Boolean Algebra For Circuits

Madiba Hudson-Quansah

Contents

Chapter 1 Logic Gates — Page 2

1.1 AND Gate 2
1.2 OR Gate 2
1.3 NOT Gate 2
1.4 NAND Gate 3
1.5 NOR Gate 3
1.6 XOR Gate 3
1.7 XNOR Gate 3

Chapter 2 Karnaugh Maps — Page 4

Chapter 3 Sequential Logic Circuits — Page 6

3.1 Combinational Logic Circuits vs Sequential Logic Circuits 6
Combinational Logic Circut — 6 • Sequential Logic Circuits — 6 • Full Adder — 6
3.1.3.0.1 Combinational Adder 6
3.1.3.0.2 Sequential Adder 6
3.2 Types of Sequential Logic Circuits 6
Synchronous Sequential Circuits — 7
3.2.1.0.1 Storage Elements 7
3.3 Latches 7
SR Latch — 7 • SR Latch with Clock — 8 • Problems with Latches — 9
3.4 Flip-Flops 9
SR Flip-Flop — 10 • D Flip-flop — 10 • JK Flip-flop — 10 • T Flip-flop — 10
3.5 Characteristic Tables and Characteristic / State equations 10
Characteristic Tables — 10 • State Equations — 11 • Excitation Tables — 11 • Direct Inputs — 12
3.6 Analysis of Sequential Circuits 12
Types of Finite State Machines — 13
3.7 State Reduction and Assignment 13

Chapter 1

Logic Gates

1.1 AND Gate

Table 1.1: And Gate Truth table

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Sum of Products - $x \cdot y$

Product of Sums - $(x + y)\ (x + \overline{y})\ (\overline{x} + y)$ or $(\overline{x} + \overline{y})$

1.2 OR Gate

Table 1.2: Or Gate Truth Table

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Sum of Products - $\overline{x}y + x\overline{y} + xy$ or $\overline{x\overline{y}}$

Product of Sums - $x + y$

1.3 NOT Gate

Table 1.3: Not Gate Truth table

x	$\overline{x}$
1	0
0	1

Sum of Products - $\overline{x}$

Product of Sums - x

1.4 NAND Gate

Table 1.4: NAND Gate Truth table

x	y	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0

Sum of Products - $\overline{x}\overline{y} + \overline{x}y + x\overline{y}$ or xy

Product of Sums - $\overline{x} + \overline{y}$

1.5 NOR Gate

Table 1.5: NOR Gates Truth table

x	y	$\overline{x + y}$
0	0	1
0	1	0
1	0	0
1	1	0

Sum of Products - $\overline{x}\overline{y}$

Product of Sums - $\left(x + \overline{y}\right)\left(\overline{x} + y\right) + \left(\overline{x} + \overline{y}\right)$ or $x + y$

1.6 XOR Gate

Table 1.6: XOR Gate Truth table

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Sum of Products - $\overline{x}y + x\overline{y}$

Product of Sums - $\left(x + y\right)\left(\overline{x} + \overline{y}\right)$

1.7 XNOR Gate

Table 1.7: XNOR Gate Truth table

x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1

Sum of Products - $\overline{x}\overline{y} + xy$

Product of Sums - $\left(x + \overline{y}\right)\left(\overline{x} + y\right)$

Chapter 2

Karnaugh Maps

Definition 2.0.1: Karnaugh Map

A method of simplifying Boolean algebra expressions using a visual representation of the truth table.

A Karnaugh map represents the truth table of a Boolean expression in a grid. The grid is divided into cells, each of which represents a possible combination of inputs. The cells are arranged so that adjacent cells differ by only one input. The cells are then grouped together to form a simplified expression. For a two variable expression the resulting truth table is as follows: And the resulting Karnaugh map is as follows: :

x_1	x_2	F
0	0	m_1
0	1	m_2
1	0	m_3
1	1	m_4

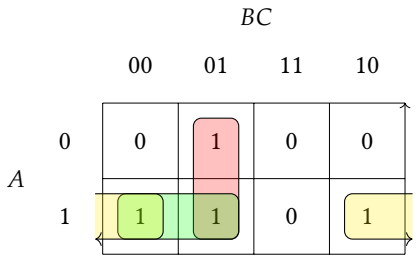
A	B	F
0	0	1
0	1	1
1	0	0
1	1	1

		A	
		0	1
B	0	1	1
	1	0	1

We can the group the cells as $00, 01$ and $01, 11$.

For a three variable expression the resulting truth table is as follows:

A	B	C	F
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	1
1	1	1	0



Simplified:

$$\begin{aligned} A \cdot \overline{B} + \overline{B} \cdot C + A \cdot B &= A\overline{B} + AB + \overline{B}C \\ &= A + \overline{B}C \end{aligned}$$

Chapter 3

Sequential Logic Circuits

3.1 Combinational Logic Circuits vs Sequential Logic Circuits

3.1.1 Combinational Logic Circut

Definition 3.1.1: Combinational Logic Circuits

A combinational logic circuit is a circuit that has no memory. The output of the circuit is determined by the current input only.

At any time the output of a combinational logic circuit depends only on the inputs, with no regard to the previous state of the circuit. Time is not a factor in the output of the circuit.

3.1.2 Sequential Logic Circuits

Definition 3.1.2: Sequential Logic Circuits

A sequential logic circuit is a circuit that has memory. The output of the circuit is determined by the current input and the previous state of the circuit.

A combinational circuit with feedback through stored memory defined as state. Outputs depend on inputs and the state of the circuit, i.e. the previous outputs, with the next state being a function of the current state and the current inputs.

3.1.3 Full Adder

3.1.3.0.1 Combinational Adder

A combinational 4 bit adder requires 4 full adders connected in series, also known as the ripple carry adder, where each carry out propagates to the next full adder and one adder is active at a time.

3.1.3.0.2 Sequential Adder

A sequential 4 bit adder uses just one full adder and a single bit memory element to store the carry out from the previous addition. The carry out is fed back into the carry in of the full adder. A clock signal is used to control the addition process and 4 clock cycles are required to add two 4 bit numbers with each bit being added each cycle.

3.2 Types of Sequential Logic Circuits

There are two types of sequential logic circuits:

- Synchronous - Where the behaviour of the circuit depends on the input signal at discrete instances of time and the output changes only at these instances, i.e. a single clock signal with one clock.
- Asynchronous - Where the behaviour of the circuit depends on the input signal at any instance of time and the order the inputs change, i.e. multiple clock signals with multiple clocks.

3.2.1 Synchronous Sequential Circuits

Definition 3.2.1: Synchronous Sequential Circuits

Employs a synchronizing signal called a clock signal to control the operation of the circuit. The clock signal is a periodic train of pulses that oscillates between high (1) and low (0) states. The clock signal determines when computation occurs and when the output is updated, while other signals determine what changes will occur.

The storage elements (memory) used in clocked sequential circuits are called flip-flops. Each flip-flop can store one bit of information 0, 1 and a circuit may use many flip-flops to define the circuit's state. Each flip-flop state updates only with pulses of the clock signal.

3.2.1.0.1 Storage Elements

A storage element can maintain a binary state indefinitely, until directed by an input signal to switch state, with the main difference between storage element types being the number of inputs they have and how the inputs affect the binary state. The two main types of storage elements are:

- Latches - Level-sensitive, used in asynchronous sequential circuits.
- Flip-Flops - Edge-sensitive, built with latches.

3.3 Latches

Definition 3.3.1: Latch

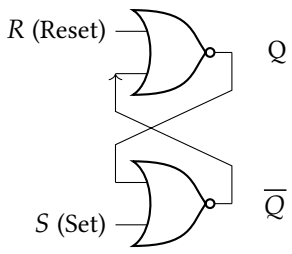
A latch is a sequential logic circuit that has two stable states and can be used to store one bit of information. Changes state during the level of the clock signal, making them level-sensitive and unstable, i.e. not suitable for high speed applications.

A latch is the most basic memory element and can store either a 0 or 1 and can be built using two NOR or NAND gates.

3.3.1 SR Latch

Definition 3.3.2: SR Latch

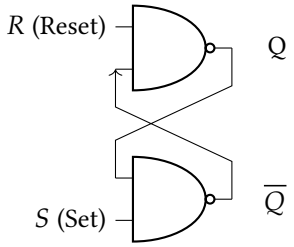
A simple latch that has two inputs, Set (S) and Reset (R), and two outputs, Q and $\overline{Q}$. The latch is level-sensitive and can be used to store one bit of information.



S	R	Q	$\overline{Q}$
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

Table 3.1: Function table

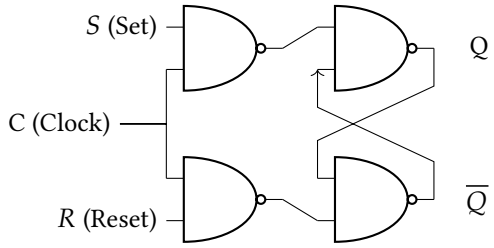
When $S = R = 0$, Q stays the same and $S = R = 1$ is undefined. And implemented with NAND gates the SR latch becomes the $\overline{SR}$ latch:



S	R	Q	$\overline{Q}$
0	1	1	0
1	1	1	0
1	0	0	1
1	1	0	1
0	0	1	1

Table 3.2: Function table

3.3.2 SR Latch with Clock

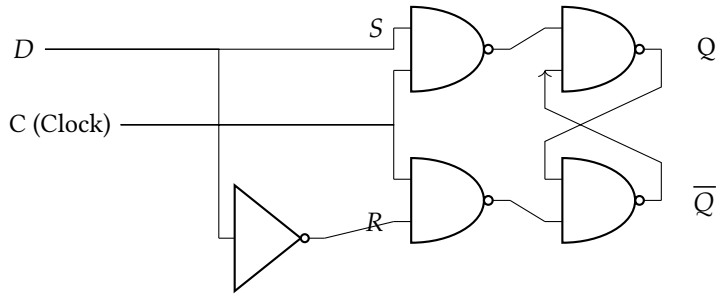


A SR latch can be modified to control when it changes, using a clock signal. The clock signal is used to enable the latch, and the latch only changes state when the clock signal is high. Given by the function table below:

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; Set state
1	1	1	Undefined

Where C is the clock signal. When $C = 0$ the values of S and R have no effect on the latch and only when $C = 1$, does the latch change state.

We can eliminate the undefined state by ensuring S and R are never high at the same time. This can be done by using a single input D for both S and R and inverting the signal going to R ensuring both signals are always inverse. This gives us the D (Data) Latch:



C	D	Next State of Q
0	X	No change
1	0	$Q = 0$; Reset state
1	1	$Q = 1$; Set state

3.3.3 Problems with Latches

- Latches are transparent, i.e. the state keeps changing as long as the clock signal is high, due to this uncertain they are not reliable for storage.
- Latches are level-sensitive, i.e. the output changes when the input changes, making them unsuitable for high speed applications.

3.4 Flip-Flops

Definition 3.4.1: Flip-Flop

A flip-flop is a sequential logic circuit that has two stable states and can be used to store one bit of information. Changes state during the edge of the clock signal, making them edge-sensitive and stable, i.e. suitable for high speed applications. All flip-flops are made from latches.

Flip-flops only change state when the clock signal changes state in either way, i.e. from 0 to 1 (**Rising Edge**) or 1 to 0 (**Falling Edge**) and do not change state when the clock signal is stable. Flip -flops can be made from latches using a slave-master configuration. Where the master latch receives external inputs and the slave latch receives input from the master latch, where depending on the clock signal only one latch is active at a time. If $C = 1$ i.e. the clock signal is high, the master latch is enabled and the inputs are stored in the master latch. If $C = 0$ i.e. the clock signal low, the slave latch is enabled and the inputs are generated using the slave latch.

The delay of logic gates inside the flip-flop is crucial to the operation of the flip-flop, as the output of the flip-flop depends on the inputs and the previous state of the flip-flop. The delay of the flip-flop is the time taken for the output to change after the clock signal changes state. There are two timings to consider when designing a flip-flop:

Setup Time T_s - The minimum time during which D input must be maintained before the clock transition occurs.

Hold Time T_h - The minimum time during which D input must be maintained after the clock transition occurs.

3.4.1 SR Flip-Flop

Definition 3.4.2: SR Flip-Flop

Built using two latches (Master $C = 1$ and Slave $C = 0$), Q is sampled at the falling edge of the clock signal. Data is entered during the rising edge of the clock pulse, but the output is not reflected until the falling edge when Q is sampled.

Master				Slave			
C	S	R	Y	$\overline{C}$	S	R	Q
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	0
1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	1	1	0	0	1	1	0

3.4.2 D Flip-flop

Definition 3.4.3: D Flip-Flop

A D flip-flop is a simple flip-flop that has one input, D (Data), and one output, Q. The D flip-flop is edge-sensitive and changes state at the rising edge of the clock signal. The D flip-flop is used to store one bit of information.

3.4.3 JK Flip-flop

3.4.4 T Flip-flop

3.5 Characteristic Tables and Characteristic / State equations

3.5.1 Characteristic Tables

Definition 3.5.1: State tables

Defines the operation of a flip-flop in a tabular form. Where the next state is defined in terms of the current state and the inputs, Where:
Current State $Q(t)$ - Refers to the current state before the clock signal causes a change in state.
Next State $Q(t+1)$ - Refers to the state after the clock signal causes a change in state.

The Characteristic table for a SR flip-flop is as follows:

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

D Flip-flop:

T Flip-flop:

D	$Q(t)$
0	0
1	1

T	$Q(t)$
0	$Q(t)$
1	$\overline{Q(t)}$

And the JK flip-flop:

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

Table 3.3: JK Flip-flop State table

3.5.2 State Equations

Definition 3.5.2: State equation

Defines the operation of a flip-flop in an algebraic form

The state equation for the SR Flip-flop is:

$$Q(t+1) = S + \overline{R}Q(t)$$

D Flip-flop is:

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

JK Flip-flop:

$$Q(t+1) = J\overline{Q(t)} + \overline{K}Q(t)$$

And the T Flip-flop:

$$Q(t+1) = T \oplus Q(t)$$

3.5.3 Excitation Tables

Definition 3.5.3: Excitation Tables

Defines the inputs required to change the state of a flip-flop. Where:
Current State $Q(t)$ - Refers to the current state before the clock signal causes a change in state.
Next State $Q(t+1)$ - Refers to the state after the clock signal causes a change in state.

D Flip-flop:

$Q(t+1)$	D	Operation
0	0	Reset
1	1	Set

SR Flip-flop:

$Q(t)$	$Q(t+1)$	S	R	Operation
0	0	0	X	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change

JK Flip-flop:

$Q(t)$	$Q(t+1)$	J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No change

T Flip-flop:

$Q(t+1)$	T	Operation
$Q(t)$	0	No change
$\overline{Q(t)}$	1	Complement

3.5.4 Direct Inputs

Some flip-flops have asynchronous inputs to set / reset their states independently of the clock.

3.6 Analysis of Sequential Circuits

Analysis describes what a given circuit will do. The behaviour of a clocked sequential circuit is determined from the inputs and outputs and the states of the Flip-flop. The steps involved in analysis are:

- Obtain state equations.
- Fill the state table.
- Draw the state diagram.

For example given the sequential circuit below:

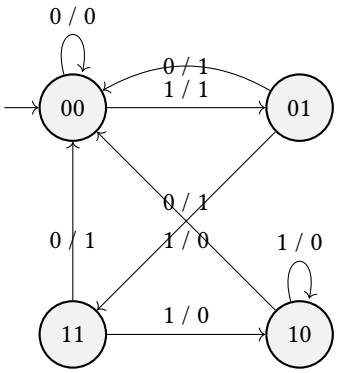
First we obtain the state equations:

$$A = xA + xB$$
$$B = \overline{A}x$$
$$y = (B + A) \cdot \overline{x}$$

Then we fill the 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	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Finally we draw the state diagram:



3.6.1 Types of Finite State Machines

There are two types of finite state machines and therefore two types of sequential circuits:

Moore Machine - The output depends only on the current state.

Mealy Machine - The output depends on both the current state and the inputs.

3.7 State Reduction and Assignment

Definition 3.7.1: State Reduction

The process of reducing the number of states of a state machine while keeping the input-output behaviour unchanged.

Given the finite state machine below: