Computer Science Department

CS504

Digital Logic & Computer Organization

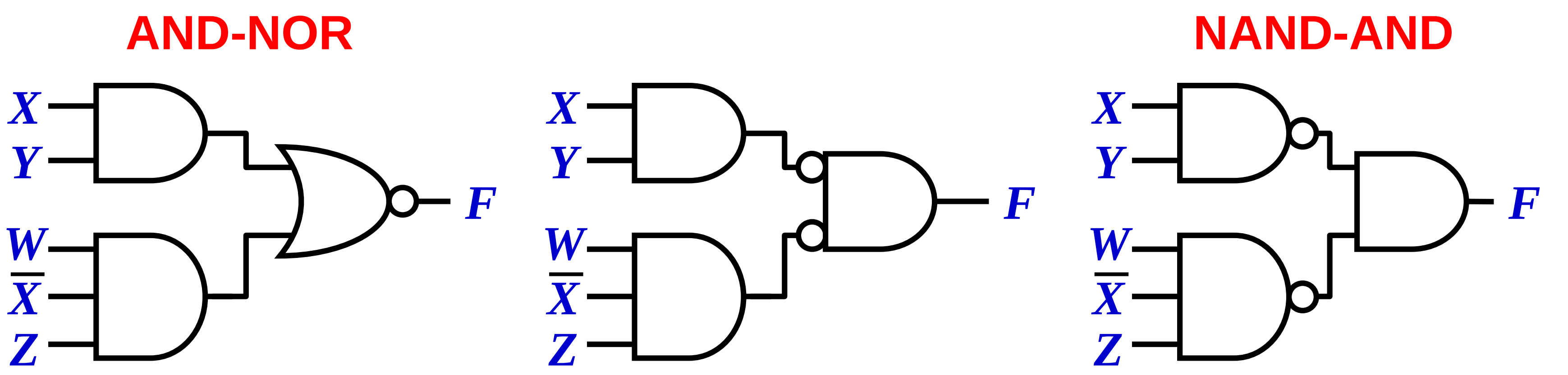
Lecture 6

Lecture Outline (Chapter 3)

- ★ **Other Types of 2-Level Circuits (Section 3.7)**
- ★ **Exclusive OR / Exclusive NOR (Section 3.8)**
  - **XOR / XNOR Tables and Symbols**
  - **Uses for XOR / XNOR**
  - **XOR Implementations**
  - **XOR / XNOR Identities**
  - **Odd Function**
  - **Odd/Even Functions**
  - **Odd/Even Function Implementation**
  - **Parity Generators And Checkers**

Other Types of 2-Level Circuits

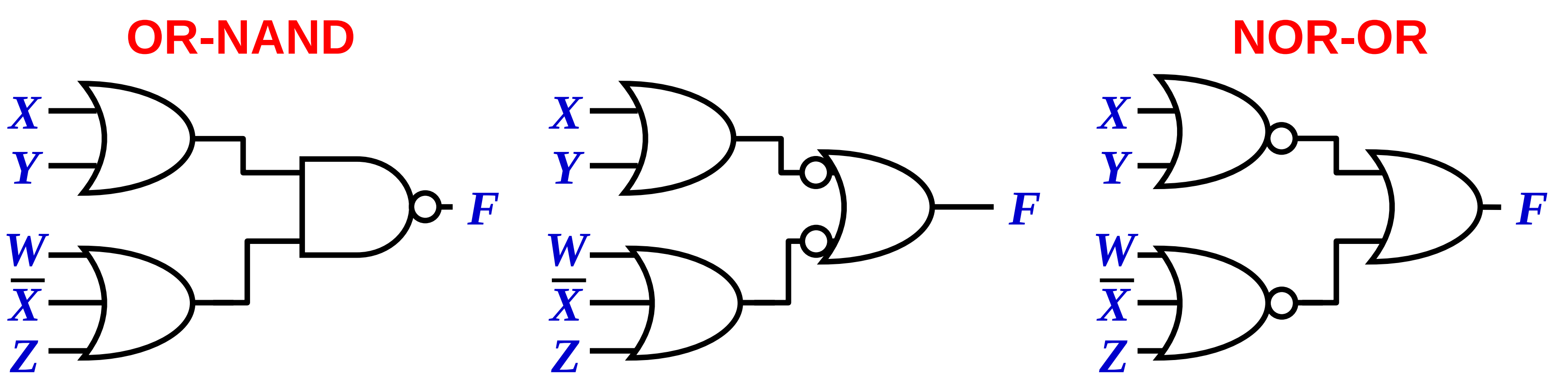
- Other useful types of 2-level circuits:
  - AND-NOR = AND-OR-INVERT = NAND-AND
  - OR-NAND = OR-AND-INVERT = NOR-OR
- AND-NOR Function: $F = \overline{XY + W\overline{X}Z}$



- AND-NOR circuits can be converted to NAND-AND

Other Types of 2-Level Circuits (2)

- Other useful types of 2-level circuits:
  - AND-NOR = AND-OR-INVERT = NAND-AND
  - OR-NAND = OR-AND-INVERT = NOR-OR
- OR-NAND Function: $F = \overline{(X+Y)(W+\overline{X}+Z)}$



- OR-NAND circuits can be converted to NOR-OR

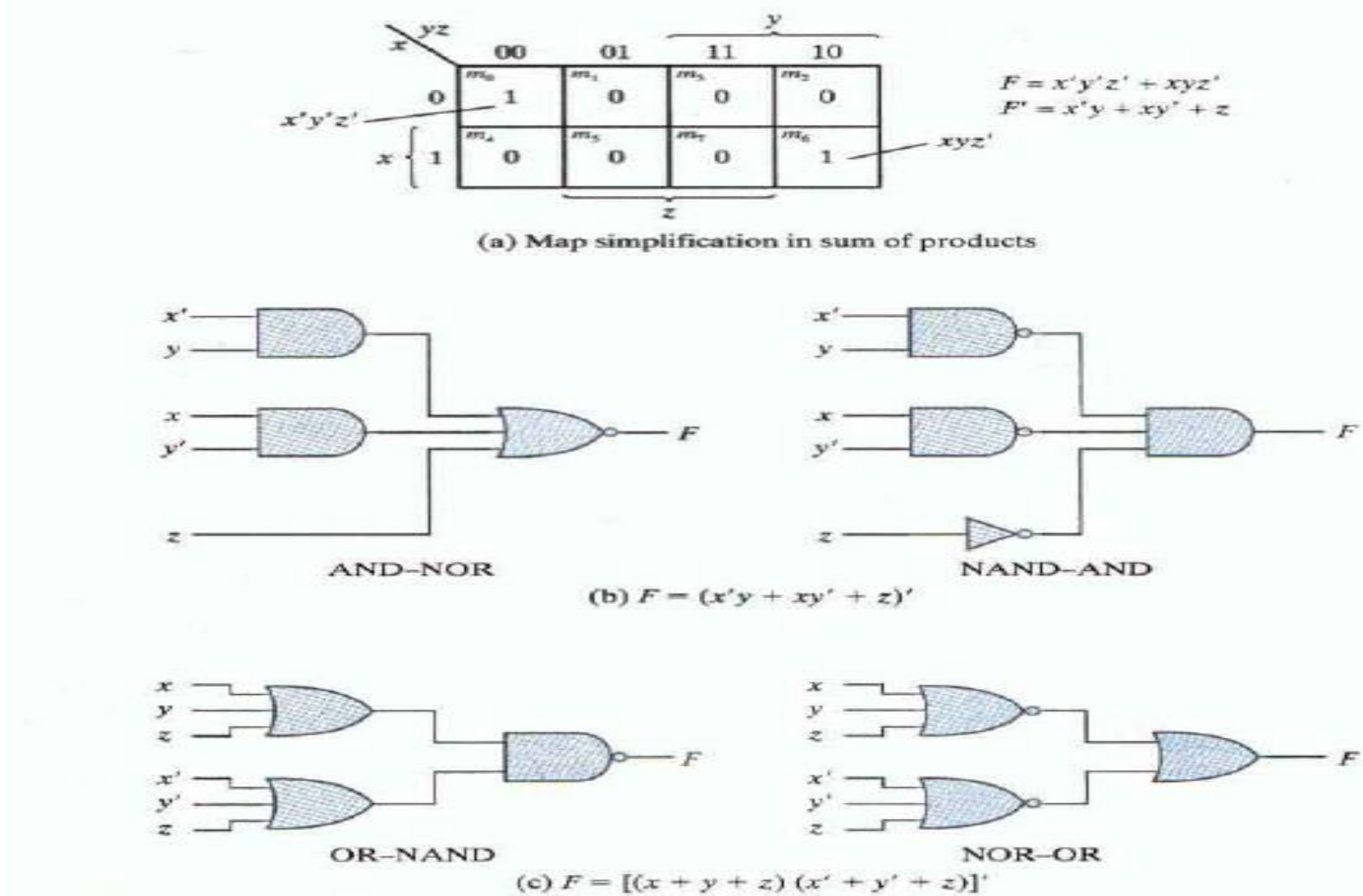
Other Types of 2-Level Circuits (3)

Implementation with Other Two-Level Forms

Equivalent Nondegenerate Form (a)	Equivalent Nondegenerate Form (b)*	Implements the Function	Simplify F' into	To Get an Output of
AND-NOR	NAND-AND	AND-OR-INVERT	Sum-of-products form by combining 0's in the map.	F
OR-NAND	NOR-OR	OR-AND-INVERT	Product-of-sums form by combining 1's in the map and then complementing.	F

*Form (b) requires an inverter for a single literal term.

Other Types of 2-Level Circuits (4)



(a) Map simplification in sum of products

(b) $F = (x'y + xy' + z)'$

(c) $F = [(x + y + z)(x' + y' + z)]'$

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 (true gate)
  - Implemented by interconnecting other gate types (XOR is used as a convenient representation)
- The **eXclusive-NOR (XNOR)** function is the complement of the XOR function
- XOR and XNOR gates are complex gates

XOR / XNOR Tables and Symbols

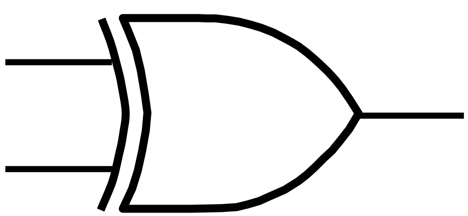
XOR

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

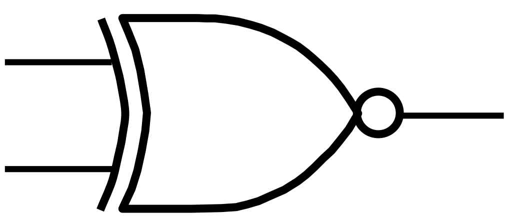
XNOR

X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

XOR Symbol



XNOR Symbol



- The XNOR is also denoted as **equivalence**

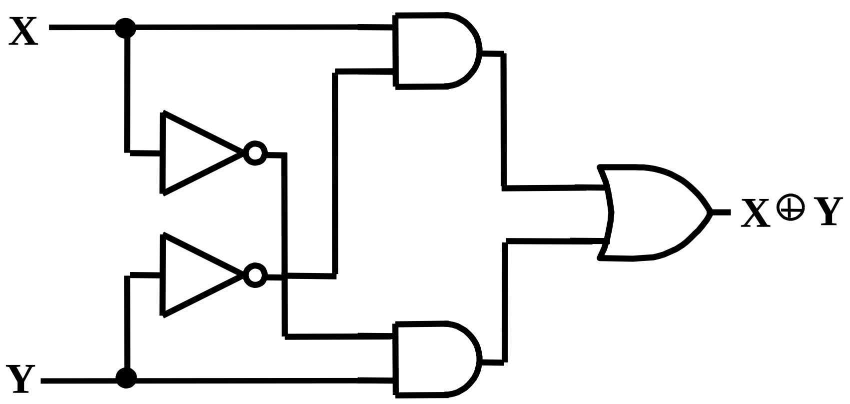
Uses for XOR / XNOR

- SOP Expressions for XOR/XNOR:
  - The XOR function is: $\mathbf{X \oplus Y = X\overline{Y} + \overline{X}Y}$
  - The eXclusive NOR (XNOR) function, know also as **equivalence** is: $\mathbf{\overline{X \oplus Y} = X\,Y + \overline{X}\,\overline{Y}}$
- Uses for the XOR and XNORs gate include:
  - Adders/subtractors/multipliers
  - Counters/incrementers/decrementers
  - Parity generators/checkers
- Strictly speaking, XOR and XNOR gates do no exist for more than two inputs. Instead, they are replaced by **odd** and **even** functions.

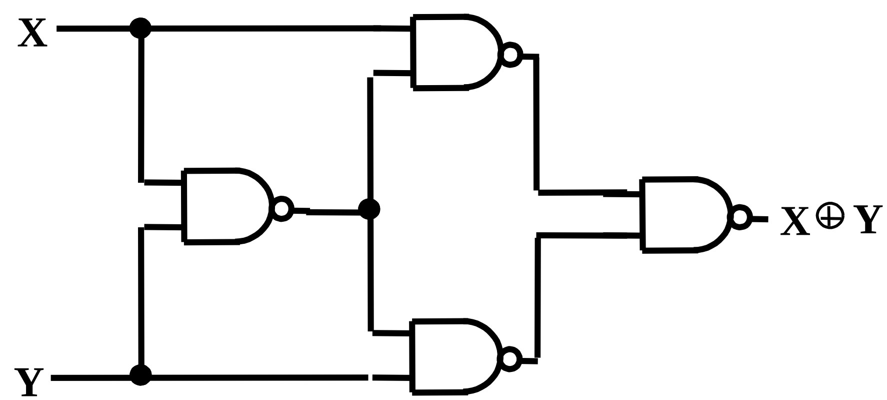
XOR Implementations

SOP implementation for XOR:

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



NAND only implementation for XOR:



XOR / XNOR Identities

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

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

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

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

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

$$\overline{\overline{(X \oplus Y)} \oplus Z} = \overline{X \oplus \overline{(Y \oplus Z)}} = X \oplus Y \oplus Z$$

- XOR and XNOR are **associative** operations

Odd Function

- The XOR function can be extended to 3 or more variables
- For 3 or more variables, XOR is called an **odd function**
  - The function is 1 if the total number of 1's in the inputs is **odd**

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

X \ YZ	00	01	11	10
0		1		1
1	1		1	

$X \oplus Y \oplus Z$

WX \ YZ	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$W \oplus X \oplus Y \oplus Z$

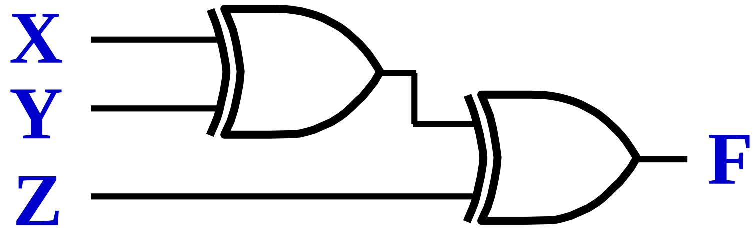
Odd/Even Functions

- The 1s of an **odd function** correspond to inputs with an **odd number of 1s**
- The **complement** of an odd function is called an **even function**
- The 1s of an **even function** correspond to inputs with an **even number of 1s**
- Implementation of odd or even functions use trees made up of 2-input XOR or XNOR gates respectively

Odd/Even Function Implementation

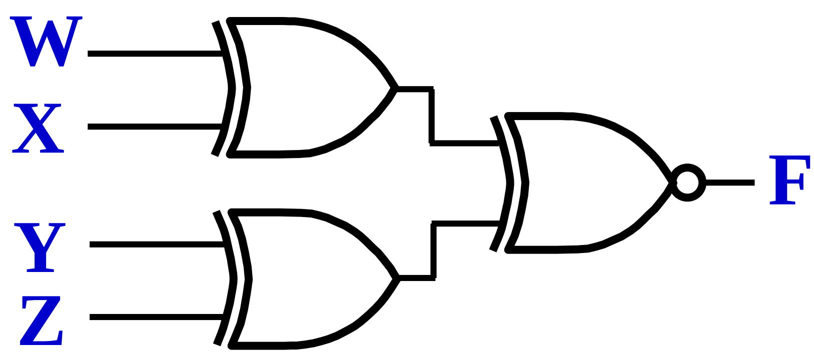
- Design a 3-input odd function with 2-input XOR:
- 3-input odd function:

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



- Design a 4-input even function with 2-input XOR and XNOR gates:
- 4-input even function:

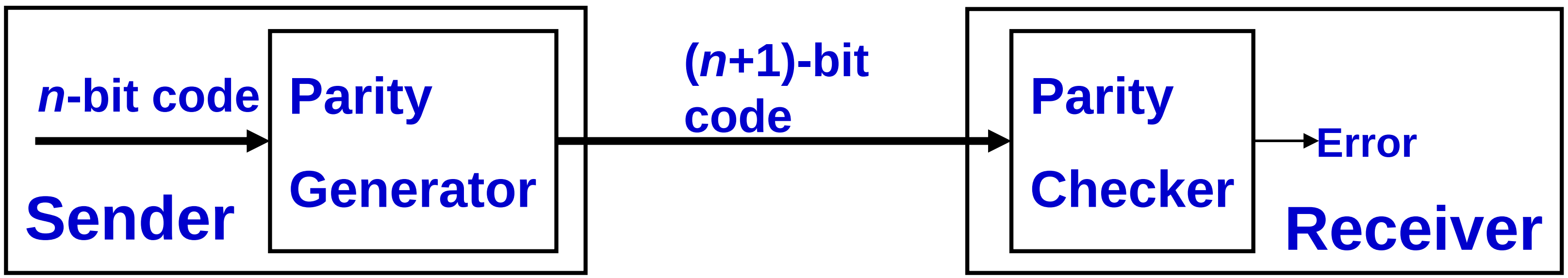
$F = \overline{(W \oplus X) \oplus (Y \oplus Z)}$



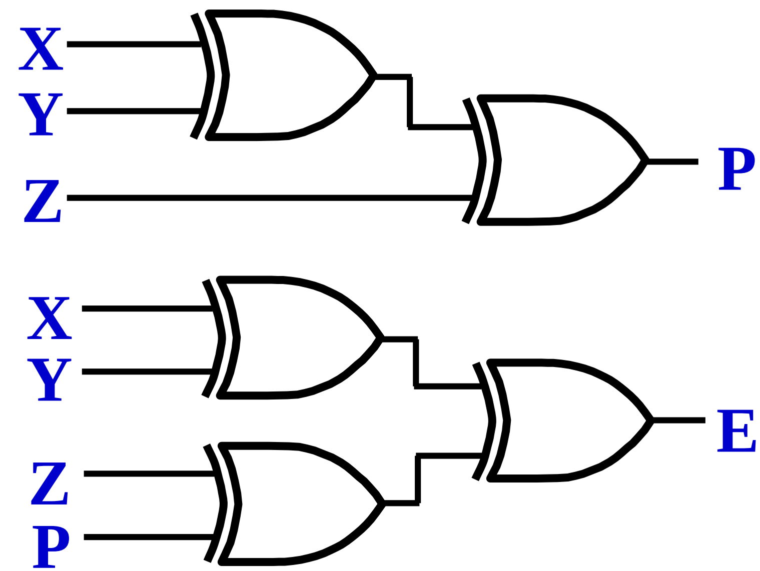
Parity Generators And Checkers

- A parity bit added to *n*-bit code produces (*n*+1)-bit code with an odd (or even) count of 1s
- Odd Parity bit: count of 1s in (*n*+1)-bit code is odd
  - So use an **even function** to generate the **odd parity bit**
- Even Parity bit: count of 1s in (*n*+1)-bit code is even
  - So use an **odd function** to generate the **even parity bit**
- To check for odd parity
  - Use an **even function** to check the (*n*+1)-bit code
- To check for even parity
  - Use an **odd function** to check the (*n*+1)-bit code

Parity Generator And Checker



- Design an even parity generator and checker for 3-bit codes
- Solution: Use 3-bit odd function to generate even parity bit
- Use 4-bit odd function to check for errors in even parity codes
- 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



Parity Generator And Checker (2)

Table 3.4
Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	*y*	*z*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Parity Generator And Checker (3)

Table 3.5
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

The End

Questions?