ISSN 2959-6157

Research and Design of Data Encryption Circuit based on D Flip-Flop

Muzhen Tie

Department of Electric and Computer Engineering, University of Rochester, Rochester, NY, US
Corresponding author: mtie@u.rochester.edu

Abstract:

Since information technology is developing so quickly, the problem of data security has become increasingly prominent. Data encryption technology, as an important means of securing data transmission, plays a vital role in resisting network attacks and protecting privacy. In this paper, a data encryption circuit based on D flip-flop is studied, aiming to improve the efficiency and security of data encryption. First, this paper introduces the basic principles of data encryption and analyzes the application of D flip-flop in encryption circuit. Subsequently, the design principle of Moore's state machine is elaborated, and encryption circuit is designed with the use of the state graph and state table. Further, this paper demonstrates the logic circuit design process., including the derivation of D-flip-flop input Circuit construction and Boolean expressions. This research provides a hardware implementation design solution for the design of data encryption circuits, which is expected to play a role in improving encryption speed and enhancing security.

Keywords: State Machines (Moore & Mealy), State Graphs, Logic Gates, D flip flop

1. Introduction

With the swift advancement of information technology, data security has become the focus of global attention [1]. Among many data protection measures, data encryption technology is particularly important because of its key role in ensuring the security of information transmission [1]. Although traditional software encryption methods secure data to a certain extent, in the face of increasing computational demands and sophisticated network attacks [2], hardware encryption schemes are favored for their high efficiency and attack resistance [2].

The core of digital circuits are logic gates and flip-flops, which are the basis for building complex logic systems [3]. D flip-flops, as a basic timing logic element, not only play an crucial part in the processing and storage of data, but also play a key role in designing flexible and powerful encryption circuits [4]. With a well-designed D flip-flop network, complex encryption algorithms can be realized to provide strong protection for data.

The purpose of this paper is to study and design a data encryption circuit based on D flip-flop. Firstly, this paper will explore the basic principles of logic operations and the application of D-flip-flops in encryption circuits. Secondly, this paper will detail the design principles of Moore's state machine and show how to utilize state diagrams and state tables to design encryption circuits. Finally, this paper will demonstrate the logic circuit design method, including the derivation of D-type flip-flop input Boolean expressions and the implementation of the circuit.

The structure of this paper is set up as follows: The introduction, which comprises the first chapter, provides an overview of the paper's major content and structure as well as the background and research importance of the thesis. The second chapter introduces the basic operations of digital circuits in detail, including the working principles of logic devices such as not gates, and gates, or gates and D flip-flops and their applications in circuit design. Chapter 3 is the encryption circuit design, which provides the Moore state machine design theory, the process for making a state table and state graph, and the particular steps needed to create a logic circuit. Finally, the entire paper is summed up in the conclusion part, which also provides an outlook for future research.

2. Basic Logical Operation, Logic Gates, and Flip-Flop

2.1 Definition of Signal States

Low voltage Level (0): In digital circuits, a low voltage level usually indicates a logic "0". It is a voltage level below the logic threshold of the circuit, depending on the circuit design, but usually below some reference voltage.

High voltage level (1): Correspondingly, a high voltage level indicates a logic "1". It is a voltage level above the circuit's logic threshold, usually above a reference voltage, indicating an active or true state.

2.2 Logic Gates Used in the Design

2.2.1 NOT Gate

A NOT gate is a simple logic circuit that performs a basic logic non-operation, i.e., inverting the state of an input signal. In digital circuit design, a NOT gate is one of the basic components for building more complex logic functions.

Functional Description:

A NOT gate has the following functional characteristics:

Single Input: A NOT gate has only one input, which receives the original signal [5].

Single Output: A NOT gate inverts the input signal and provides the result through the output [5].

Signal inversion: In the event that the input is a low voltage level (0), the NOT gate will produce a high voltage level (1); in the event that the input is a high voltage level (1), on the other hand, the NOT gate will produce a low voltage level (0) [5].

The NOT gate icon is shown in Figure 1:

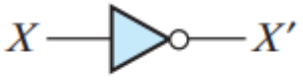

Fig. 1 NOT gate icon [6]

The NOT gate truth table is shown in Table 1:

Table 1. NOT gate truth table

Input	Output
0	1
1	0

2.2.2 AND Gate

An AND gate is a fundamental building block in digital logic systems, performing the logical “AND” operation in Boolean algebra.

Functional Description:

Multiple Inputs: an AND gate has two or more inputs and is used to receive multiple signals [7].

Single output: an AND gate has only one output, which is used to provide the outcome of each input signal’s “and” operation [7].

All high input is high: When all the input signals are high, the AND gate’s output is high; otherwise, it is low. [7].

Application Example:

In a multiconditional logic determination, AND gate output only becomes high when every requirement is met, indicating that the logic is true.

In control systems, and gate can be used to ensure that an operation is activated only when multiple signals are correct.

Figure 2 displays the icon for the two-input AND gate:

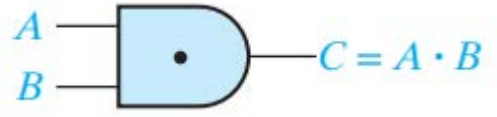

Fig. 2 two-input AND gate icon [6]

Table 2 displays the two-input AND gate truth table:

Table 2. two-input AND gate truth table

Input		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

2.2.3 OR Gate

OR gates are logic gates that implement the “or” operation in Boolean logic.

Functional Characteristics:

Multiple Inputs: Or gates also have two or more inputs for receiving multiple signals [8].

Single Output: An OR gate’s output gives the outcome of the “or” operation for each and every input signal. [8].

Output high if anyone is high: An OR gate has a high out-

put if at least one of its inputs is high. If each input is low, the output is also low [8].

Application Example:

In signal selection, an OR gate can be used to select a high-level signal from multiple input signals and the output is the selected high-level signal.

In a fault detection system, OR gate can be used to detect any one of multiple fault signals and output a high level as a warning once a fault occurs.

Figure 3 displays the icon for the two-input OR gate:

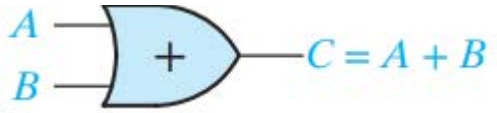

Fig. 3 two-input OR gate icon [6]

Table 3 displays the truth table for the two-input OR gate.:

Table 3. two-input OR gate truth table

Input		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

2.3 Flip-Flop Used in the Design

2.3.1 D Flip-Flop

A D-type flip-flop is a bistable memory cell that can be controlled by a clock signal. It can store a single binary data bit and update its output under the control of a clock signal [9].

Functional Characteristics:

Data Input (D): Data signals are received and stored via the data input of a D flip-flop. [9].

Clock input (CLK): A clock input on the D flip-flop regulates how data is stored and updated [9].

Output (Q): The current status of the data that is being saved is provided by the D flip-flop's output [9].

Complementary Output (Q'): Additionally, the D flip-flop has a complementary output whose state is the opposite of Q [9].

Principle of operation:

The D flip-flop stores the current data input signal D into the internal state when it receives a clock signal, such as one that transitions from low to high. [9].

The output Q will be updated to the same state as the data input D [9].

The complementary output Q' will be updated to the opposite state of the data input D [9].

Application Example:

Register design: Data processing and storing multi-bit registers could be created through connecting D flip-flops in series.

Counter design: by returning output of a D-type flip-flop to the input, a counter can be constructed to realize the digital counting function.

Memory design: Arrays of large numbers of D flip-flops can form memories, such as random access memory (RAM).

Figure 4 displays the rising edge D flip-flop symbol:

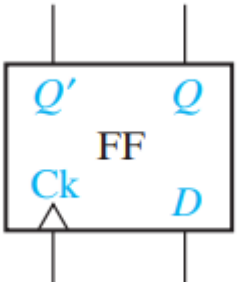

Fig. 4 rising edge D flip-flop icon [5]

Table 4 displays the rising edge D flip-flop truth table.:

Table 4. rising edge D flip-flop truth table

D	Q	Q+
0	0	0
0	1	0
1	0	1
1	1	1

3. Encryption Circuit Design

In this section, the paper will detail the design process of encryption circuits, including principles of Moore state machine design, the creation of logic circuit designs as well as state graph and state tables.

3.1 Moore State Machine Design

One kind of timing logic circuit is a Moore state machine where the output is solely determined by the present state, without any influence from the current input [10]. In encryption circuit design, Moore state machine is used to recognize specific bit streams and control the encryption process of data based on these bit streams.

Design Principle:

Moore state machine consists of a finite number of states, each corresponding to a unique output [10]. The State Transition Diagram describes how the state machine transfers from one state to another given the inputs.

The State Table lists all possible state transitions, comprising the inputs, outputs, next state, and present state..

Application in encryption circuits:

A Moore state machine is intended to identify two distinct Bit streams in this architecture: “111” and “101”. When “111” is detected, the state machine controls the circuit to start encryption (inverting the input bits) and stops encryption when “101” is detected.

3.2 State Graphs and State Tables

A Moore state machine can be seen visually as a state graph, showing the transition relationships between states. A state table, on the other hand, is a detailed list of state transitions.

State Graph Drawing:

Based on the design requirements, a state graph is drawn which includes six states: S0, S1, S2, S3, S4 and S5.

The state graph shows how the state machine transfers from one state to another in case a specific Bit is received.

Figure 5 displays the design of the state graph:

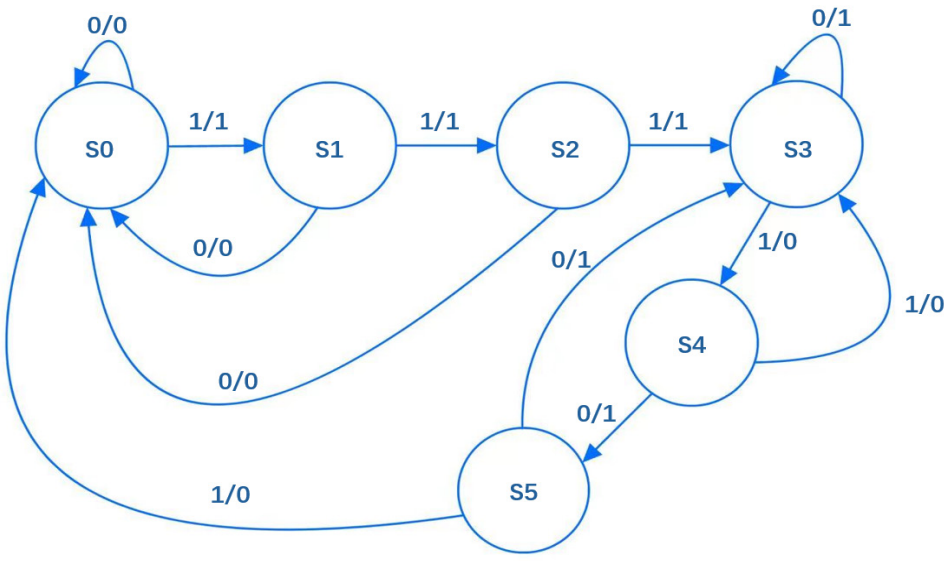

Fig. 5 state graph design

Construction of state table:

The next state and output are specified in the state table of each state when a “0” or “1” input is received. For example, state S0 transfers to S1 and starts the encryption process when “111” is received, and stays at S0 when “101” is received.

Table 5 displays the design of the state table:

Table 5. state table design

Present	Next State 0 1	Out Put 0 1
S0	S0 S1	0 1
S1	S0 S2	0 1
S2	S0 S3	0 1
S3	S3 S4	1 0
S4	S5 S3	1 0
S5	S3 S0	1 0

3.3 Design of Logic Circuit

The process of translating state tables into real circuit components that carry out a state machine's logic operations is known as logic circuit design.

Derivation of Boolean expression for D-type flip-flop input:

Based on the state table, the Boolean expressions that control the inputs to the D flip-flops are derived. These expressions determine how the data inputs (D) of each flip-flop should be determined using the input bits and the state at that moment.

Table 6 displays the design of the truth table:

Table 6. truth table design

A	B	C	X	A+	B+	C+	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	0	0	0	0
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

3.4 Logic Circuit Implementation

The circuit design also included the use of Karnaugh diagrams (K-maps) to simplify Boolean expressions and optimize circuit design.

Logic circuits were designed using logic gates (e.g., AND, OR, NOT) and D-type flip-flops to implement the logic functions of state machines.

Karnaugh map and simplification is shown in Figure 6:

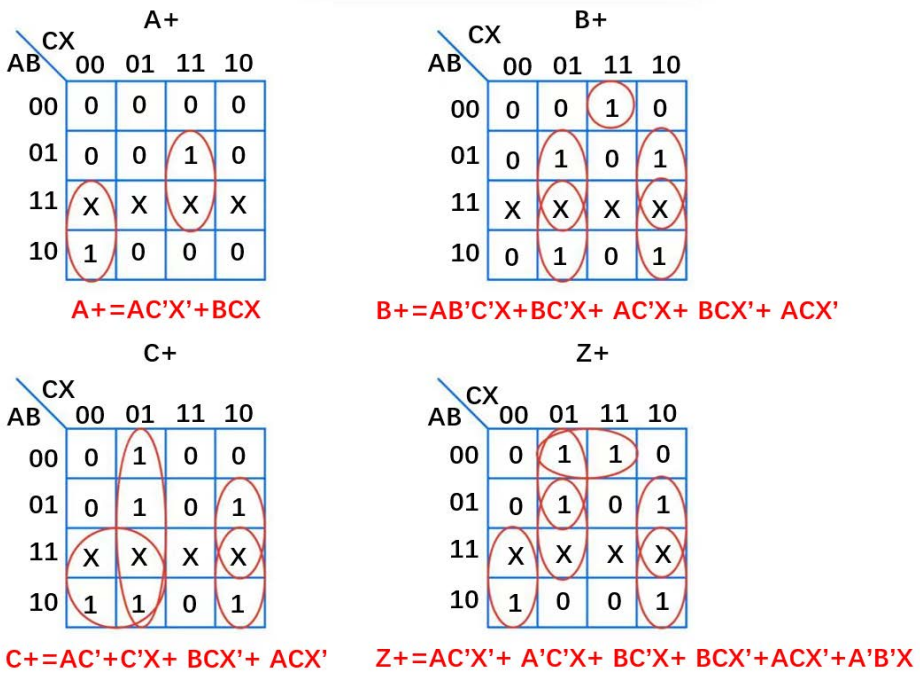

Fig. 6 Karnaugh map and simplification

Encryption Circuit Schematic is shown in Figure 7:

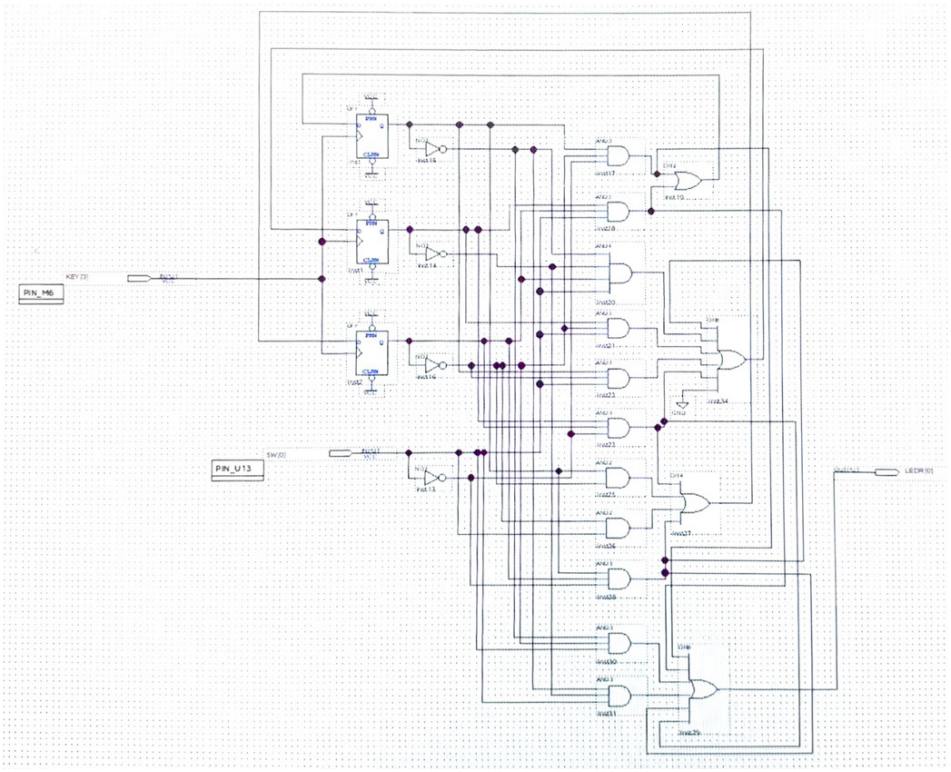

Fig. 7 Encryption Circuit Schematic

4. Summary

In this paper, an efficient data encryption method is successfully realized by studying and designing a data encryption circuit based on D flip-flop. Based on an in-depth analysis of the importance of data encryption technology

and the role of D flip-flop, this paper describes in detail the design principle of Moore state machine and utilizes state graph and state table to construct the encryption circuit. The encryption logic of the circuit is realized through logic circuit design and the derivation of Boolean expressions.

This study not only provides a hardware implementation scheme for the design of data encryption circuits. Future work will focus on further optimizing the circuit design, which can be done by calculating the logic effort and delay of the critical path, optimizing the circuit to increase the encryption speed and reduce the power consumption, and exploring more complex encryption algorithms to meet the increasing demand for data security. In addition, the expandability and compatibility of the circuits will be considered to adapt to different application scenarios and system requirements.

References

[1] S William and W. Stallings, "Cryptography and Network Security 4/E [M]", *Pearson Education India*, 2006.

[2] M. Iavich, R. Bocu, G. Iashvili and S. Gnatyuk, "Novel Method of Hardware Security Problems Identification," 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2020, pp. 427-431

[3] D. Harris and S. L. Harris, "Digital design and computer architecture", *Morgan Kaufmann*, 2010.

[4] Y. Liang, C. C. Boon, D. Kissinger and Y. Wang, "A Low-Power D-type Flip-flop with Active Inductor and Forward Body Biasing Techniques in 40-nm CMOS," 2019 IEEE 19th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems (SiRF), Orlando, FL, USA, 2019, pp. 1-4

[5] Not gate (inverter). NOT Gate - Logicly Documentation. (n.d.). https://logic.ly/lessons/not-gate/#:~:text=A%20NOT%20gate%2C%20often%20called,results%20in%20a%20true%20output.

[6] Roth, C. H., & Kinney, L. L. (2014). *Fundamentals of Logic Design, 7*. Cengage Learning.

[7] Harris, D. M., & Harris, S. L. (2010). *Digital Design and computer architecture*. Morgan Kaufmann Publishers.

[8] WAKERLY, J. F. (2006). Digital Design: Principles and practices John F. Wakerly. Pearson Prentice Hall.

[9] GeeksforGeeks. (2023, June 14). D Flip Flop. https://www.geeksforgeeks.org/d-flip-flop/

[10] Moore and mealy machines. Tutorialspoint. (n.d.-b). https://www.tutorialspoint.com/automata_theory/moore_and_mealy_machines.htm