|  |  |
| --- | --- |
| **Course code and name:** | B38DF Computer Architecture and Embedded Systems |
| **Type of assessment:** | **Group** |
| **Coursework Title:** | Lab2 report |
| **Student Name:** | Zetong Yang Xunchi Ma |
| **Student ID Number:** | H00392699 H00362669 |

|  |
| --- |
| **Declaration of authorship. By signing this form:**   * **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section. * I confirm that I have read, understood and followed the University’s Regulations on plagiarism as published on the [University’s website](https://www.hw.ac.uk/uk/students/studies/examinations/plagiarism.htm), and that I am aware of the penalties that I will face should I not adhere to the University Regulations. * I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](https://heriotwatt.sharepoint.com/sites/skillshub/SitePages/Academic-Integrity-and-Plagiarism.aspx)   **Student Signature** *(type your name):* Zetong Yang Xunchi Ma  **Date**: *04-21-2023* |



**SCHOOL OF ENGINEERING and PHYSICAL SCIENCES**

**ELECTRICAL AND ELECTRONIC ENGINEERING**

|  |  |  |
| --- | --- | --- |
| **No.** | **HWUM ID** | **Name** |
| 1 | H00392699 | Zetong Yang |
| 2 | H00362669 | Xunchi Ma |

|  |  |  |
| --- | --- | --- |
| **Campus** | **Xidian Campus** | |
| **Date** | **Due Date: 21/4/2023** | **Submitted Date: 19/4/2023** |

|  |  |
| --- | --- |
| **Lab Sheet No.** | **1** |
| **Title of Lab Sheet** | **LAB 2 REPORT** |

Table of Contents

|  |  |  |
| --- | --- | --- |
|  | [Introduction](#_Introduction) | 1 |
|  | [Results](#_Results) & Discussion | 1-9 |
|  | [Conclusions](#_Conclusions) & Recommendations | 10 |
|  | [References](#_References) | 10 |
|  | [Appendix](#_APPENDIX) | 10-11 |

**1.INTRODUCTION**

In this experiment, we were asked to use Verilog language to design and implement different combination and sequential circuits, and there were five questions to complete. The first two questions are relatively simple, while the last three questions are related to FSM and are more complex. In addition to writing program code, when necessary, we can write test bench code to verify experimental results and simulate waveforms.

# **2.RESULT AND DISCUSSION**

**Problem 1**

Write a Verilog HDL module called minority. It receives three inputs, **a**, **b**, and **c**. It produces one output, **y**, that is TRUE if at least two of the inputs are FALSE. Use Quartus II functional simulation for simulating your circuit and demonstrate its correct working.

Solution:

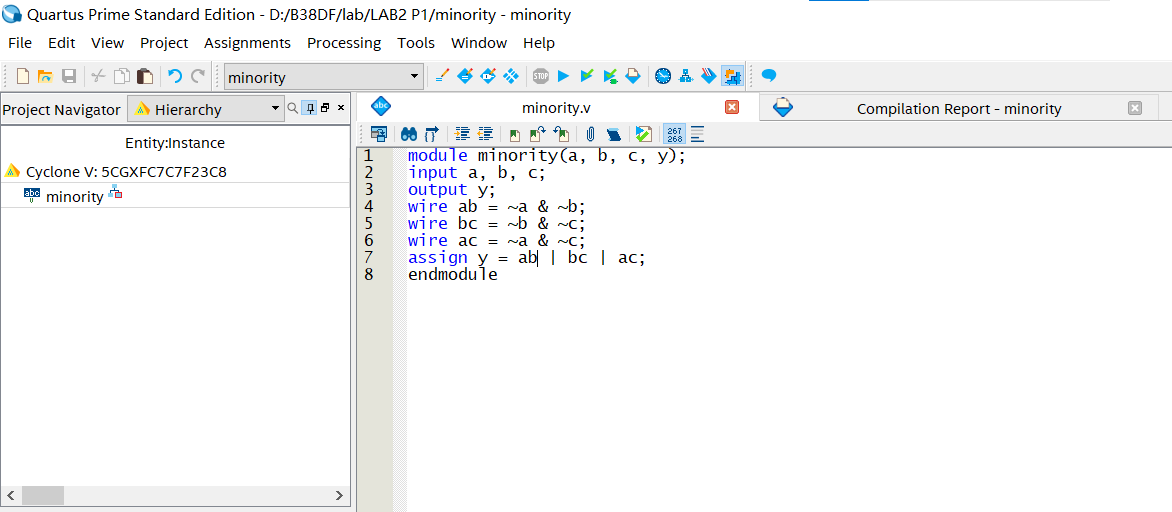


Figure 1: Verilog code of P1

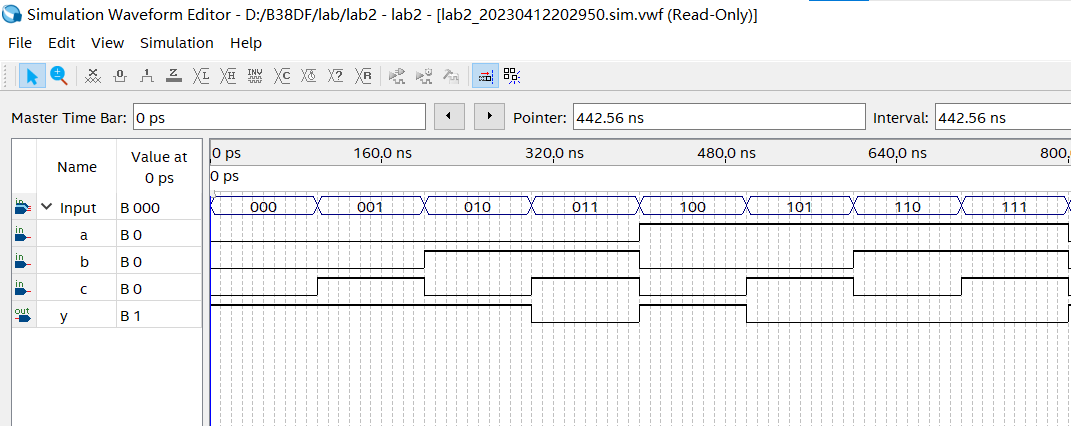


Figure 2: Simulation result of P1

**Problem 2**

Given an 8-bit bus, the task is to use Verilog to build an encoder that can identify the location of the first '1' in the bus, starting from the most significant bit (MSB). For example, if the input is [0 0 1 0 1 1], then the logic should return 3. (Hint: you can use a for loop, but probably a simpler solution can be achieved with casez.)

Solution:

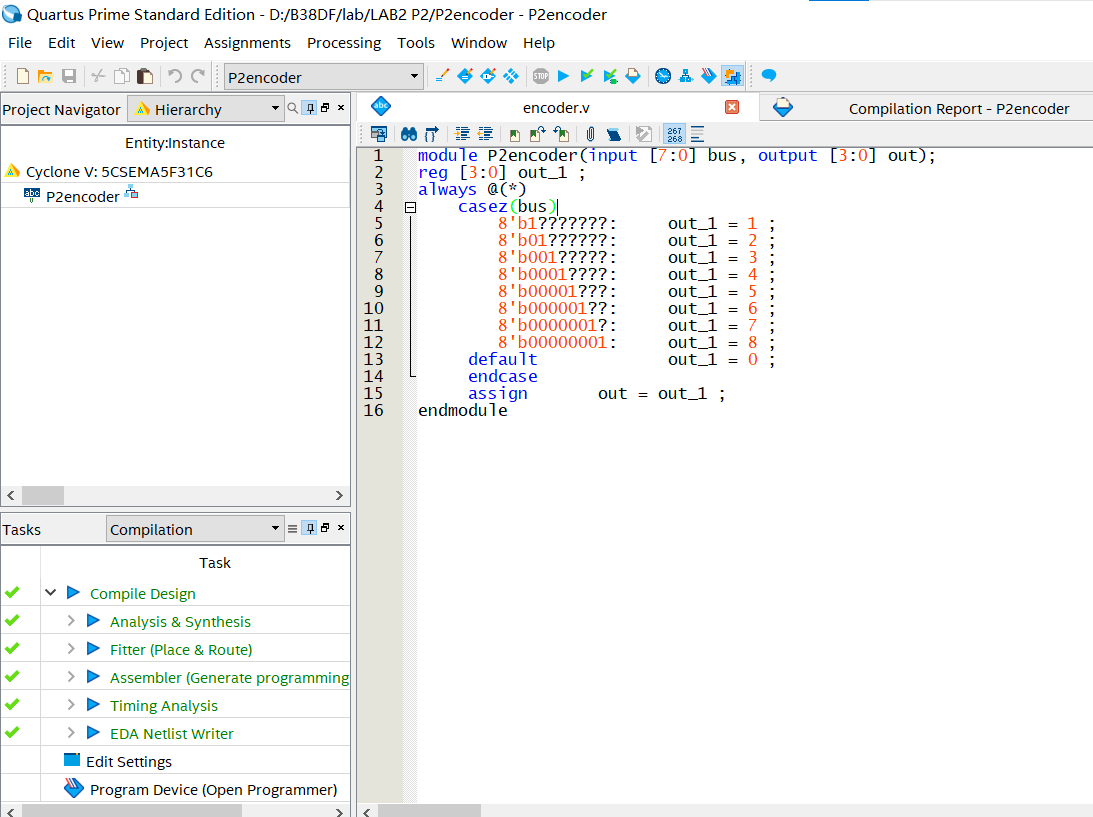
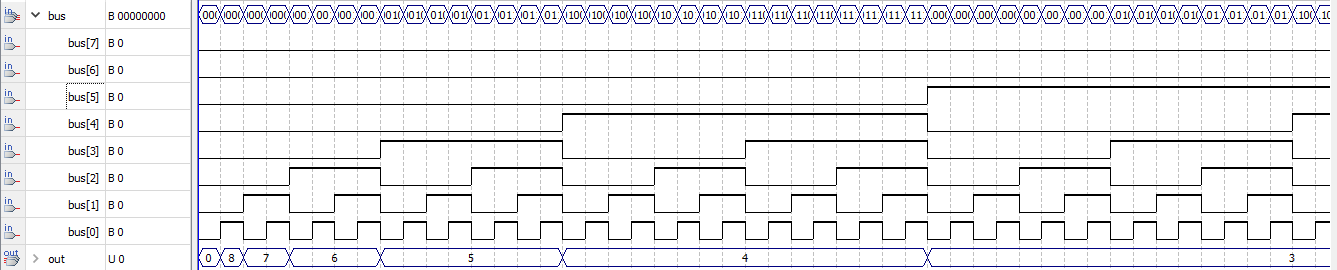
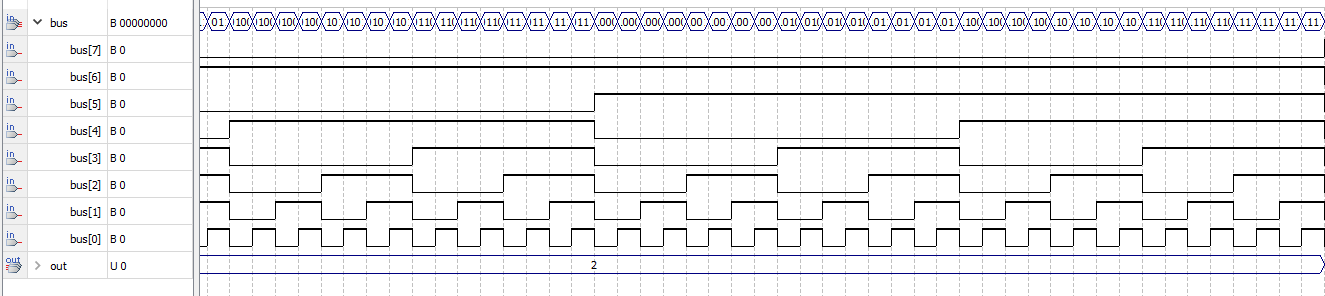


Figure 3: Verilog code of P2





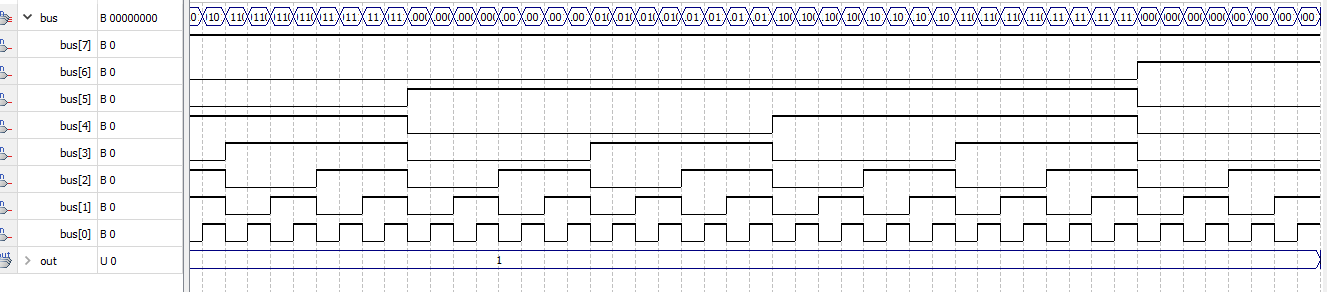


Figure 4: Simulation result of P2

**Problem 3**

A parity check is a core component of the Hamming error-correcting codes. An even/odd parity checker can be implemented as an FSM that receives bit sequence as input and generates a 1 if the number of 1's received so far is even, or 0 otherwise. For example, if the input stream is 0110100101...., then the corresponding output stream of an even parity checker is 1011000110.... Based on the parity checker FSM shown in Figure Q3(in Appendix), implement the parity checker in Verilog.

Solution:

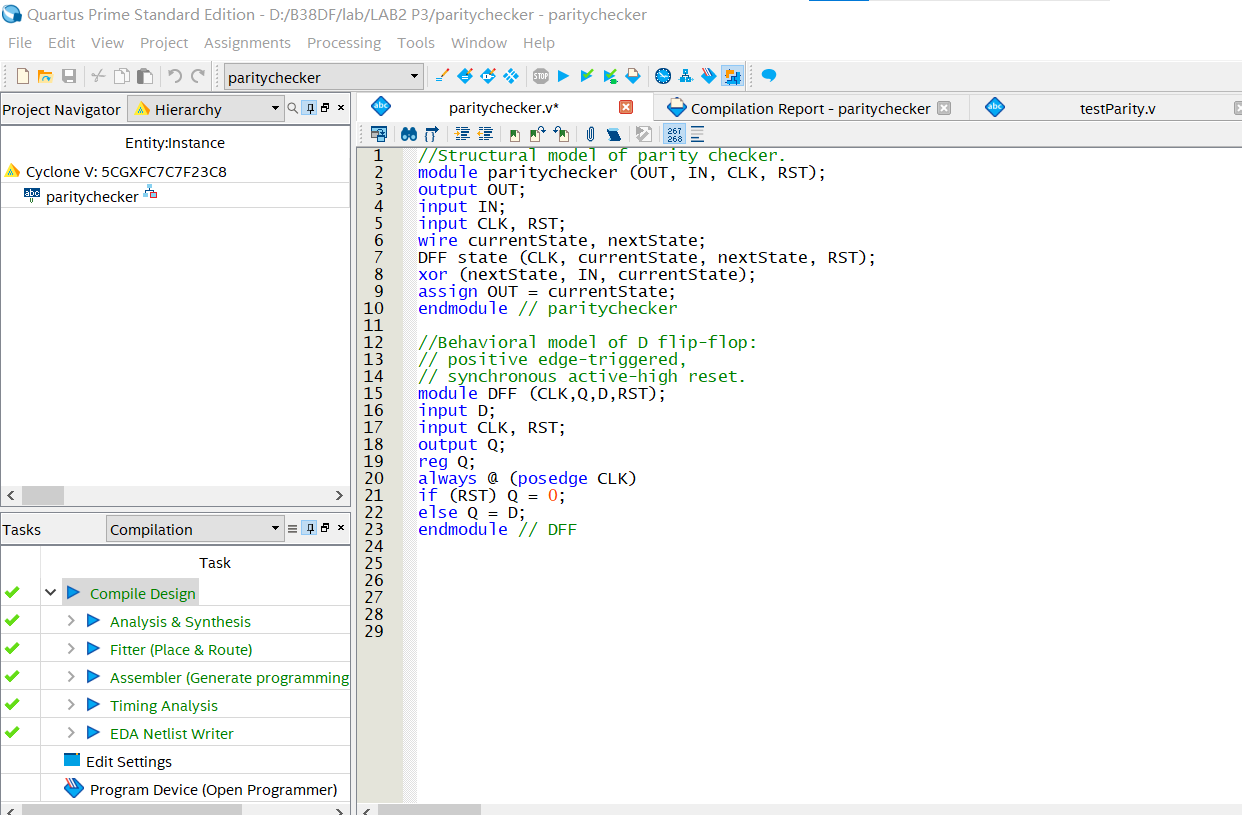


Figure 5: Verilog code of P3

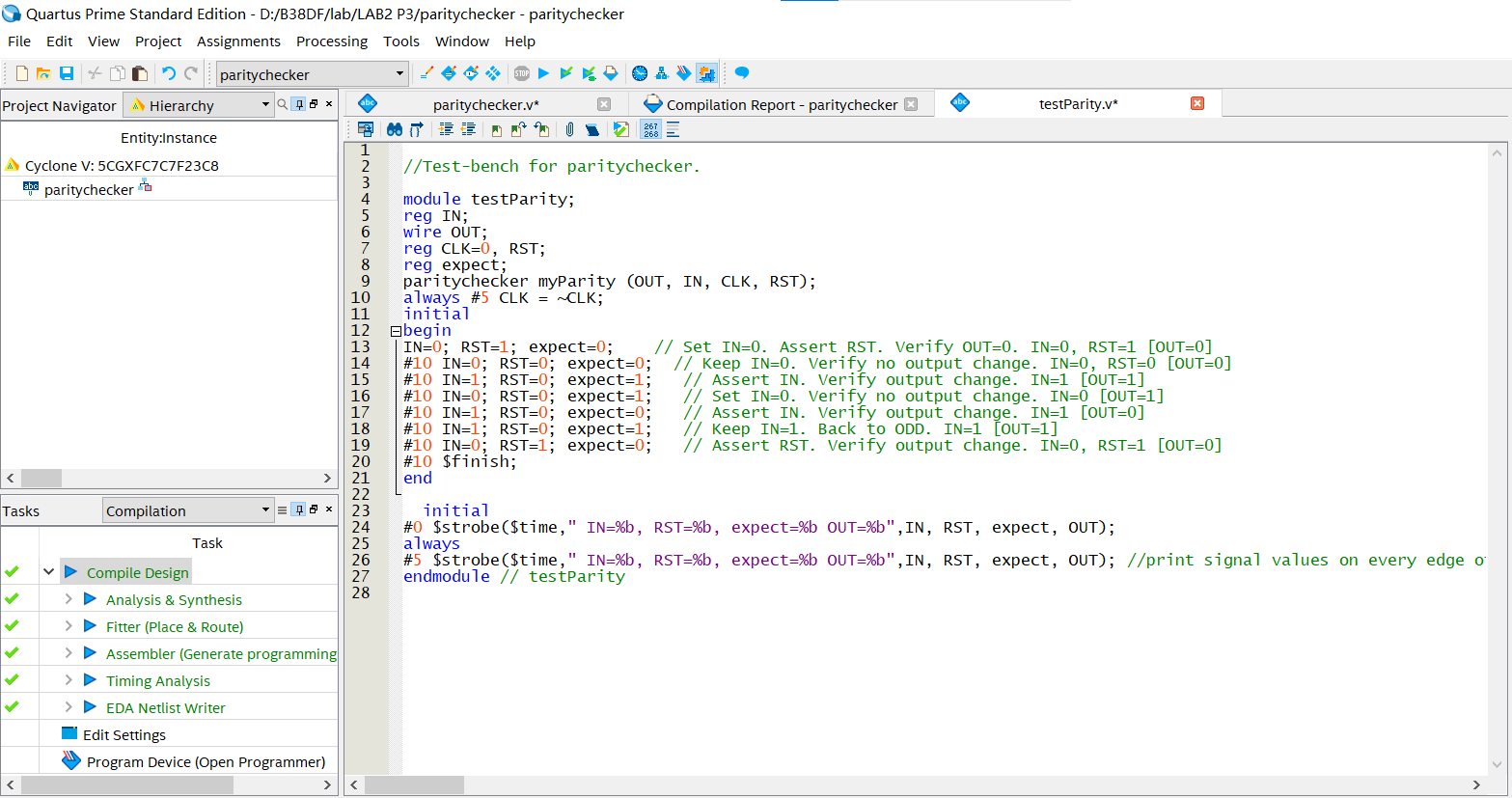


Figure 6: Testbench code of P3

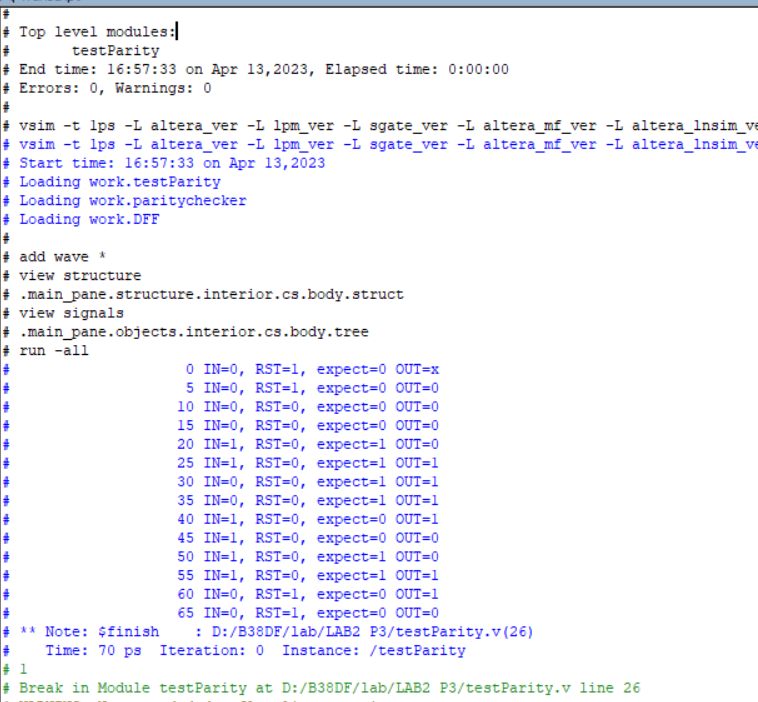


Figure 7:Transcript result of testbench

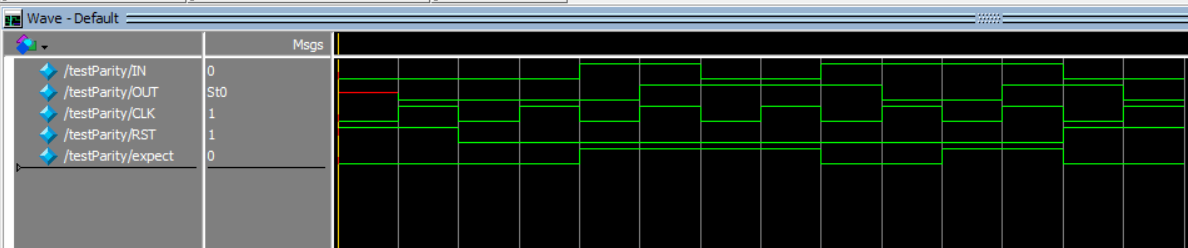


Figure 8: Waveform result of testbench

**Problem 4**

Based on the state diagram provided in Figure Q4a (in Appendix), write a Verilog code for a 111 Sequence Detector that outputs 1 when a sequence of three consecutive 1’s is applied to input, and 0 otherwise. As example input and output is given in Figure Q4b (in Appendix).

Solution:

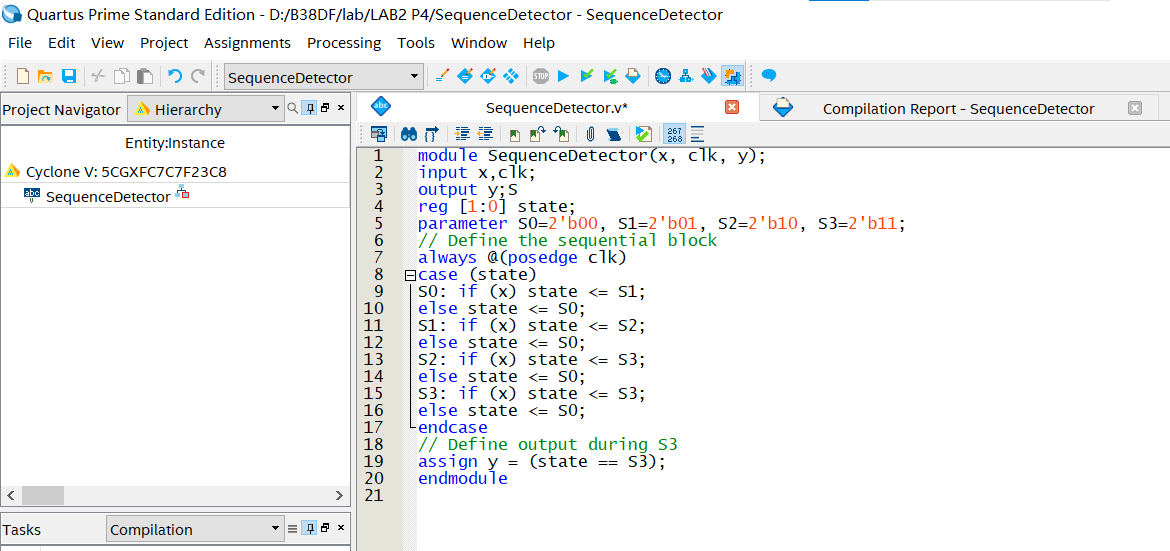


Figure 9: Verilog code of P4

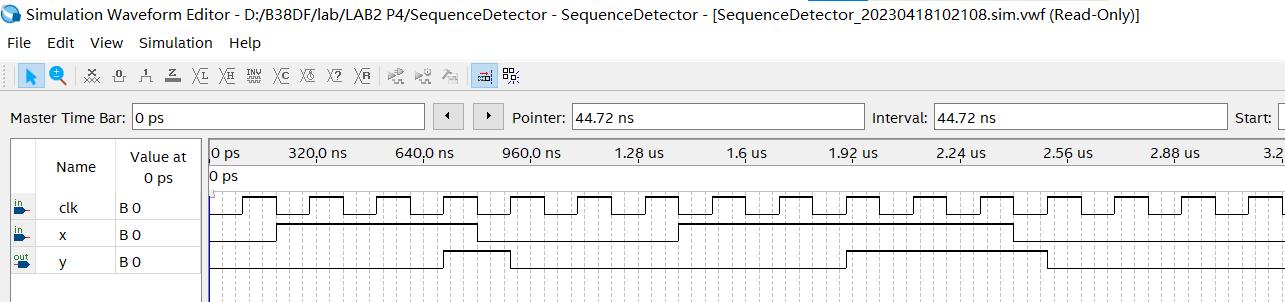


Figure 10: Example Simulation Waveform of P4

**Problem 5**

Figure Q5a (in Appendix) shows an electronic combination lock with a reset button, two number buttons (0 and 1), and an unlock output. The combination of 01011 will unlock the lock. Based on the state diagram shown in Figure Q5b (in Appendix), write a Verilog code to implement the design.

Solution:

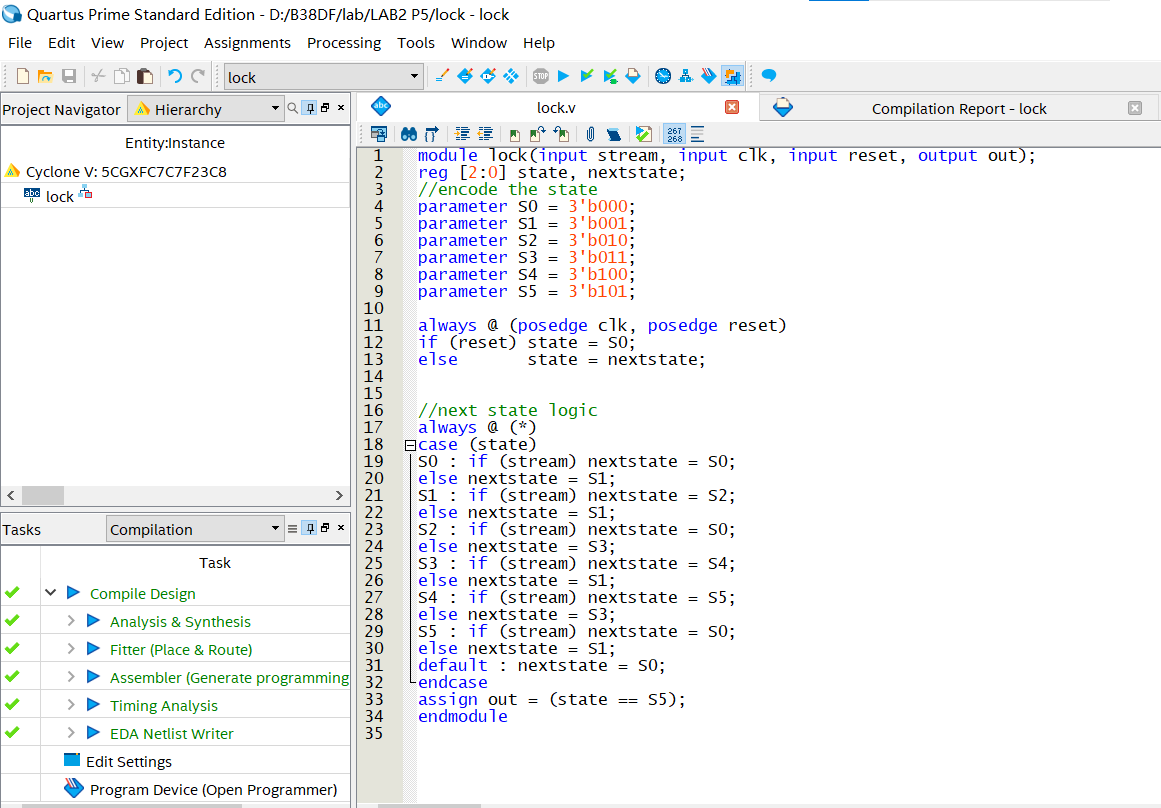


Figure 11:Verilog code of P5

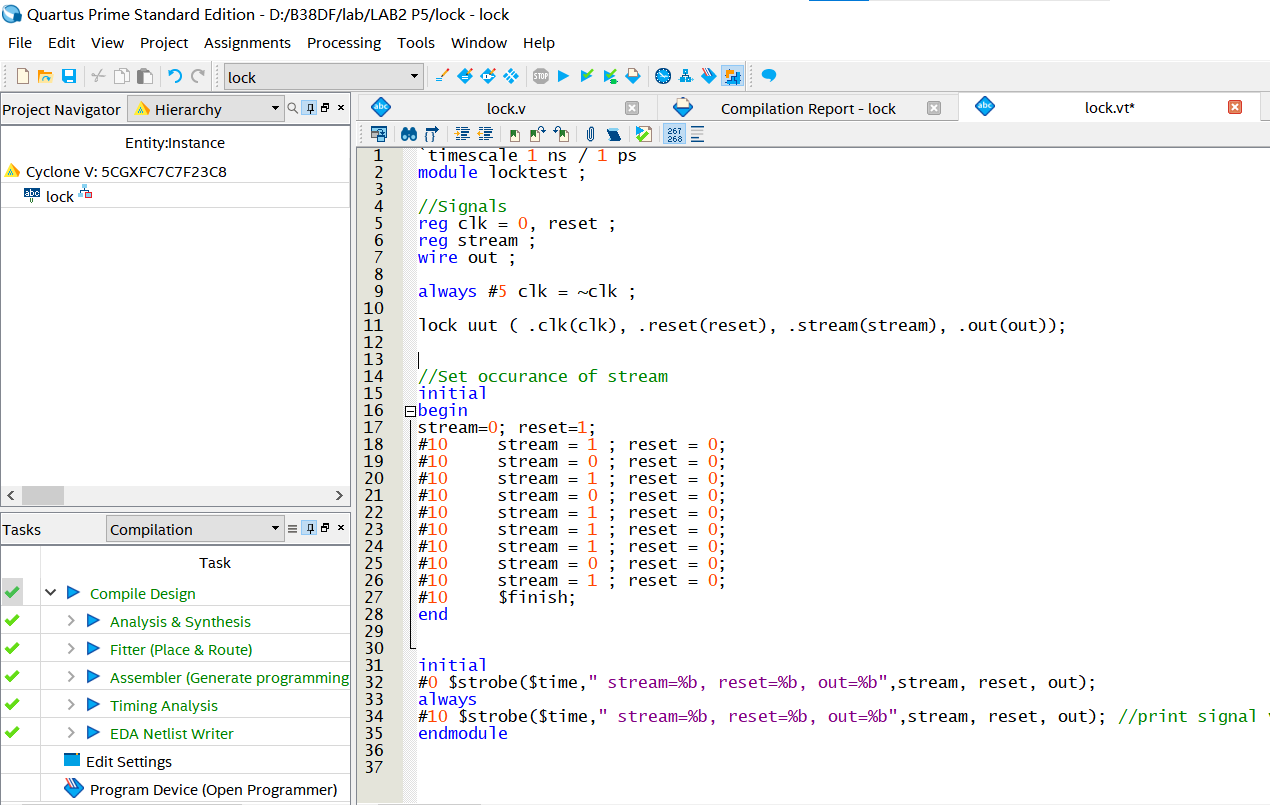


Figure 12: Testbench code of P5

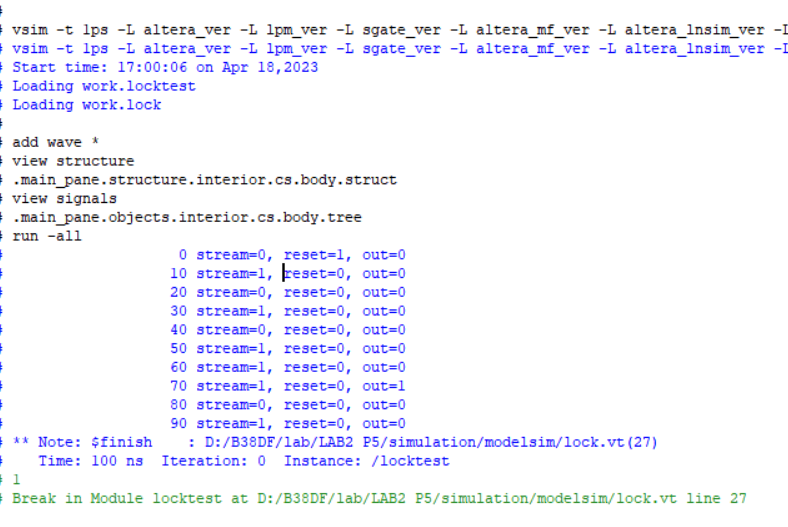


Figure 13: Transcript result of testbench

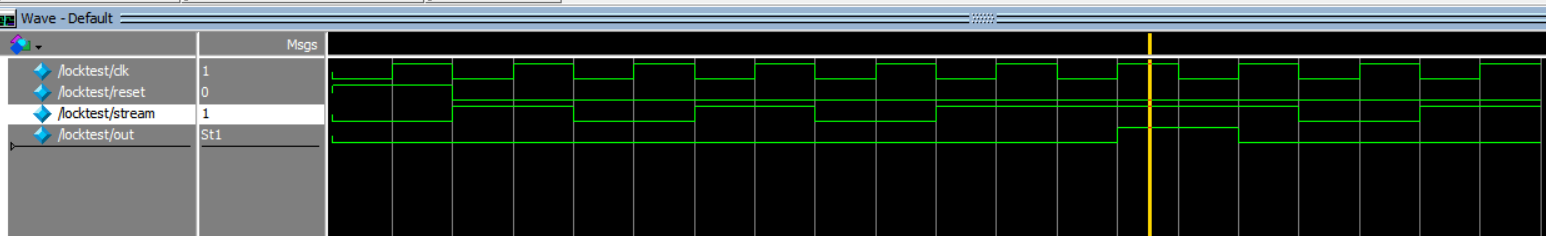


Figure 14: Waveform result of testbench

**Discussion**: From questions 3 to 5, it is not difficult to see that the FSMs designed for the three questions are all related to the sequence detector. For these three problems, I all used Moore machine, which means that the output value is only related to the current state.

Five step controller design process[1]:

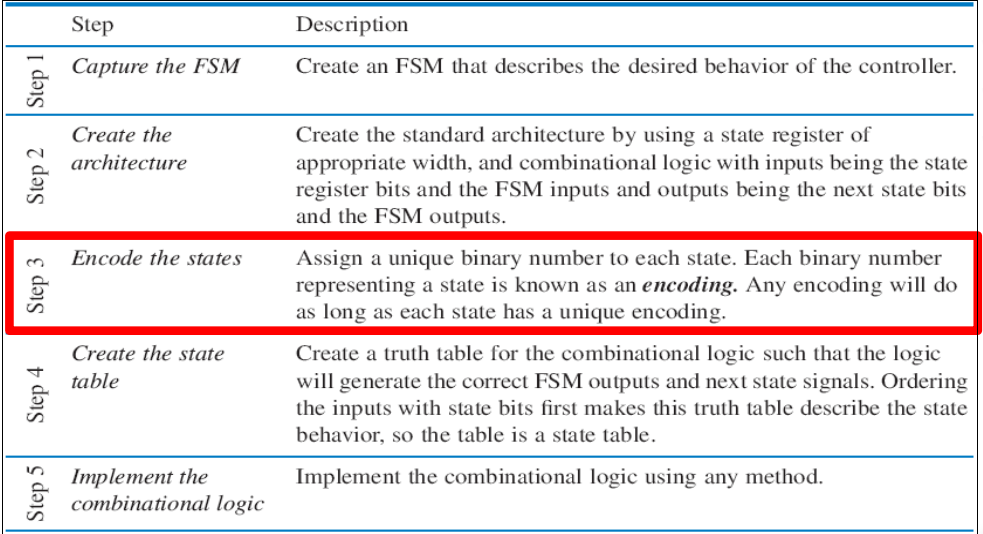
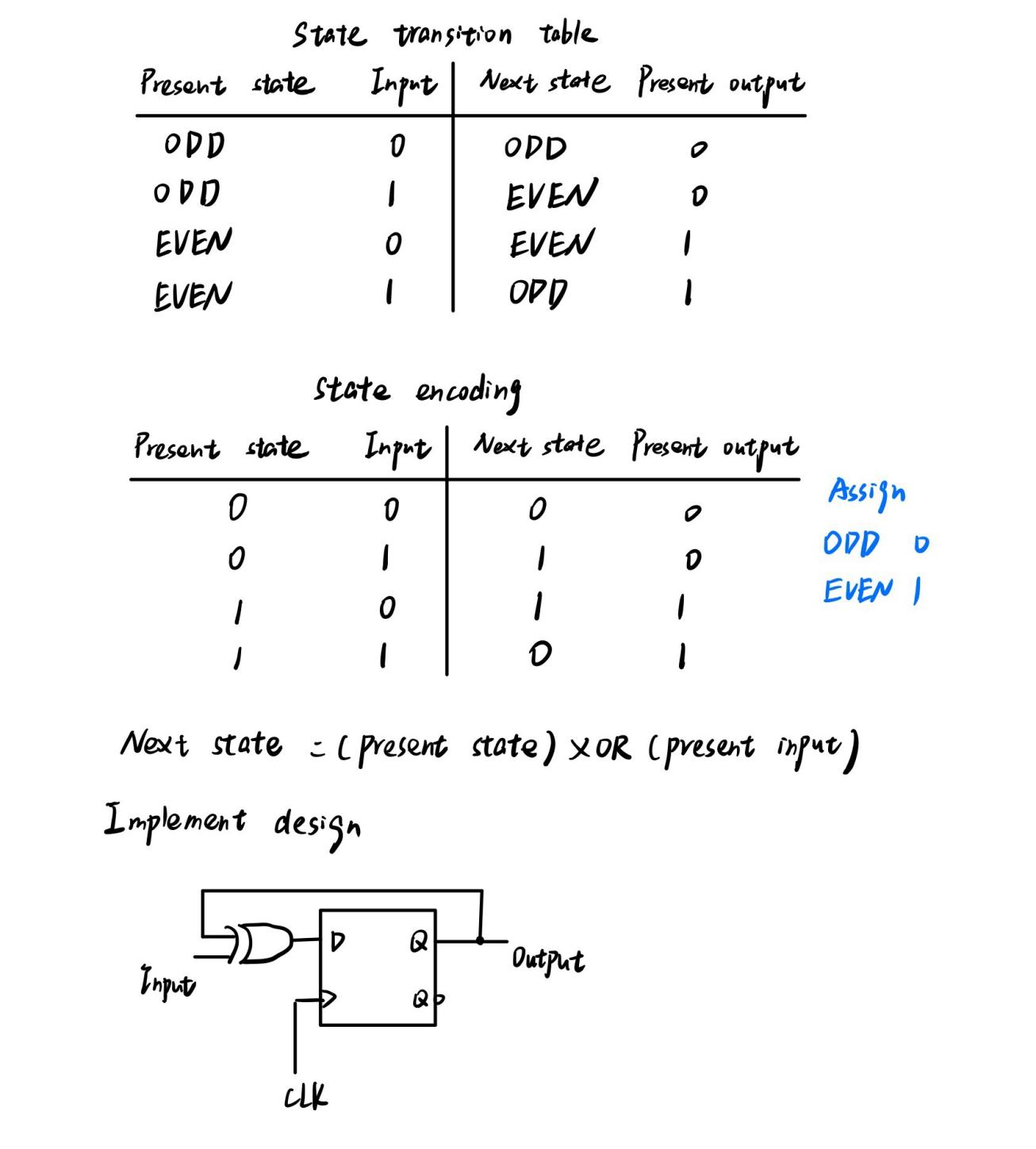


Figure 15: Design process for sequential logic

Consider P3 as an Example, the design process should be as follows:

****

## In addition, through the problem 4,5, we can learn about the concept of Overlapping and Non- Overlapping. Take problem 4 as an example, the sequence to be detected is 111.

## Take input = 011100011111000 as an example

  In the case of Non overlapping, the system will generates first ‘one’ at the 4th bit ( sequence is detected at the third bit 11****1**** ).The system then starts fresh to check the sequence. The next sequence is detected at 10th bit.

So Output = 000100000100000

However, in the case of overlapping, the system generates first ‘one’ at the 4th bit. we see that last bit is 1 in the sequence, and the staring bit in the sequence is also ‘one’ i.e both are same. So the system will consider this bit as the starting bit of next sequence , if sequence is continued in further bits.[2]

So Output = 000100000111000

1. **CONCLUSIONS AND RECOMMENDATIONS**

This experiment not only involved Verilog code writing, but also taught us how to write and use Test bench in Quartus, and reviewed knowledge about finite state machines (FSM). Meanwhile, we also learn about the concept of overlapping and non-overlapping which are used for sequence detection.

# **REFERENCES**

[1] B38DB L08**----“**Controller (Sequential Logic) Design Process**”**

[2]https://alljobs.co.in/study-material/sequence-detector-verilog-code/#3bit\_Sequence\_Detectors

**APPENDIX**

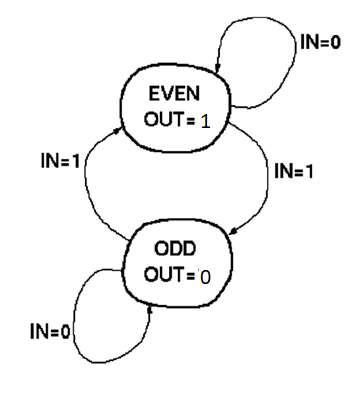
****

Figure Q3 FSM for parity checker

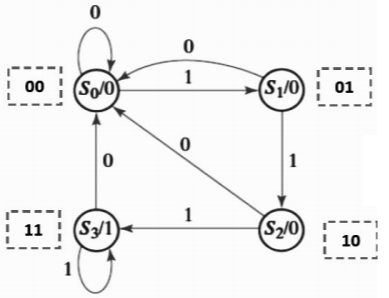


Figure Q4a State diagram

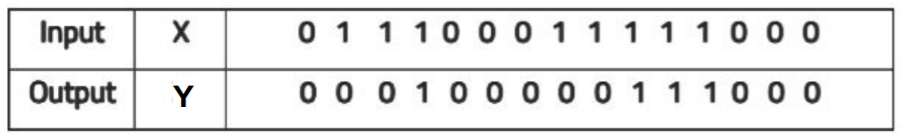


Figure Q4b Example input/output

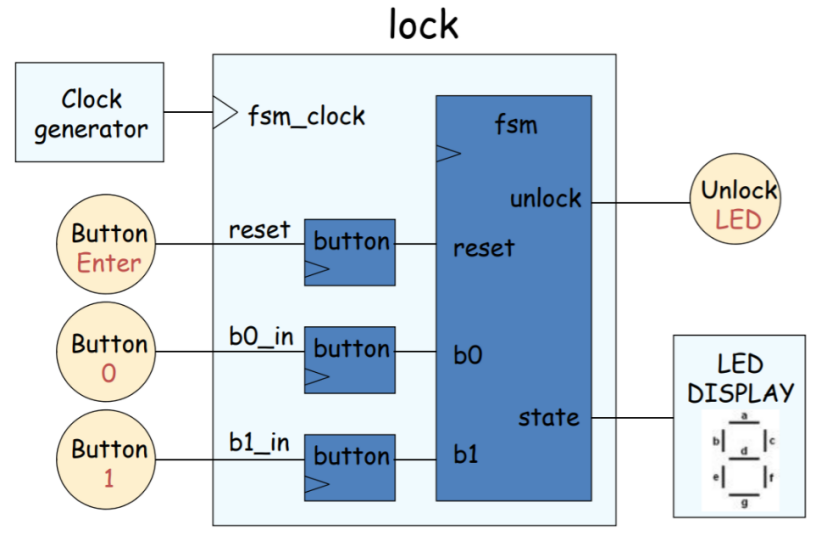


Figure Q5a Electronic combination lock

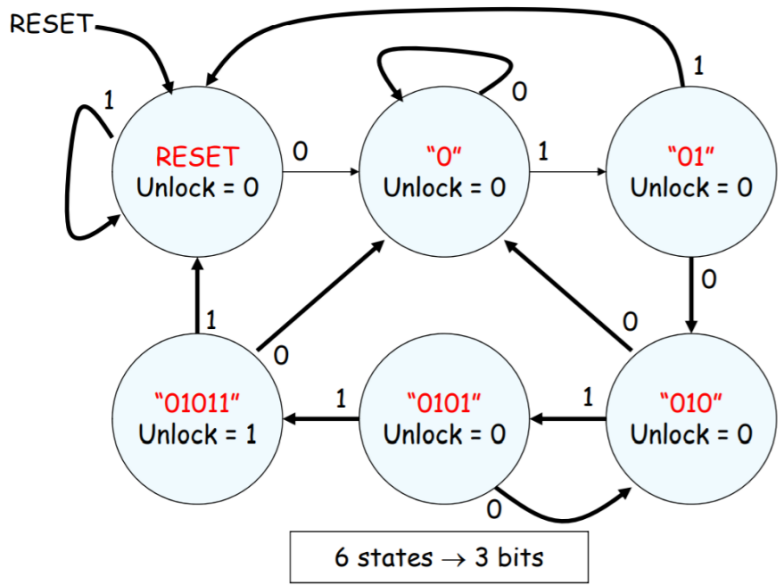


Figure Q5b State diagram for the electronic combination lock