

CSC258 Computer Organizations, Fall Term, 2019



Bulletpoint notes by [Tingfeng Xia](#). Work licensed under a creative commons non-commercial share-alike 4.0 license. This is not meant to be a set of comprehensive notes, rather it is my pick of bullet list of material from the course.

Table of Contents

1. [Transistors](#)

2. [Circuits](#)

Transistors

Doping

Semiconductors, such as silicon, they themselves do not conduct electricity, so we have to add impurities to the material. (We call this process doping) There are two types of doping, n-type and p-type.

n-type doping

We add Phosphorus which is a Group XV element meaning it has 5 electrons but only 4 of them are needed to form a covalent bond with silicon so we have one extra electron floating around.

IMPORTANT: Notice that here the doped material does not carry any overall charge! It is Phosphorus atom being added not ion, so the overall charge is zero.

A word on naming: Since in this case, the substance has negative charge floating around, we call it n-type.

p-type doping

We add Boron to Silicon in this case. Since Boron has three electrons on the outmost shell, it would make the Si structure lack one electron to form all 4 covalent bonds. In this case we say that the structure has a hole in it, which is basically a space for an electron. Notice that, again, we don't have an overall charge on our material.

Diffusion and Drift - Equilibria

These are two currents formed by a pn-junction. We will first need to explore the electric field formed by this. So before we put the pn junction into one piece, they are neutral individually by themselves. At the time we put them together, a **diffusion current** occurs. Then the positively doped side has electrons from the other side so the p-type side is now overall negative in charge

(with extra electrons), and the n-type material is now overall positive in charge (short in electron).

... **This causes an electric field.** In an electric field, an electron is given a force to move from the negatively charged side to the positively charged side, and the current caused by this movement of electrons is called a **drift current**.

A Diffusion is the current of electron moving from n-type doped material to p-type doped material. (一开始电子填补空洞而产生的电流) In conventional current this goes from p to n. **A Drift** is the current formed by electron being 'sucked' back under the force of the electric field. (后来电子因为电场的作用而从负极到正极所产生的电流) In terms of conventional current, it goes n → p.

A word on naming: A way to think of the naming is: diffusion means to spread, to even out so it is the process of evening out the electrons/holes in the materials. And a drift could be thought in the way that electrons have moved from p → n now they have to move back n → p, so the electron needs to 'drift' to make a u-turn.

An Equilibrium will be achieved eventually, this is intuitive. We call the layer of 'neutral' layer in a pn-junction a **depletion layer**.

Bias - Basis of Transistors

Forward Bias (Conducting)

+ [P-type | n-type] -

In this case:

- The p-type end, which is originally more negative (since electrons are gained) gets connected to positive, making this half more neutral, and hence decrease the width of the depletion layer.
- The n-type end, which is originally more positive (since electrons are lost) gets connected to negative, where it can gain electrons, so this half is more neutral now and this decreases the width of the depletion layer.

Hence the application of voltage decreased the depletion layer and made the pn-junction conductive.

Reverse Bias (Insulating)

- [P-type | n-type] +

In this case the p-type end, which is originally more negative, gets negative, which means gaining more electron so it further increases the width of the depletion layer. (Similar for the n-type end) So all in all, the depletion layer increases and hence the pn-junction becomes insulating.

MOSFET

Metal Oxide Semiconductor Field Effect Transistor; There are two types of nMOS and pMOS; Notice that in both nMOS and pMOS, if we don't apply any voltage to them, the transistor acts as a two way insulator since at least one way the depletion layer will increase.

nMOS (n-type channel formed; voltage apply -> conduct)

The nMOS is formed by a npn combination of doped semi-conductors. There is a metal connection (The M in MOSFET) called gate at the top of the p-type substrate. When a 5V voltage is applied to the gate, electrons will gather at the top of the p-type substrate and then the two n-type substrates are connected through this n-type channel.

pMOS (p-type channel formed; logic zero -> conduct; w/ circ)

The pMOS transistor is formed by a pnp combination of doped semi-conductors. Generally the same as the above case. **Note:** logic zero is different from not applying voltage!!! Only zero voltage applied will cause the positive charges to gather at the top and allow a p-type channel hence allowing conducting electricity.

Circuits

Minterms & Maxterms

[This](#) quora answer explained these two concepts well and I hereby acknowledge that I quoted this answer in my explanation.

Standard Truth Table (Format)

A standard truth table is a truth table such that the first column of input has first half equal to zero and second half equal to one; the second column is divided into two parts the upper half and the lower one and in each half: the top half is zero and lower half is filled with one; and on and on and so on... The verbal description may be hard to understand, we shall see this through an example. Say we have three inputs A, B and C with one output Y.

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Minterms

Minterms are rows in a standard truth table that has a one as output. They are labeled with index, starting from the first row as index zero. A minterm is a product (AND) of all variables in the function, in direct or complemented form. (Intuition behind naming:) A minterm has the property that it is equal to 1 on exactly one row of the truth table.

Formal: an AND expression *with every input present* in true or complemented form. So for example given inputs A, B, C and D, valid minterms could look like $A \cdot \bar{B} \cdot C \cdot D$ or $\bar{A} \cdot B \cdot \bar{C} \cdot D$ et cetera, et cetera.

Maxterms

Maxterms are rows in a standard truth table that has zero as output. A maxterm is a sum (OR) of all the variables in the function, in direct or complemented form. A maxterm has the property that it is equal to 0 on exactly one row of the truth table.

Formal: an OR expression *with every input present* in true or a complemented form.

Memorizing Min&Max 0's and 1's

```
min 0\bar{1}
```

In case of min term, if it is with bar, then it is one. W/o bar means zero. Min has first syllable **m** so 0 and 1中的min'没'有bar

```
max \bar{0}1
```

In case of max term, if it is with bar, then it is zero. W/o bar means one. Max has first syllable **M** so 0 and 1中的Max'没'有bar

Sum-of-Minterms & Product-of-Maxterms

Sum of Minterm is a way of expressing which inputs cause the output to go high under the assumption that the index is the index of the row in a standard truth table. This method gives us a more compact way to display the entire truth table. Sum of Minterms are useful in cases with very few input combinations that produce high output. While Product of Maxterms is useful when expressing truth tables that have very few low output cases.

An interesting property is that $m_x = \bar{M}_x$, Minterm x is the complement of Maxterm x. For example: $m_0 = \bar{A} \cdot \bar{B} \implies M_0 = A + B$. We can also convert min/max term back and forth. For example a 2-input XOR gate has SOM form $F = m_1 + m_3$ and this is equivalent to the POM form $F = M_0 \cdot M_3 (= (A + B)(\bar{A} + \bar{B}))$ which indeed the XOR gate.

Reducing Circuits

Boolean Algebra**Axioms**

1. $0 \cdot 0 = 0$
2. $0 \cdot 1 = 1 \cdot 0 = 0$
3. $1 \cdot 1 = 1$
4. $x = 1 \implies \bar{x} = 0$

Consequence of Axioms

1. $x \cdot 0 = 0$
2. $x \cdot 1 = x$
3. $x \cdot x = x$
4. $x \cdot \bar{x} = 0$
5. $\bar{\bar{x}} = x$
6. $x + 1 = 1$
7. $x + 0 = x$
8. $x + x = x$
9. $x + \bar{x} = 1$
10. $x + \bar{x} = 1$

Commutative Law

$$x \cdot y = y \cdot x \text{ and } x + y = y + x$$

Associative Law

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \text{ and } x + (y + z) = (x + y) + z$$

Distributive Law

$$x \cdot (y + z) = x \cdot y + x \cdot z \text{ and } x + (y \cdot z) = (x + y) \cdot (x + z)$$

Consensus Law ?

$$x \cdot y + \bar{x} \cdot z + y \cdot z = x \cdot y + \bar{x} \cdot z$$

Absorption Law

$$x \cdot (x + y) = x \text{ and } x + (x \cdot y) = x$$

DeMorgan's Law

$$\bar{x} \cdot \bar{y} = \overline{x + y} \text{ and } \bar{x} + \bar{y} = \overline{x \cdot y}$$

Karnaugh (K-)Maps

K-maps are 2 dimensional grid of minterms, where adjacent minterm locations in the grid differ by a single literal and the values in each grid is the output for that *combined* minterm.

Using K-Map

Once a K-map is created, draw boxes over groups of high output values subject to the following constraints:

- Boxes must be rectangular and aligned with the map.
- Number of values contained within each box be a power of 2.
- Boxes **may** overlap with each other.
- Boxes may wrap across edges of the map. **Reading the result:** We use a sum of minterms to get the overall output equation.

Logic Devices

Multiplexers

Adders

There are two types of adders, the half adder and the full adder respectively. We will take a look at them respectively below.

Half Adders

Half adders are cheaper to implement. They have two bits for the two input bits that are to be added and two bits for output, one for carry out and one for the added bit output. (adds two bits to produce a two bit sum). One might find that the logic circuit is overwhelming in terms of memorizing, but it falls apart if we look at the logic expression: $C = X \cdot Y$ and $S = X \oplus Y = X \cdot \bar{Y} + \bar{X} \cdot Y$ where we remember that C represents the carry bit and S is the main output bit.

Full Adders

Full adders are slightly more complicated than half adders since they have an additional bit of input that takes care of the carry input. Same as above, we can use the logic expression to remember the logic circuit level design, which is as follows $C = X \cdot Y + (X \oplus Y) \cdot Z$ and $S = X \oplus Y \oplus Z$ where we remember that S , C represents the output and carry respectively.

Subtractors

Subtractors are an extension of adders. Basically we perform addition on a negative number. Before we do subtraction, we need to understand how negative numbers work in binary settings.

1. Unsigned = a separate bit exists for the sign and the data bits store the positive version of the number

2. Signed = all its are used to store a 2's complement negative number

Two's Complement

Two's complement of a number produces the 'negative version' of that number. We can get the 2's complement of a number using two steps. The first step is to get the 1's complement where we bitwisely negate the original binary number. And the second step is to simply add one to the 1's complement result and this procedure yields us the 2's complement. The signed subtraction can be performed by using the binary adder circuit with negative numbers.

Rules about Signed Binary Numbers

- The largest positive binary number is zero followed by all ones
- The binary value for -1 has one in all the digits (000001 → 111110 → 111111)
- The most negative binary number is one followed by all zeros.
- There are 2^n possible values that can be stored in an n -digit binary number, in which 2^{n-1} are negative, $2^{n-1} - 1$ are positive and we have zero.

Subtraction Circuits

Since we perform subtraction in the form of adding a negative (2's complement) number, we can create a subtraction circuit based on this. To do so, we will rely on the same set up that we had in the case of ripple adder. We will add a negation gate to each of the bits from the number to be subtracted (1's complement) and input a 1 at the carry bit for the first full adder (2's complement).

Addition/Subtraction Circuit

This is rather hard to explain in plain english, so I borrowed Prof. Steve Engels' [slide](#)