Liam Tang

202135591

Electrical and Electronic Engineering

EE270

Digital Electronics

4-bit Bouncing Counter

18/11/22

**I confirm and declare that this report and the assignment work is entirely the product of my own efforts and I have not used or presented the work of others herein.**

[1. Executive Summary 1](#_Toc120737943)

[2. Introduction 1](#_Toc120737944)

[3. Review of Specification 1](#_Toc120737945)

[4. Design 2](#_Toc120737946)

[4.1 Design Improvements 5](#_Toc120737947)

[5. Testing, Simulation & Results 6](#_Toc120737948)

[5.1 Test Strategy 6](#_Toc120737949)

[5.2 Results and Analysis 6](#_Toc120737950)

[6. Conclusions 8](#_Toc120737951)

[7. References 8](#_Toc120737952)

# Executive Summary

The aim of this report is to explore the functionality and construction of a 4-bit Bouncing Counter which should generate a specific output sequence. The 4-bit counter should reset to an initial value when the Reset input is called upon. Using a State Transition Table and Karnaugh Maps, equations for J and K values for each flip-flop were derived which served as blueprints for the circuit assembly process. The design included 2 inputs for counting and resetting, a clock pulse to trigger the flip-flops, a concatenator which combined the 4 binary outputs and converted them into a total decimal output, and logic gates which generated the correct inputs for the flip-flops. The design functioned correctly and met the requirements specified but could be improved by defining the invalid states to ensure that the design can withstand a wider range of inputs.

# Introduction

This report aims to explore the functionality and construction of a 4-bit Bouncing Counter by using Karnaugh Maps and State Transition Tables to generate next state equations. The Microsoft Excel application was used to construct the State Transition Table and the Simulink application was used to build the final circuit design. Once the final design was built, a range of appropriate test scenarios were used to examine the robustness and functionality of the design.

# Review of Specification

Timeline

Description automatically generated with low confidence

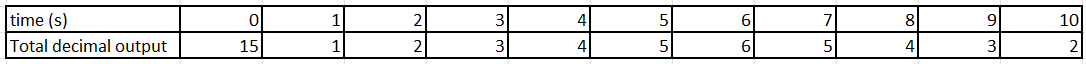
Figure 1: 4-Bit bouncing counter diagram

[1]

Figure 1 contains a diagram of the 4-Bit counter that was required to be designed. The design features 2 input signals, Count (X1) and Reset (X0), and a clock signal which is used to trigger the flip-flops on falling edges. Z3, Z2, Z1 and Z0 are outputs, with Z3 being the most significant. Each of the outputs represent a single bit of the total binary output. The design must be a of Moore machine type. A Moore machine is where the inputs are not tied to the outputs.

When the Count input signal is constantly high and the Reset input is constantly low, the counter must generate an output sequence that has an initial value of 15. In increments of 1, it then must count up 5 from a start value then count back down to the start value in reductions of 1. The start value is determined by the last digit in the registration number which, in this instance, is 1. Table 1 displays the total decimal output for one cycle of the circuitry response. Note that the initial output, 15, only features when counting begins at 0 seconds or when the Reset input is high.

Table 1: 1 cycle of counting output sequence



# Design

The design required 11 states to satisfy the output sequence as shown in Table 1. Each state has an output that corresponds to a member in the output sequence. 2 states would be required for outputs 2, 3, 4 and 5 as there are two members of the list for each of those outputs – one with a next state that outputs a number above (counts up) and one with a next state that outputs a number below (counts down).

To accommodate the 11 states required in the design, negative-edge-triggered J K flip-flops were used since the design specified that the flip-flops should trigger on falling edges of the clock pulse. The number of flip-flops governs the number of states that exist. This is shown in (1).

, *where n denotes the number of flip-flops*  (1)

Since the design only required 11 states, and 4 J K flip-flops produces 16 states, the remaining 5 states were omitted from the State Transition Table and are left undefined as ‘don’t care’ terms. The 5 unused states will never be reached since they are not defined as the next state of the 11 used states.

A State Transition Table was constructed in order to visualise the behaviour of the system and determine the necessary J and K values required in the design, as shown in Table 2 below.

A picture containing text, shoji, building

Description automatically generatedTable 2: State Transition Table

The State Transition Table shown in Table 2 displays the logic for each flip-flop output (Q3, Q2, Q1, Q0 with Q3 being the flip-flop associated with the output of the most significant bit). It also displays the logic for the next state for each flip-flop as well as the J and K values required to achieve the next state. It was logical to define each state such that its next state had an output that was the subsequent number in the counting output sequence (i.e. state ‘4 – up’ has an output of 4 and has next state ‘5 – up’ which outputs 5).

Two inputs, X1 (Count) and X0 (Reset) were used to control the counting process. It was decided that the invalid states, when X1 = X0 = 1, were defined as ‘don’t cares’ in the State Transition Table as the design brief did not specify that the test sequence demanded coverage of all input combinations. The J and K columns in the transition table were populated using the truth table shown in Figure 2. Using the IF and AND logic functions in Excel, a formula was derived to autocomplete the J and K columns. (2) displays the formula used for the column J3. The cell coordinates and outputs were modified accordingly for each J and K column.

(2)

A picture containing chart

Description automatically generated

Table

Description automatically generated with low confidence

Figure 2: JK truth table and next state equation

Figure 3: 3x3 K map for J3

The truth table for a J K flip-flop, shown in Figure 2, was determined by analysing the behaviour of J and K by using the different binary combinations of Q and Q\*, where Q denotes the current state and Q\* denotes the next state. Figure 3 contains the equation for J3 as well as the K map used to calculate it. To plot the 6 variable K map, headings Q3Q2Q1 were placed on the x-axis and Q0X1X0 were placed on the y-axis. The logic for each Q3Q2Q1Q0X1X0 combination, found in the State Transition Table, was used to populate the K map. To generate an equation for J3, the cells with a value of 1 must be grouped in sizes that are powers of 2 (1,2,4,8…). Neighbouring cells can be grouped together so long as they are vertically or horizontally adjacent to each other. Cells at the border of the table can be grouped with cells at the opposite border of the table. Don’t care terms are useful as they are not required to be part of a group but can be grouped together with 1s in order to make groups larger and hence make the equation simpler. Once all the 1s in the K map have been grouped, a simple Boolean equation is generated which acts as the final Sum of Products equation for J3. This describes the logic for J3 as it responds to different Q3Q2Q1Q0X1X0 logic. In instances where more than 1 group is formed, the group equations are ‘ORed’ together to produce the Sum of Products equation.

This process of plotting a 6 variable K map was repeated for the other J and K values and outputs Z3, Z2, Z1, Z0 in the state transition table. This resulted in the following equations:

(3) (4)

(5) (6)

(7) (8)

(9) (10)

(11) (12)

(13) (14)

Diagram, engineering drawing

Description automatically generated

Figure 4: Circuitry Diagram

Simulink, a circuit analysis application, was used to build the circuit in Figure 4. Equations 3-14 were used as the blueprints to build the circuit as they describe the inputs required in each J and K terminal for the flip-flops and terminals for the Concat operator. The Concat operator was used to combine the individual bit outputs into a total decimal output. This made it easier to analyse the results of circuit.

## Design Improvements

An improvement to the design would be to define the invalid state. This could be defined such that the next state of any invalid state would be the same as its present state – ensuring that nothing changes when the invalid state is reached. The invalid states would also be defined so that the outputs are the same as the other outputs in their subsection of the table. The new definition of the invalid state would effectively serve as a ‘don’t change’ state.

However, defining the invalid state would lead to more complicated equations for the next state, JK, and output equations since less don’t care terms would appear in the K maps. This would require a more complicated circuit design and would be more costly than the initial design if constructed in real life.

Use of more logic gates would also lead to a longer propagation delay. Propagation delay is the time taken for a signal to pass through a component and is usually a very small time value. With more components in the circuit, the propagation delay is larger and could result in signals arriving at the terminals of logic gates asynchronously which could potentially lead to errors. However, this issue would only arise if the design was implemented in real life. Propagation delay does not exist in the simulations conducted in this report.

# Testing, Simulation & Results

A variety of test inputs were used to test the robustness and functionality of the 4-Bit Bouncing counter. The signal editor component was used for the X1 and X0 input signals. A range of test scenarios were constructed within the signal editor component.

## Test Strategy

Multiple test strategies were implemented to test the functionality of the circuitry design. The simulation results were compared with theoretical results in order to identify errors in the design. The test strategies can be found in Table 3.

A screenshot of a computer

Description automatically generated with medium confidenceTable 3: Test Strategies

‘Count’ was designed to test only the counting functionality of the circuit. The output sequence should align with Table 1. ‘Stay’ was used to test the response of the circuit when X1 = X0 = 0. The output should not change from the initial state. ‘Count to 6’ tests the reset input X0. The output should count to 6 and stay. ‘Count to 5 – reset’ tests the count input (X1) after the reset input (X0) is applied. The output should count to 5 before resetting to the initial state (output 15) and should commence counting again from 15.

The invalid input X1=X0=1 is never tested since states with this input were not defined in the State Transition Table and instead is replaced by don’t care terms. Testing the invalid input would lead to unpredictable next states so the test strategy was designed to only test defined states.

## Results and Analysis

The clock pulse waveform in Figure 5 was used to activate the flip-flops. The pulse was configured to have a period of 1 second. This ensured that the falling edges would occur at 1 second intervals, triggering the flip-flops at 1 second intervals, which made it easier to visualise the behaviour of the circuit.

Chart, histogram

Description automatically generated

Figure 5: clock pulse waveform

The J K flip-flops are negatively-edge-triggered and will trigger at the falling edges of the clock pulse (namely at seconds 1, 2, 3, 4, 5, 6, 7, 8, 9, 10). This will enable 11 states to be reached during testing.

Figures 6, 7, 8, and 9 display the total binary outputs for each of the test inputs.

Chart, table

Description automatically generated with medium confidenceChart, scatter chart

Description automatically generated

Figure 7: total binary output for ‘Stay’

Figure 6: total binary output for ‘Count’

Chart

Description automatically generatedChart

Description automatically generated

Figure 9: total binary output for ‘Count to 5 – reset’

Figure 8: total binary output for ‘Count to 6’

Figure 6 displays the output for the ‘Count’ test input. It shows that the 4-Bit Bouncing counter outputs the sequence 15,1,2,3,4,5,6,5,4,3,2 which is expected since the Count input (X1) is always high, and the Reset input (X0) is always low.

Figure 7 displays the output for the ‘Stay’ test input. The output sequence of 15,15,15,15,15,15,15,15,15,15,15 is expected as both the Count and Reset inputs are always low – hence the next state should equal the current state, and the output should remain constant.

Figure 8 contains the output sequence for the ‘Count to 6’ test input. The output sequence of 15,1,2,3,4,5,6,6,6,6,6 is expected and shows that the circuit counts to 6 and remains at 6 until the simulation is over. This is because the Count input toggles to low at 6 seconds and remains low while the Reset does not change its logic.

Figure 9 displays the waveform for the ‘Count to 5 – reset’ test input. The figure displays the correct output sequence of 15,1,2,3,4,5,15,1,2,3,4 and shows that the circuit counts up to 5 before resetting to 15 and begins the counting process again.

All the output waveforms align with the predictions made in the Test Strategy section which indicates that the circuit is functioning correctly.

# Conclusions

The purpose of this report was to explore the functionality and design of a 4-Bit bouncing counter. Using powerful techniques like plotting Karnaugh Maps and State Transition Tables, a functioning 4-Bit bouncing counter was designed which satisfied the Specification of Design. The proposed circuitry design included 2 input signals Count (X1) and Reset (X0) and also included a clock pulse signal to activate 4 negatively-edge-triggered JK flip-flops. A State Transition Table was constructed such that each state had an output that was a member of the output sequence in Table 1. J and K binary values were obtained using a truth table which was derived from the next state equation of a JK flip-flop. Equations for each J, K, and Z values were generated using Karnaugh Maps. These equations were used as a blueprint to build the final circuit design on Simulink. The Test Strategies show that the 4-Bit counter responds to its Count and Reset inputs correctly and can generate the output sequence mentioned in the Specification of Design which can be found in Table 1.

Potential improvements to the design process include defining the invalid states (when X1=X0=0) so that errors will not occur when invalid states are reached. However, this would lead to more complicated J, K, and output equations which would result in heavier circuitry design with more logic gates required in the design.

# References

[1] Department of Electrical and Electronic Engineering, EE270 Digital Electronic Systems: Assignment Brief Cover Sheet, Page 1, Glasgow: University of Strathclyde, 2022