Digital Components: Sequential Circuits

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

October 10, 2023

Outline

Acknowledgement

The content of most slides come from the authors of the textbook:

Null, Linda, & Lobur, Julia (2018). The essentials of computer organization and architecture (5th ed.). Jones & Bartlett Learning.

Table of Contents

Lesson Objectives

Students are expected to be able to

1. Apply Boolean algebra and functions.
2. *Understand the relationship between Boolean logic and digital computer circuits.*
3. *Learn how to design simple logic circuits.*
4. *Understand how digital circuits work together to form complex computer systems.*

Table of Contents

Concept of Sequential Circuits

- A combinational logic circuit defines its output as a function of its current inputs.
- A sequential circuit defines its output as a function of both its current inputs and its previous inputs
  - To remember previous inputs, sequential circuits must have some sort of storage element.
  - We typically refer to this storage element as a flip-flop.
- Flip-flops
  - SR flip-flop
  - JK flip-flop
  - D flip-flop

Asynchronous and Synchronous

A sequential circuit uses past inputs to determine present outputs indicates that we must have event ordering.

- Asynchronous sequential circuits.
  - They become active the moment any input value changes.
- Synchronous sequential circuits.
  - Synchronous sequential circuits are easier to understand than their asynchronous counterparts.
  - In this lesson, we only examine synchronous sequential circuits
  - From this point, when we refer to "sequential circuit," we are implying "synchronous sequential circuit" unless otherwise stated.

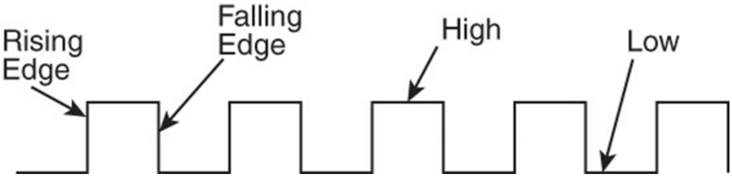
Clock

Synchronous sequential circuits use “clocks” to order events

- A clock is a circuit that emits a series of pulses with a precise pulse width and a precise interval between consecutive pulses.
- This interval is called the clock cycle time.
- Clock speed is generally measured in Herz (Hz), e.g., megahertz (MHz) or gigahertz (GHz).

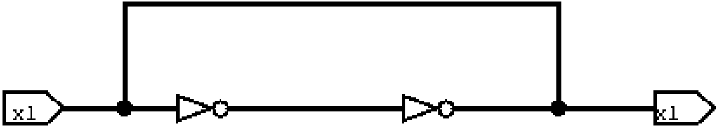
Establishing Timing Order

- Discrete instances of time (i.e., state changes) are established via clock ticks.
- Edge- and Level-Triggered State Changes. Circuits can change state on the rising edge, falling edge, or when the clock pulse reaches its highest voltage.
  - Circuits that change state on the rising edge, or falling edge of the clock pulse are called edge-triggered.
  - Level-triggered circuits change state when the clock voltage reaches its highest or lowest level.



Feedback and State

- To retain their state values (output corresponding to previous input), sequential circuits rely on feedback.
- Feedback in digital circuits occurs when an output is looped back to the input.
- A simple example of this concept is shown below.

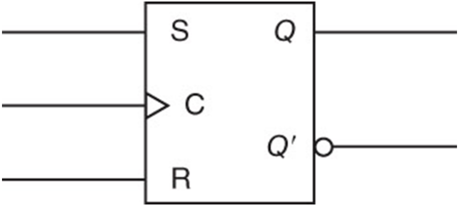


  - If Q is 0 it will always be 0, if it is 1, it will always be 1. Why?

SR Flip-Flop

One of the most basic sequential logic components is the SR flip-flop.

- The block diagram:
  - S: set; R: rest; C: clock; Q: output (state)
  - View them as a function of time (t)



SR Flip-Flop: Truth Table

- The SR flip-flop actually has three inputs: S, R, and its current output, Q.
- Thus, we can construct a truth table for this circuit.
- Notice the two undefined values. When both S and R are 1, the SR flip-flop is unstable.

S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	undefined
1	1	1	undefined

Characteristic Table of SR Flip-Flop

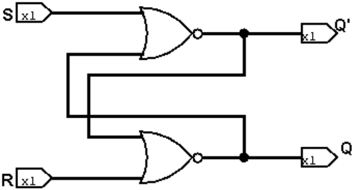
The behavior of an SR flip-flop can be described by a characteristic table.

- $Q(t)$ means the value of the output at time t. $Q(t+1)$ is the value of Q after the next clock pulse.

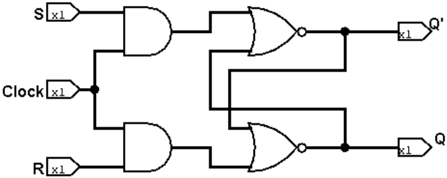
S	R	$Q(t+1)$
0	0	$Q(t)$ – no change
0	1	0 – reset to 0
1	0	1 – set to 1
1	1	undefined

Internals of SR Flip-Flop

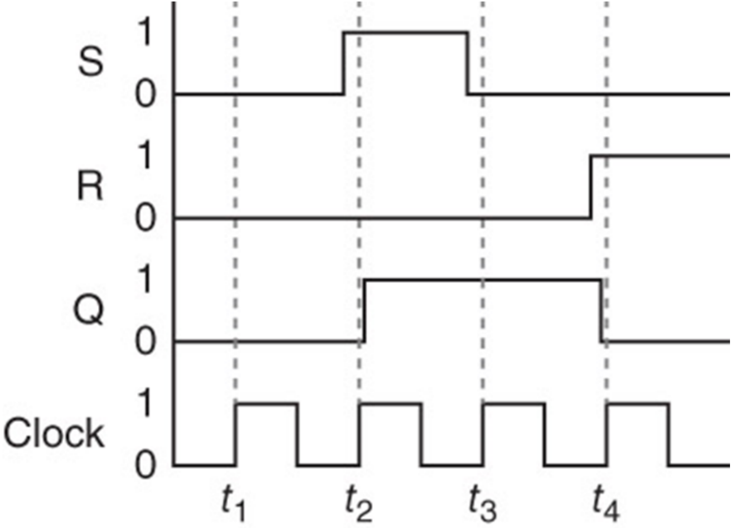
SR Flip-flop



SR Flip-flop with clock

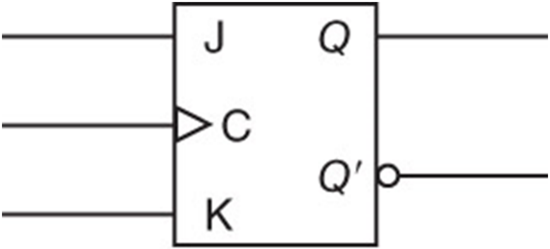


Timing Diagram



JK Flip-Flop

- If we can be sure that the inputs to an SR flip-flop will never both be 1, we will never have an unstable circuit. This may not always be the case.
- The SR flip-flop can be modified to provide a stable state when both inputs are 1.
- This modified flip-flop is called a JK flip-flop.



From SR Flip-Flop to JK Flip-Flop

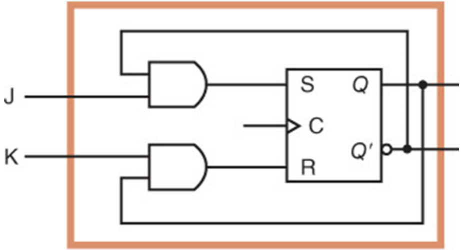
- The characteristic table indicates that the flip-flop is stable for all inputs.

J	K	$Q(t+1)$
0	0	$Q(t)$ – no change
0	1	0 – reset to 0
1	0	1 – set to 1
1	1	$Q'(t)$

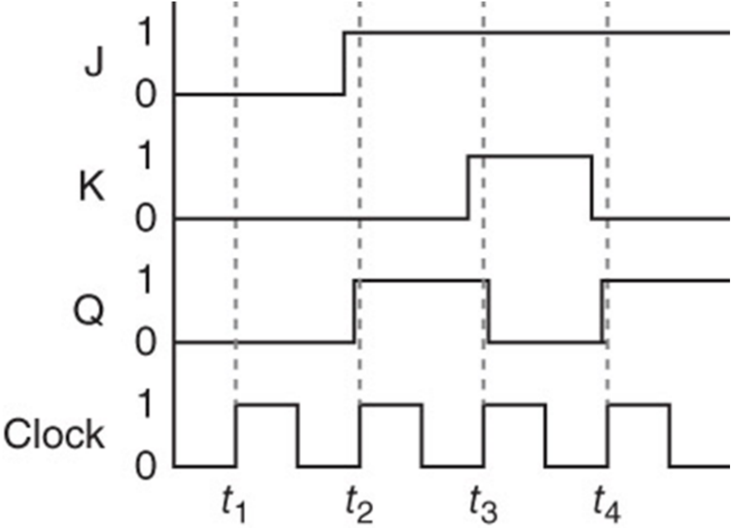
- we see how an SR flip-flop can be modified to create a JK flip-flop.

Internals of JK Flip-Flop

JK Flip-flop



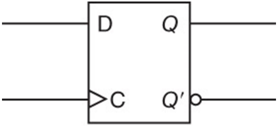
Timing Diagram



D Flip-Flop

- Another modification of the SR flip-flop is the D flip-flop.
- The output of the flip-flop remains the same during subsequent clock pulses. The output changes only when the value of D changes.
- The D flip-flop is the fundamental circuit of computer memory.

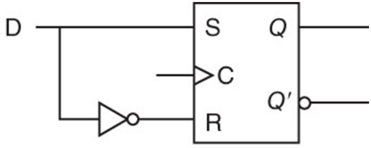
JK Flip-flop



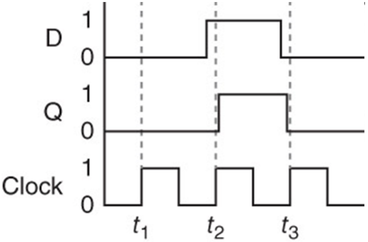
D	$Q(t+1)$
0	0
1	1

D Flip-Flop Internal and Timing

D Flip-flop



Timing Diagram of D Flip-flop



Models for Sequential Circuits – FSMs

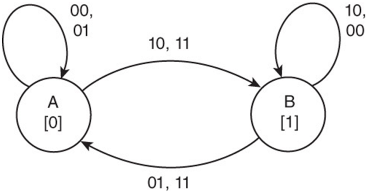
The behavior of sequential circuits can be expressed using characteristic tables or finite state machines (FSMs).

- FSMs consist of a set of nodes that hold the states of the machine and a set of arcs that connect the states.
- Moore and Mealy machines are two types of FSMs that are equivalent.
- They differ only in how they express the outputs of the machine.
- Moore machines place outputs on each node, while Mealy machines present their outputs on the transitions.

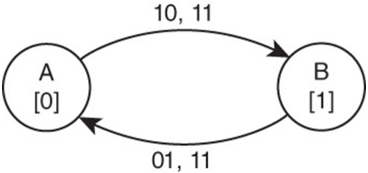
FSM: Examples

The behavior of a JK flop-flop is depicted below by a Moore machine (left) and a Mealy machine (right).

Moor Machine of JK Flip-flop



Mealy Machine of JK Flip-flop



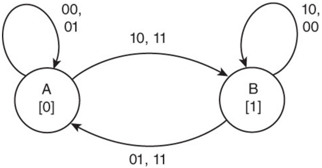
From FSMs to Circuits

Although the behavior of Moore and Mealy machines is identical, their implementations differ.

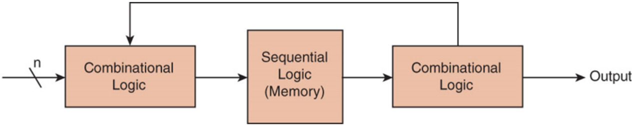
- Moore vs. Mealy machines

From FSMs to Circuits: From Moore Machine

Moor Machine of JK Flip-flop

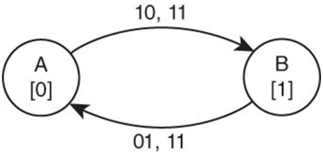


Mealy Machine of JK Flip-flop

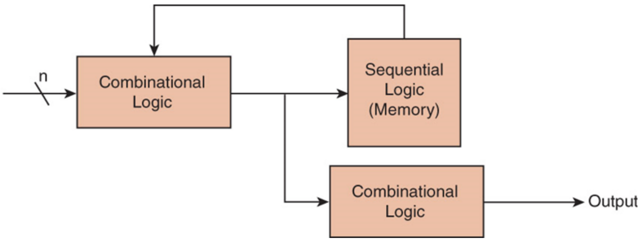


From FSMs to Circuits: From Mealy Machine

Moor Machine of JK Flip-flop



Mealy Machine of JK Flip-flop



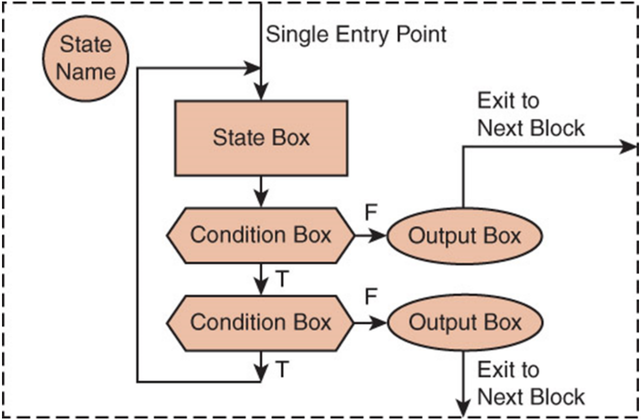
ASMs

It is difficult to express the complexities of actual implementations using only Moore and Mealy machines.

- They do not address the intricacies of timing very well.
- It is often the case that an interaction of numerous signals is required to advance a machine from one state to the next.

The algorithmic state machine (ASM) addresses these shortcomings. The next slide illustrates the components of an ASM.

Components of ASMs



ASM: Example

This is an ASM for a microwave oven.

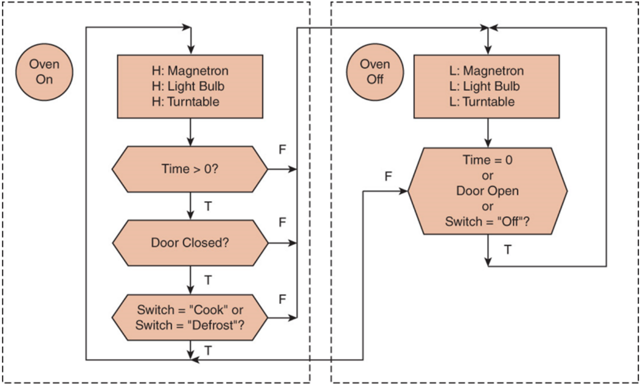


Table of Contents

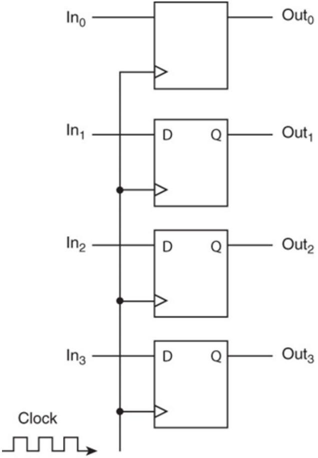
Example Circuits

Sequential circuits are used anytime that we have a “stateful” application.

- A stateful application is one where the next state of the machine depends on the current state of the machine and the input.
- A stateful application requires both combinational and sequential logic.

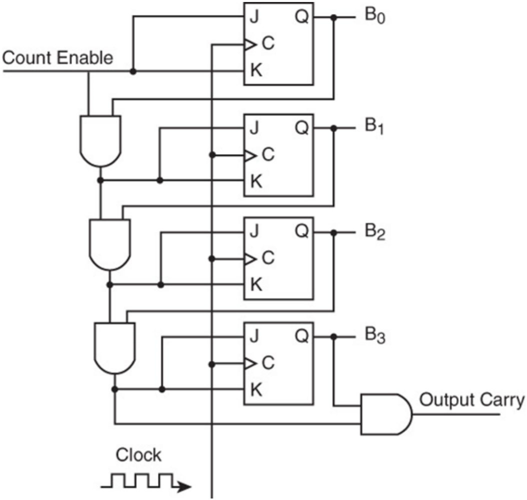
Example Circuit: 4-bit Registers

This illustration shows a 4-bit register consisting of D flip-flops.



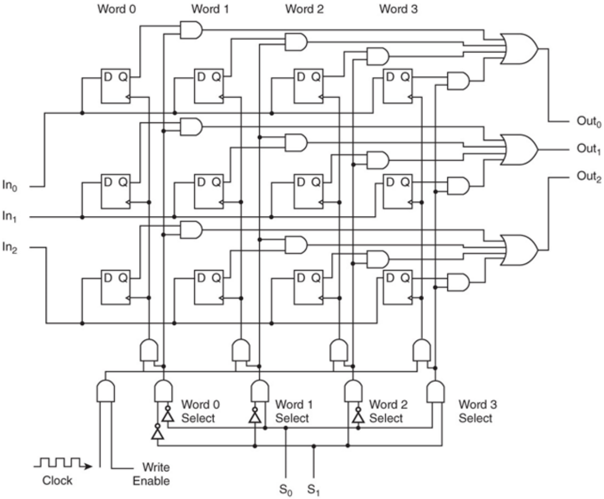
Example Circuit: Binary Counter

This illustration shows a 4-bit binary counter using JK Flip-Flops



Example Circuit: 4×3 Memory

This illustration shows a 4×3 Memory (4 words, each 3 bits)



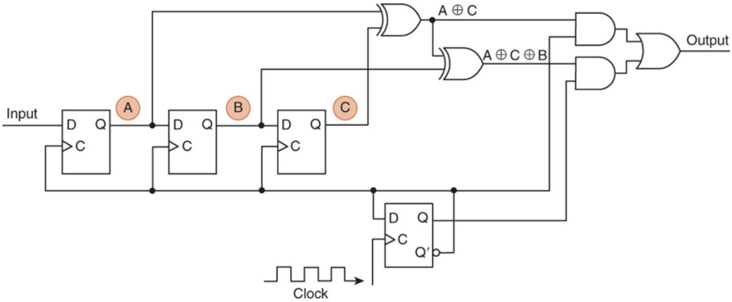
Example Circuit: Convolutional Encoding and Decoding

Convolutional coding and decoding requires sequential circuits.

- One important convolutional code is the (2,1) convolutional code that underlies the PRML code.
- A (2, 1) convolutional code is so named because two symbols are output for every one symbol input.
- A convolutional encoder for PRML with its characteristic table is shown on the next slide.

Encoder Characteristic Table and Circuit

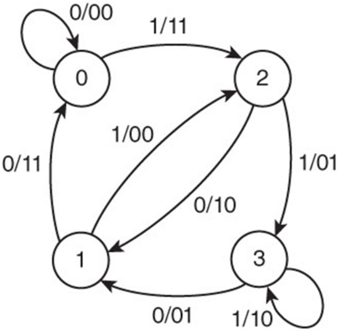
A	B C (t)	B C (t+1)	O	A	B C (t)	B C (t+1)	O
0	00	00	00	0	10	01	10
1	00	10	00	1	10	11	01
0	01	00	11	0	11	01	01
1	01	10	00	1	11	11	10



Encoder Model (Mealy Machine)

The fact that there is a limited set of possible state transitions in the encoding process is crucial to the error correcting capabilities of PRML.

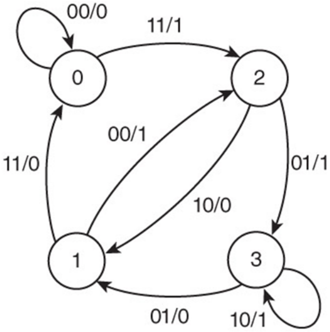
You can see by our Mealy machine for encoding that:



Decoder Model (Mealy Machine)

The decoding of our code is provided by inverting the inputs and outputs of the Mealy machine for the encoding process.

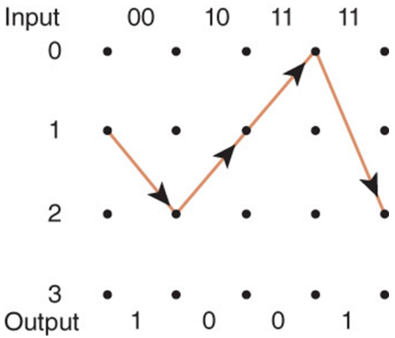
You can see by our Mealy machine for decoding that:



Lattice Diagram

Yet another way of looking at the decoding process is through a lattice diagram.

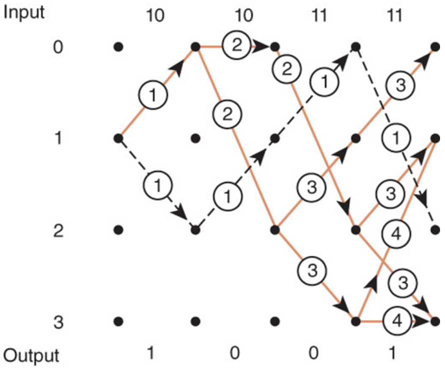
Here we have plotted the state transitions based on the input (top) and showing the output at the bottom for the string 00 10 11 11.



Lattice Diagram

Suppose we receive the erroneous string: 10 10 11 11. We have plotted the accumulated errors based on the allowable transitions.

The path of least error outputs 1001, thus 1001 is the string of maximum likelihood.



Considerations for Circuit Design

We have seen digital circuits from two points of view

- digital analysis
  - explores the relationship between a circuits inputs and its outputs.
- digital synthesis
  - creates logic diagrams using the values specified in a truth table.

Additional consideration:

- be mindful of minute propagation delays that occur between the time when a circuit's inputs are energized and when the output is accurate and stable.

Tool Support for Circuit Design

Digital designers rely on specialized software, such as VHDL and Verilog, to create efficient circuits.

- Software is an enabler for the construction of better hardware.

But, recall the Principle of Equivalence of Hardware and Software — software is in reality a collection of algorithms that could just as well be implemented in hardware.

Circuits (Hardware) or Software?

- When we need to implement a simple, specialized algorithm and its execution speed must be as fast as possible, a hardware solution is often preferred.
- This is the idea behind embedded systems, which are small special-purpose computers that we find in many everyday things.
- Embedded systems require special programming that demands an understanding of the operation of digital circuits, the basics of which you have learned in this chapter.

Table of Contents

Summary and Q&A

You are expected to be able to

1. *Understand the relationship between Boolean logic and digital computer circuits.*
2. *Learn how to design simple logic circuits.*
3. *Understand how digital circuits work together to form complex computer systems.*

Any questions on sequential circuits:

- Clocks and feedbacks
- SR Flip-Flop
- JK Flip-Flop
- D Flip-Flop
- Modeling circuits (FSMs and ASMs)
- Design consideration

Table of Contents

Summary: I

- Computers are implementations of Boolean logic.
- Boolean functions are completely described by truth tables.
- Logic gates are small circuits that implement Boolean operators.
  - The basic gates are AND, OR, and NOT.
  - The XOR gate is very useful in parity checkers and adders.
  - The "universal gates" are NOR, and NAND.

Summary: II

- Computer circuits consist of combinational logic circuits and sequential logic circuits.
- Combinational circuits produce outputs (almost) immediately when their inputs change.
- Sequential circuits require clocks to control their changes of state (i.e., current and past inputs).
- The basic sequential circuit unit is the flip-flop
  - SR, JK, and D flip-flops

Summary: III

- The behavior of sequential circuits can be expressed using characteristic tables or through various finite state machines.
- Moore and Mealy machines are two finite state machines that model high-level circuit behavior.
- Algorithmic state machines are better than Moore and Mealy machines at expressing timing and complex signal interactions.
- Examples of sequential circuits include memory, counters, and Viterbi encoders and decoders.