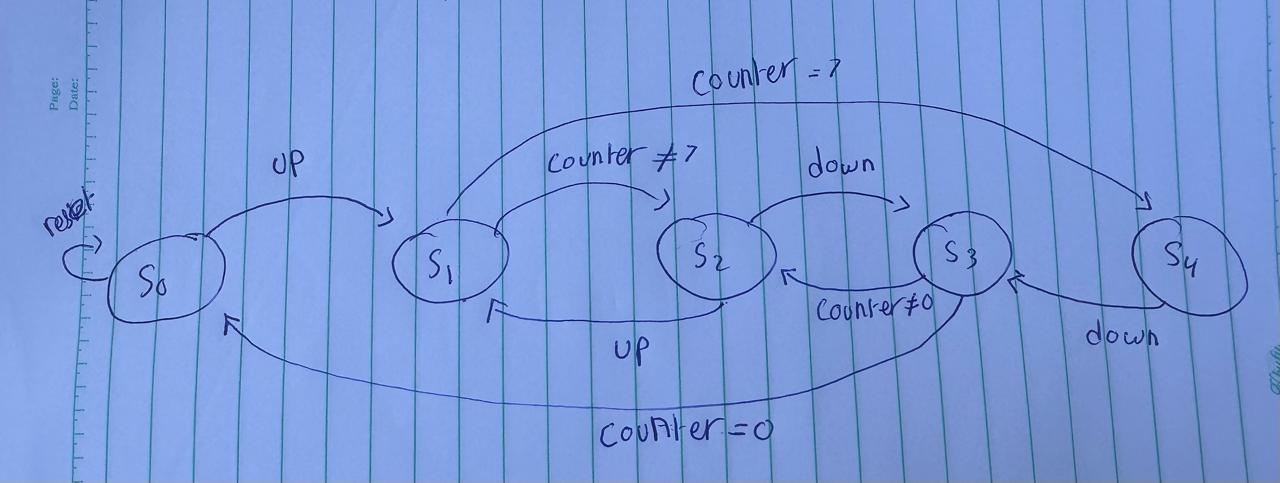
**4.Finite State Machine (FSM):**

This module defines a finite state machine with five states (s0 to s4) to control the behavior of the counter.

It transitions between states based on the current state and the inputs (up and down).

It sets flags (full and empty) based on the current state and the counter value.

At s0 the flag is empty and in s1 to check if the counter is full or not when someone gets in if not full it goes to s2 else it goes to s4, in s2 the flag is not full and not empty ,in s3 it check if the counter is empty or not when someone gets out if empty it return to s0



**5.Read-Only Memory (ROM):**

This module serves as a ROM, storing values for wtime based on the inputs pcount and tcount.

The output wtime depends on the values of pcount and tcount, which are used as address inputs to fetch data from ROM.

7.**Edge detector (posege\_detector):**

This module implements an edge detector has two inputs and one output. The design aims to detect the positive edge of input (siginal), and output (out). So we expect to see a pulse on (out) whenever (siginal) changes from value 0 to 1. It's used to synchronize signals in digital circuits. A positive edge detector will send out a pulse whenever the signal it is monitoring changes from 0 to 1 (positive edge). The idea behind a positive edge detector is to delay the original signal by one clock cycle, take its inverse and perform a logical AND with the original signal.

A screenshot of a computer

Description automatically generated

A diagram of a circuit

Description automatically generated

**6.Decoder for 7-Segment Display (decoder\_7seg):**

This module decodes the input binary values into signals for a 7-segment display.

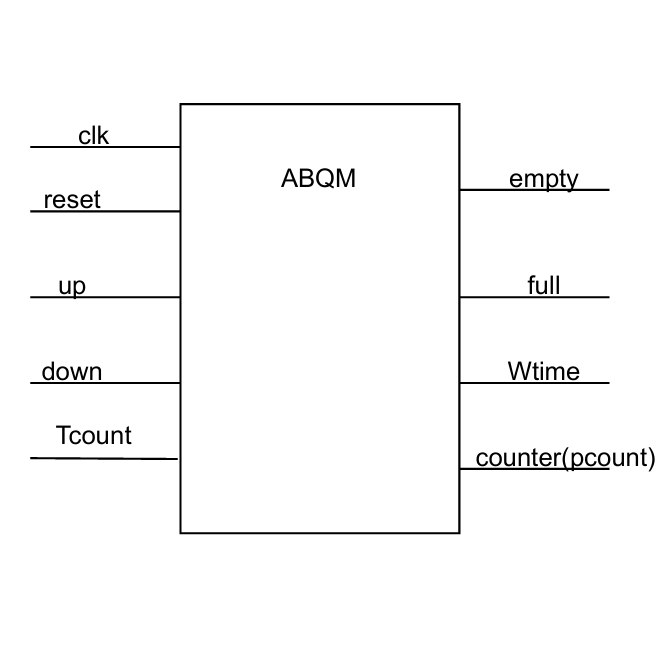
It takes input signals corresponding to binary digits and outputs signals for the segments of the display.

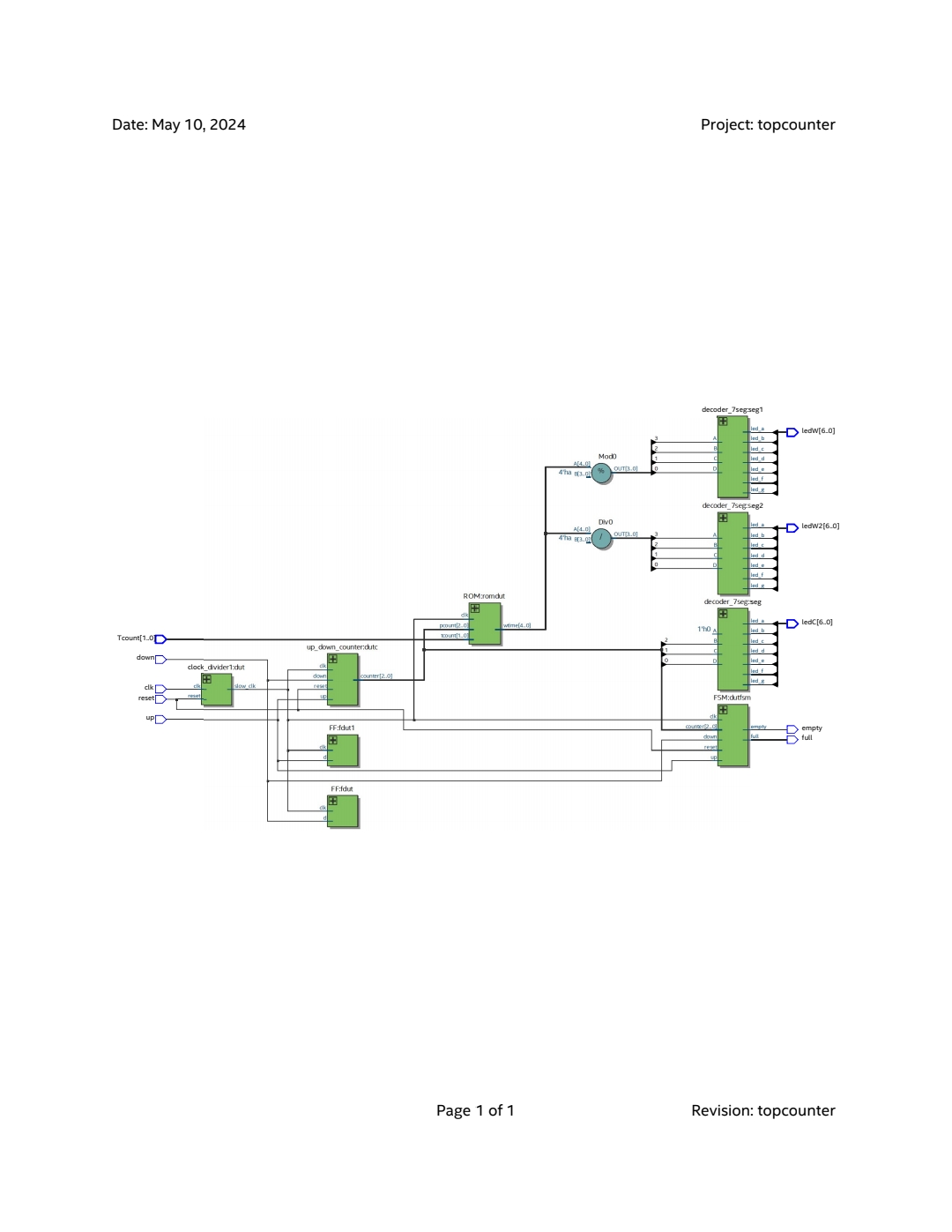
**7.Top Module (ABQM and topcounterTest):**

ABQM integrates all the modules together.

ABQMTest is a testbench module used to simulate the behavior of ABQM.

It instantiates the other modules and provides stimulus to observe the functionality.



It instantiates the other modules and provides stimulus to observe the functionality.

**8.Test Bench (test\_ABQM):**

This module contains the test scenario for the design.

It provides stimulus to the DUT (Design Under Test) and monitors its outputs.Test Bench (test\_ABQM):

This module contains the test scenario for the design.

It provides stimulus to the DUT (Design Under Test) and monitors its outputs.

A screenshot of a computer

Description automatically generated

**conclusion** : The overall functionality appears to be a counter with up and down control, displaying the count on a 7-segment display and managing flags based on the count and the state of the counter.

**Team Members:** Malak yasser aly 22100400

Omar Ashraf 22102210

Fares Hassan Hamed 221010963