Hacettepe University Computer Engineering Department

Verilog Practice Part 2 - Sequential Circuits

Weeks 10-11 - UNGRADED

BBM233 Digital Design Lab - 2024 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.

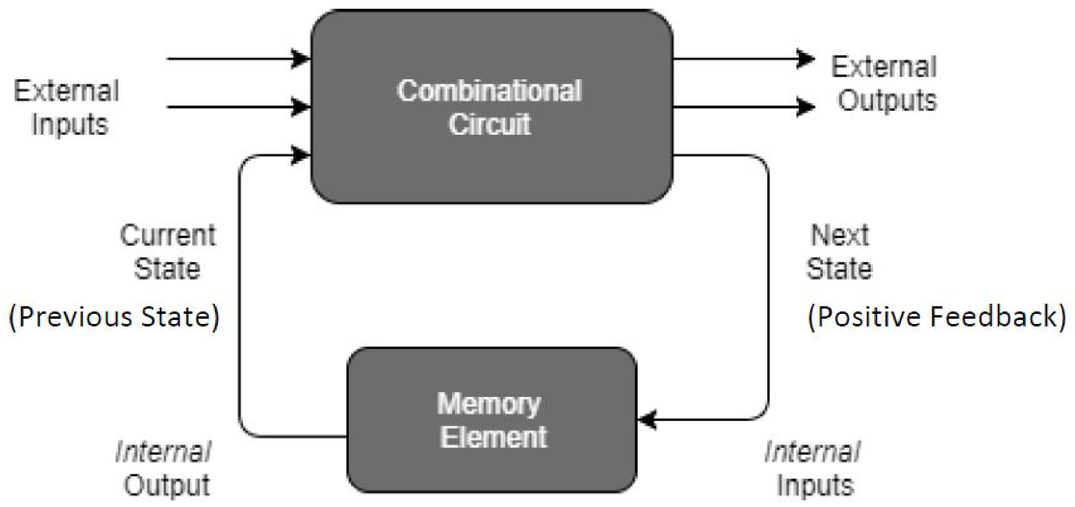
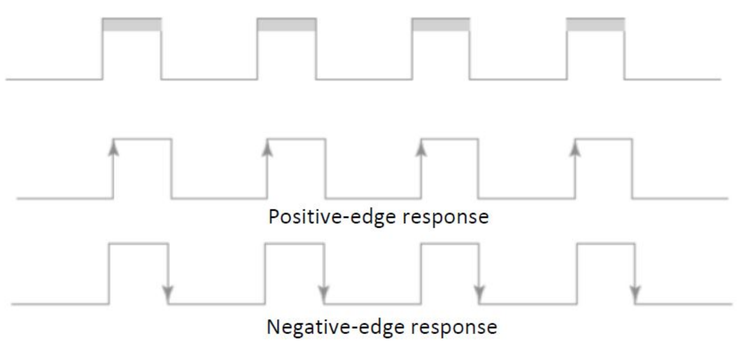


Figure: Sequential Circuit

Storage Elements

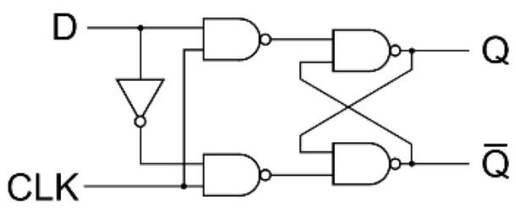
Latches: level-sensitive

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

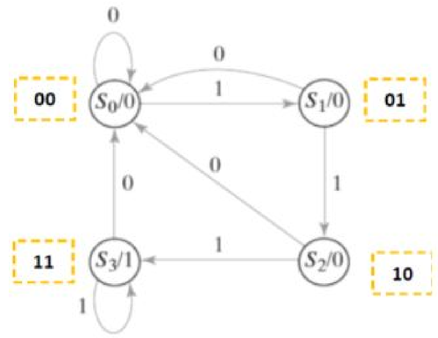
Characteristic Equation
Q(t+1) = D

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:

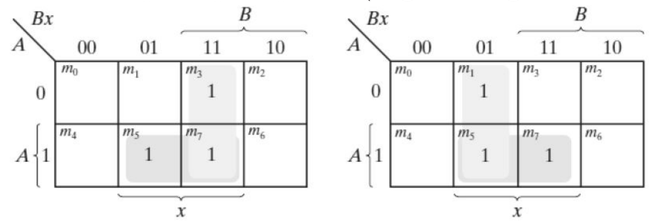
1 Create a state transition diagram from the description. Reduce the number of states if necessary, and assign binary values to the states.



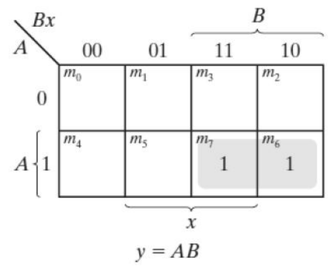
2 Convert the state transition diagram into a state transition table (binary coded state table).

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

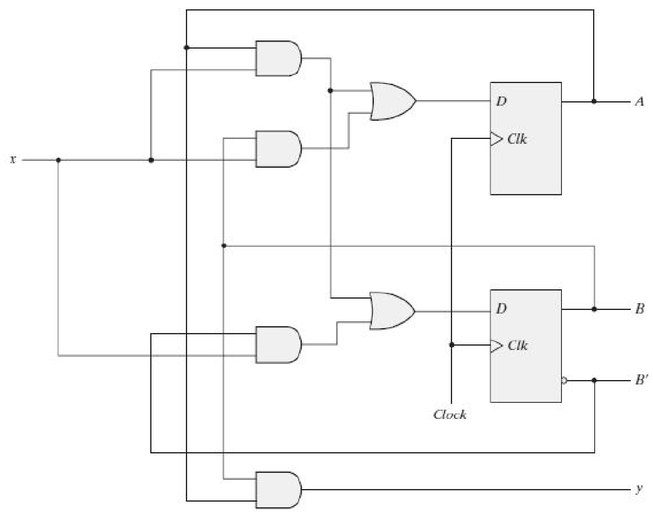
3 Determine the number of flip-flops needed and choose flip-flop types. Derive their excitation tables (if designing a sequential circuit with flip-flops other than the D type), and derive input and output equations from the state table. Minimize the functions for the flip-flop inputs (e.g. using Karnaugh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



Verilog Practice Part 2 - Sequential Circuits

Weeks 10-11 - UNGRADED

BBM233 Digital Design Lab - 2024 Fall

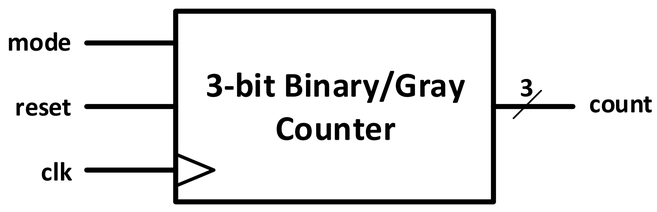
Experiment Introduction

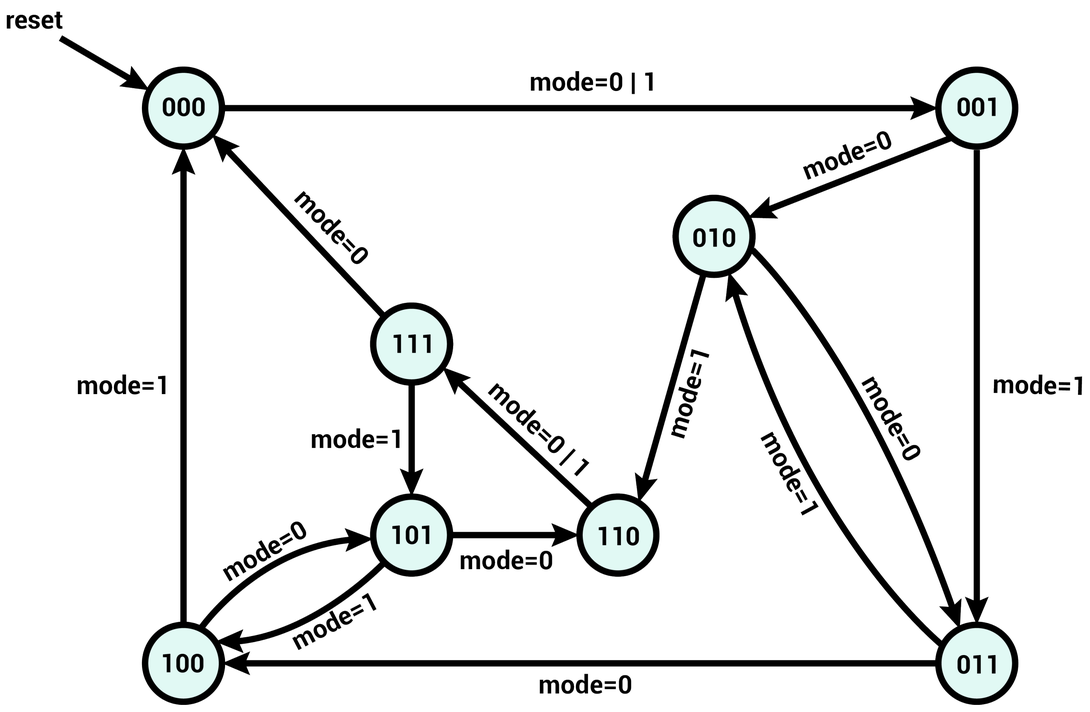
Gray code is a form of binary that uses a different method of incrementing from one number to the next. With gray code, only one bit changes state from one position to another. Below table illustrates the difference between binary and gray code.

Decimal	Gray Code	Natural Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Even though gray code seems like an innocent and theoretical encoding system invented only for fun, it has a lot of important use cases. Here are some example use cases of gray code: **Axes of Karnaugh Maps, mathematical puzzles, telegraphy codes, ADC conversion, error correction, genetic algorithms**.

For this experiment, you are supposed to design a **Binary/Gray Code Counter** circuit using Verilog. The circuit should be designed to **count up** in **binary or gray code** depending on the 1-bit `mode` input variable, as illustrated in the state diagram below. If `mode` is `0`, it should count in natural binary, otherwise it should count in gray code. The circuit you design should also have another 1-bit input variable `reset`, which resets the circuit state to the start state `000`.



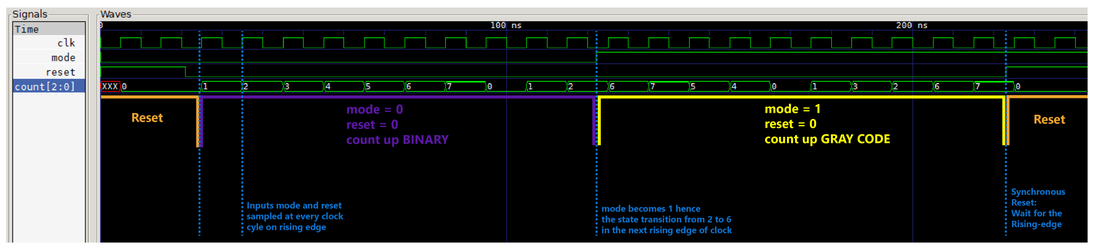


The counter state diagram is illustrated in the figure above. It has eight states, three 1-bit inputs, `reset`, `mode`, and `clk`, and a 3-bit output `count` that outputs the current state of the counter.

Part 0: Implementation With D Flip-Flops

- 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 for both experiment parts.** Name your D flip-flop module as **dff_sync_reset.v** and machine module as **counter_d.v**
- You must use a Rising Edge D Flip-Flop with Synchronous Reset on High Level. I.e., it should be triggered on the rising edges of the **clk** signal, 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, order, or width!**
- Verify the Verilog model of the counter by filling the necessary parts in the given testbench **counter_tb.v** for all possible test cases.

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



Part 1: Implementation With JK Flip-Flop

Using the state transition diagram of the counter given on the left, 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 **jk_sync_reset.v** and **counter_jk.v**. Verify the Verilog model of the counter by commenting out the instantiation part of the **counter_d** module and uncommenting the instantiation part of the **counter_jk** module in the given testbench **counter_tb.v**. (**you MUST use these starter codes!**). Both of your implementations should match the waveform given above.

In this implementation you must also use a positive-edge-triggered JK Flip-Flop with Synchronous Reset on High Level. I.e., it should be triggered on the rising edges of the **clk** signal.

You can test the functionality and design of your codes via our Tur^{6}Bo Grader. Note that the testbench tests are not included for this practice example, but they will be for the assignment.