Digital Circuit Theory - Laboratory						
Academic year	Laboratory exercises on	Mode of studies	Field of studies	Supervisor	Group	Section
2024/2025	Wednesday 11:45 – 13:15	SSI	Informatics	DP	1	1

Report from Exercise No 5

Performed on: 06.11.2024

Exercise Topic : Synchronous Sequential Circuits

Performed by:

Piotr Copek
Zuzanna Micorek

Introduction

In this report, we present the design and implementation of various synchronous sequential circuits, each fulfilling a specific digital logic function. These tasks involved creating Moore state diagrams, encoding states using binary values, and simplifying Boolean expressions for state transitions and outputs through Karnaugh maps. Each circuit was built using either D or JK flip-flops, NAND, and NOR gates, chosen according to the requirements of each design task. This report summarizes our approach, challenges, and solutions, providing insights into sequential circuit design and practical applications.

Task 2

Design a synchronous sequential circuit detecting than a binary 2-bit number corresponding to the 2-bit words sent to the serial input X, is an odd number. The first bit of the word is assumed to be MSB. Detection should cause the output to become set for the time not longer than one clock period.

Solution

We started by drawing the Moore state diagram to represent the behavior of the sequential circuit. Each state represented the progress through the bits of the 2-bit input, with transitions based on the serial input X.

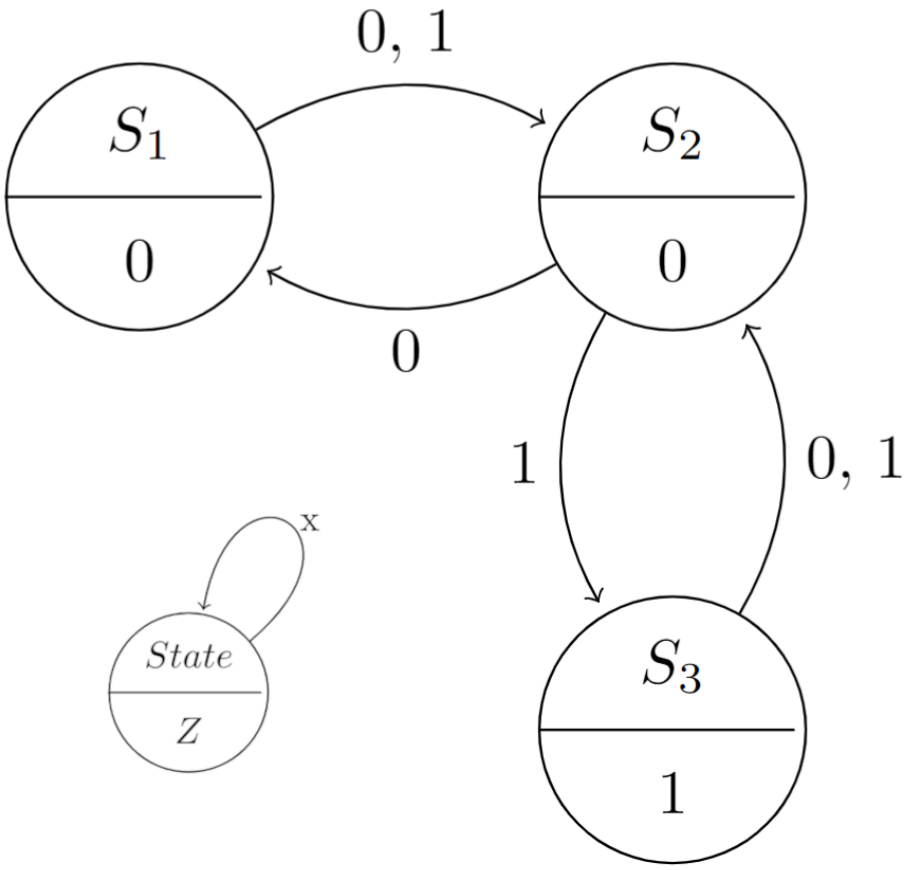

Figure 1 - Graph for Moore Machine implementation of task.

We encoded the states using binary values.

$$State \rightarrow Q_1Q_2$$
$$S_1 \rightarrow 00$$
$$S_2 \rightarrow 01$$
$$S_3 \rightarrow 11$$

We constructed Karnaugh maps for the state transitions and a separate K-map for the output. This allowed us to identify groups and simplify the Boolean expressions for the next states and the output.

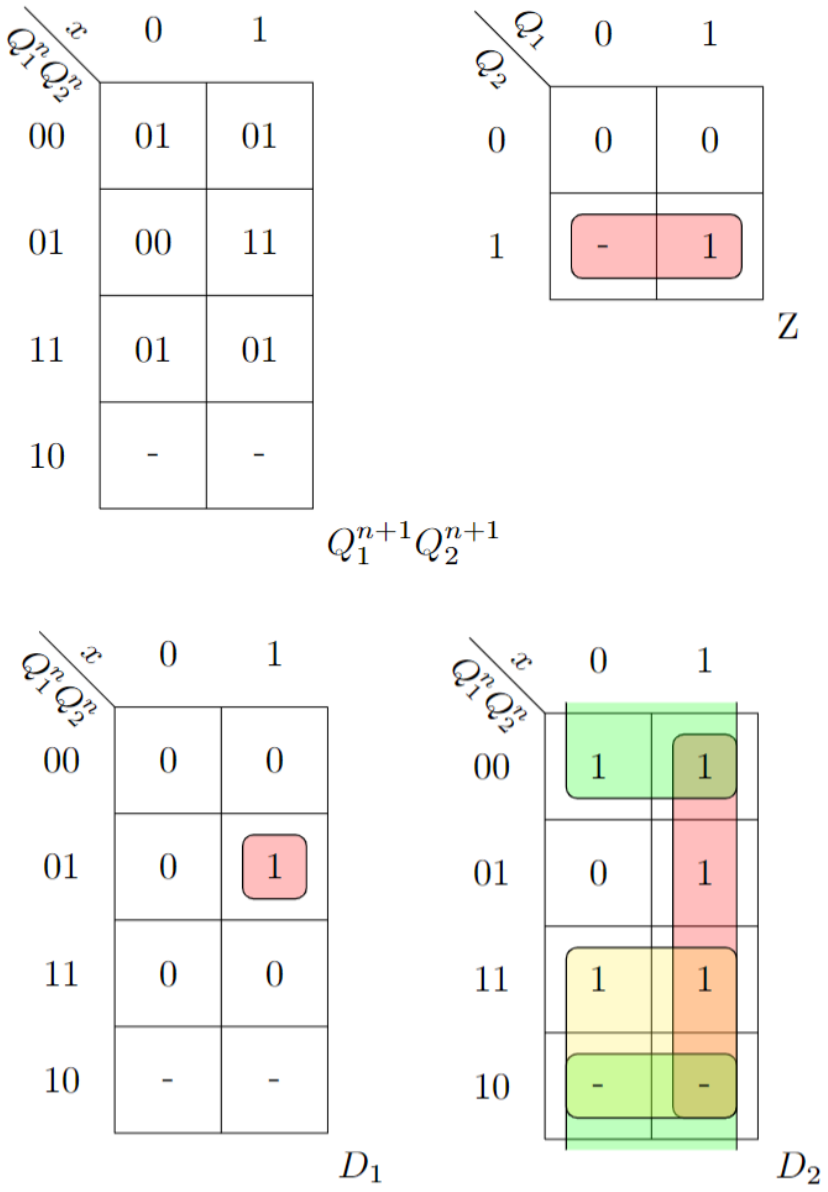

Figure 2 - K-maps obtained from graph used to obtain solution for the task.

From the K-maps we obtained equations, which allowed us to build circuit based on D type flip-flops and NAND gates:

First D flip-flop implemented with NAND gates:

$$D_1 = \overline{Q_1} \cdot Q_2 \cdot x = \overline{\overline{\overline{Q_1} \cdot Q_2 \cdot x}}$$

First D flip-flop implemented with NAND gate:

$$D_2 = x + Q_1^n + \overline{Q_2^n} = \overline{\overline{x} \cdot \overline{Q_1^n} \cdot Q_2^n}$$

$$Z = Q_1^n$$

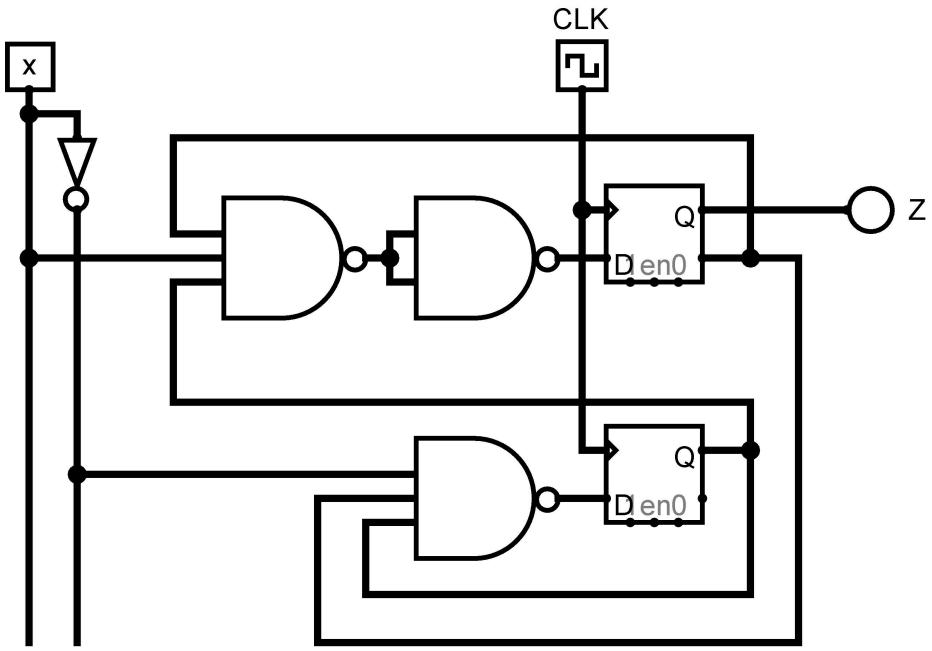

Figure 3 - Implementation of task 2.

Summary

In this task, we designed a synchronous sequential circuit to detect odd 2-bit numbers using flip-flops gates. We started with a Moore state diagram, encoded the states, used Karnaugh maps for simplification, and chose D flip-flop. We successfully implemented the design, gaining valuable insights into sequential circuit design and practical implementation.

Task 3

Design a synchronous sequential circuit detecting than a binary 2-bit number corresponding to the 2-bit words sent to the serial input X, is an odd number. The first bit of the word is assumed to be LSB. Detection should cause the output to become set for the time not longer than one clock period.

Solution

We constructed Karnaugh maps for the state transitions and a separate K-map for the output. This allowed us to identify groups and simplify the Boolean expressions for the next states and the output.

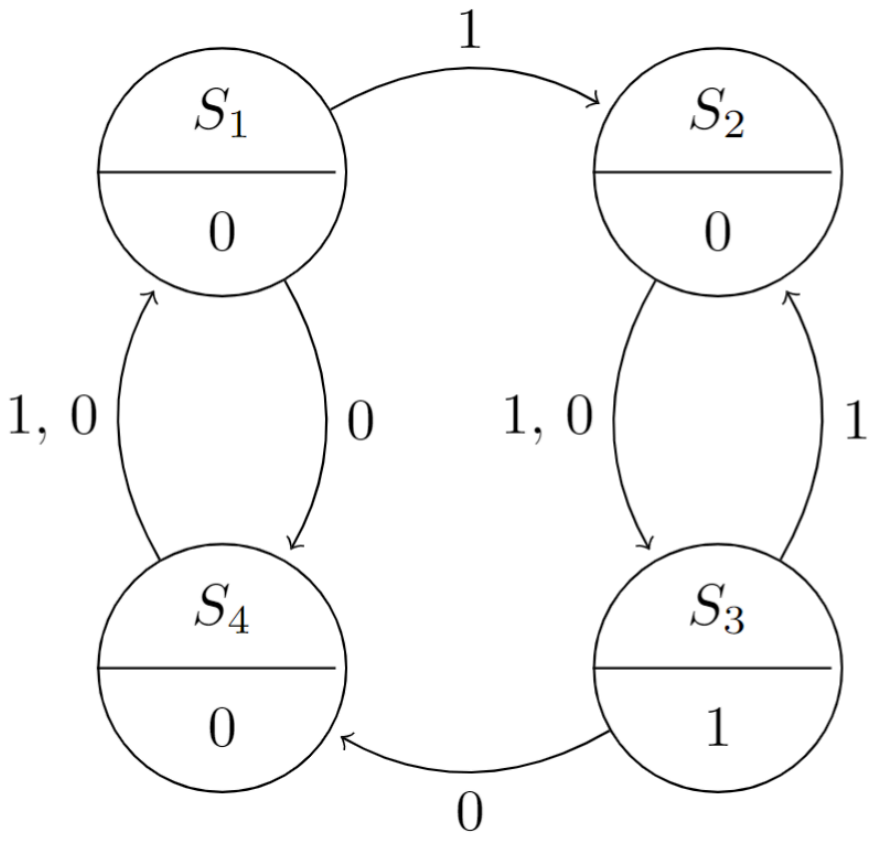

Figure 4 - Graph for Moore Machine implementation of task.

We encoded the states using binary values.

$$State \rightarrow Q_1Q_2$$
$$S_1 \rightarrow 00$$
$$S_2 \rightarrow 01$$
$$S_3 \rightarrow 11$$
$$S_4 \rightarrow 10$$

We constructed Karnaugh maps for the state transitions and a separate K-map for the output.

Considering our design requirements, we chose the JK flip-flop as the most optimal flip-flop for our circuit.

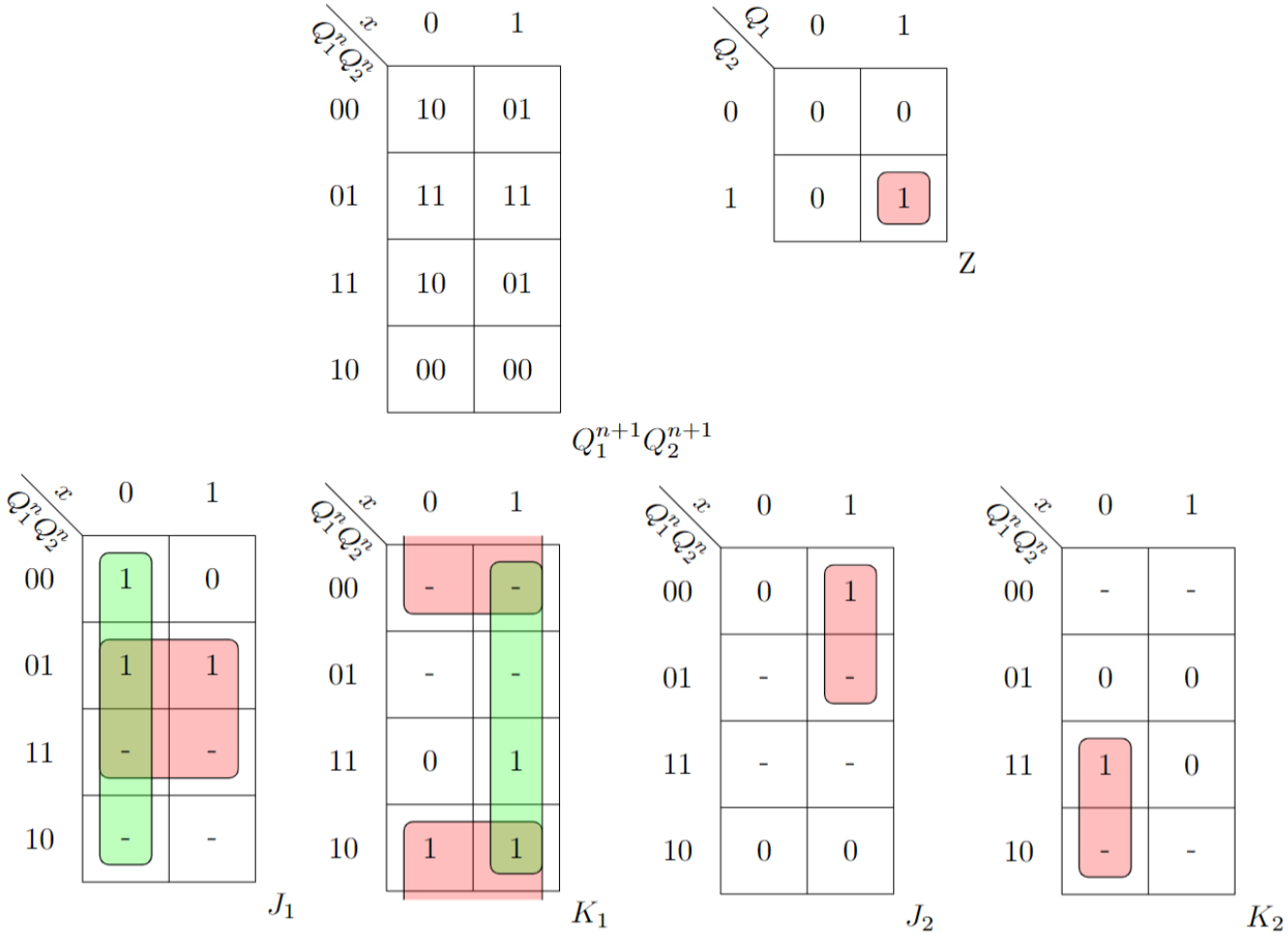

Figure 5 - K-maps obtained from graph used to obtain solution for the task.

From the K-maps we obtained equations, which allowed us to build circuit based on JK type flip-flops, NAND and NOR gates:

$$J_1 = \overline{x} + Q_2^n = \overline{x \cdot \overline{Q_2^n}}$$

$$K_1 = \overline{Q_2^n} + x = \overline{Q_2^n \cdot \overline{x}}$$

$$J_2 = \overline{Q_1^n} \cdot x = \overline{Q_1^n + \overline{x}}$$

$$K_2 = Q_1^n \cdot \overline{x} = \overline{\overline{Q_1^n} + x}$$

$$Z = Q_1^n \cdot Q_2^n = \overline{\overline{Q_1^n} \cdot \overline{Q_2^n}}$$

After transforming the expressions, we translated the expressions into the circuit diagram, which was build during laboratories.

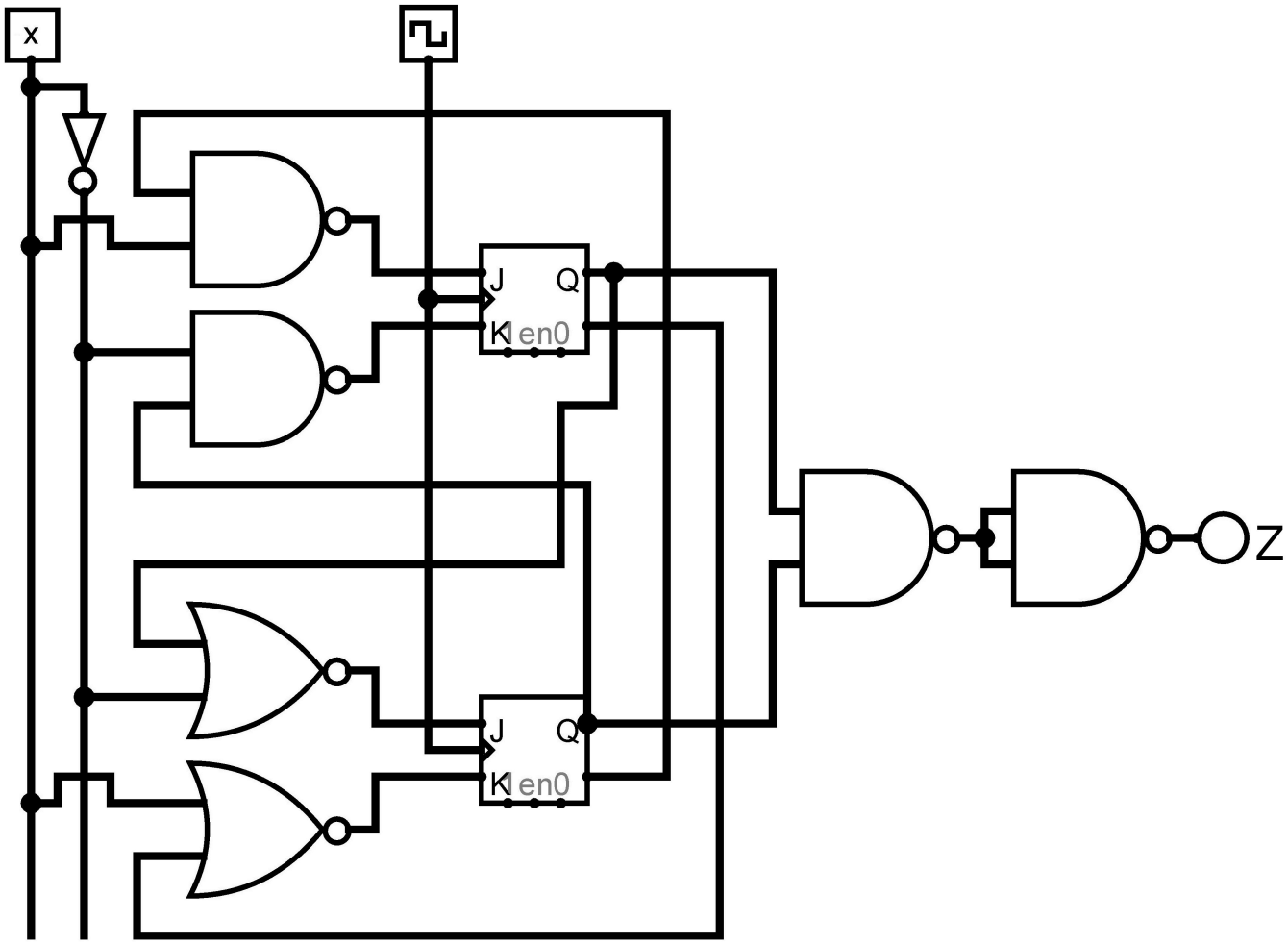

Figure 6 - Implementation of task 3.

Summary

In this task, we designed a synchronous sequential circuit to detect odd 2-bit numbers, with the first bit being the LSB. We started with a Moore state diagram, encoded the states, used Karnaugh maps for simplification, and chose the JK flip-flop as the optimal flip-flop. We successfully implemented the design using both NAND and NOR gates, gaining valuable insights into sequential circuit design and practical implementation.

Task 4

Design a synchronous parallel frequency divider with the filling of the signal equal 1/2. For the values of programming signal equal 0 there should be obtained division by 2, for 1 by 4.

Solution

We started by creating a timing chart to represent the desired behavior of the frequency divider. The chart illustrated the clock signal CLK and the corresponding outputs for different programming signal p values.

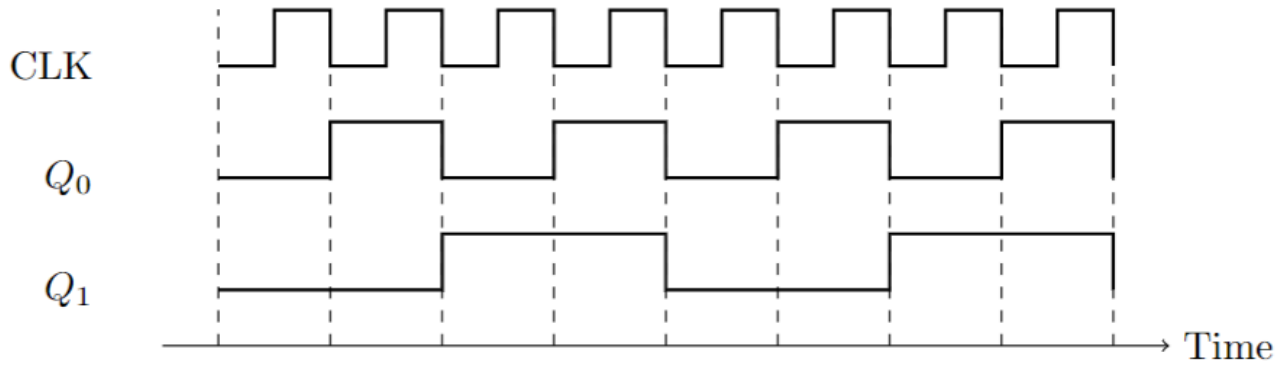

Figure 7 - Timing chart for exercise 4 used to build circuit.

We selected JK flip-flops for their ability to toggle states, which is essential for frequency division. This choice helped in designing a simple and efficient circuit.

When the programming signal p is 0, the first JK flip-flop (Q_0) toggles on every clock pulse. This configuration results in Q_0 having half the frequency of the clock signal CLK.

When the programming signal p is 1, the second JK flip-flop $(Q1)$ is added to the configuration. Q_1 toggles on every second pulse of Q_0, resulting in Q_1 having one-fourth the frequency of the clock signal CLK.

We used NAND gates to control the toggling behavior of the flip-flops based on the programming signal p.

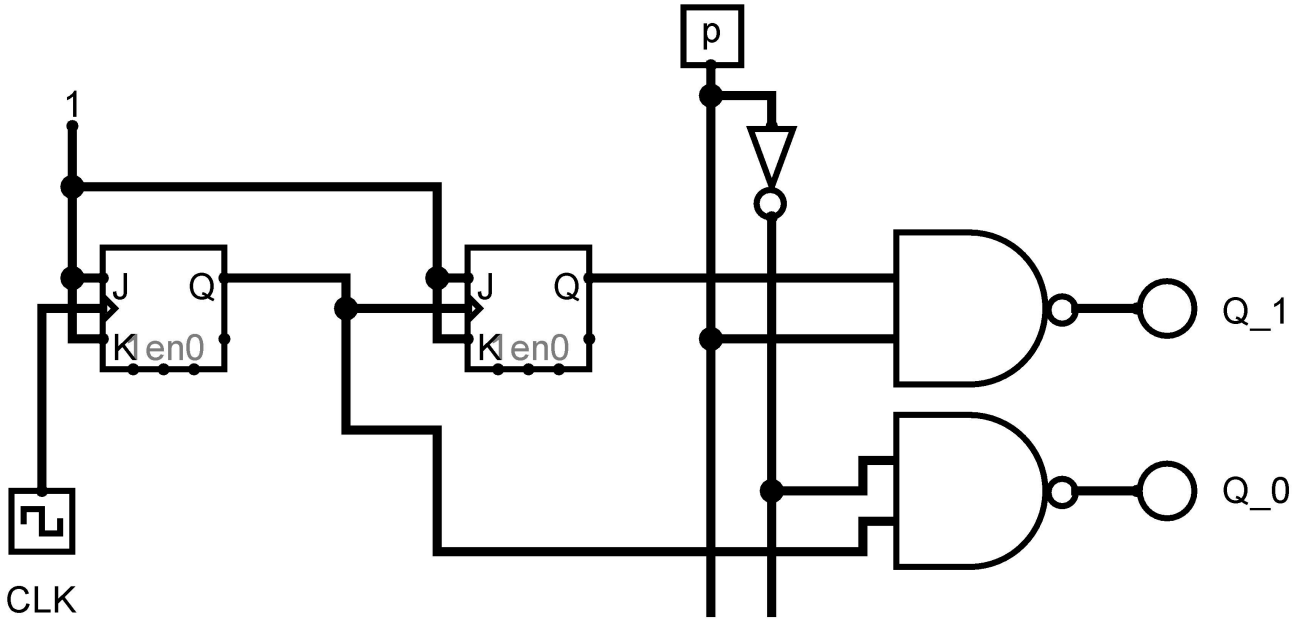

Figure 8 - Implementation of task based on timing chart with use of JK flip-flops and NAND gates.

Summary

In this task, we designed a synchronous parallel frequency divider using JK flip-flops and NAND gates. We began by creating a timing chart to visualize the desired behavior, chose JK flip-flops for their toggling ability, and configured the circuit for both division by 2 and division by 4. By integrating NAND gates, we controlled the flip-flops based on the programming signal, ensuring correct operation. This design effectively met the task requirements, providing a practical implementation of a frequency divider with a 50% duty cycle.

Task 5

Design a synchronous parallel frequency divider with the smallest possible filling of the signal. For the values of programming signals equal 0 there should be obtained division by 2, for 1 by 4.

Solution

We began by creating a timing chart to visualize the desired output waveforms. The chart showed how the outputs Q_0 and Q_1 should behave in relation to the clock signal CLK for both

division by 2 and division by 4.

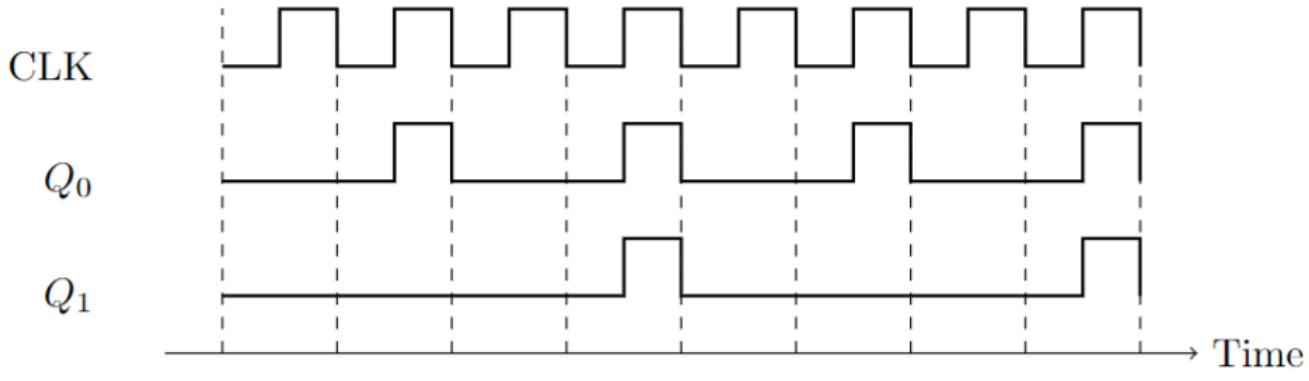

Figure 9 - Timing chart for exercise 4 used to build circuit.

We selected JK flip-flops for the design due to their ability to toggle states, which simplifies the frequency division process.

When the programming signal p is 0, the first JK flip-flop (Q_0) is set to toggle on every clock pulse. This ensures that Q_0 outputs a signal with half the frequency of the clock signal CLK, achieving division by 2.

When the programming signal p is 1, the second JK flip-flop (Q_1) is included in the configuration. Q_1 toggles on every second pulse of Q_0, resulting in an output frequency that is one-fourth of the clock signal CLK, achieving division by 4.

To minimize the duty cycle, we integrated NAND gates to control the flip-flops' toggling behavior based on the programming signal p. The logic was designed to ensure that the outputs Q_0 and Q_1 have the smallest possible duty cycle while maintaining the correct frequency division.

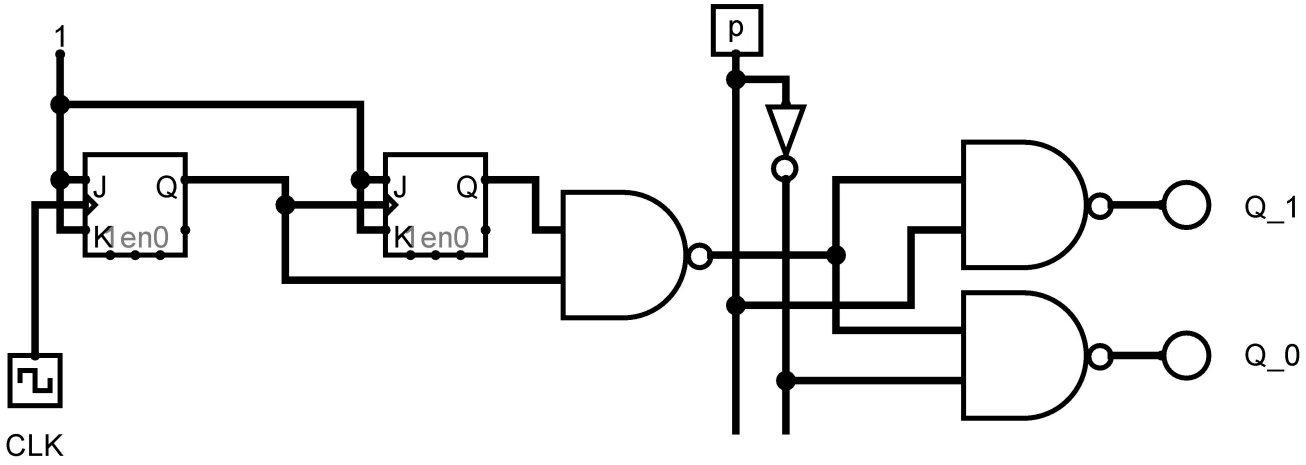

Figure 10 - Implementation of task based on timing chart with use of JK flip-flops and NAND gates.

Summary

In this task, we designed a synchronous parallel frequency divider with minimal signal duty cycles. We began by creating a timing chart to illustrate the required output waveforms, chose JK flip-flops for their toggling capability, and configured the circuit for division by 2 and division by 4 based on the programming signal. By integrating NAND gates, we minimized the duty cycles of the outputs while ensuring correct frequency division. This design effectively met the task requirements, providing an efficient implementation of a frequency divider with the smallest possible signal duty cycle.

Task 7

Design a synchronous circuit executing operation of multiplication by 2 of N-bit binary number, sent to the serial input X starting from the least significant position.

Solution

We started by drawing a Moore state diagram to represent the behavior of the sequential circuit.

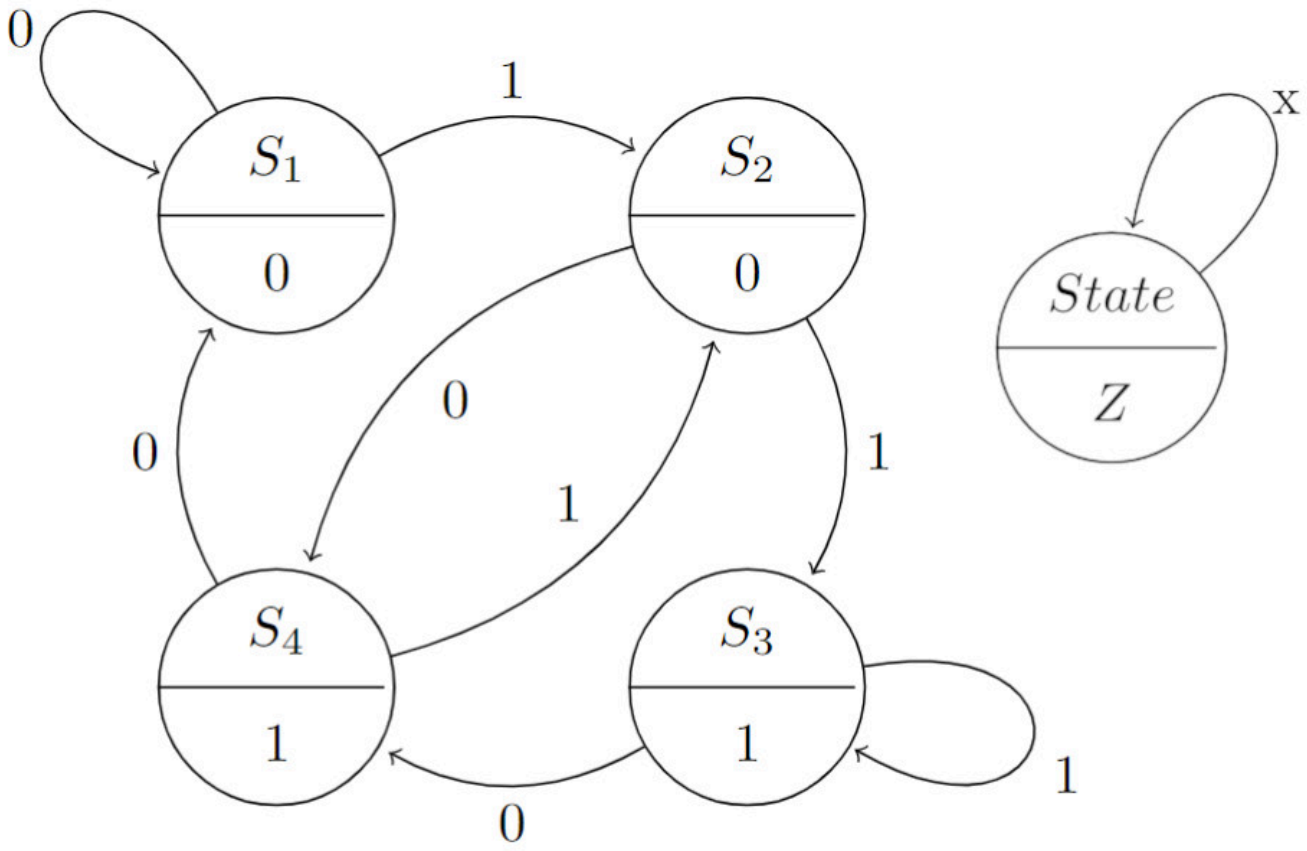

Figure 11 - Graph for Moore Machine implementation of task.

We encoded the states using binary values.

$$State \rightarrow Q_1^n Q_2^n$$
$$S_1 \rightarrow 00$$
$$S_2 \rightarrow 01$$
$$S_3 \rightarrow 11$$
$$S_4 \rightarrow 10$$

We constructed Karnaugh maps for the state transitions and a separate K-map for the output. This allowed us to identify groups and simplify the Boolean expressions for the next states and the output.

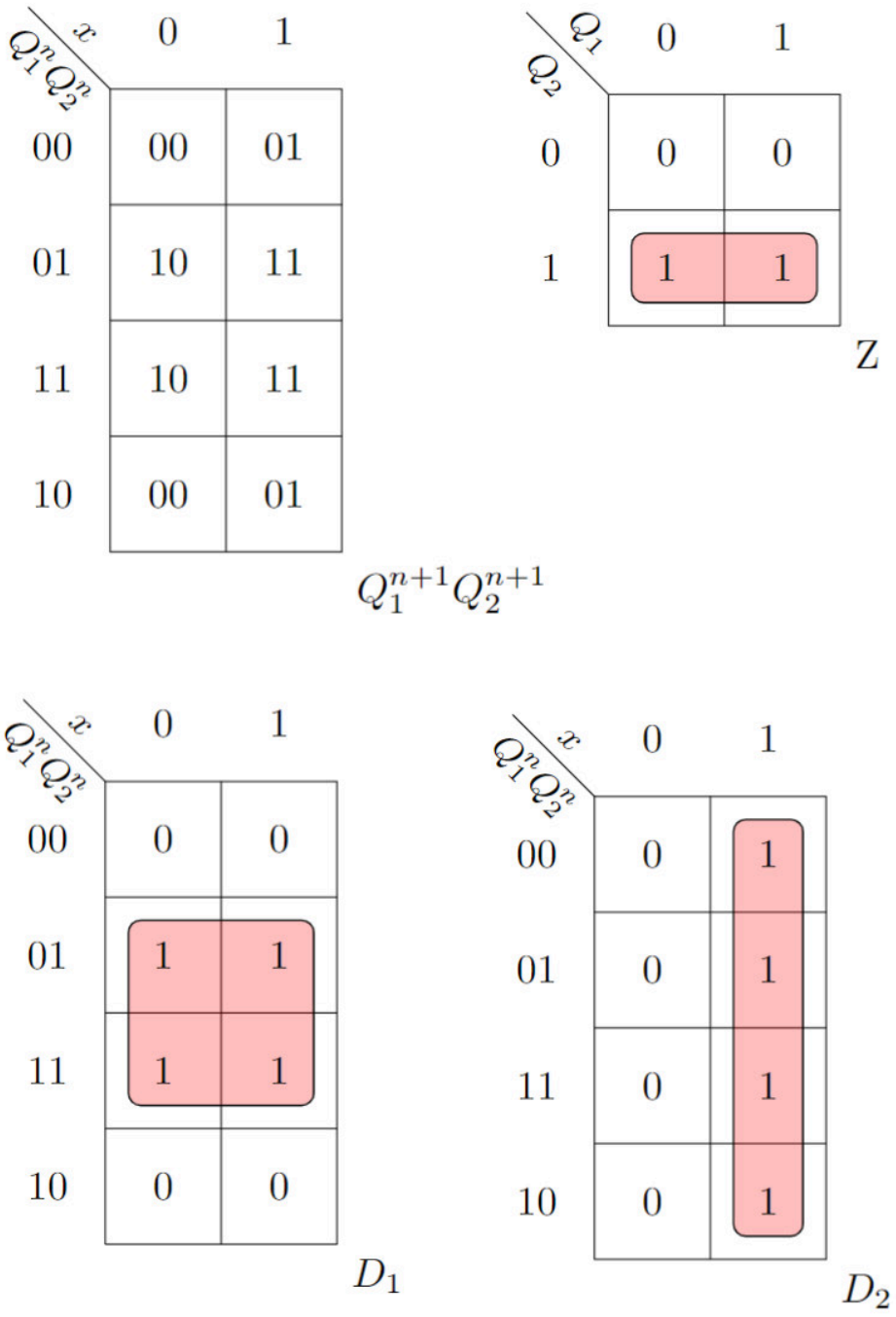

Figure 12 - K-maps obtained from graph used to obtain solution for the task..

From the K-maps we obtained equations, which allowed us to build circuit based on D flip-flops:

$$D_1 = Q_2^n$$

$$D_2 = x$$

$$Z = Q_1^n$$

D flip-flops and NAND gates were employed to create the necessary control logic for the shift register. This included managing the input signal and ensuring that the bits were correctly shifted with each clock pulse.

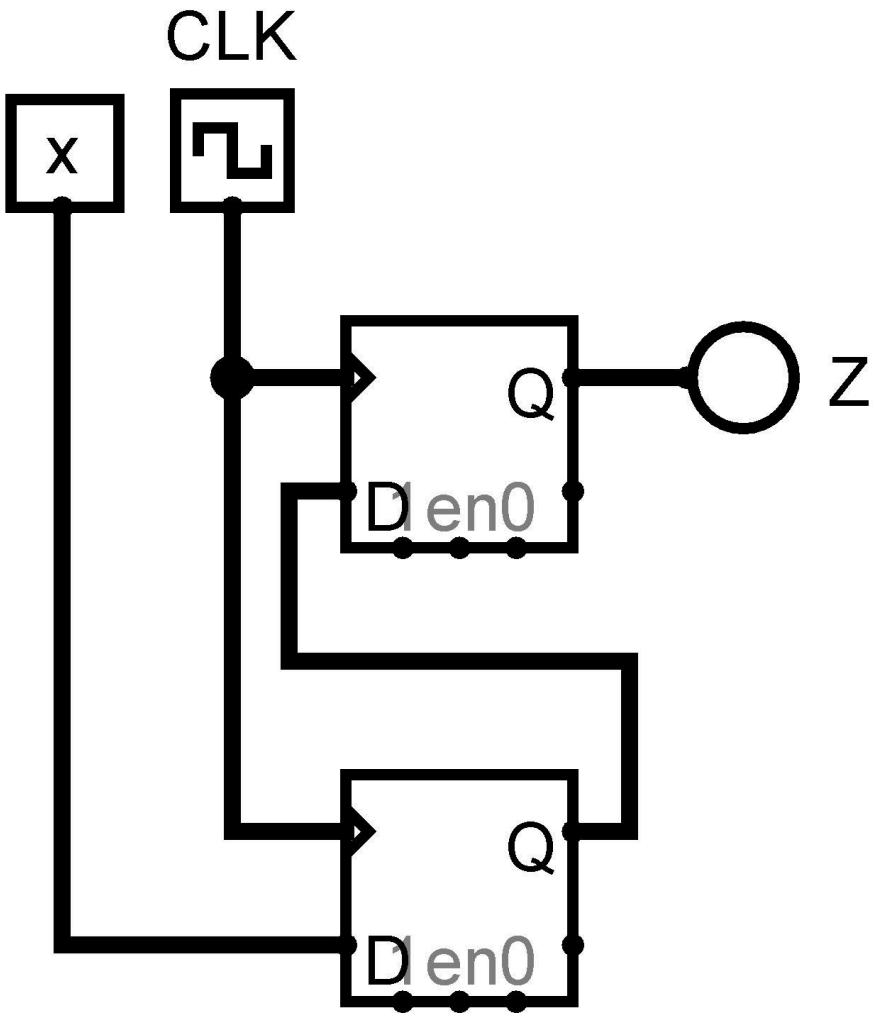

Figure 12 - Implementation of task 7 using D flip-flops.

Summary

In this task, we designed a sequential circuit using a Moore state machine. We started by creating a state diagram, encoding the states, and using Karnaugh maps for simplification. We then selected D flip-flops and implemented the control logic using them and NAND gates as well. This task enhanced our understanding of synchronous sequential circuit design and practical implementation.

Final Conclusions and Observations

In these tasks, we designed and implemented various sequential circuits to achieve specific digital logic functions. We started by creating Moore state diagrams and encoding states using binary values. We utilized Karnaugh maps to obtain Boolean expressions for both state transitions and outputs. Different flip-flops, such as D and JK, were chosen based on the requirements and practicality of each task, and the circuits were implemented with help of using NAND and NOR gates to realize the logic. These exercises allowed us to gain practical experience in sequential circuit design, from theory to real-world implementation.