

COM 251 Logic Design and Circuits

Combinational Logic Design: More Gates

Prof. Dr. Halûk Gümüşkaya

haluk.gumuskaya@gediz.edu.tr

haluk@gumuskaya.com http://www.gumuskaya.com

Computer Engineering Department

GEDİZÜNİVERSİTESİ
izmir

Thursday, October 27, 2011

1

Acknowledgement

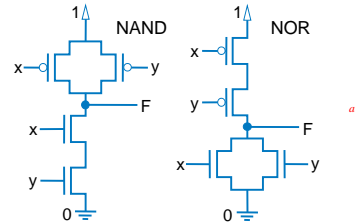
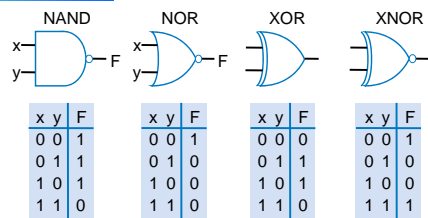
The slides have been based in-part upon original slides of a number of books including:

- *Digital Design with RTL Design, Verilog and VHDL*, 2nd ed., Frank Vahid, John Wiley, 2011.
- *Logic and Computer Design Fundamentals*, 4th Ed., M. Morris Mano, C. Kime, Prentice Hall, 2008.

2

2.8

More Gates



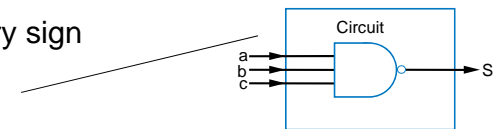
- NAND: Opposite of AND (“NOT AND”)
- NOR: Opposite of OR (“NOT OR”)
- XOR: Exactly 1 input is 1, for 2-input XOR. (For more inputs -- odd number of 1s)
- XNOR: Opposite of XOR (“NOT XOR”)
- NAND same as AND with power & ground switched
 - nMOS conducts 0s well, but not 1s (reasons beyond our scope) – so NAND is more efficient
- Likewise, NOR same as OR with power/ground switched
- NAND/NOR more common
- AND in CMOS: NAND with NOT
- OR in CMOS: NOR with NOT

3

More Gates: Example Uses

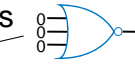
- Aircraft lavatory sign example

– $S = (abc)'$



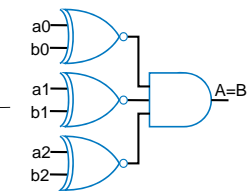
- Detecting all 0s

– Use NOR



- Detecting equality

– Use XNOR



- Detecting odd # of 1s

– Use XOR

– Useful for generating “parity” bit common for detecting errors

4

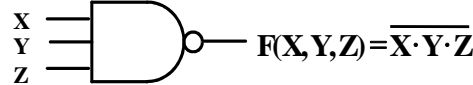
Completeness of NAND

- Any Boolean function can be implemented *using just NAND gates*. Why?
 - Need AND, OR, and NOT
 - NOT: 1-input NAND (or 2-input NAND with inputs tied together)
 - AND: NAND followed by NOT
 - OR: NAND preceded by NOTs
- Thus, **NAND is a universal gate**
- *Universal gate* - a gate type that can implement any Boolean function.
 - Can implement any circuit using just NAND gates
- Likewise for NOR



5

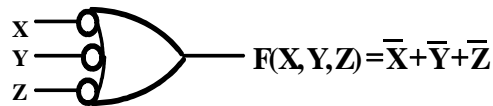
NAND Gate

- The basic NAND gate has the following symbol, illustrated for three inputs:
 - **AND-Invert (NAND)**
- 
- NAND represents NOT AND, i. e., the AND function with a NOT applied. The symbol shown is an AND-Invert.
 - The small circle (“**bubble**”) represents the *invert function*.

6

NAND Gates (continued)

- Applying DeMorgan's Law gives **Invert-OR (NAND)**



- This NAND symbol is called **Invert-OR**, since inputs are inverted and then ORed together.
- *AND-Invert* and *Invert-OR* both represent the NAND gate.
- Having both makes visualization of circuit function easier.
- A NAND gate with one input degenerates to an inverter.

7

NAND Gates (continued)

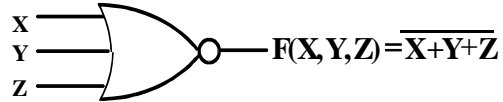
- The NAND gate is the **natural implementation for CMOS technology** in terms of **chip area and speed**.
- NAND **usually does not have a operation symbol** defined since
 - the **NAND operation is not associative**, and
 - we have difficulty dealing with non-associative mathematics!

8

NOR Gate

- The basic NOR gate has the following symbol, illustrated for three inputs:

- **OR-Invert (NOR)**

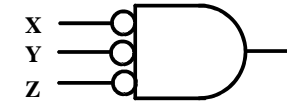


- NOR represents NOT - OR, i. e., the OR function with a NOT applied.
- The symbol shown is an OR-Invert.

9

NOR Gate (continued)

- Applying DeMorgan's Law gives Invert-AND (NOR)



- This NOR symbol is called Invert-AND, since inputs are inverted and then ANDed together.
- OR-Invert and Invert-AND both represent the NOR gate. Having both makes visualization of circuit function easier.
- A NOR gate with one input degenerates to an inverter.

10

NOR Gate (continued)

- The NOR gate is a natural implementation for some technologies other than CMOS in terms of chip area and speed.
- The NOR gate is also a universal gate
- **NOR usually does not have a defined operation symbol** since
 - the **NOR operation is not associative**, and
 - we have difficulty dealing with non-associative mathematics!

11

Technology Mapping

- Mapping Procedures
 - To NAND gates
 - To NOR gates

12

Mapping to NAND gates

▪ **Assumptions:**

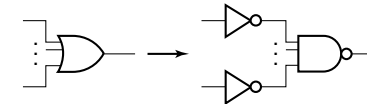
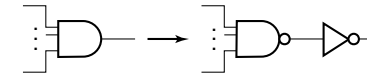
- Gate loading and delay are ignored
- Cell library contains an inverter and n -input NAND gates, $n = 2, 3, \dots$
- An AND, OR, inverter schematic for the circuit is available

▪ **The mapping is accomplished by:**

- Replacing AND and OR symbols,
- Pushing inverters through circuit fan-out points, and
- Canceling inverter pairs

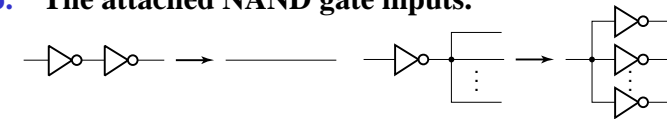
NAND Mapping Algorithm

1. **Replace ANDs and ORs:**

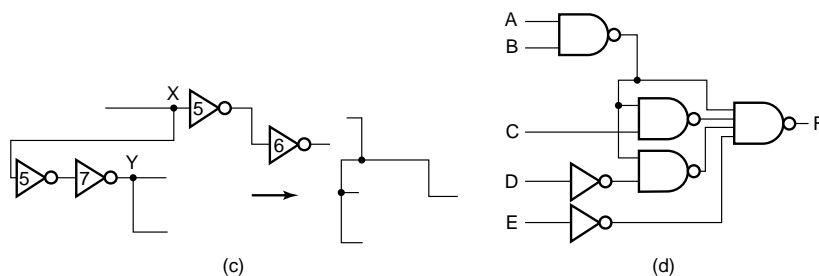
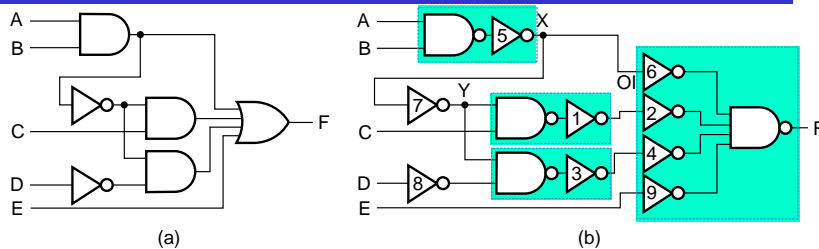


2. **Repeat the following pair of actions until there is at most one inverter between :**

- A circuit input or driving NAND gate output, and
- The attached NAND gate inputs.



NAND Mapping Example



Mapping to NOR gates

▪ **Assumptions:**

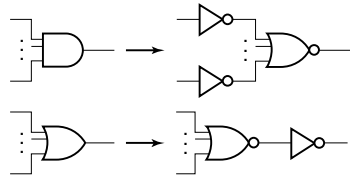
- Gate loading and delay are ignored
- Cell library contains an inverter and n -input NOR gates, $n = 2, 3, \dots$
- An AND, OR, inverter schematic for the circuit is available

▪ **The mapping is accomplished by:**

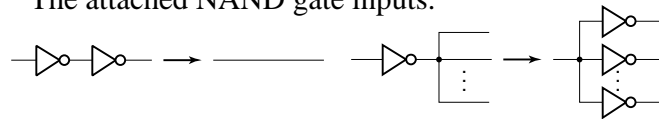
- Replacing AND and OR symbols,
- Pushing inverters through circuit fan-out points, and
- Canceling inverter pairs

NOR Mapping Algorithm

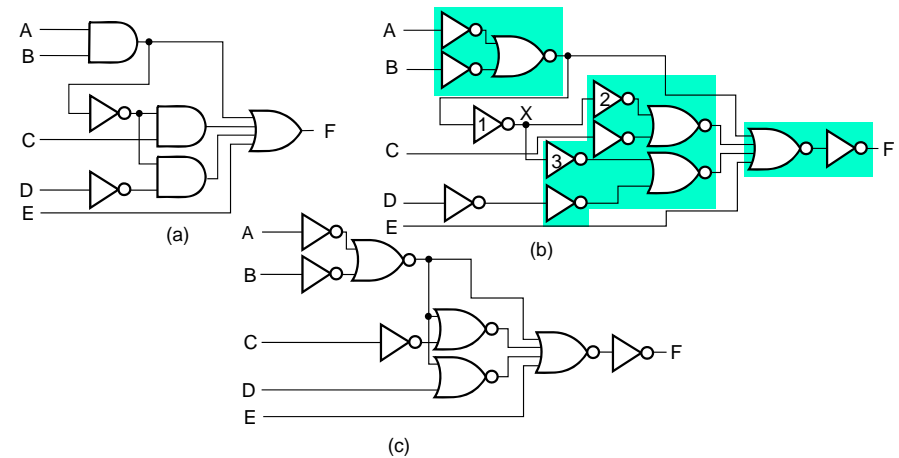
1. Replace ANDs and ORs:



2. Repeat the following pair of actions until there is at most one inverter between :
 - a. A circuit input or driving NAND gate output, and
 - b. The attached NAND gate inputs.



NOR Mapping Example



Exclusive OR/ Exclusive NOR

- The *eXclusive OR (XOR)* function is an important Boolean function used extensively in logic circuits.
- The XOR function may be;
 - implemented directly as an electronic circuit (truly a gate) or
 - implemented by interconnecting other gate types (used as a convenient representation)
- The *eXclusive NOR* function is the complement of the XOR function
- By our definition, XOR and XNOR gates are complex gates.

Exclusive OR/ Exclusive NOR

- Uses for the XOR and XNORs gate include:
 - Adders/subtractors/multipliers
 - Counters/incrementers/decrementers
 - Parity generators/checkers
- Definitions
 - The XOR function is: $X \oplus Y = X\bar{Y} + \bar{X}Y$
 - The *eXclusive NOR (XNOR)* function, otherwise known as *equivalence* is: $\bar{X \oplus Y} = XY + \bar{X}\bar{Y}$
- Strictly speaking, XOR and XNOR gates do not exist for more than two inputs. Instead, they are replaced by odd and even functions.

Truth Tables for XOR/XNOR

- Operator Rules: XOR

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

- XNOR

X	Y	$\overline{(X \oplus Y)}$ or $X \equiv Y$
0	0	1
0	1	0
1	0	0
1	1	1

- The XOR function means:
X OR Y, but NOT BOTH
- Why is the XNOR function also known as the *equivalence* function, denoted by the operator \equiv ?

21

XOR/XNOR (Continued)

- The XOR function can be extended to 3 or more variables. For more than 2 variables, it is called an *odd function* or *modulo 2 sum* (Mod 2 sum), not an XOR:

$$X \oplus Y \oplus Z = \overline{X} \overline{Y} Z + \overline{X} Y \overline{Z} + X \overline{Y} \overline{Z} + X Y Z$$

- The complement of the odd function is the even function.

- The XOR identities:

$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$

$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$

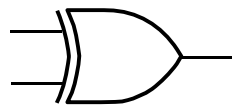
$$X \oplus Y = Y \oplus X$$

$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

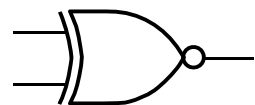
22

Symbols For XOR and XNOR

- XOR symbol:



- XNOR symbol:

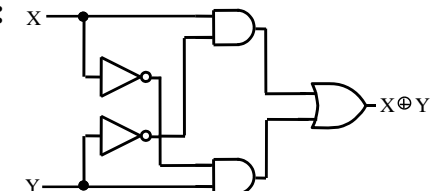


- Shaped symbols exist only for two inputs

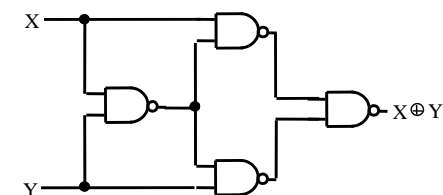
23

XOR Implementations

- The simple SOP implementation uses the following structure:



- A NAND only implementation is:



24

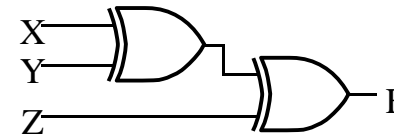
Odd and Even Functions

- The odd and even functions on a K-map form “checkerboard” patterns.
- The 1s of an odd function correspond to minterms having an index with an odd number of 1s.
- The 1s of an even function correspond to minterms having an index with an even number of 1s.
- Implementation of odd and even functions for greater than four variables as a two-level circuit is difficult, so we use “trees” made up of :
 - 2-input XOR or XNORs
 - 3- or 4-input odd or even functions

25

Example: Odd Function Implementation

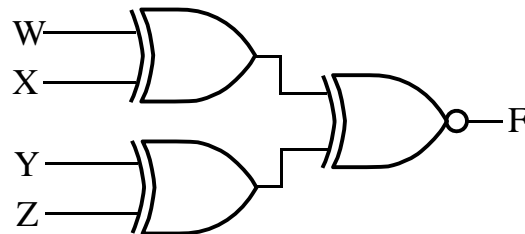
- **Design a 3-input odd function $F = X \oplus Y \oplus Z$ with 2-input XOR gates**
- **Factoring, $F = (X \oplus Y) \oplus Z$**
- **The circuit:**



26

Example: Even Function Implementation

- **Design a 4-input odd function $F = W \oplus X \oplus Y \oplus Z$ with 2-input XOR and XNOR gates**
- **Factoring, $F = (W \oplus X) \oplus (Y \oplus Z)$**
- **The circuit:**



27

Parity Generators and Checkers

- A *parity* bit added to n-bit code to produce an n + 1 bit code:
 - Add odd parity bit **to generate** code words with even parity
 - Add even parity bit to generate code words with odd parity

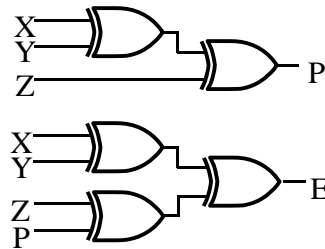
- Use odd parity circuit **to check** code words with even parity
- Use even parity circuit to check code words with odd parity

28

Example: Parity Generator and Checker

- $n = 3$. Generate even parity code words of length 4 with odd parity generator:
- Check **even parity** code words of length 4, with **odd parity checker**:
- Operation: $(X, Y, Z) = (0, 0, 1)$ gives
 $(X, Y, Z, P) = (0, 0, 1, 1)$ and $E = 0$.

If Y changes from 0 to 1 between generator and checker,
 then $E = 1$ indicates an error.



29

Number of Possible Boolean Functions

- How many possible functions of 2 variables?
 - 2^2 rows in truth table, 2 choices for each
 - $2^{(2^2)} = 2^4 = 16$ possible functions
- N variables
 - 2^N rows
 - $2^{(2^N)}$ possible functions

a	b	F
0	0	0 or 1 2 choices
0	1	0 or 1 2 choices
1	0	0 or 1 2 choices
1	1	0 or 1 2 choices

$2^4 = 16$
possible functions

a	b	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	a AND b		a		b		a XOR b	a OR b	a NOR b	a XNOR b	b'		a'		a NAND b
																	1

30