CSSE2010 Assignment 1 – Part 2

**Name:** Hugo Burton

**Student Number:** 46985123 (s4698512)

**Due Date:** 1pm, Thursday 26­th August 2021

# Design

## State Diagram

In the process of designing a circuit to detect the length of a word entered by a user, the different states of the machine first need to be determined.

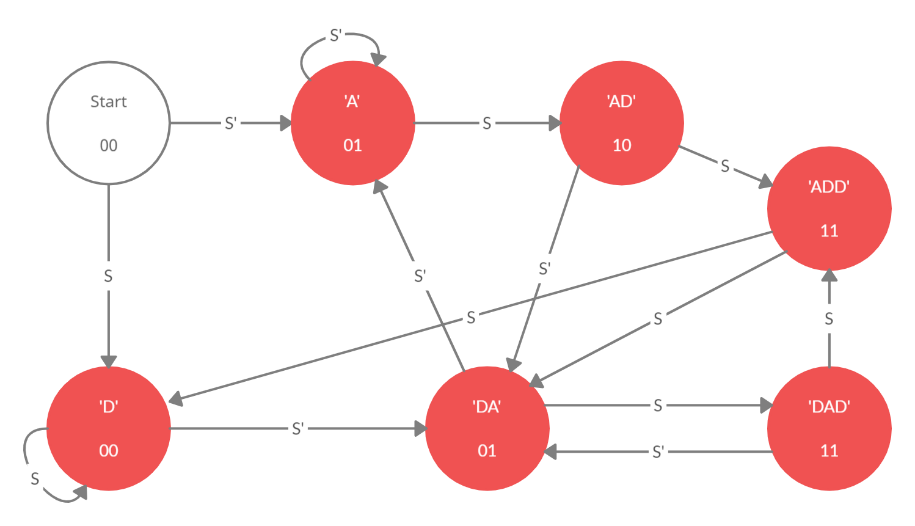
As specified in the task sheet, the circuit must detect the following words (as described in the table) and output the length of these words:

|  |  |
| --- | --- |
| **Word** | **Word Length (Output)** |
| “A” | 1 |
| “AD” | 2 |
| “ADD” | 3 |
| “DAD” | 3 |

However, these are **not** all the states the circuit can exhibit. For example, in order to reach a state of “DAD”, the circuit must be in the state of “D” then “DA” before it can be in a state of “DAD”. Additionally, there must be a “Start” or empty state when the circuit is first powered on which indicates no characters have been inputted (i.e. the clock has not yet had its first rising edge).

Therefore, the possible states this machine can be in are defined below in the state diagram. The arrows represent the possible transitions that occur with and representing the inputs of “D” and “A” respectively. Note that the output for the length will use two bits and follow the following encoding

|  |  |
| --- | --- |
| Word Length | Encoding |
| 0 | 00 |
| 1 | 01 |
| 2 | 10 |
| 3 | 11 |



## Two-Dimensional (2D) State Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current State** | **Next State** | | **Outputs** | |
|  | **S** | **S'** | **X1** | **X0** |
| Start | "D" | "A" | 0 | 0 |
| "A" | "AD" | "A" | 0 | 1 |
| "AD" | "ADD" | "DA" | 1 | 0 |
| "ADD" | "D" | "DA" | 1 | 1 |
| "D" | "D" | "DA" | 0 | 0 |
| "DA" | "DAD" | "A" | 0 | 1 |
| "DAD" | "ADD" | "DA" | 1 | 1 |

## State Encodings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| One-Hot Encoding | |  | Binary Encoding | |
| State | Encoding |  | State | Encoding |
| Start | 0000001 |  | Start | 000 |
| "A" | 0000010 |  | "A" | 001 |
| "AD" | 0000100 |  | "AD" | 010 |
| "ADD" | 0001000 |  | "ADD" | 011 |
| "D" | 0010000 |  | "D" | 100 |
| "DA" | 0100000 |  | "DA" | 101 |
| "DAD" | 1000000 |  | "DAD" | 110 |
|  |  |  | Undefined | 111 |

## State Tables for Encodings

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| One Hot-Encoding | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| State Name | **Encoding** | | | | | | | Input | **Next State** | | | | | | | **Outputs** | |
|  | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 | S | D6 | D5 | D4 | D3 | D2 | D1 | D0 | X1 | X0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 1 | S | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ~S | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| "A" | 0 | 0 | 0 | 0 | 0 | 1 | 0 | S | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| "A" | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ~S | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| "AD" | 0 | 0 | 0 | 0 | 1 | 0 | 0 | S | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| "AD" | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ~S | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| "ADD" | 0 | 0 | 0 | 1 | 0 | 0 | 0 | S | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| "ADD" | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ~S | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| "D" | 0 | 0 | 1 | 0 | 0 | 0 | |  | | --- | | 0 | | S | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| "D" | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ~S | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| "DA" | 0 | 1 | 0 | 0 | 0 | 0 | 0 | S | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| "DA" | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ~S | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| "DAD" | 1 | 0 | 0 | 0 | 0 | 0 | 0 | S | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| "DAD" | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ~S | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Binary Encoding | |  |  |  |  |  |  |  |  |
| **State Name** | **Encoding** | | | **Input** | **Next State** | | | **Outputs** | |
|  | Q2 | Q1 | Q0 | S | D2 | D1 | D0 | X1 | X0 |
| Start | 0 | 0 | 0 | S | 1 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | ~S | 0 | 0 | 1 | 0 | 0 |
| "A" | 0 | 0 | 1 | S | 0 | 1 | 0 | 0 | 1 |
| "A" | 0 | 0 | 1 | ~S | 0 | 0 | 1 | 0 | 1 |
| "AD" | 0 | 1 | 0 | S | 0 | 1 | 1 | 1 | 0 |
| "AD" | 0 | 1 | 0 | ~S | 1 | 0 | 1 | 1 | 0 |
| "ADD" | 0 | 1 | 1 | S | 1 | 0 | 0 | 1 | 1 |
| "ADD" | 0 | 1 | 1 | ~S | 1 | 0 | 1 | 1 | 1 |
| "D" | 1 | 0 | 0 | S | 1 | 0 | 0 | 0 | 0 |
| "D" | 1 | 0 | 0 | ~S | 1 | 0 | 1 | 0 | 0 |
| "DA" | 1 | 0 | 1 | S | 1 | 1 | 0 | 0 | 1 |
| "DA" | 1 | 0 | 1 | ~S | 0 | 0 | 1 | 0 | 1 |
| "DAD" | 1 | 1 | 0 | S | 0 | 1 | 1 | 1 | 1 |
| "DAD" | 1 | 1 | 0 | ~S | 1 | 0 | 1 | 1 | 1 |
| Undefined | 1 | 1 | 1 | S | 1 | 0 | 0 | 0 | 0 |
| Undefined | 1 | 1 | 1 | ~S | 0 | 0 | 1 | 0 | 0 |

## Boolean Logic

### One Hot Encoding

when

when

never

when

when

when

when

when

when

### Binary Encoding

when

when

when

when

when

## Chosen Encoding

Choose Binary Encoding:

* ­Uses fewer flip-flops (3) over one-hot encoding (7)
* Boolean logic for the outputs as well as for the next state logic to is far less complex with binary encoding. This is because the logic for one-hot encoding must deal with more bits than in binary encoding which is inefficient for this circuit.
* One-hot encoding would require a lot of not gates because only one can be a at any given time. Therefore, in the sum of products, each of the remaining outputs need to be negated (if not using flip flops with a output.

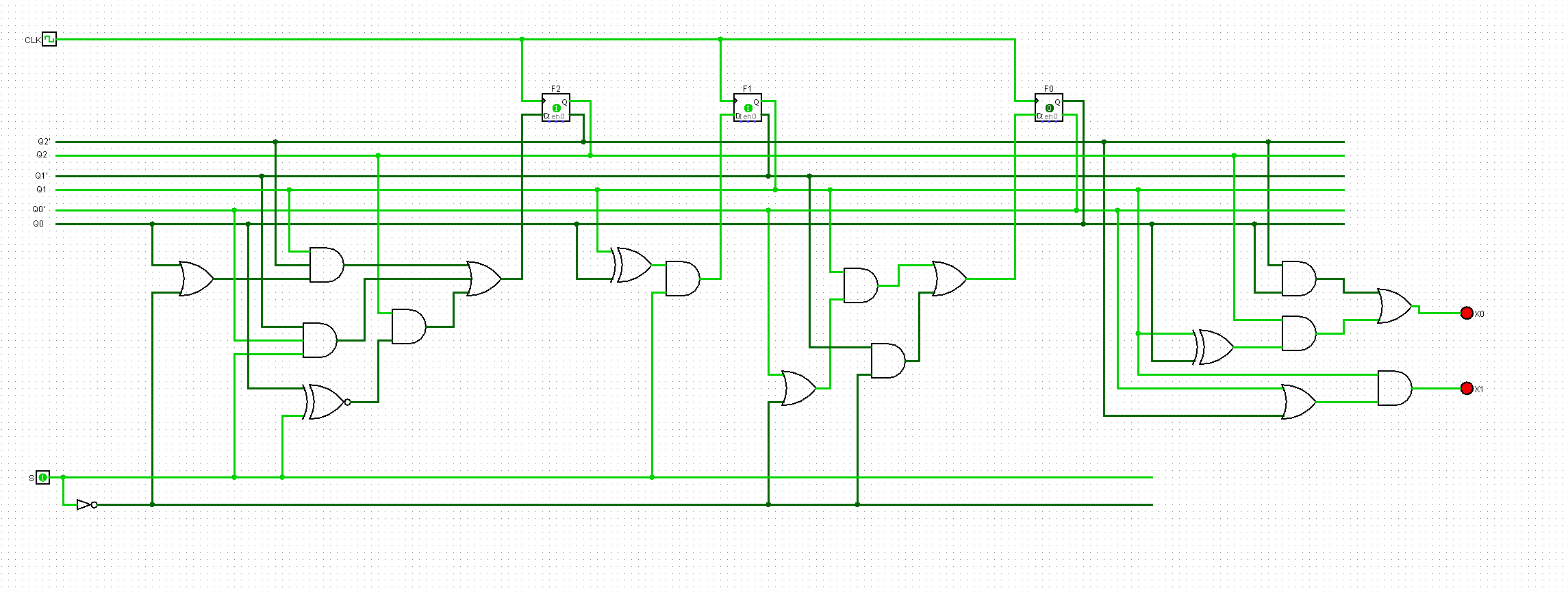
## Logic Diagram

# Simulation

The circuit was replicated in Logisim – an open source program to simulate logical circuits. The file is attached below:



A screenshot of the circuit in Logisim is also shown below



# Questions

1. Why did you choose the encoding you did in step 6?
2. If you were to build this circuit physically using 74 series chips in the CSSE2010 kit, how many chips would you need to use? Justify.
3. How did you test your Logisim circuit to ensure it works properly?