Digital circuit design

Vikas Dhiman for ECE275

May 9, 2024

Contents

Chapter 1

Course improvements: TODO

ToDo

Chapter 2

Number System

2.1 Place value number system

- What is a place value number system?
- What are some examples?
- What are some non-examples?
- What is a radix (or base)?
- How to convert between different radix in place value system?
- What are some commonly used number systems in computer engineering?

2.1.1 Decimal number system

2.1.2 Binary numbers

2.1.3 Conversion between different radix

Problem 1. *Convert the following binary numbers to decimal:* $(11110)_2$, $(100111)_2$.

Conversion from decimal to binary The value is in decimal because we find it easy to do calculations in decimal numbers. Decimal values can be converted back to Binary representation by *repeated division* by 2 while noting down the remainder. Allow me to use / sign to denote both quotient and remainder after division. Let's convert $(22)_{10}$ back to binary:

$22/2 = (11, 0)$	11 is the quotient and 0 is the remainder
$11/2 = (5, 1)$	5 is the quotient and 1 is the remainder
$5/2 = (2, 1)$	
$2/2 = (1, 0)$	
$1/2 = (0, 1)$	

Read the remainders from bottom to top and right them as left to right, to form the resultant binary number $(22)_{10} = (10110)_2$.

Problem 2. *Find the binary representation for decimal numbers:* 123 *and* 89*. Show your work.*

2.2 Hexadecimal numbers

Numbers with base 16 are called Hexadecimal numbers. From 0 to 9 the symbols are same as decimal numbers. From 10 to 15, Hexadecimal numbers use A to F.

$$A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$$

. Example, $(10AD)_{16} = 1 \times 16^3 + 10 \times 16^1 + 13 = 4096 + 160 + 13 = 4269$.

2.3 Octal numbers

Numbers with base 8 are called octal numbers. Example, $(354)_8 = 3 \times 8^2 + 5 \times 8 + 4 = 192 + 40 + 4 = 236$.

2.4 Hexadecimal/octal to binary and vice-versa

Normally, if you have to convert between a number of base r_1 to a number of base r_2, we will have to convert it via decimal numbers. Convert from base r_1 to decimal and then from decimal to r_2.

Since Hexadecimal base 16 is an exact power of 2 ($16 = 2^4$). Conversion between Hexadecimal to binary is easy. You can group 4 binary digits from right to left and convert each group of 4 binary digits to a single Hexadecimal digit and back. Example, $(10110)_2 = (0001\_0110)_2 = (16)_{16}$. To convert back. Take example, $(10AD)_{16} = (0001\_0000\_1010\_1101)_2 = (1\_0000\_1010\_1101)_2$.

Problem 3. *Find the binary and decimal values of the following Hexadecimal numbers* $(A25F)_{16}$*,* $(F0F0)_{16}$*.*

Similarly octal to binary can proceed by grouping 3-binary digits at a time. Example, $(354)_8 = (011\_101\_100)_2$.

Problem 4. *Find the binary and decimal values of the following Octal numbers* $(3751)_8$ *and* $(722)_8$*.*

2.5 Signed binary numbers

Signed numbers include both negative and positive numbers. There three common signed number representations

1. Sign magnitude representation
2. One's complement
3. Two's complement

2.5.1 Sign-magnitude representation

The Most significant (left most) *bit* (binary digit) represents sign (0 = + and 1 = −), the rest represent the magnitude. Example, a 5-bit number $(11010)_2$ in signed magnitude representation has the value of $(-1010)_2 = -10$. Note that +10 has to be represented by a leading 0 at the most significant bit (MSB) $+10 = (01010)_2$. Hence, the number of bits have to be specified.

Problem 5.
- *Write down all possible 4-digit binary numbers and corresponding decimal values if they are in signed magnitude format? What is the minimum and maximum value?*
- *What is the minimum and maximum value of n-digit signed binary number in sign-magnitude format?*

2.5.2 One's complement negation

You can convert a positive number (say +10) to negative number by applying a negative sign in front of it (-(+10) = -10). It is more evident from taking negative of a negative number (-(-10) = +10). In case of sign-magnitude representation, the "negative operator" flips the sign bit. The next two signed number representations (1's complement and 2's complement) are designed around specific negative operator definitions.

Negate $13_{10} = 01101_2$ using 5-bit one's complement.

Negate -13_{10} using 5-bit one's complement.

2.5.3 One's complement binary numbers

In one's complement representation, the negative operation is obtained by flipping all the bits of the binary number. Example, a 5-bit one's complement of $+10 = (01010)_2$ is $(10101)_2 = -10$. Note that flipping bits is equivalent to subtracting the number from $(11111)_2$, hence the name. You can also confirm that double negative operator yields back the same number.

Problem 6.

- *Write down all possible 4-digit binary numbers and corresponding decimal values if they are in sign magnitude format? What is the minimum and maximum value?*

- *What is the minimum and maximum value of n-digit signed binary number in one's complement?*

Problem 7. *Determine the decimal values of the following 1's complement 6-digit binary numbers :*

1. *01101110*

2. *10101101*

Problem 8. *Convert the decimal numbers -17 and +23 into the 6-digit one's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.*

2.5.4 Two's complement negation

In two's complement representation, the n-digit negative number is obtained by subtracting the positive number from 2^n. Example, two's complement of 5-digit binary number $+10 = (01010)_2$ is $2^5 - 10 = 22 = (11000)_2$. An easier algorithm to get two's complement goes via one's complement. Note that $(11111)_2 = 2^5 - 1$. We can get two's complement by adding 1 to one's complement. To get two's complement:

1. Flip all the bits. (Same as taking one's complement).
2. Add 1 to the number.

Negate $13_{10} = 01101_2$ using 5-bit two's complement.

Negate -13_{10} using 5-bit two's complement.

How to convert one's complement number representation into sign-magnitude numbers?

1. Check if the number is positive or negative. Even for one's complement representation, or two's complement representation, if the MSB (Most-significant bit) is 1, then the number is negative, otherwise positive.

2. If positive: For positive numbers, two's complement, one's complement and sign magnitude are the same. No conversion between different representation is needed. 2.b If negative: For negative numbers. Flip the bits of 1's complement. Once you flip the 1's complement bits of a negative number, you get the corresponding positive number.

3. We still want to represent the original negative number. So we set the MSB of sign-magnitude representation to 1. Since the range (min and max) for both n-bit 1's complement and sign-magnitude are the same (between $-(2^{n-1}-1)$ and $2^{n-1}-1$), you can always represent 8-bit 1's complement numbers with needing to extend the 8-bit number to 9-bits.

Example: Convert 8-bit one's complement 10101010 to 8-bit sign-magnitude Let number n = 10101010

1. Is the number +ve or -ve: It is negative because it starts with 1.

2. The number is not positive.

3. Take the 1's complement of the negative number to get the positive part. i.e. Flip the bits: -n = 01010101 or n = -(01010101)

4. We got the positive part of the number, but we want to represent the original negative number, so we set the MSB bit one. Hence, the equivalent sign-magnitude representation is: n = 11010101

2.5.5 Two's complement representation

Problem 9. *Determine the decimal values of the following 2's complement 6-digit numbers :*

1. *01011110*

2. *10010111*

Problem 10. *Convert the decimal numbers -17 and +23 into the 6-digit two's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.*

Problem 11. *Convert the decimal numbers 73, 23, -17, and -163 into signed 8-bit numbers in the following representations:*

1. *Sign and magnitude*
2. *1's complement*
3. *2's complement*

2.5.6 Arithmetic overflow

Problem 12. *Consider addition of 4-digit two's complement binary numbers*

1. $1010_2 + 1101_2$
2. $1011_2 + 1100_2$

In which of the two case overflow happens? Can you come up with a rule to "easily" detect overflow?

Rules for detecting arithmetic overflow:

1. Adding numbers of different signs never produces an overflow.

2. Adding numbers of the same sign may produce an overflow

   (a) Wrong approach: Adding two negative 2's complement numbers always produces an additional carry-over 1, but that in itself isn't an overflow. An example, the range of 4-bit 2's complement numbers is between -8 to +7. Adding -3 to -4 in 2's complement is 1101 + 1100 produces an additional carry over 1. You can ignore the additional carry-over 1 to get the correct answer 1001 = -7 which is within range -8 to 7.

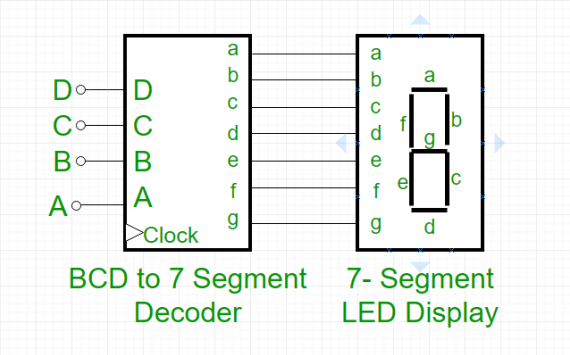
   (b) Approach 1: The easiest way for now to detect overflow is if adding two -ve numbers results in a +ve number, or adding +ve numbers results in a -ve number.

   (c) Approach 2: You can also do a range test in decimal based range test. The range of n-bit 2's complement numbers is between -2^{n-1} and $2^{n-1}-1$. For 5-bit 2's complement numbers, it is between -16 and 15. For 6-bit 2's complement numbers, it is between -32 and 31.

   (d) Approach 3: You can also check the carry-overs of the most significant two bits. If they match, i.e. 0 and 0, or 1 and 1, then there is no overflow. If they do not match, i.e. 0 and 1 or 1 and 0, then there is an overflow.

2.6 Binary coded decimal

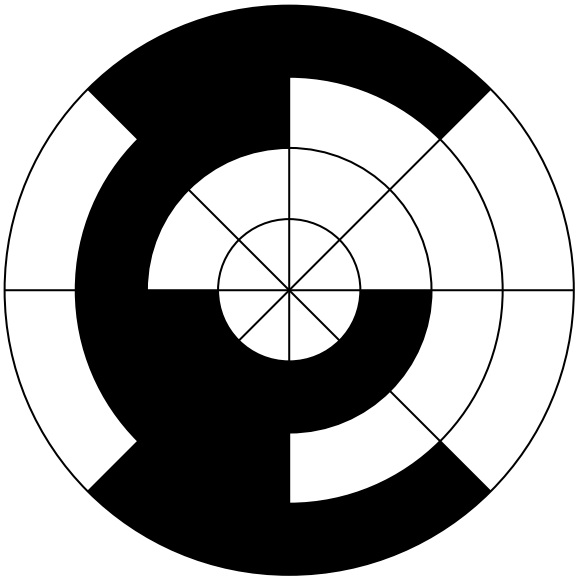
In Binary coded decimal (BCD), each decimal digit is represented by 4 bits. For example, $1047 = (0001\_0000\_0100\_0111)_{\text{BCD}}$. It is useful in input-output applications where the number has to be either displayed as decimal or received as decimal.



Problem 13. *Convert 11, 23, 35, 57 and 103897 to BCD?*

2.7 Gray code

A sequence of binary numbers where only one bit changes when the number increases by 1. It is helpful in applications like wheel encoders



Problem 14. *Write all possible 3-bit binary numbers in gray-code*

Chapter 3

Boolean Algebra

3.1 Learning objectives

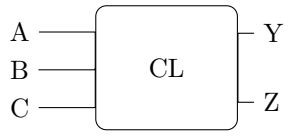
1. Introduce truth tables as Behavioral verilog right away

1. Representing digital circuits
2. Converting between different notations: Boolean expression, logic networks and switching circuits
3. Converting between different logic network specifications: truth table, minterm, maxterms, product of sums canonical form and sum of product canonical form.

3.2 Basic Gates and notations summary

Name	C/Verilog	Boolean expr.	Truth Table	Switching circuit	(ANSI) symbol	Venn diagram
AND Gate	`L = x1 & x2`	$L = x_1 \cdot x_2 = x_1 x_2$	x_1 x_2 | $x_1 \cdot x_2$ 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1			
OR Gate	`L = x1 | x2`	$L = x_1 + x_2$	x_1 x_2 | $x_1 + x_2$ 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1			
NOT Gate	`L = ~ x1`	$L = \bar{x}_1 = x_1'$	x_1 | $\bar{x}_1$ 0 | 1 1 | 0			

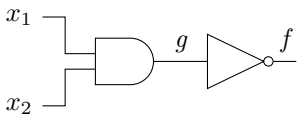
3.3 Digital circuits or networks



$$Y = F(A, B, C) \qquad Z = G(A, B, C)$$

3.4 Two input networks

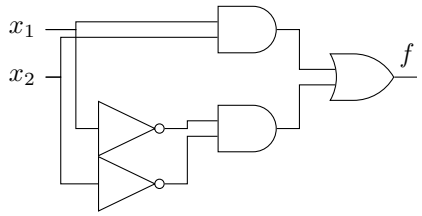
Example 1. *Convert the following (ANSI) network into a Boolean expression, a truth table and a Venn diagram.*



Example 2. *Convert the following Boolean expression into a (ANSI) network, a truth table and a Venn diagram:*

$$f = \overline{x_1 + x_2}$$

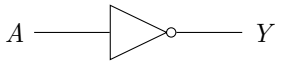
Problem 15. *Convert the following (ANSI) network into a Boolean expression, a truth table and a Venn diagram.*

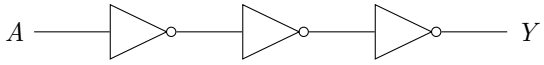


Example 3. *Convert the following Boolean expression into a network, a truth table and a Venn diagram:*

$$f = x_1\bar{x}_2 + \bar{x}_1x_2$$

Problem 16. *Can two different circuits have the same truth table? Can two different truth tables have the same circuit? Consider the following two circuits for example*



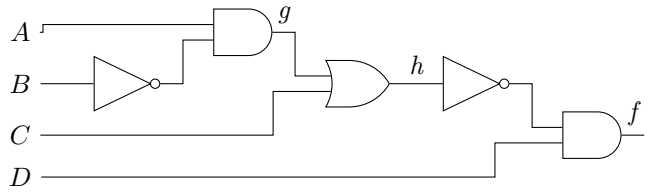


How about Venn digrams?

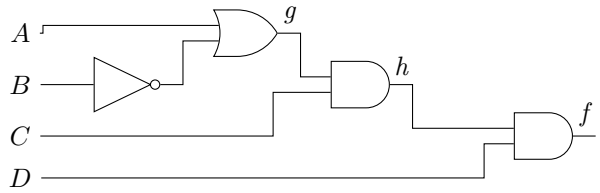
Remark 1. *Truth tables and Venn diagrams define* what *the combinational circuit should do. Truth tables define output for every input. Boolean expression and networks define* how *to achieve the desired input output relationship.*

3.5 Multi-input networks

Example 4. *Convert the following (ANSI) network into a Boolean expression and a truth table.*



Problem 17. *Convert the following (ANSI) network into a Boolean expression and a truth table.*



3.6 Minterms and Maxterms

3.6.1 Minterms

Minterm is a product involving all inputs (or complements) to a function. Every row of a truth table has a corresponding minterm. Minterm is true if and only if the corresponding row in the table is active.

Minterms defined as follows for each row of a two input truth table:

A	B	minterm	minterm name
0	0	$\bar{A}\bar{B}$	m_0
0	1	$\bar{A}B$	m_1
1	0	$A\bar{B}$	m_2
1	1	AB	m_3

Consider a two input circuit whose output Y is given by the truth table:

A	B	Y	minterm	minterm name
0	0	0	$\bar{A}\bar{B}$	m_0
0	1	1	$\bar{A}B$	m_1
1	0	0	$A\bar{B}$	m_2
1	1	1	AB	m_3

then $Y = \bar{A}B + AB = m_1 + m_3 = \sum(1, 3)$.
This also gives the *sum of products canonical form.*

Example 5. *What is the minterm m_{13} for a 4-input circuit with inputs x, y, z, w (ordered from MSB to LSB).*

Problem 18. *What is the minterm m_{23} for a 5-input circuit with inputs a, b, c, d, e (ordered from MSB to LSB).*

Example 6. *Convert the following 4-input truth table into sum of minterms and sum of products canonical form.*

minterm name	*A*	*B*	*C*	*D*	*f*
m_0	*0*	*0*	*0*	*0*	*0*
m_1	*0*	*0*	*0*	*1*	*1*
m_2	*0*	*0*	*1*	*0*	*0*
m_3	*0*	*0*	*1*	*1*	*0*
m_4	*0*	*1*	*0*	*0*	*0*
m_5	*0*	*1*	*0*	*1*	*1*
m_6	*0*	*1*	*1*	*0*	*0*
m_7	*0*	*1*	*1*	*1*	*0*
m_8	*1*	*0*	*0*	*0*	*0*
m_9	*1*	*0*	*0*	*1*	*0*
m_{10}	*1*	*0*	*1*	*0*	*0*
m_{11}	*1*	*0*	*1*	*1*	*0*
m_{12}	*1*	*1*	*0*	*0*	*0*
m_{13}	*1*	*1*	*0*	*1*	*1*
m_{14}	*1*	*1*	*1*	*0*	*0*
m_{15}	*1*	*1*	*1*	*1*	*0*

Problem 19. *Convert the following 4-input truth table into sum of minterms and sum of products canonical form.*

minterm name	*A*	*B*	*C*	*D*	*f*
m_0	*0*	*0*	*0*	*0*	*0*
m_1	*0*	*0*	*0*	*1*	*0*
m_2	*0*	*0*	*1*	*0*	*0*
m_3	*0*	*0*	*1*	*1*	*1*
m_4	*0*	*1*	*0*	*0*	*0*
m_5	*0*	*1*	*0*	*1*	*0*
m_6	*0*	*1*	*1*	*0*	*0*
m_7	*0*	*1*	*1*	*1*	*1*
m_8	*1*	*0*	*0*	*0*	*0*
m_9	*1*	*0*	*0*	*1*	*0*
m_{10}	*1*	*0*	*1*	*0*	*0*
m_{11}	*1*	*0*	*1*	*1*	*1*
m_{12}	*1*	*1*	*0*	*0*	*0*
m_{13}	*1*	*1*	*0*	*1*	*1*
m_{14}	*1*	*1*	*1*	*0*	*1*
m_{15}	*1*	*1*	*1*	*1*	*0*

3.6.2 Maxterms

Maxterm is a sum involving all inputs (or complements) to a function. Every row of a truth table has a corresponding maxterm. Minterm is false if and only if the corresponding row in the table is active.

Maxterms are defined as follows for each row of a two input truth table:

A	B	maxterm	maxterm name
0	0	$A + B$	M_0
0	1	$A + \bar{B}$	M_1
1	0	$\bar{A} + B$	M_2
1	1	$\bar{A} + \bar{B}$	M_3

Consider a two input circuit whose output Y is given by the truth table:

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \bar{B}$	M_1
1	0	0	$\bar{A} + B$	M_2
1	1	1	$\bar{A} + \bar{B}$	M_3

then $Y = (A + B)(\bar{A} + B) = M_0 M_2$.

Writing a functional specification in terms of minterms is also called product of sums canonical form.

Example 7. *Write the maxterm M_{11} for 4-input Boolean function with the ordered inputs A, B, C, D.*

Example 8. *Convert the following 4-input truth table into product of maxterms and product of sums canonical form.*

maxterm name	*A*	*B*	*C*	*D*	*f*
M_0	*0*	*0*	*0*	*0*	*0*
M_1	*0*	*0*	*0*	*1*	*0*
M_2	*0*	*0*	*1*	*0*	*0*
M_3	*0*	*0*	*1*	*1*	*1*
M_4	*0*	*1*	*0*	*0*	*0*
M_5	*0*	*1*	*0*	*1*	*0*
M_6	*0*	*1*	*1*	*0*	*0*
M_7	*0*	*1*	*1*	*1*	*1*
M_8	*1*	*0*	*0*	*0*	*0*
M_9	*1*	*0*	*0*	*1*	*0*
M_{10}	*1*	*0*	*1*	*0*	*0*
M_{11}	*1*	*0*	*1*	*1*	*1*
M_{12}	*1*	*1*	*0*	*0*	*0*
M_{13}	*1*	*1*	*0*	*1*	*1*
M_{14}	*1*	*1*	*1*	*0*	*1*
M_{15}	*1*	*1*	*1*	*1*	*0*

Problem 20. *Convert the following 4-input truth table into product of maxterms and products of sums canonical form.*

maxterm name	*A*	*B*	*C*	*D*	*f*
M_0	*0*	*0*	*0*	*0*	*0*
M_1	*0*	*0*	*0*	*1*	*1*
M_2	*0*	*0*	*1*	*0*	*1*
M_3	*0*	*0*	*1*	*1*	*1*
M_4	*0*	*1*	*0*	*0*	*1*
M_5	*0*	*1*	*0*	*1*	*0*
M_6	*0*	*1*	*1*	*0*	*1*
M_7	*0*	*1*	*1*	*1*	*1*
M_8	*1*	*0*	*0*	*0*	*0*
M_9	*1*	*0*	*0*	*1*	*1*
M_{10}	*1*	*0*	*1*	*0*	*1*
M_{11}	*1*	*0*	*1*	*1*	*1*
M_{12}	*1*	*1*	*0*	*0*	*0*
M_{13}	*1*	*1*	*0*	*1*	*1*
M_{14}	*1*	*1*	*1*	*0*	*1*
M_{15}	*1*	*1*	*1*	*1*	*0*

Example 9. *Write the 3-input truth table for the function* $f = m_2 + m_3 + m_7$.

Problem 21. *Write the 3-input truth table for the function* $f = M_4 M_5 M_7$.

Problem 22. *Write the truth table for the function* $f = \bar{A}B\bar{C} + AB\bar{C}$.

3.7 Karnaugh maps

3.7.1 Two input K-maps

B \ A	0	1
0	m_0	m_2
1	m_1	m_3

3.7.2 Three input K-maps

C \ AB	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

3.7.3 Four input K-maps

CD \ AB	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

3.7.4 Five input K-maps

A = 0

DE \ BC	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

A = 1

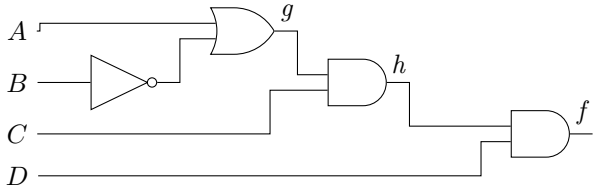
DE \ BC	00	01	11	10
00	m_{16}	m_{20}	m_{28}	m_{24}
01	m_{17}	m_{21}	m_{29}	m_{25}
11	m_{19}	m_{23}	m_{31}	m_{27}
10	m_{18}	m_{22}	m_{30}	m_{26}

3.8 More Gates and notations summary

Name	C/Verilog	Boolean expr.	Truth Table	(ANSI) symbol	K-map
NAND Gate	`Q = ~(x1 & x2)`	$Q = \overline{x_1 \cdot x_2} = \overline{x_1}\,\overline{x_2}$	x_1 x_2 | $\overline{x_1 \cdot x_2}$ 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0		
NOR Gate	`Q = ~(x1 | x2)`	$Q = \overline{x_1 + x_2}$	x_1 x_2 | $\overline{x_1 + x_2}$ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0		
XOR Gate	`Q = x1 ^ x2`	$Q = x_1 \oplus x_2$	x_1 x_2 | $x_1 \oplus x_2$ 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0		
XNOR Gate	`Q = ~(x1 ^ x2)`	$Q = \overline{x_1 \oplus x_2}$	x_1 x_2 | $\overline{x_1 \oplus x_2}$ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1		

Example 10. *Convert the following Boolean expression into a K-map.* $f = \overline{A\bar{B} + C}D$

Problem 23. *Convert the following logic circuit into a K-map.*



3.9 Boolean Algebra

3.9.1 Axioms of Boolean algebra

1. $0 \cdot 0 = 0$
2. $1 + 1 = 1$

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

3.9.2 Single variable theorems (Prove by drawing K-maps)

1. $x \cdot 0 = 0$

2. $x + 1 = 1$

3. $x \cdot 1 = x$

4. $x + 0 = x$

5. $x \cdot x = x$

6. $x + x = x$

7. $x \cdot \bar{x} = 0$

8. $x + \bar{x} = 1$

9. $\bar{\bar{x}} = x$

Remark 2 (Duality)**.** *Swap* $+$ *with* $\cdot$ *and 0 with 1 to get another theorem*

3.9.3 Two and three variable properties (Prove by K-maps)

1. Commutative: $x \cdot y = y \cdot x$, $x + y = y + x$

2. Associative: $x \cdot (y \cdot z) = (x \cdot y) \cdot z$, $x + (y + z) = (x + y) + z$

3. Distributive: $x \cdot (y + z) = x \cdot y + x \cdot z$, $x + y \cdot z = (x + y) \cdot (y + z)$

4. Absorption: $x + x \cdot y = x$, $x \cdot (x + y) = x$

5. Combining: $x \cdot y + x \cdot \bar{y}$, $(x + y) \cdot (x + \bar{y}) = x$

6. DeMorgan's theorem: $\overline{x \cdot y} = \bar{x} + \bar{y}$, $\overline{x + y} = \bar{x} \cdot \bar{y}$.

7. Concensus:

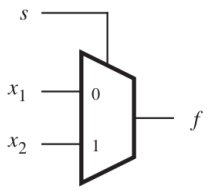
   (a) $x + \bar{x} \cdot y = x + y$

   (b) $x \cdot (\bar{x} + y) = x \cdot y$

   (c) $x \cdot y + y \cdot z + \bar{x} \cdot z = x \cdot y + \bar{x} \cdot z$

   (d) $(x + y) \cdot (y + z) \cdot (\bar{x} + z) = (x + y) \cdot (\bar{x} + z)$

Example 11 (Multiplexer)**.** *Multiplexer is a circuit used to select one of the input lines x_1 and x_2 based only select input s. When $s = 0$, x_1 is selected, x_2 is selected otherwise. Find a boolean expression and a circuit for multiplexer*



s	$f(s, x_1, x_2)$
0	x_1
1	x_2

Example 12. *Simplify $f = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}\bar{C}$ using boolean algebra.*

Example 13. *Simplify $f = \bar{A}\bar{A}\bar{C} + \bar{A}\bar{B}C$ using K-maps.*

3.10 Logic minimization

3.11 Logic minimization

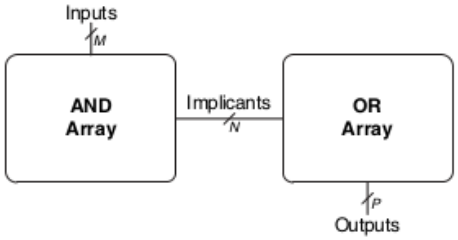
A general optimization criteria for multi-level logic are to Minimize some combination of:

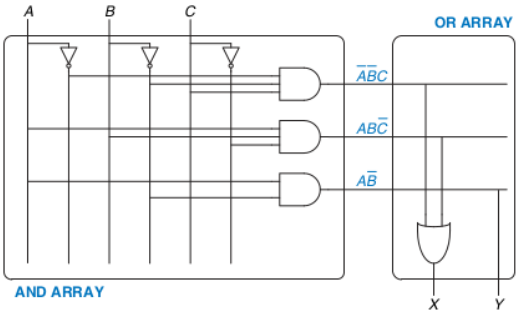
1. Area occupied by the logic gates and interconnect;
2. the Critical Path Delay of the longest path through the logic;
3. the Degree of Testability of the circuit, measured in terms of the percentage of faults covered by a specified set of test vectors, for an appropriate fault model (Eg., single stuck faults, multiple stuck faults, etc.);

4. Power consumed by the logic gates.

In this course, we will start with two-level multi-input circuits and a criteria based on the number of gates/transistors/diodes.

3.12 Programmable Logic Arrays





3.13 Two-level circuits

The cost that we are going to consider in this class depend upon:

1. Number of gates.
2. Number of input to the gates.

More gates need more transistors, more area on the chip. More-inputs the gate need more transistors within each gate. Number of gate inputs can be considered secondary criterion to the number of gates.

Example 14. *Find the cost of the following Boolean expression* $X = \bar{A}\bar{B}C + AB\bar{C} + A\bar{B}$.

Problem 24. *Find the cost of the following Boolean expression* $X = A\bar{B}C + \bar{A}B\bar{C} + \bar{B}C$.

3.14 Terminology for K-maps

Running Example: $f = \sum m(0,1,2,3,7) = \bar{x}_1 + x_1x_2x_3$.

Literal A single variable or its complement. Example: $\bar{x}, x_1, x_2, x_3$

Implicant A product term which is true for a function. All minterms are implicants. Example: $x_1x_2x_3$, $\bar{x}_1$, $m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$, $\bar{x}_1x_3$, $\bar{x}_1\bar{x}_3$.

Prime Implicant An implicant that cannot be combined into fewer literals. Example: $\bar{x}_1, x_2x_3$.

Essential Prime Implicant An implicant that cannot be combined into fewer literals. Example: x_2x_3.

Cover : List of Prime Implicants that account for all $f = 1$.

Cost : Number of gates (excluding not gate on literals) and number of inputs to each gate.

Example 15. *Find minimum cost expression for the function* $f(x_1, x_2, x_3) = \prod M(4,5,6)$

Problem 25. *Find minimum cost expression for the function* $f(x_1, x_2, x_3) = \prod M(2, 5, 6)$

3.14.1 Incompletely specified functions or Don't cares

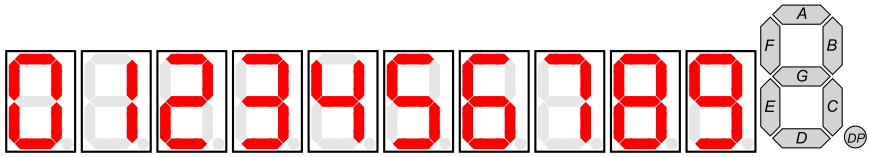


Figure 3.1: 7 Segment Representations of Each Integer

BCD Value				LED Segment
D_3	D_2	D_1	D_0	E
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	d
1	0	1	1	d
1	1	0	0	d
1	1	0	1	d
1	1	1	0	d
1	1	1	1	d

Example 16. *Find minimum cost expression for the function*

$$f(x_1, \ldots, x_4) = \sum m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$$

Problem 26. *Find minimum cost expression for the function*

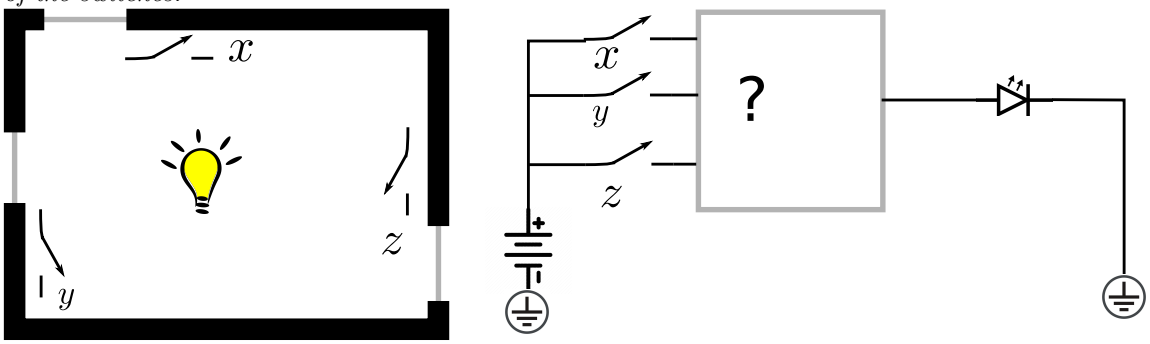
$$f(x_1, \ldots, x_4) = \sum m(0, 2, 4, 6, 7, 8, 9, 13) + D(1, 12, 15)$$

3.15 A few more Boolean problems

Example 17. *Simplify the following Boolean expression:*

$$f = x_1\bar{x}_3\bar{x}_4 + x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3$$

Example 18. *Assume that a large room has three doors and that a switch near each door controls a light in the room. It has to be possible to turn the light on or off by changing the state of any one of the switches.*



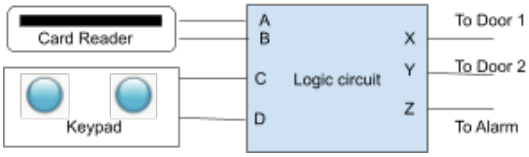
Problem 27. *A simple security system for two doors consists of a card reader and a keypad.*

A person may open a particular door if he or she has a card containing the corresponding code and enters an authorized keypad code for that card. Note that card-code and keypad-code are different. The outputs from the card reader are given in the table below.

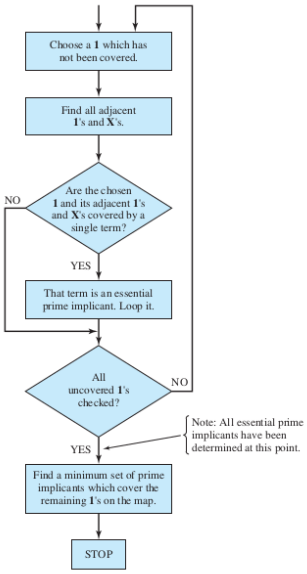
To unlock a door, a person must hold down the proper keys on the keypad and, then, insert the card in the reader. The authorized keypad code for door 1 is 10, and the authorized keypad code for door 2 is 11. If the card has an invalid code or if the wrong keypad code is entered, the alarm will ring when the card is inserted. If the correct keypad code is entered, the corresponding door will be unlocked when the card is inserted.

Design the logic circuit for this simple security system. Your circuit's inputs will consist of a card code AB, and a keypad code CD. The circuit will have three outputs XYZ (if X is 1, door 1 will be opened; if Y is 1, door 2 will be opened; if Z 1, the alarm will sound).

Find the minimal cost two-level circuit using K-maps for X, Y, Z. Provide the minimal cost. (It can be either of SOP/POS forms)



	A	B
No card inserted	0	0
Valid card-code for door 1	0	1
Valid card-code for door 2	1	1
Invalid card code	1	0



Example 19.

CD \ AB	00	01	11	10
00	d	1		1
01		1	1	1
11		1	d	
10		1		1

Problem 28. *Find the minimum SOP (sum of products) and POS (product of sum) expression for the function* $f(a,b,c,d) = \prod M(5,7,13,14,15) \cdot \prod D(1,2,3,9)$

Chapter 4

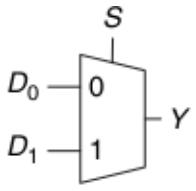
Adders, Muxes and Decoders

4.1 Objectives

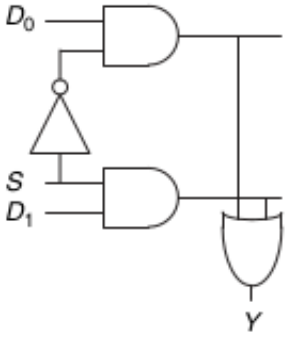
1. Design combinational circuits using multiplexers and decoders

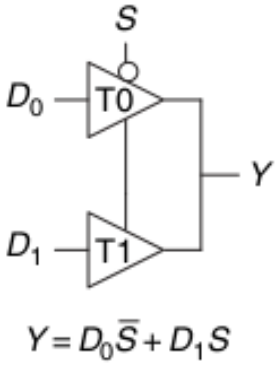
4.2 Design combinational circuit using multiplexers [1, Section 2.8.1]

4.2.1 Review: 2to1 Multiplexer (MUX)



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





4.2.2 Wider multiplexers

Draw the symbol for a 4:1 MUX, an 8:1 MUX and a $2^N : 1$ MUX and write corresponding Boolean expressions.

Example 20. *Design a circuit for* $Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$ *using a 8:1 MUX.*

Remark 3. *A* $2^N : 1$ *MUX can be used to program any N-input logic function.*

Example 21. *Design a circuit for* $Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$ *using a 4:1 MUX and NOT gates only.*

Remark 4. *A* $2^{N-1} : 1$ *MUX can be used to program any N-input logic function, if we use literals on the input side.*

Example 22. *Design a circuit for* $Y = \bar{A}C + \bar{A}B + B\bar{D}$ *using a 8:1 MUX and NOT gates only. Also design using 4:1 MUX and other gates. fewest gates.*

4.3 Encoders and Decoders

Example 23. *Draw the symbol and the truth table for 2:4 decoder. Also write the logic expressions.*

Example 24. *Draw the symbol and the truth table for 3:8 decoder, 4:16 decoder and* $N : 2^N$ *decoder. Also write the logic expressions.*

Example 25. *Design a circuit for a XOR gate using a 2:4 decoder and an OR gate.*

Example 26. *Design a circuit for* $Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$ *using a 3:8 decoder and an OR gate.*

4.3.1 (Priority) Encoders

Example 27. *Draw symbol and truth table for a 4:2 priority encoder.*

Example 28. *Draw symbol and truth table for a 8:3 priority encoder.*

Chapter 5

Sequential Logic

4.Provide Verilog idioms for each of the sequential circuits

5.The build up of SR latch is interesting from here. https://www.youtube.com/watch?v=BYN8Zmk6HJY

6.JK latch and T ff are not that interesting

7.Metastability must be introduced

5.1 Objectives

1. Understand timing diagrams, gate delays and critical path
2. Design Hazard-free two level circuits
3. Building blocks of sequential circuits
4. Analyze a sequential circuit and derive a state-table and a state-graph
5. Derive a state graph or state table from a word description of the problem
6. Understanding the structure of an FPGA

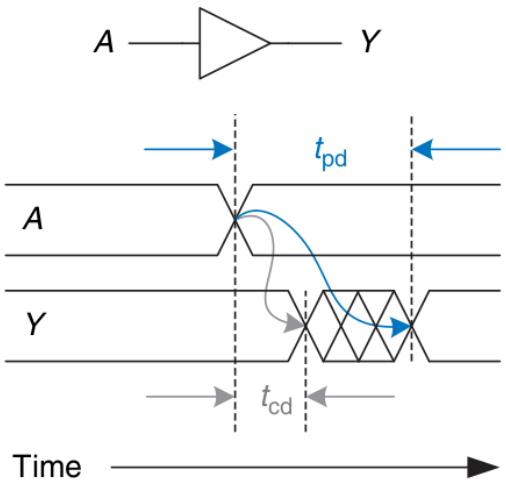
5.2 Why do we need sequential circuits?

Example 29. *Think about this problem: Design an occupancy counter that depends on a sensor S at the class door. The sensor is triggered every time a person passes through the door. The counter can be reset to zero with a reset button. Assume we only need up to two bit counter C_1C_0. Draw a truth table for this circuit. Do you have requisite knowledge for designing this circuit? Can this circuit be designed without a memory element?*

5.3 Timing diagrams and propagation delays

Example 30 (Timing diagram)**.** *Draw a timing diagram for an ideal NAND gate.*

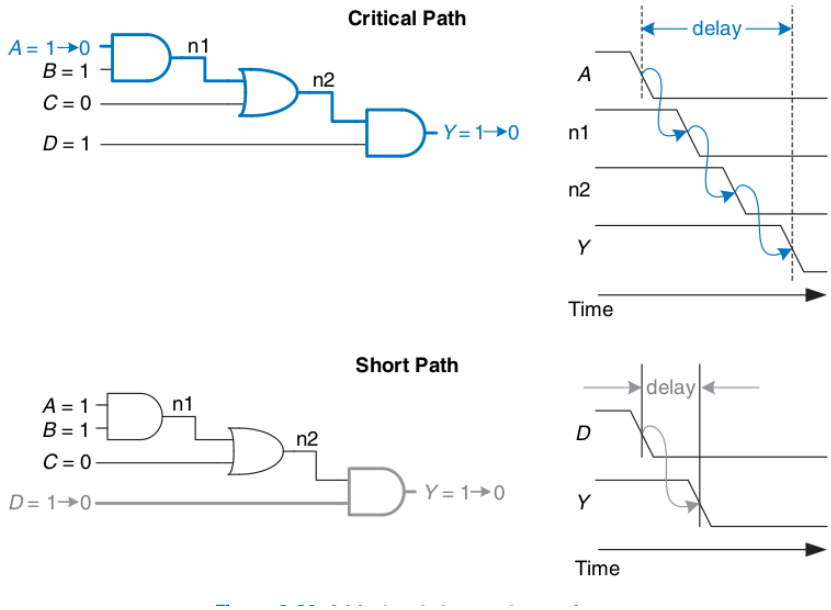
5.3.1 Delays



Definition 1 (Propagation delay (t_{pd}))**.**

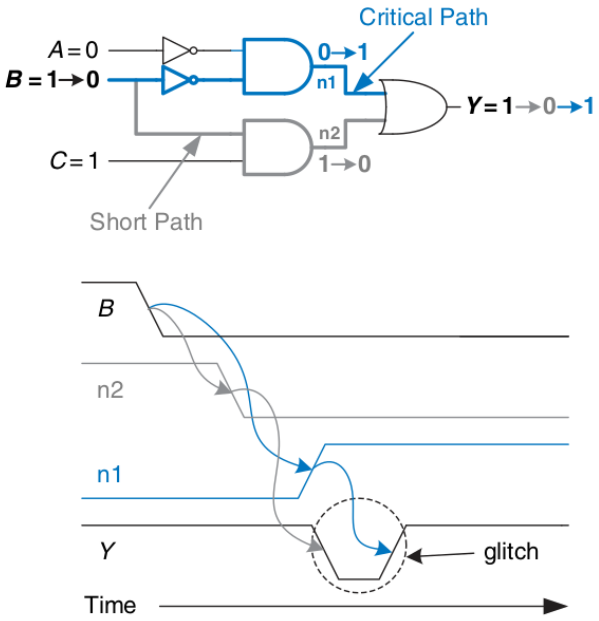
Definition 2 (Contamination delay (t_{cd}))**.**

5.3.2 Paths



Example 31. *Find the propagation delay of the circuit above given that propagation delay of each gate is* 100*ps add contamination delay of* 60*ps.*

5.4 Glitches or Hazards



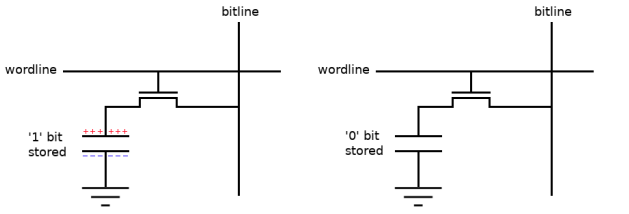
Definition 3 (Glitch or Hazard)**.**

Example 32. *Design a circuit that fixes the glitch in the above circuit (also known as glitch-free or hazard-free circuit).*

5.5 How to create memory element from circuits

Two types of memory

1. Volatile memory. For example, RAM, CPU registers.
2. Non-volatile memory. For example, SSD, Flash drives. (Not covered in this course)
   (a) Memories that require periodic refreshing. For example, DRAM: Dynamics Random Access memory (Not covered in this course)



[1]

(b) Memories that are always refreshing. For example, SRAM: Static Random Access memory [3, Appendix B.64]

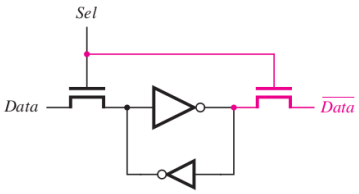
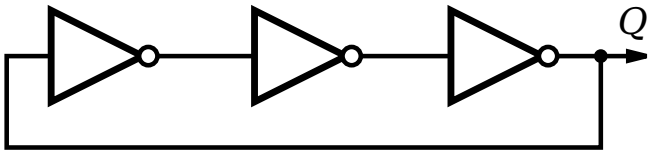


Figure B.64 An SRAM cell.

5.6 Latches and Flip-Flops [1, Sec 3.2]

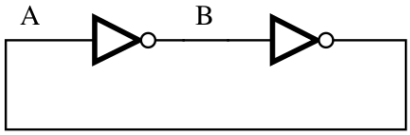
Example 33 (Ring oscillator)**.** *[1, Sec 3.31] How many stable states does the following circuit have?*



Definition 4 (Astable circuits)**.**

Example 34. *Analyze the timing diagram of the following circuit.*

[1] Image source: `allaboutcircuits.com/technical-articles/introduction-to-dram-dynamic-random-access-memory/`

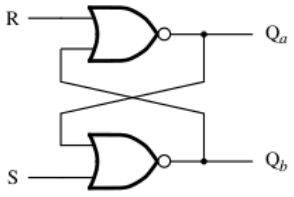


Definition 5 (Bistable circuits)**.**

Definition 6 (Characteristic or state table)**.** *Draw the characteristic or state table of the above circuit.*

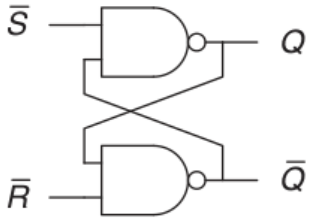
5.6.1 SR (Set-Reset) latch [1, Sec 3.2.1]

Definition 7 (SR latch)**.** *The following circuit is called the SR latch.*



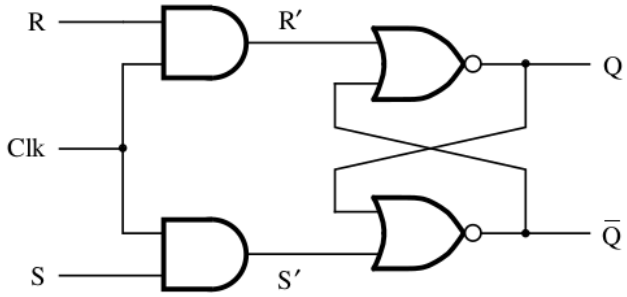
1. *How many stable states does this circuit have?*
2. *Draw its characteristic or state table.*
3. *Draw SR latch symbol*

Problem 29 (SR latch using NAND gates)**.** *Draw the characteristic or state table for the following circuit*



5.6.2 Gated SR latch [3, Sec 5.2]

Definition 8 (Gated SR latch)**.** *The following circuit is called the Gated SR latch.*

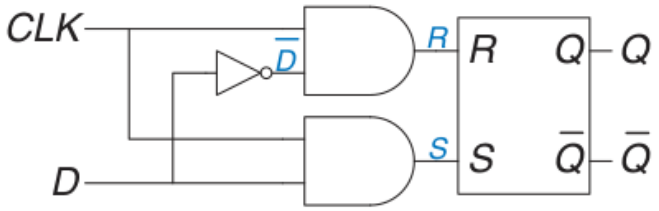


1. *Draw its characteristic table.*

2. *Draw the Gated SR latch symbol*

5.6.3 D (Data) latch [1, Sec 3.2.2]

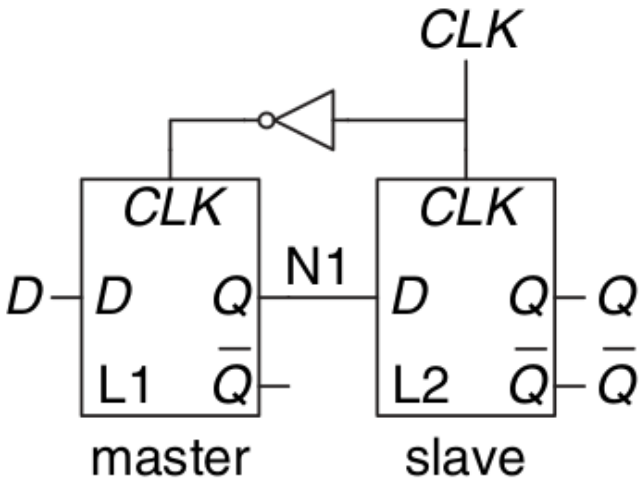
Definition 9 (D latch). *The following circuit is called the D latch.*



1. *Draw its characteristic table.*
2. *Draw the D latch symbol*

5.6.4 D flip-flop [1, Sec 3.2.2]

Definition 10 (D flip-flop). *The following circuit is called the D flip-flop.*



1. *Draw its timing diagram*
2. *Draw its characteristic table.*
3. *Draw the D flip-flop symbol*

Remark 5. *What is the difference between a latch and a flip-flop?*

Example 35. *Add a* RESET *signal to the D flip-flop that resets the state of flip-flop to 0.*

Example 36. *The toggle (T) flip-flop has one input, CLK, and one output, Q. On each rising edge of CLK, Q toggles to the complement of its previous value. Draw a schematic for a T flip-flop using a D flip-flop and an inverter.*

Problem 30. *A JK flip-flop receives a clock and two inputs, J and K. On the rising edge of the clock, it updates the output, Q. If J and K are both 0, Q retains its old value. If only J is 1, Q becomes 1. If only K is 1, Q becomes 0. If both J and K are 1, Q becomes the opposite of its present state.*

1. *Construct a JK flip-flop using a D flip-flop and some combinational logic.*

2. *Construct a D flip-flop using a JK flip-flop and some combinational logic.*

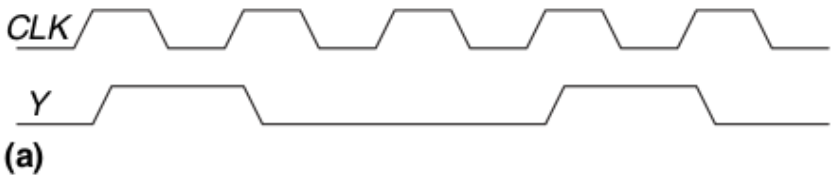
3. *Construct a T flip-flop (see Exercise 3.9) using a JK flip-flop.*

5.7 Finite State Machines [1, Sec 3.4]

[2]

Example 37. *Design an occupancy counter that depends on a sensor S at the class door. The sensor is triggered every time a person passes through the door. Assume that the counter starts at zero. Assume we only need up to two bit counter C_1C_0. Draw a state table for this circuit.*

Problem 31. *A divide-by-N counter has one output and no inputs. The output Y is HIGH for one clock cycle out of every N. In other words, the output divides the frequency of the clock by N. The waveform for a divide-by-3 counter is shown here:*



Sketch circuit designs for such a counter

Problem 32. *Design a 3-bit counter which counts in the sequence: 001, 011, 010, 110, 111, 100, (repeat) 001, ...*

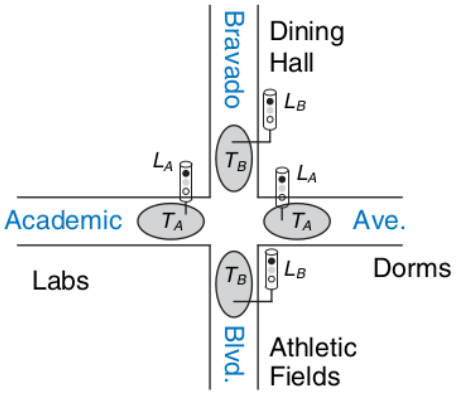
Example 38. *Design an odd-even counter for an single bit input. The output of this circuit should be 1 if the number of 1s to the input have been odd so far and 0 otherwise.*

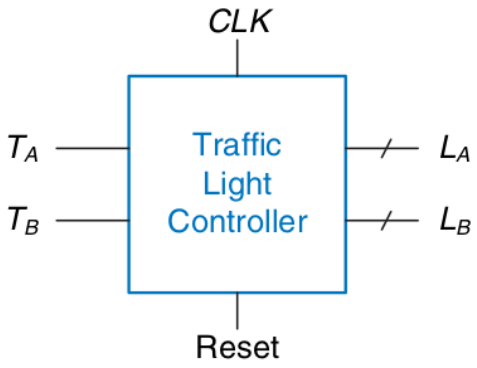
Example 39 (Sequence detectors)**.** *A sequential circuit has one input and one output. The output becomes 1 and remain 1 thereafter when at least two 0's and at least two 1's have occurred as inputs regardless of the order of*

Example 40. *Consider the problem of inventing a controller for a traffic light at a busy intersection on campus. There are two traffic sensors, T_A and T_B , on Academic Ave. and Bravado Blvd., respectively. Each sensor indicates TRUE if students are present and FALSE if the street is empty. There are two traffic lights, L_A and L_B, to control traffic. Each light receives digital inputs specifying whether it should be green, yellow, or red. When the system is reset, the lights are green on Academic Ave. and red on Bravado Blvd. As long as traffic is present on Academic Ave., the lights do not change. When there is no longer traffic on Academic Ave., the light on Academic Ave. becomes yellow for 5 seconds before it turns red and Bravado Blvd.'s light turns green. Similarly, the Bravado Blvd. light remains green as long as traffic is present on the boulevard, then turns*

[2] These notes will not fit on your note sheet.

yellow and eventually red.





1. *Draw a state transition diagram*
2. *Draw a state table*
3. *Assign binary encodings to each of the states*
4. *Redraw the state table with binary encodings. Design a minimal SOP boolean expression.*
5. *Assign binary encodings to each of the output and redraw the output table. Design a minimal SOP boolean expression for the outputs.*

Problem 33. *Design a circuit for a 2x2 pixel resolution pong game, where the ball can only occupy 4 possible pixels and a single paddle occupies another 2 pixels. The ball bounces of the paddle when the paddle is in the correct row. To keep it interesting, the ball takes a different path from the source path. Track the score with a single bit counter.*

5.8 Mealy Moore Sequence detector

5.9 Objectives

1. Analyse and design both Mealy and Moore sequential circuits with multiple inputs and multiple outputs
2. Convert between Mealy and Moore designs

5.10 Mealy vs Moore Finite State Machines

Definition 11 (Finite State Machines (FSM))**.** *[1, Sec 3.4]*

Definition 12 (Mealy FSM)**.** *[1, Sec 3.4.3]*

Definition 13 (Moore FSM)**.** *[1, Sec 3.4.3]*

Example 41. *A sequential circuit has one input (X) and one output (Z). The circuit examines groups of four consecutive inputs and produces an output Z=1 if the input sequence 0010 or 0001 occurs. The sequences can overlap. Draw both Mealy and Moore timing diagrams. Find the Mealy and Moore state graph.*

Problem 34. *A sequential circuit has one input (X) and one output (Z). The circuit examines groups of four consecutive inputs and produces an output Z=1 if the input sequence 0101 or 1001 occurs. The circuit resets after every four inputs. Draw both Mealy and Moore timing diagrams. Find the Mealy and Moore state graph.*

5.11 State reduction via Implication tables

8.Skip this section

To minimize the number of states, we will identify "equivalent states" and eliminate any redundancy found. Two states are equivalent if they have equivalent next states and the same output for each possible input condition. To find equivalent states we will create an "implication table" which looks at pairs of states and identifies which states have to be equivalent if this pair is to be equivalent. We can use a table to hold information about each pair.

To investigate all possible pairs, we could use a square table such as this to record information about pairs of states. But note every pair represented in the upper right "triangle" of the table is also listed in the lower left "triangle" of the table. Furthermore, the diagonal of the table will only present information about a state being equivalent to itself. Only the part of the table in bold is needed to investigate all possible pairs of states:

PS	NS		Z	
	X=0	X=1	X=0	X=1
A	B	C	0	1
B	A	E	1	0
C	D	A	0	1
D	C	E	1	0
E	A	F	1	0
F	E	F	0	1

Figure 5.1: Example state table

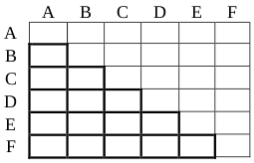


Figure 5.2: Implication table

5.12 State assignment

9.Skip this section

5.13 Finite State Machine Optimization

5.14 Objectives

10.highlight one-hot state vector 11.highlight the timing vs space trade-offs of mealy vs moore FSM in context of verilog 12.Consider describing the stages of Synthesis

1. Analyse and design both Mealy and Moore sequential circuits with multiple inputs and multiple outputs
2. Convert between Mealy and Moore designs
3. Perform a state assignment using the guideline method
4. Reduce the number of states in a state table using row reduction and implication tables

5.15 Mealy vs Moore Finite State Machines

Definition 14 (Finite State Machines (FSM)). *[1, Sec 3.4]*

Definition 15 (Mealy FSM). *[1, Sec 3.4.3]*

Definition 16 (Moore FSM). *[1, Sec 3.4.3]*

Example 42. *A sequential circuit has one input (X) and one output (Z). The circuit examines groups of four consecutive inputs and produces an output Z=1 if the input sequence 0010 or 0001 occurs. The sequences can overlap. Draw both Mealy and Moore timing diagrams. Find the Mealy and Moore state graph.*

Problem 35. *A sequential circuit has one input (X) and one output (Z). The circuit examines groups of four consecutive inputs and produces an output Z=1 if the input sequence 0101 or 1001 occurs. The circuit resets after every four inputs. Draw both Mealy and Moore timing diagrams. Find the Mealy and Moore state graph.*

5.16 Full procedure for designing sequential logic circuit

1. Convert the word problem to a state transition diagram. Let the states be $S_0, S_1, S_2, \ldots, S_n$.

2. Draw state transition table with named states. For example,

Present State	Next State		Outputs	
	X = 0	X = 1	X=0	X=1
S_0	S_1	S_2	0	0
S_1	S_2	S_0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

3. State reduction step: Reduce the number of required states to a minimum. Eliminate unnecessary or duplicate states.

4. State assignment step: Assign each state a binary representation. For example,

State name	State assignments ($Q_2Q_1Q_0$)
S_0	000
S_1	001
$\vdots$	$\vdots$

5. Draw State assigned transition table. For example,

Inputs (X_1X_0)	Present State (Q_1Q_0)	Next State ($Q_1^+Q_0^+$)	Outputs (Z_1Z_0)
0 0	00	01	0 0
0 0	01	10	0 0
$\vdots$ $\vdots$	$\vdots$	$\vdots$	$\vdots$ $\vdots$

(a) Use excitation tables to find truth tables for the combinational circuits. For example, the excitation table for J-K ff is

Q	Q^+	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

5.17 State assignment by guideline method [2, Section 8.2.5]

13.skip this section

5.17.1 State Maps

Example 43. *Draw a state map for a sequential assignment of the states*

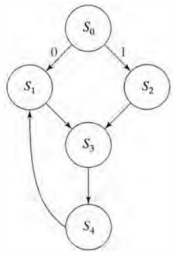


Figure 8.27 Five-state finite state machine.

5.17.2 Guideline method

Guideline method states that the following states should be adjacent in the state map according the following priorities:

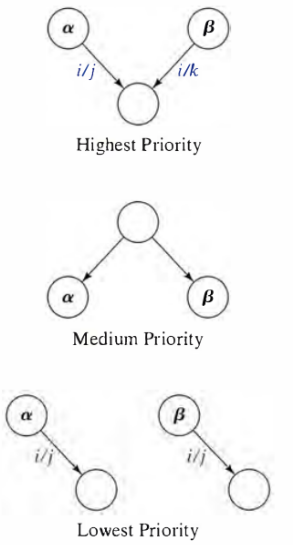


Figure 8.29 Adjacent assignment priorities.

Example 44. *A state transition table is given. Find optimal state assignment by using the guideline method.*

Input Sequence	Present State	Next State $X=0$	Next State $X=1$	Output $X=0$	Output $X=1$
Reset	S_0	S_1'	S_1'	0	0
0 or 1	S_1'	S_3'	S_4'	0	0
00 or 10	S_3'	S_0	S_0	0	0
01 or 11	S_4'	S_0	S_0	1	0

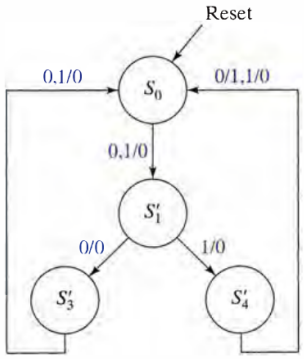
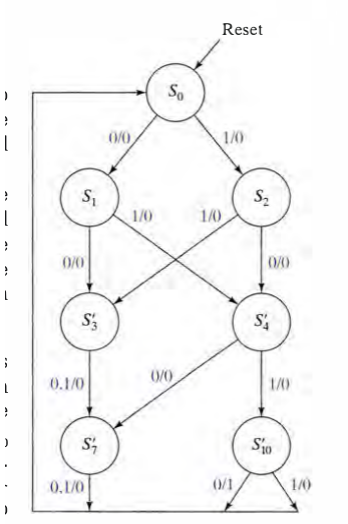


Figure 8.30 Reduced state diagram for 3-bit sequence detector.

Example 45. *Draw a Mealy FSM for detecting binary string 0110 or 1010. The machine returns to the reset state after each and every 4-bit sequence. Draw the state transition diagram on your own as practice problem. The state transition diagram is given here. Find optimal state assignment by using the guideline method.*



5.18 State reduction by implication chart

Example 46. *Design a Mealy FSM for detecting binary sequence 010 or 0110. The machine returns to reset state after each and every 3-bit sequence. For now the state transition table is given. Reduce the following state transition table*

Input Sequence	Present State	Next State $X=0$	Next State $X=1$	Output $X=0$	Output $X=1$
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

5.18.1 Implication chart Summary

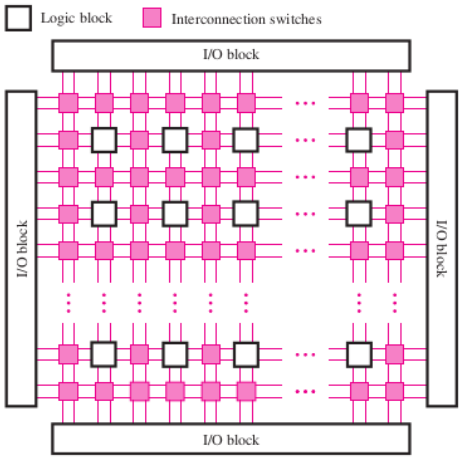
The algorithms for state reduction using the implication chart method consists of the following steps

1. Construct the implication chart, consisting of one square for each possible combination of states taken two at a time.
2. For each square labeled by states S_i and S_j, if the outputs of the states differ, mark the square with an X; the states are not equivalent. Otherwise, they may be equivalent. Within the square write implied pairs of equivalent next states for all input combinations.
3. Systematically advance through the squares of the implication chart. If the square labeled by states S_i, S_j contains an implied pair S_m, S_n and square S_m, S_n is marked with an X, then mark S_i, S_j with an X. Since S_m, S_n are not equivalent, neither are S_i, S_j.
4. Continue executing Step 3 until no new squares are marked with an X.
5. For each remaining unmarked square S_i, S_j, we can conclude that S_i, S_j are equivalent.

Chapter 6

More definitions

6.1 FPGA [3, Section B.6.5]



(a) General structure of an FPGA

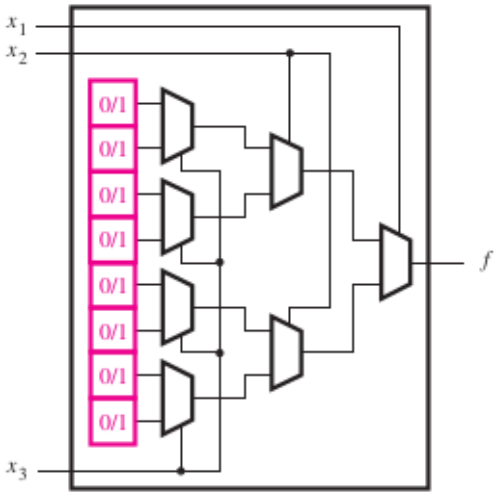


Figure B.37 A three-input LUT.

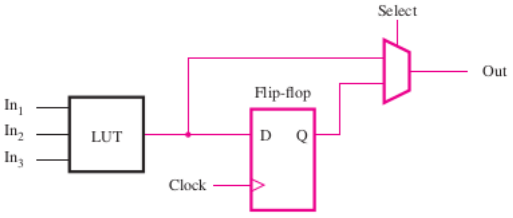


Figure B.38 Inclusion of a flip-flop in an FPGA logic element.

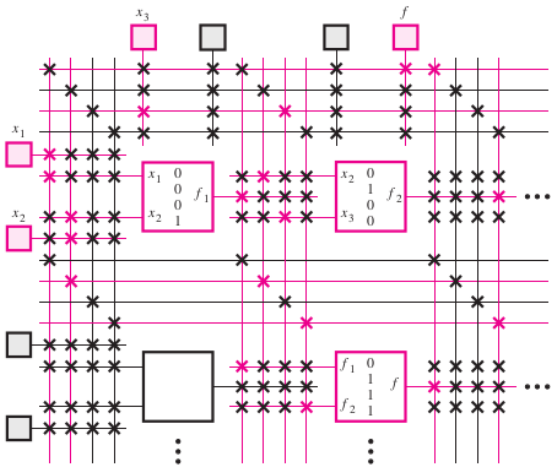
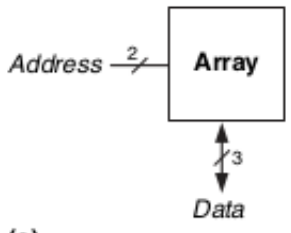


Figure B.39 A section of a programmed FPGA.

Definition 17 (Random Access Memory (RAM)). *Structure of a RAM is as follows:*



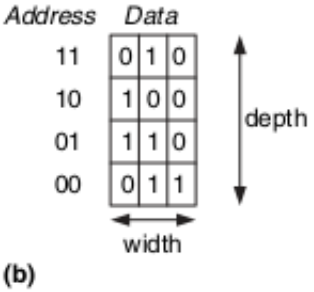


Figure 5.39 4 × 3 memory array: (a) symbol, (b) function

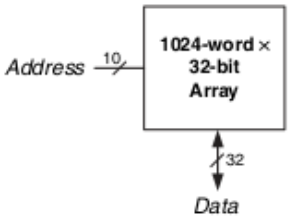


Figure 5.40 32 Kb array: depth = 2^{10} = 1024 words, width = 32 bits

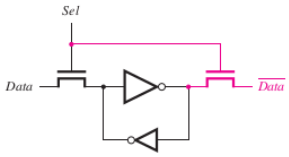


Figure B.64 An SRAM cell.

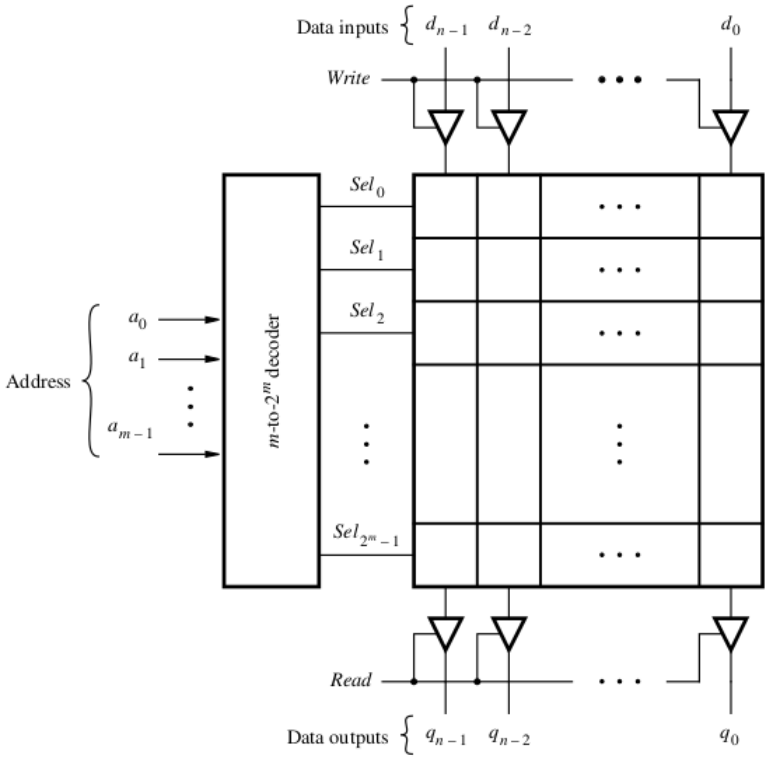


Figure B.66 A $2^m \times n$ SRAM block.

Definition 18 (Read Only Memory (ROM)). *Structure of a ROM is as follows:*

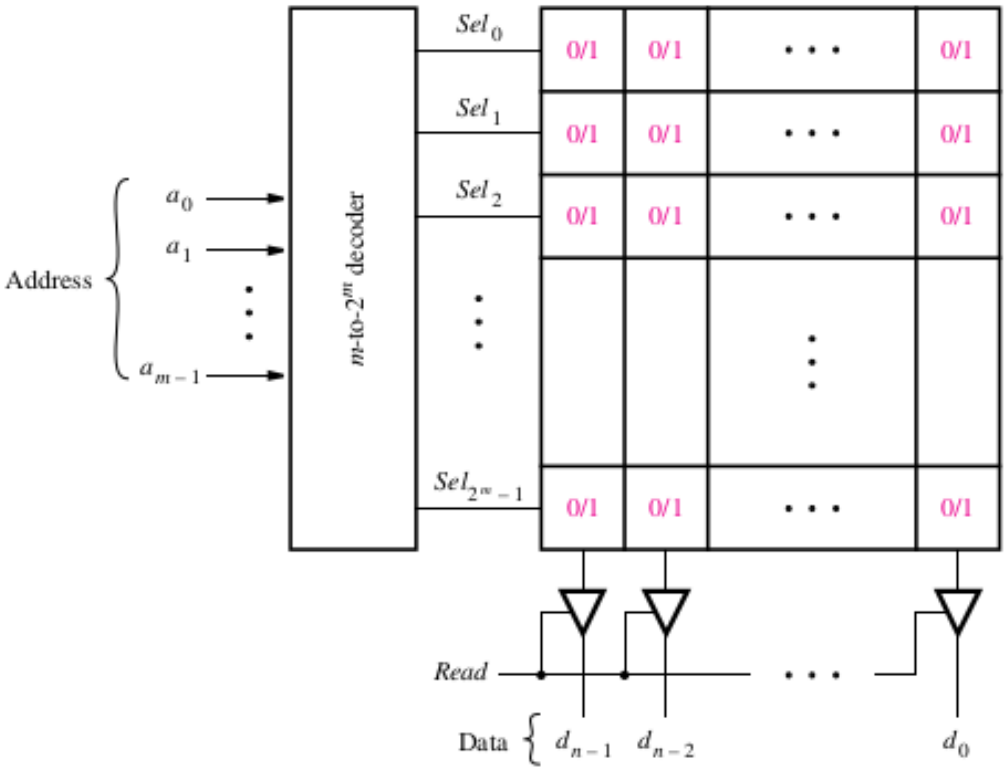
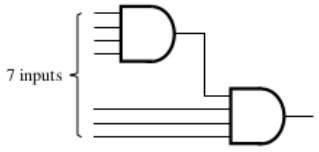


Figure B.72 A $2^m \times n$ read-only memory (ROM) block.

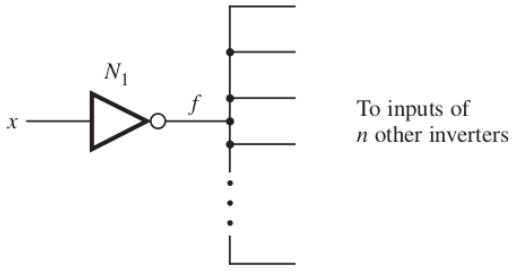
Definition 19 (Fan-in)**.** *The fan-in of a logic gate is number of inputs to a logic gate. [3, Section B.8.9]*



Remark 6 (Fan-in)**.** *The fan-in of a gate is limited by the propagation delay t_p. Higher the fan-in, higher the t_p. The output voltage thresholds like V_{OL} and V_{OH} also limit fan-in. Higher the fan-in, higher is V_{OL} (and lower is the V_{OH}).*

Example 47. *Implement an OR gate with fan-in of 7 using OR gates with fan-in of 3.*

Definition 20 (Fan-out)**.** *The fan-out of a logic gate is the maximum number of other gates that can be connected to output of a gate. [3, Section B.8.9]*



Definition 21 (Programmable Logic Array (PLA))**.** *Structure of a PLA:*

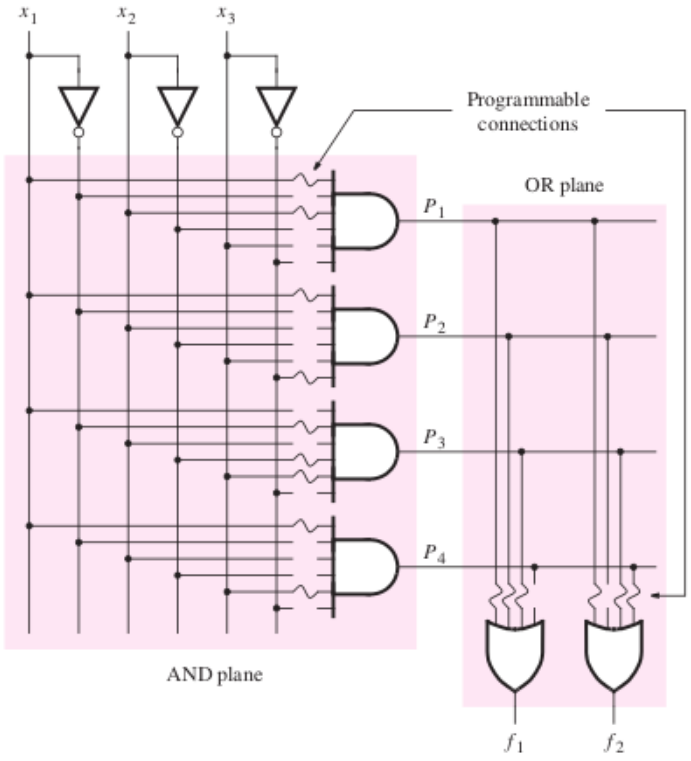


Figure B.26 Gate-level diagram of a PLA. *[3, Section B.6.1]*

Definition 22 (Programmable Array Logic (PAL))**.** *Structure of a PAL:*

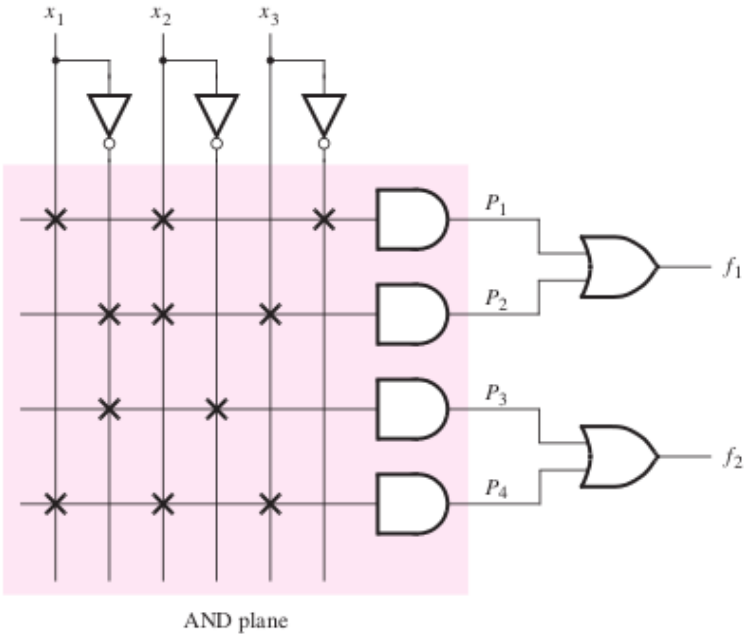
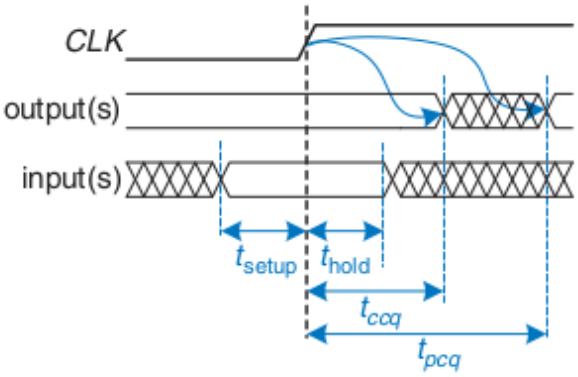


Figure B.28 An example of a PAL.

[3, Section B.6.2]

Example 48. *What is the difference between PLA and PAL?*

6.2 Timing parameters for sequential circuit [1, Section 3.5]



Definition 23 (Setup time t_{su} of a latch/flip-flop)**.** *Time for which input must be stable before the clock edge.*

Definition 24 (Hold time t_h of a latch/flip-flop)**.** *Time for which input must be stable after the clock edge.*

Definition 25 (Clock-to-Q contamination delay t_{ccq} of a latch/flip-flop)**.** *Time taken to influence (contaminate) the Q output after the clock edge.*

Definition 26 (Clock-to-Q propagation delay t_{ccq} of a latch/flip-flop)**.** *Time taken for Q output to stabalize after the clock edge.*

Chapter 7

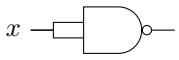
Quine McCluskey

7.1 Circuit design using NAND/NOR gates

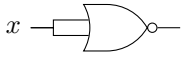
Example 49. *Implement the function* $f(x_1, x_2, x_3) = \sum m(2, 3, 4, 6, 7)$ *using (1) NAND gates only and (2) NOR gates only.*

Remark 7. *NAND-NAND logic is generated from SOP form. NOR-NOR logic is generated from POS form.*

Remark 8. *NOT gate can also be created from a NAND gate* $\bar{x} = \overline{x \cdot x}$.



Remark 9. *NOT gate can also be created from a NOR gate* $\bar{x} = \overline{x + x}$.



Problem 36. *Design the simplest circuit that implements the function* $f(x_1, x_2, x_3) = \sum m(3, 4, 6, 7)$ *using (1) NAND gates only (2) NOR gates only.*

7.2 PI Table reduction and Petrick's method

This is not in the text-book. For additional reading, please refer to the linked resources on the website.

Definition 27 (Implicant)**.** *Given a function f of n variables, a product term P is an implicant of f if and only if for every combination of values of the n variables for which $P = 1$, f is also equal to 1.*

Definition 28 (Prime Implicant)**.** *A prime implicant of a function f is an implicant which is no longer an implicant if any literal is removed from it.*

There are 4 main steps in the Quine-McCluskey algorithm/PI Table reduction and Petrick's method:

1. Generate Prime Implicants
2. Construct Prime Implicant Table. PIs as columns, and minterms as rows (don't cares are excluded).
3. Reduce Prime Implicant Table by repeating following steps until they it cannot be reduced further
   (a) Remove Essential Prime Implicants
   (b) Row Dominance: Remove *dominating* rows. (i.e. unnecessary minterms)
   (c) Column Dominance: Remove *dominated* columns. (i.e. remove unnecessary PIs)
4. Solve Prime Implicant Table by Petrick's method

7.2.1 Generate Prime Implicants

Example 50. *Generate prime implicants of the function $F(A, B, C, D) = \sum m(0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15)$ using Quine-McCluskey method*

Steps:

1. Start with writing minterms in binary format (include don't cares as minterms).

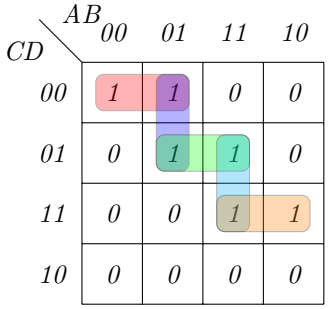
2. Create potential groups of minterms that can be combined (merged). The only minterms that can be combined differ only be single 1. Create a new list of combined minterms as n-1 literal implicants.
3. Check off the minterms that could be combined. Unchecked minterms are prime implicants (PIs).
4. Repeat the grouping process with n-1 literal implicants.

Problem 37. *Generate PIs for the function* $F(A,B,C,D) = \sum m(0,2,3,4,5,6,7,8,9,10,11,12,13)$.

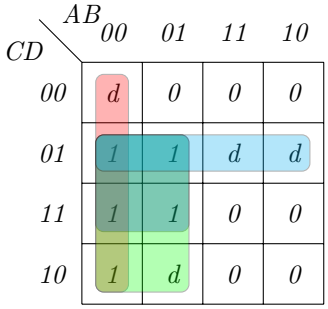
7.2.2 Prime Implicants table and reduction

Example 51. *Reduce the prime implicants* $\{\bar{B}\bar{D}, C\bar{D}, BD, BC, A\bar{D}, AB\}$ *using prime implicants table.*

Example 52.



Example 53.



Example 54. *Reduce the following PI table*

	$\bar{A}\bar{D}$	$\bar{B}\bar{D}$	$\bar{C}\bar{D}$	$\bar{A}C$	$\bar{B}C$	$\bar{A}B$	$B\bar{C}$	$A\bar{B}$	$A\bar{C}$
0	X	X	X						
2	X	X		X	X				
3				X	X				
4	X		X			X	X		
5						X	X		
6	X			X		X			
7				X		X			
8		X	X					X	X
9								X	X
10		X			X			X	
11					X			X	
12			X		X		X		X
13							X		X

7.2.3 Petrick's method

Example 55. *Solve the Prime Implicant table using Petrick's method*

	$p_1 = \bar{A}C$	$p_2 = \bar{B}C$	$p_3 = \bar{A}B$	$p_4 = B\bar{C}$	$p_5 = A\bar{B}$	$p_6 = A\bar{C}$
3	X	X				
5			X	X		
7	X		X			
9					X	X
11		X			X	
13				X		X

Example 56. *Find the minimum SOP expression for the function* $F(A, B, C, D) = \sum m(2, 3, 7, 9, 11, 13) + \sum d(1, 10, 15)$ *using Quine-McCluskey method.*

Chapter 8

Analog details

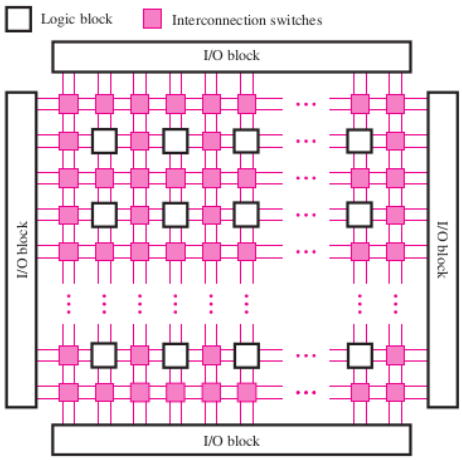
Some of the material is out of the textbook. Additional resources include Appendix B of Brown and Vranesic book, "Fundamentals of digital logic."

8.1 Objectives

1. Describe how tri-state and open-collector outputs are different from totem-pole outputs

2. Compute noise margin of one device driving the same time

8.2 FPGA [3, Section B.6.5]



(a) General structure of an FPGA

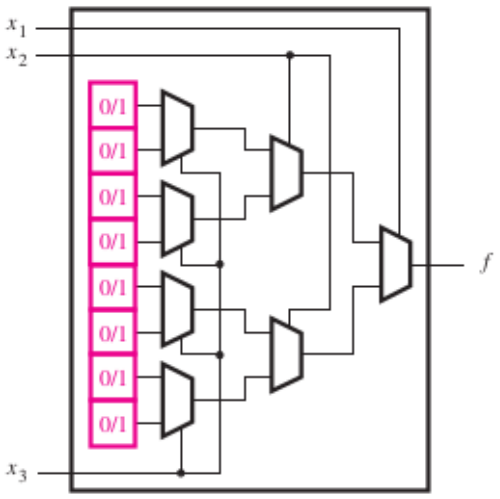


Figure B.37 A three-input LUT.

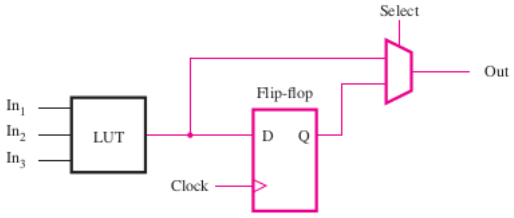


Figure B.38 Inclusion of a flip-flop in an FPGA logic element.

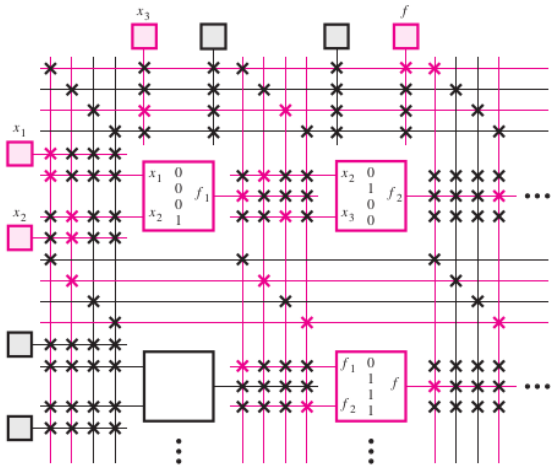
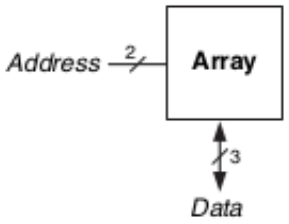
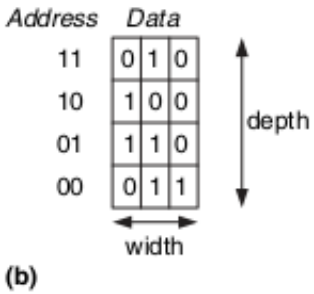


Figure B.39 A section of a programmed FPGA.

Definition 29 (Random Access Memory (RAM))**.** *Structure of a RAM is as follows:*



(a)



(b)

Figure 5.39 4 × 3 memory array: (a) symbol, (b) function

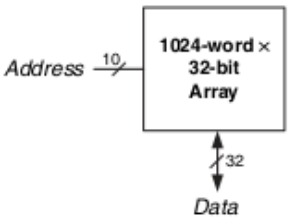


Figure 5.40 32 Kb array: depth = 2^{10} = 1024 words, width = 32 bits

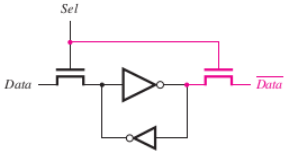


Figure B.64 An SRAM cell.

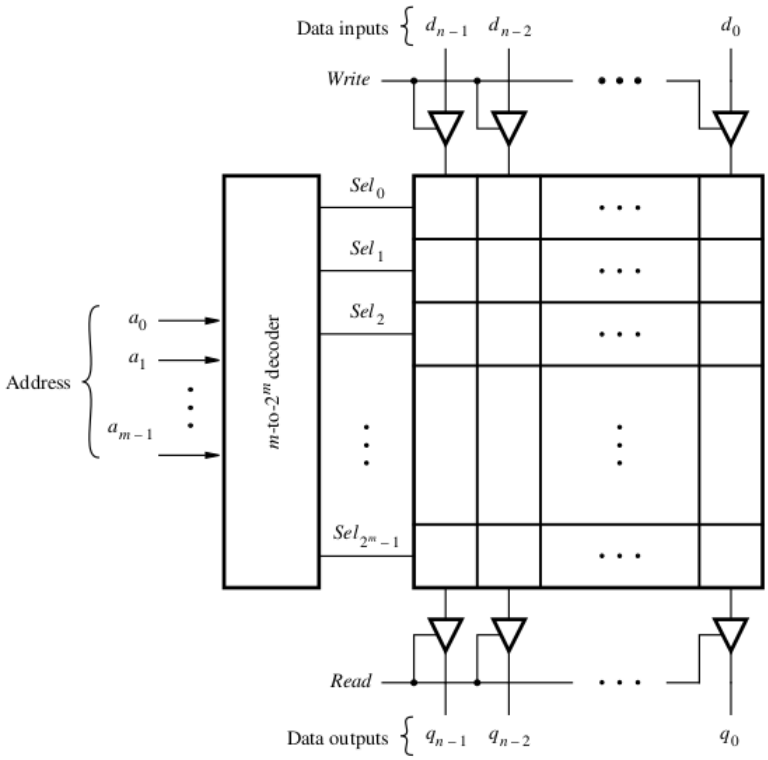


Figure B.66 A $2^m \times n$ SRAM block.

Definition 30 (Read Only Memory (ROM))**.** *Structure of a ROM is as follows:*

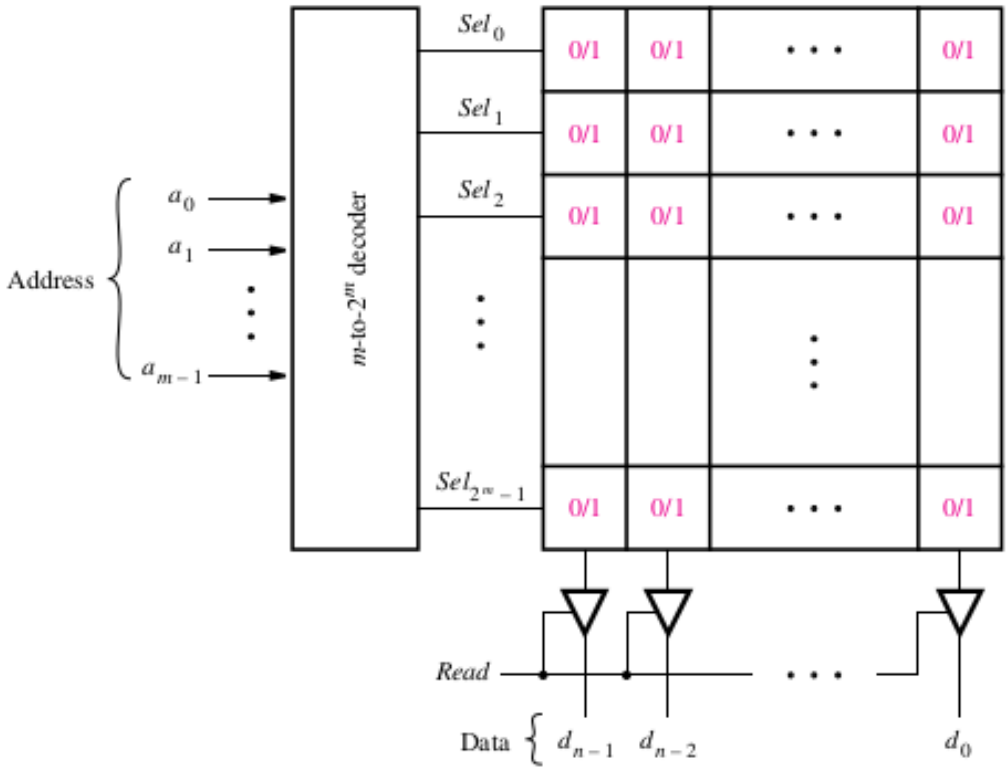
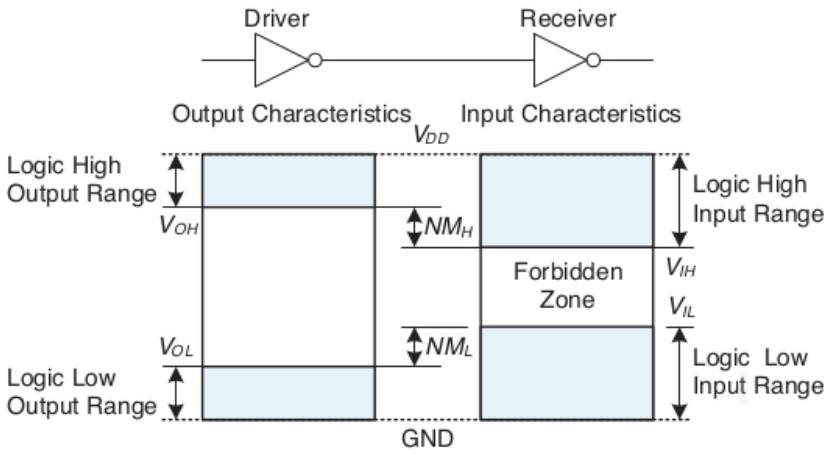


Figure B.72 A $2^m \times n$ read-only memory (ROM) block.

Example 57. *Draw a Multiplexer using sum of products form.*

8.3 Logic levels and Noise Margins



Definition 31 (Supply Voltage ($V_{DD}/V_{CC}/V_{SS}$)).

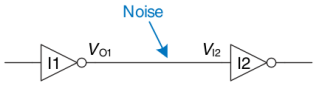
Definition 32 (Ground Voltage (V_{GND})).

Definition 33 (Input high (V_{IH}) and Input Low (V_{IL}) of a gate).

Definition 34 (Output high (V_{OH}) and Output low (V_{OL}) of gate).

Definition 35 (Positive logic and Negative logic).

Definition 36 (Noise margins (NM_L and NM_H) of a channel).



Example 58.
If $V_{DD} = 5V$, $V_{IL} = 1.35V$, $V_{IH} = 3.15V$, $V_{OL} = 0.33V$ and $V_{OH} = 3.84V$ for both the "inverters", then what are the low and high noise margins? Can the circuit tolerate 1V of noise at the channel?

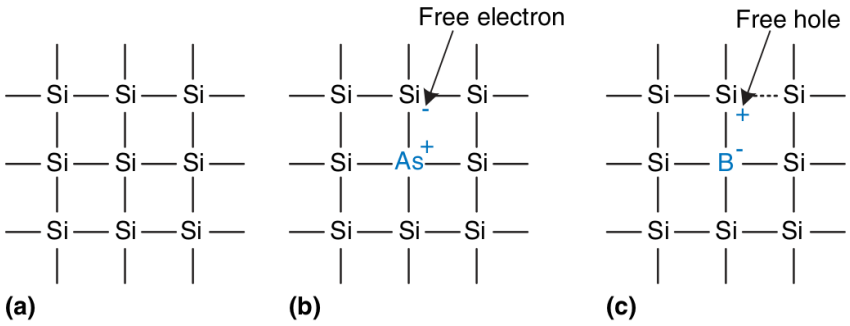
8.4 Semiconductors and Doping

Not in syllabus but good to know

Elements recognized as metalloids V • T • E

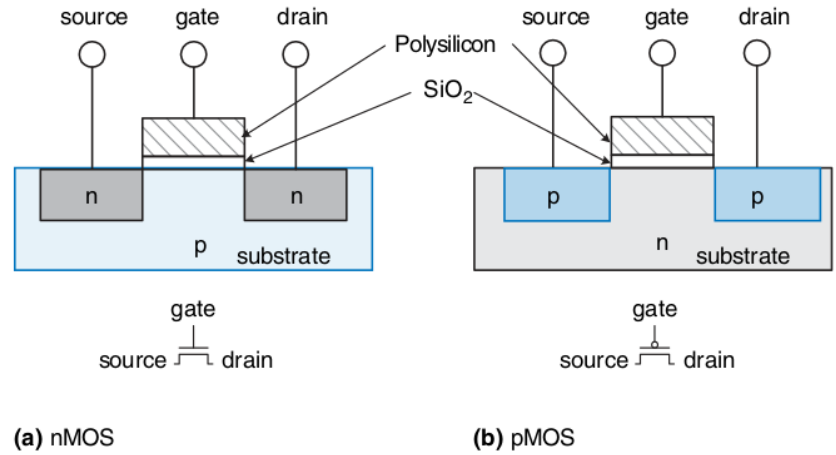
	13	14	15	16	17
2	B Boron	C Carbon	N Nitrogen	O Oxygen	F Fluorine
3	Al Aluminium	Si Silicon	P Phosphorus	S Sulfur	Cl Chlorine
4	Ga Gallium	Ge Germanium	As Arsenic	Se Selenium	Br Bromine
5	In Indium	Sn Tin	Sb Antimony	Te Tellurium	I Iodine
6	Tl Thallium	Pb Lead	Bi Bismuth	Po Polonium	At Astatine

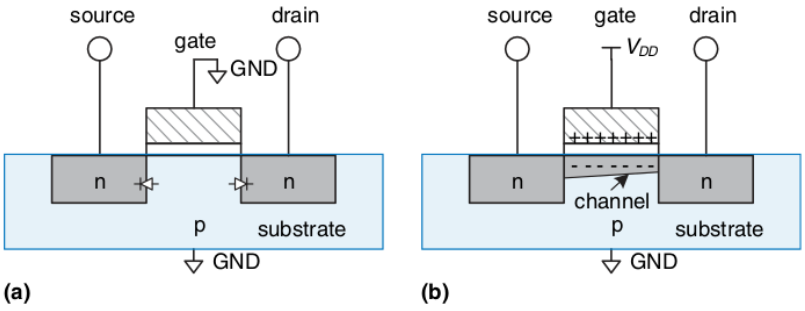
Commonly recognized (86–99%): B, Si, Ge, As, Sb, Te
Irregularly recognized (40–49%): Po, At
Less commonly recognized (24%): Se
Rarely recognized (8–10%): C, Al
(All other elements cited in less than 6% of sources)
Arbitrary metal-nonmetal dividing line: between Be and B, Al and Si, Ge and As, Sb and Te, Po and At



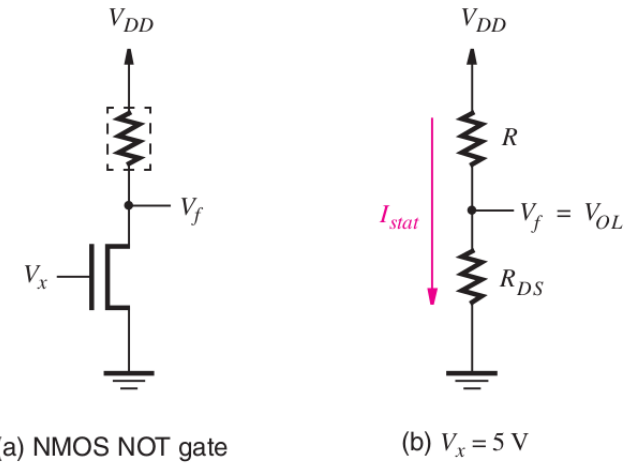
8.5 MOSFET: Metal Oxide Field Effect Transistors

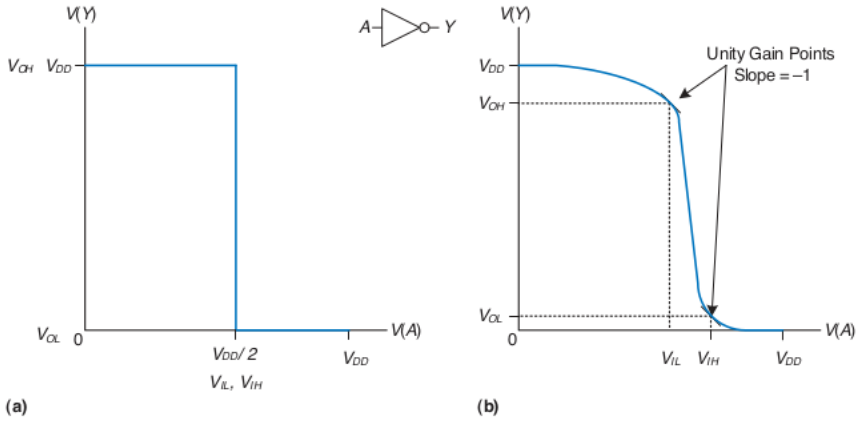
Not in syllabus but good to know

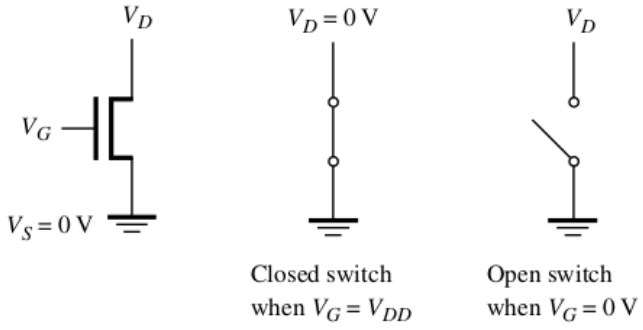




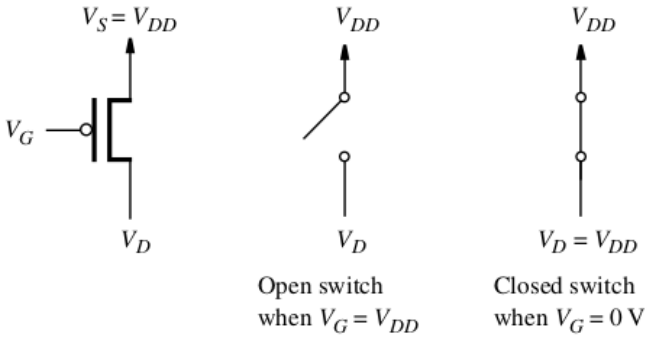
8.6 DC Transfer characteristic







(a) NMOS transistor



(b) PMOS transistor

Example 59. *Draw a NOT gate using nMOS transistors.*

Example 60. *Draw a NOT gate using pMOS transistors.*

Remark 10. *nMOS transistors pass 0's well (output between 0 and $V_{DD} - V_t$). pMOS transistors pass 1's well (output between V_t and V_{DD}).*

Example 61. *Draw CMOS NOT Gate.*

Example 62. *Draw a two input CMOS NAND Gate*

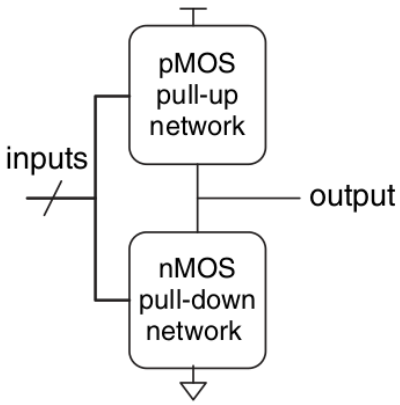
Definition 37 (Negative logic)**.**

Example 63. *Analyze the above circuit under negative logic.*

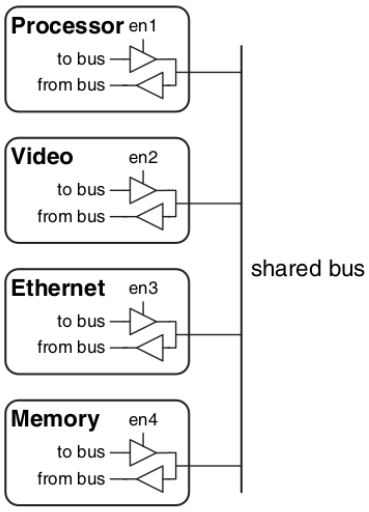
Example 64. *Draw a three input NAND using CMOS.*

Example 65. *Draw a three input NOR using CMOS.*

Example 66. *Draw a two input AND gate using CMOS.*



8.6.1 Gates with floating output



Definition 38 (Transmission gate)**.** *Draw a schematic of transmission gate and truth table for transmission gate. What is its commonly used symbol?*

Definition 39 (Tristate buffer)**.** *What is tristate buffer? Draw it's symbol and truth table? Where is it used?*

Example 67. *Draw a Multiplexer using transmission gates.*

Example 68. *Draw a Multiplexer using tristate buffers.*

Definition 40 (Totem-pole)**.** *Draw a Push-pull (or Totem-pole) output NAND gate using CMOS. Can you connect this gate to a shared bus?*

Definition 41 (Tristate)**.** *Draw a Tristate output NAND gate using CMOS with an output enable (OE) input. Can you connect this gate to a shared bus?*

Definition 42 (Open-collector)**.** *Draw a open-collector output NAND gate. Can you connect this gate to a shared bus?*

8.7 Verilog truth tables

Table 11-11—Bitwise binary AND operator

&	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

Table 11-12—Bitwise binary OR operator

|	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

Bibliography

[1] Sarah L Harris and David Harris. *Digital design and computer architecture*. Morgan Kaufmann, 2022.

[2] Randy Katz and Gaetano Barriello. *Contemporary Logic Design*. Prentice Hall, 2004.

[3] Brown Stephen and Vranesic Zvonko. *Fundamentals of digital Logic with Verilog design*. McGraw Hill, 2022.