Hacettepe University Computer Engineering Department

Experiment 5 - Sequential Circuits in Verilog

Due date: 17/12/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2023 Fall

AIM

Designing and simulating sequential circuits in Verilog HDL.

BACKGROUND

Sequential Circuits

Sequential circuits are circuits whose outputs depend on both the present inputs and the sequence of past inputs (sequential circuits include memory elements). That is, the previous inputs or state of the circuit will have an effect on its present state.

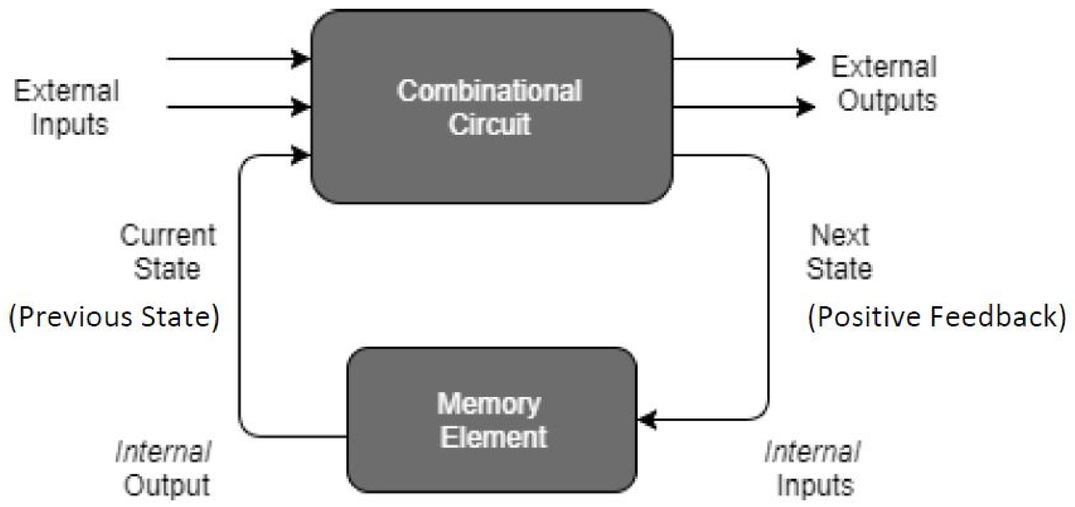
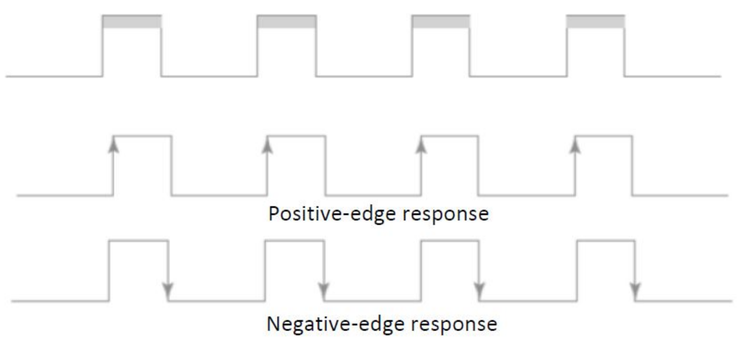


Figure: Sequential Circuit

Storage Elements

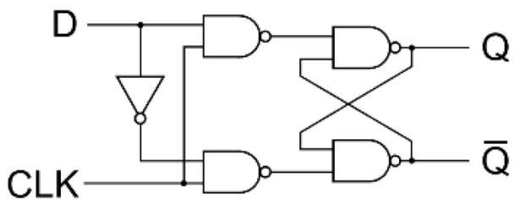
Latches: level-sensitive

Flip-flops: edge-sensitive



A flip flop is a basic building block of sequential logic circuits. It is a circuit that has two stable states and can store one bit of state information. The output changes state by signals applied to one or more control inputs.

Edge-Triggered D Flip Flop



Inputs	Present Output	Next Output
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

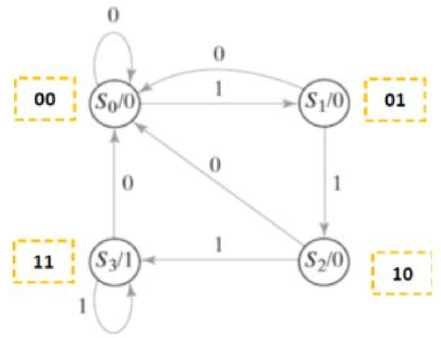
Characteristic Equation
Q(t+1) = D

A basic D Flip Flop has a D (data) input, a clock (CLK) input and outputs Q and Q' (the inverse of Q). Optionally it may also include the PR (Preset) and CLR (Clear) control inputs.

Sequential Circuit Design Steps

To design a sequential circuit, start with the problem definition and use the following steps:

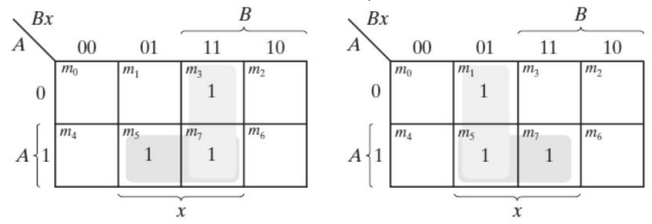
1 Create a state transition diagram from the description. Reduce the number of states if necessary, and assign binary values to the states.



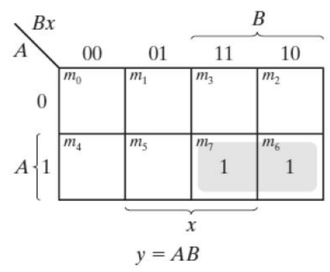
2 Convert the state transition diagram into a state transition table (binary coded state table).

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

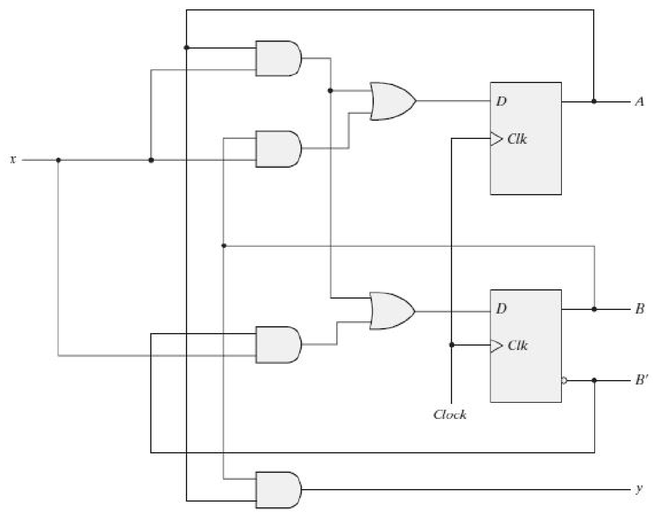
3 Determine the number of flip-flops needed and choose flip-flop types. Derive their excitation tables (if designing a sequential circuit with flip-flops other than the D type), and derive input and output equations from the state table. Minimize the functions for the flip-flop inputs (e.g. using Karnaugh Maps).



4 Determine the combinatorial circuit to represent the output (if any).



5 Finally, use simplified functions to design sequential circuit and obtain the design schematic.



Experiment 5 - Sequential Circuits in Verilog

Due date: 17/12/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2023 Fall

Introduction

In this experiment, we aim to design a finite state machine (FSM) that models a simple automated parking system. The system sequentially checks for available parking spots and guides vehicles to the appropriate location. The states and transitions are based on whether parking spots are available or occupied.

Figure 1: Parking Lot System

The FSM begins in an initial state and progresses through a series of path states. When an empty parking spot is detected, the FSM transitions to a parking state, outputs a signal indicating successful parking, and moves to a parked state. The parked state is maintained as long as the vehicle remains parked. Once the vehicle vacates the spot, the FSM resets to the initial state, ready for a new vehicle.

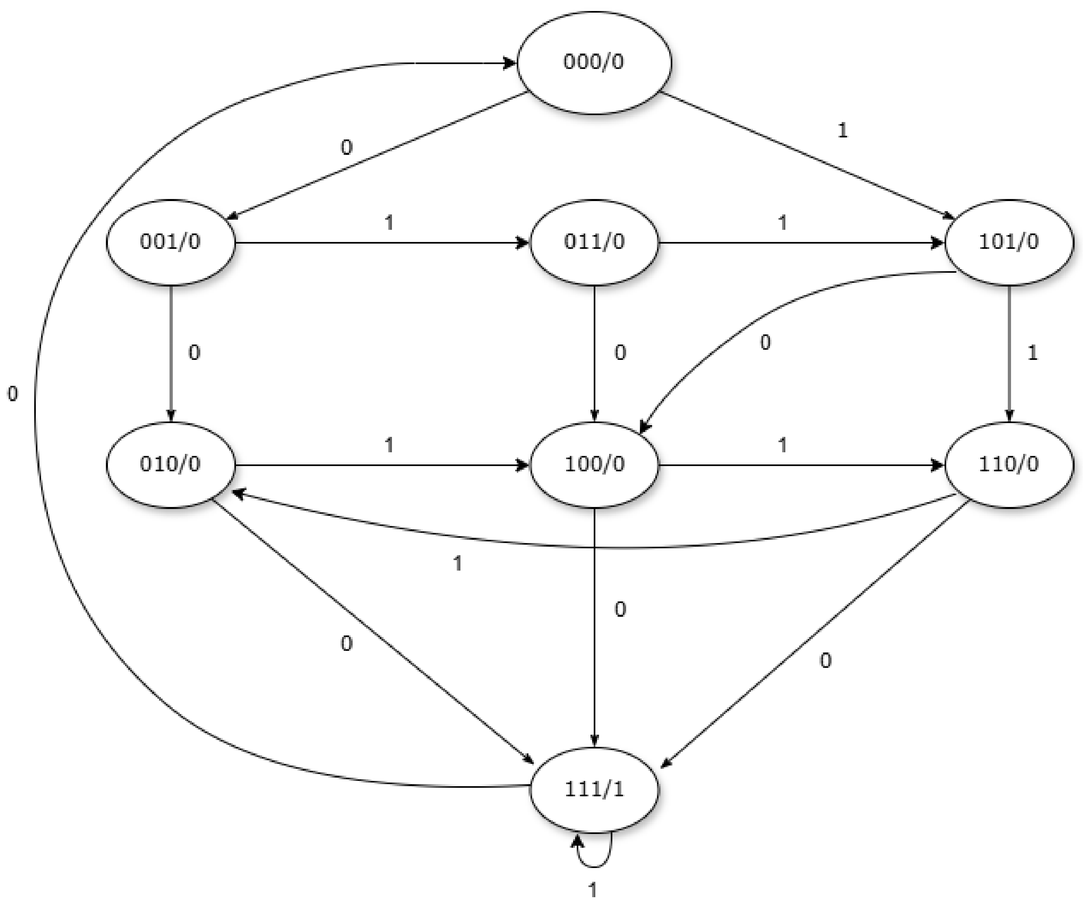


Figure 2: Moore Machine Diagram for Parking

The states below represent their purpose, categorized into paths or parking:

- **Initial State (000):** The starting point of the parking lot.
- **Path States (001, 011, 101):** These states lead to other parking states without providing an output.
- **Parking States (010, 100, 110):** These states remain in a loop until a parking spot is found.
- **Final State (111):** This state outputs a **1** when parking is successfully completed.

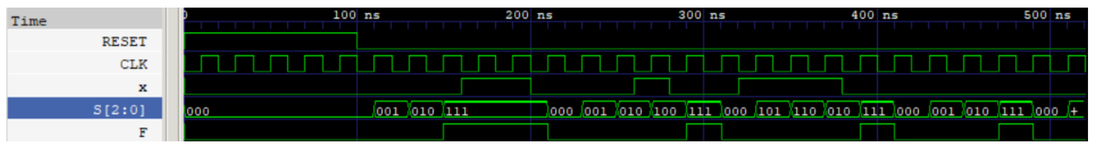
System Specifications

The state diagram of the machine is illustrated in Figure 2. It has eight states, one 1-bit input x, and one 1-bit output F.

You should follow the instructions below:

- Use sequential circuit design steps described on the previous page to obtain the design schematic. Use D flip flops as storage elements to store the state information.
- Implement the design in Verilog using **STRUCTURAL design approach following the circuit schematic. Behavioral design will be graded as 0.** Name your D flip flop module as **dff.v** and machine module as **machine_d.v**
- You must use a Rising Edge D Flip-Flop with Asynchronous Reset on High Level. I.e., it should be triggered on the rising edges of the both **clk** and the **rst** signals, as can be seen in the waveform below.
- **You MUST download and use these starter code files before you start working! Do NOT change the I/O port names!**
- Verify the Verilog model of the machine by writing an appropriate testbench **machine_d_tb.v** for all possible test cases.

Make sure to obtain a similar waveform which shows the correctness of your design. **You MUST test for all possible state transitions.**



System Specifications [BONUS for Extra Credit]

Using the diagram illustrated in Figure 2, follow the instructions given in the first part but use **JK flip flops** instead of D flip flops and name your Verilog source code files as **jkff.v**, **machine_jk.v** and **machine_jk_tb.v** (you MUST use starter code!). In this implementation too you should use a positive-edge-triggered JK Flip-Flop with Asynchronous Reset on High Level. I.e., it should be triggered on the rising edges of the both **clk** and the **rst** signals.

For full credit, you must include all design steps as for the mandatory part, especially **the excitation tables for JK flip flops** to show your work.

Hacettepe University Computer Engineering Department

Experiment 5 - Sequential Circuits in Verilog

Due date: 17/12/2024, 23:59:59 **Via: https://submit.cs.hacettepe.edu.tr/**

BBM233 Digital Design Lab - 2023 Fall

Automatic grading [VERY IMPORTANT!]

Your submissions will be graded using an automatic grading script. There **will not be** any manual grading. So, you are expected to **match the given waveforms 100%**, to receive the full-credit.

You MUST download and use these starter code files before you start working! Do NOT change the I/O ports (names, order, bit width, etc.)!

You must test your codes via our Tur^{6}Bo Grader before the submission to see which tests you are passing. **Note that this is used for testing purposes only**, and **you still have to submit** your full codes via https://submit.cs.hacettepe.edu.tr/ to get graded.

Grading Policy

Verilog codes for D flip flop-based implementation: 100%

- D flip-flop module: 10%
- Implementation of the recognizer machine module using D flip-flops: 90% (behavioral design will be graded with 0!)

Bonus: 10%

PLAGIARISM CONTROL NOTICE

Students must implement their solutions individually. All submissions will be submitted to a plagiarism check. Any submissions that show a high level of similarity will be reported as plagiarism attempts to the ethics committee.

Submission via submit.cs

Submissions will be accepted via https://submit.cs.hacettepe.edu.tr/. Note that the deadline for submission is 10/12/2024 at 23:59:59.

Your submission will include the Verilog codes and it must be in the following format to be accepted:

- b<studentID>.zip
  - dff.v
  - machine_d.v
  - machine_d_tb.v
  - jkff.v
  - machine_jk.v
  - machine_jk_tb.v

Note that only ZIP archives are accepted!