

Faculty of Engineering

Computer and Systems Engineering Department

**Advance Computer Organization**

Project

**Test Benches**

**Submitted by**

Mina Sherif Henry Tadros

Fall 2018

Table of Contents

[Latch 3](#_Toc535600953)

[Introduction to Latches and Flipflops 3](#_Toc535600954)

[Test Scenarios 3](#_Toc535600955)

[T Flipflop 4](#_Toc535600956)

[Introduction 4](#_Toc535600957)

[Test Scenarios 4](#_Toc535600958)

[Priority 5](#_Toc535600959)

[Test Scenarios 5](#_Toc535600960)

[Mux 6](#_Toc535600961)

[Introduction 6](#_Toc535600962)

[Test Scenarios 7](#_Toc535600963)

[Half Adder 8](#_Toc535600964)

[Introduction 8](#_Toc535600965)

[Test Scenarios 8](#_Toc535600966)

[Full Adder 9](#_Toc535600967)

[Test Scenarios 10](#_Toc535600968)

[Demux 11](#_Toc535600969)

[Introduction 11](#_Toc535600970)

[Test Scenarios 12](#_Toc535600971)

[Decoder 13](#_Toc535600972)

[Inroduction 13](#_Toc535600973)

[Test Scenarios 14](#_Toc535600974)

[Counter 14](#_Toc535600975)

[Introduction 14](#_Toc535600976)

[Test Scenarios 15](#_Toc535600977)

[And 16](#_Toc535600978)

[Introduction 16](#_Toc535600979)

[Test Scenarios 17](#_Toc535600980)

[References 19](#_Toc535600981)

Table Of Figures

[Figure 1: D Latch 3](#_Toc535600530)

[Figure 2:D Flipflop 4](#_Toc535600531)

[Figure 3: Mux circuit 7](#_Toc535600532)

[Figure 4:Full Adder Truth table 9](#_Toc535600533)

[Figure 5: Full Adder Circuit 10](#_Toc535600534)

[Figure 6: Demux Diagram 12](#_Toc535600535)

[Figure 7: Decoder Diagram 14](#_Toc535600536)

[Figure 8: Counter Circuit 15](#_Toc535600537)

[Figure 9: And gate circuit 17](#_Toc535600538)

# Latch

## Introduction to Latches and Flipflops

Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted. In other words, when they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes. There are basically four main types of latches and flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations. In this chapter, we will look at the operations of the various latches and flipflops.

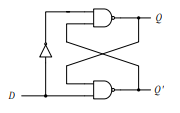


Figure 1: D Latch

## Test Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test scenario | D | Clk | Q | Nq |
|  | 1 | 0 | X | X |
| When clock is high, the output matches the input (1) | 1 | 1 | 1 | 0 |
| When clock is high, the output matches the input (zero) | 0 | 1 | 0 | 1 |
| When clock is zero, input zero is not propagated to output | 0 | 0 | 0 | 1 |
| When clock is zero, input one is not propagated to output | 1 | 0 | 0 | 1 |

# T Flipflop

## Introduction

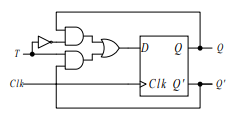


Figure 2:D Flipflop

The T flip-flop has one input in addition to the clock. T stands for toggle for the obvious reason. When T is asserted (T = 1), the flip-flop state toggles back and forth, and when T is de-asserted, the flip-flop keeps its current state. The T flip-flop can be constructed using a D flip-flop with the two outputs Q and Q' feedback to the D input through a multiplexer that is controlled by the T input as shown in Figure 2.

## Test Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test scenario | T | Clk | Clear | Q |
|  | 0 | 0 | 0 | X |
| Clear asynchronously set the output to zero | 0 | 0 | 1 | 0 |
| T change to one, Output does not change | 1 | 0 | 1 | 0 |
| Clear to zero, Output does not change | 1 | 0 | 0 | 0 |
| T at one toggles output from zero to 1 on rising edge clock | 1 | 1 | 0 | 1 |
| Clear asynchronously set the output to zero clearing 1 output | 1 | 1 | 1 | 0 |
| T to zero, Output does not change | 0 | 1 | 1 | 0 |
| Clear to zero, Output does not change | 0 | 1 | 0 | 0 |
| Clock falling edge, Output does not change | 0 | 0 | 0 | 0 |
| T change to one, Output does not change | 1 | 0 | 0 | 0 |
| T at one toggles output from zero to 1 on rising edge clock | 1 | 1 | 0 | 1 |
| Clock falling edge, Output does not change | 1 | 0 | 0 | 1 |
| T at 1 toggles output from 1 to zero on rising edge clock | 1 | 1 | 0 | 0 |

# Priority

## Test Scenarios

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test scenario | Sel0 | Sel1 | Sel2 | Sel3 | Z0 | Z1 | Z2 |
| Out of bounds Input set priority to 111 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Sel is 1000 priority is 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sel is 0100  priority is 001 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Out of bounds Input set priority to 111 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| Sel is 0010 priority is 010 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| Out of bounds Input set priority to 111 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Sel is 0001  Priority is 011 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Out of bounds Input set priority to 111 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Out of bounds Input set priority to 111 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Mux

## Introduction

Multiplexer means many into one. A multiplexer is a circuit used to select and route any one of the several input signals to a signal output. A simple example of an non electronic circuit of a multiplexer is a single pole multiposition switch.

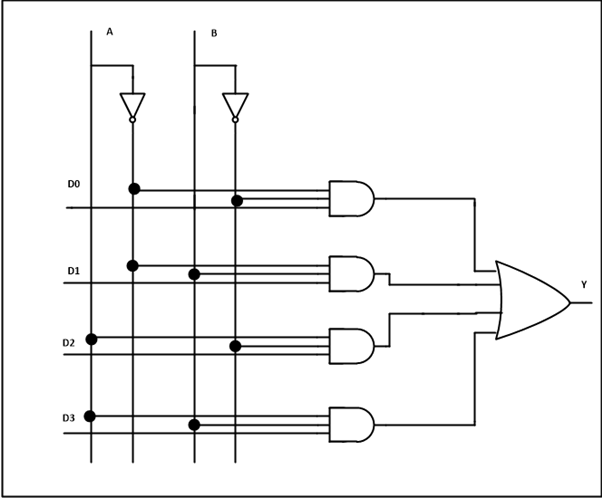


Figure 3: Mux circuit

## Test Scenarios

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test scenario | S1 | S0 | A | B | C | D | z |
| S is 00 and A (zero) is propagated to output | 0 | 0 | 0 | X | X | X | 0 |
| S is 00 and A (1) is propagated to output | 0 | 0 | 1 | X | X | X | 1 |
| S is 01 and B (zero) is propagated to output | 0 | 1 | 1 | 0 | X | X | 0 |
| S is 0 and B (1) is propagated to output | 0 | 1 | X | 1 | X | X | 1 |
| S is 10 and C (zero) is propagated to output | 1 | 0 | X | 1 | 0 | X | 0 |
| S is 10 and C (1) is propagated to output | 1 | 0 | X | X | 1 | X | 1 |
| S is 11 and D (zero) is propagated to output | 1 | 1 | X | X | 1 | 0 | 0 |
| S is 11 and D (1) is propagated to output | 1 | 1 | X | X | X | 1 | 1 |

# Half Adder

## Introduction

From the verbal explanation of a half adder, we find that this circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. We assign symbols a and b to the two inputs and S (for sum) and C (for carry) to the outputs. The C output is 1 only when both inputs are 1. The S output represents the least significant bit of the sum.

The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum-of-products expressions are

(TEI of Crete: Asynchronous eLearning Platform)

## Test Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test scenario | a | B | S | c |
| For Zero input sum and carry is zero | 0 | 0 | 0 | 0 |
| Input bit a is one, sum is one and carry is zero | 1 | 0 | 1 | 0 |
| Both inputs are one, both output carry is set and sum is not set | 1 | 1 | 0 | 1 |
| Input bit b is one and the other is zero, sum is one carry is zero | 0 | 1 | 1 | 0 |

# Full Adder

This type of adder is a little more difficult to implement than a half-adder. The main difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs. The first two inputs are A and B and the third input is an input carry designated as CIN. When a full adder logic is designed we will be able to string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.

The output carry is designated as COUT and the normal output is designated as S. Take a look at the truth-table.

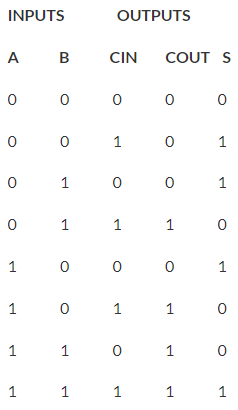


Figure 4:Full Adder Truth table

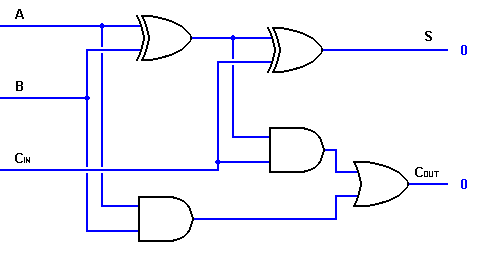


Figure 5: Full Adder Circuit

## Test Scenarios

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test scenario | a | B | c\_in | S | C\_out |
| For Zero input sum and carry is zero | 0 | 0 | 0 | 0 | 0 |
| Input bit a is one, sum is one and carry is zero | 1 | 0 | 0 | 1 | 0 |
| Both inputs are one and input carry is zero, output carry is set and sum is not set | 1 | 1 | 0 | 0 | 1 |
| Input bit b is one and the other is zero, sum is one carry is zero | 0 | 1 | 0 | 1 | 0 |
| Input bit b is one and input carry is one, only output carry is set | 0 | 1 | 1 | 0 | 1 |
| All inputs are set, all outputs are set as a result | 1 | 1 | 1 | 1 | 1 |
| Only bit a and the input carry are set, only output carry is set | 1 | 0 | 1 | 0 | 1 |
| Only input carry is set, only sum bit is set | 0 | 0 | 1 | 1 | 0 |

# Demux

## Introduction

Demultiplexer means one to many. A demultiplexer is a circuit with one input and many output. By applying control signal, we can steer any input to the output. Few types of demultiplexer are 1-to 2, 1-to-4, 1-to-8 and 1-to 16 demultiplexer.

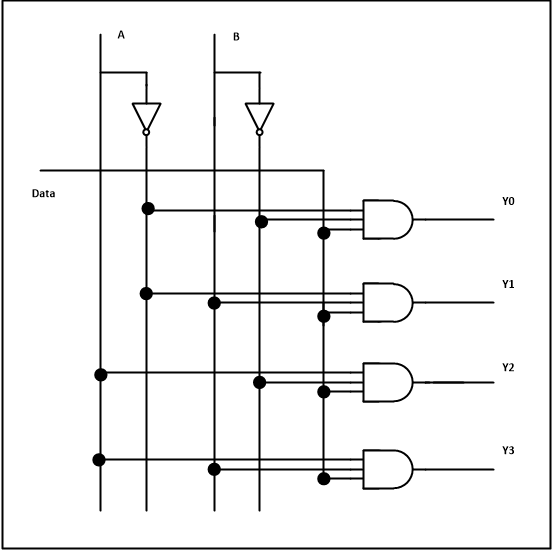


Figure 6: Demux Diagram

## Test Scenarios

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test scenario | A1 | A0 | D | Z3 | Z2 | Z1 | Z0 |
| Select first output and input is zero, only first output is zero | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Select first output and input is one, first output is one | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Select second input and input is one, only second output is one | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Select second output and input is zero, output is zero | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Select fourth output and input is zero, output is zero | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Select fourth output and input is one, only fourth output is one | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Select third output and input is one, only third output is one | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| Select third output and input is zero, output is zero | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

# Decoder

## Inroduction

As its name indicates, a decoder is a circuit component that decodes an input code. Given a binary code of n-bits, a decoder will tell which code is this out of the 2n possible codes.

Thus, a decoder has n- inputs and 2n outputs. Each of the 2n outputs corresponds to one of the possible 2n input combinations.

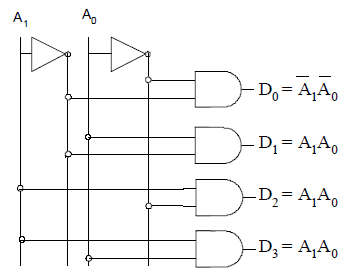


Figure 7: Decoder Diagram

## Test Scenarios

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test scenario | Address2 | Address1 | Address0 | Decode7 | Decode6 | Decode5 | Decode4 | Decode3 | Decode2 | Decode1 | Decode0 |
| Default case | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Designed output | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Default case | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Designed output | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Designed output | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Designed output | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Designed output | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Default case | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Counter

## Introduction

Binary Counters are one of the applications of sequential logic using flip-flops. A counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, in form of a clock pulse. Counters can be formed by connecting individual flip-flops together. On application of pulses, the flipflops in the counter undergo a change of state in such a manner that the binary number stored in the flip-flops represents the number of pulses applied at input. When clock pulses are applied to a counter, the counter progresses from one state to another and the final output of the flip-flop in the counter indicates the pulse count. If all the flip-flops are not clocked at the same time, the counter is asynchronous (or Ripple) and if they are clocked simultaneously, the counter is synchronous. In practice, there are two types of counters:

• Up counters, for increment in value

• Down counters, for decrement in value

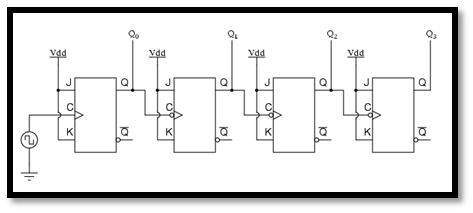


Figure 8: Counter Circuit

## Test Scenarios

|  |  |  |
| --- | --- | --- |
| Test scenario | Clk | count |
|  | 0 | X |
| Clock high, counter incremented | 1 | 1 |
| Clock low, output does not change | 0 | 1 |
| Clock high, counter incremented | 1 | 2 |
| Clock low, output does not change | 0 | 2 |
| Clock high, counter incremented | 1 | 3 |
| Clock low, output does not change | 0 | 3 |
| Clock high, counter incremented | 1 | 4 |
| Clock low, output does not change | 0 | 4 |
| Clock high, counter incremented | 1 | 5 |
| Clock low, output does not change | 0 | 5 |
| Clock high, counter incremented | 1 | 6 |
| Clock low, output does not change | 0 | 6 |
| Clock high, counter incremented | 1 | 7 |
| Clock low, output does not change | 0 | 7 |
| Clock high, counter incremented | 1 | 8 |
| Clock low, output does not change | 0 | 8 |
| Clock high, counter incremented | 1 | 9 |
| Clock low, output does not change | 0 | 9 |
| Clock high, counter incremented | 1 | 10 |
| Clock low, output does not change | 0 | 10 |
| Clock high, counter incremented | 1 | 11 |
| Clock low, output does not change | 0 | 11 |
| Clock high, counter incremented | 1 | 12 |
| Clock low, output does not change | 0 | 12 |
| Clock high, counter incremented | 1 | 13 |
| Clock low, output does not change | 0 | 13 |
| Clock high, counter incremented | 1 | 14 |
| Clock low, output does not change | 0 | 14 |
| Clock high, counter incremented | 1 | 15 |
| Clock low, output does not change | 0 | 15 |

# And

## Introduction

AND operation is represented by C = A • B

Its associated TRUTH TABLE is shown below. A truth table gives the value of output variable (here C) for all combinations of input variable values (here A and B). Thus in an AND operation, the output will be 1 (True) only if all of the inputs are 1 (True).

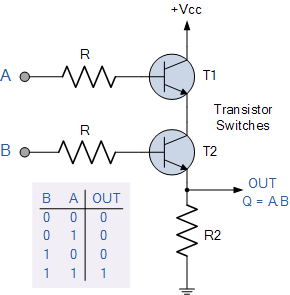


Figure 9: And gate circuit

## Test Scenarios

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Scenario | X4 | X3 | X2 | X1 | y |
| Not all inputs are set to one, output is zero | 0 | 0 | 0 | 0 | 0 |
| Not all inputs are set to one, output is zero | 0 | 0 | 0 | 1 | 0 |
| Not all inputs are set to one, output is zero | 0 | 0 | 1 | 1 | 0 |
| Not all inputs are set to one, output is zero | 0 | 0 | 1 | 0 | 0 |
| Not all inputs are set to one, output is zero | 0 | 1 | 1 | 0 | 0 |
| Not all inputs are set to one, output is zero | 0 | 1 | 1 | 1 | 0 |
| Not all inputs are set to one, output is zero | 0 | 1 | 0 | 1 | 0 |
| Not all inputs are set to one, output is zero | 0 | 1 | 0 | 0 | 0 |
| Not all inputs are set to one, output is zero | 1 | 1 | 0 | 0 | 0 |
| Not all inputs are set to one, output is zero | 1 | 1 | 0 | 1 | 0 |
| All inputs are set to one, output is one | 1 | 1 | 1 | 1 | 1 |
| Not all inputs are set to one, output is zero | 1 | 1 | 1 | 0 | 0 |
| Not all inputs are set to one, output is zero | 1 | 0 | 1 | 0 | 0 |
| Not all inputs are set to one, output is zero | 1 | 0 | 1 | 1 | 0 |
| Not all inputs are set to one, output is zero | 1 | 0 | 0 | 1 | 0 |
| Not all inputs are set to one, output is zero | 1 | 0 | 0 | 0 | 0 |

# References

(n.d.). Retrieved from Computer Engineering | Yeditepe University, Computer Engineering: https://cse.yeditepe.edu.tr/~ayildiz/attachments/flipflops.pdf

(n.d.). Retrieved from TEI of Crete: Asynchronous eLearning Platform.

(n.d.). Retrieved from http://www.circuitstoday.com/half-adder-and-full-adder

(n.d.). Retrieved from King Fahd University of Petroleum and Minerals: http://faculty.kfupm.edu.sa/COE/abouh/Lesson3\_4.pdf

(n.d.). Retrieved from National Institute of Science Education and Research: http://www.niser.ac.in/sps/sites/default/files/basic\_page/Counter%20circuits.pdf

(n.d.). Retrieved from http://www.ni.com/example/14496/en/

(n.d.). Retrieved from University of Washington Computer Science & Engineering.: https://courses.cs.washington.edu/courses/cse370/08wi/pdfs/lectures/04-Logic%20gates.pdf

(n.d.). Retrieved from https://www.electronics-tutorials.ws/logic/logic\_2.html