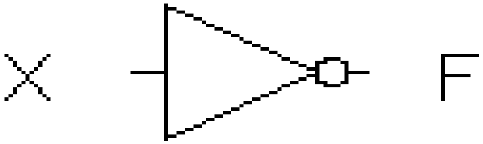
Section 2

Combinational Logic Design
Chapter 2

Basic gates

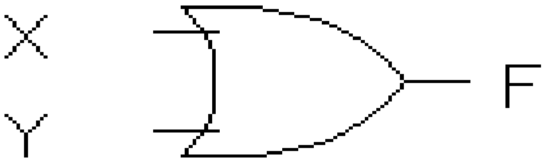
- NOT Gate (Inverter)



X	F
0	1
1	0

Basic Gates

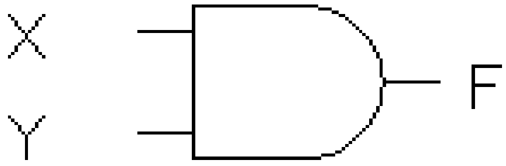
- OR Gate



X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Basic Gates

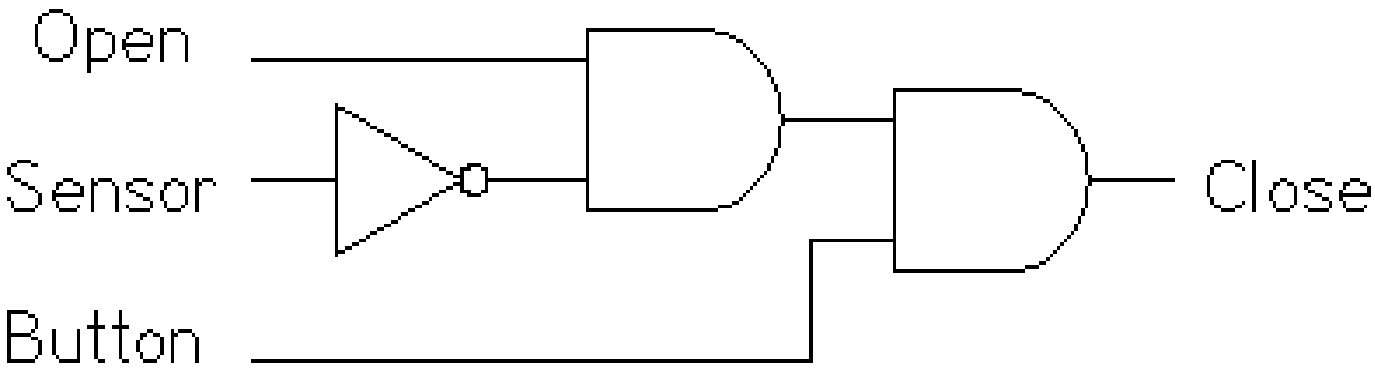
- AND Gate



X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

A Simple Circuit

- Design a garage close circuit that activates the garage door motor when the door is open, the object sensor is off, and Close button is pressed.



Boolean Algebra

Symbol	Name	Description	Priority
( )	Parenthesis	Evaluate expression inside parenthesis first	Highest
'	NOT	Evaluate left to right	
*	AND	Evaluate left to right	
+	OR	Evaluate left to right	Lowest

Boolean Algebra (Terms)

- Variable – A term that represents a value
- Literal – The usage of a variable in a formula
- Sum-Of-Products (SOP)

$$F = XYZ + X'Y'Z + XY'Z'$$

- Product-Of-Sums (POS)

$$G = (X + Y + Z)(X' + Y' + Z)(X + Y' + Z')$$

Boolean Algebra (Axioms)

(A1) X = 0 if X != 1

(A1') X = 1 if X != 0

(A2) If X = 0, then X' = 1

(A2') if X = 1, then X' = 0

(A3) 0 * 0 = 0

(A3') 1 + 1 = 1

(A4) 1 * 1 = 1

(A4') 0 + 0 = 0

(A5) 0 * 1 = 1 * 0 = 0

(A5') 1 + 0 = 0 + 1 = 1

Boolean Algebra (Postulates)

- Commutative

$$A + B = B + A$$

$$A * B = B * A$$

- Distributive

$$A * (B + C) = A * B + A * C$$

$$A + (B * C) = (A + B) * (A + C)$$

Pay attention to the second distributive

Boolean Algebra (Postulates)

- Associative

$$(A + B) + C = A + (B + C)$$

$$(A * B) * C = A * (B * C)$$

- Identity

$$0 + A = A + 0 = A$$

$$1 * A = A * 1 = A$$

Boolean Algebra (Postulates)

- Complement

$$A + A' = 1$$

$$A * A' = 0$$

Boolean Algebra (Theorems)

- Null Elements

$$A + 1 = 1$$

$$A * 0 = 0$$

- Idempotent Law

$$A + A = A$$

$$A * A = A$$

Boolean Algebra (Theorems)

- Involution Law

$$(A')' = A$$

- DeMorgan's Law

$$(A + B)' = A'B'$$

$$(AB)' = A' + B'$$

DeMorgan's can be extended to more variables

Boolean Algebra (Theorems)

- Covering

$$X + X * Y = X$$

$$X * (X + Y) = X$$

- Combining

$$X * Y + X * Y' = X$$

$$(X + Y) * (X + Y') = X$$

Boolean Algebra (Theorems)

- Consensus

$$X * Y + X' * Z + Y * Z = X * Y + X' * Z$$

$$(X + Y) * (X' + Z) * (Y + Z) = (X + Y) * (X' + Z)$$

Boolean Algebra

- Prove the following using algebraic manipulation

$X'Y' + X'Y + XY = X' + Y$

$A'B + B'C' + AB + B'C = 1$

$Y + X'Z + XY' = X + Y + Z$

$X'Y' + Y'Z + XZ + XY + YZ' = X'Y' + XZ + YZ'$

Boolean Algebra

- Prove the following using algebraic manipulation

$AB + BC'D' + A'BC + C'D = B + C'D$

Boolean Algebra

Given that A * B = 0 and A + B = 1, use algebraic manipulation to prove that

$$(A + C) * (A' + B) * (B + C) = B * C$$

Principle of Duality

- Any theorem or identity in Boolean Algebra remains true if 0 and 1 are swapped and the AND and OR operations are swapped throughout

Truth Tables

- A truth table may be expressed by many different equations.
- Prove two functions are equal by induction.
- Optimizing a function usually requires creating a truth table.

Standard Forms

A	B	C	MinTerm		MaxTerm	
0	0	0	A'B'C'	m0	A+B+C	M0
0	0	1	A'B'C	m1	A+B+C'	M1
0	1	0	A'BC'	m2	A+B'+C	M2
0	1	1	A'BC	m3	A+B'+C'	M3
1	0	0	AB'C'	m4	A'+B+C	M4
1	0	1	AB'C	m5	A'+B+C'	M5
1	1	0	ABC'	m6	A'+B'+C	M6
1	1	1	ABC	m7	A'+B'+C'	M7

Standard Forms

- Sum of Minterms
  - Product terms
  - Each term contains each variable.
  - A term is one line or element on a truth table.
  - For each line in a truth table that is 1, that term is part of the final equation.
  - Write the Sum of Minterms for table 2.3 on page 67.
  - Can be written as $\Sigma m(mx, my, ...)$

Standard Forms

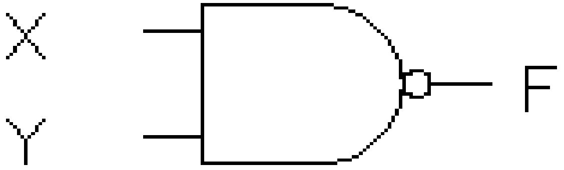
- Product of Maxterms
  - Sum terms.
  - Each term contains each variable.
  - A term is one line or element on a truth table.
  - For each line in a truth table that is 0, that term is part of the final equation.
  - Write the Product of Maxterms for table 2.3 on page 67.
  - Can be written as $\prod M(Mx, My, ...)$

Combinational Logic Design Process

- Create a truth table.
- Write optimized equations. (We still need to cover optimization in 6.2).
- Draw schematic or create hardware description from optimized equations.

Additional Gates

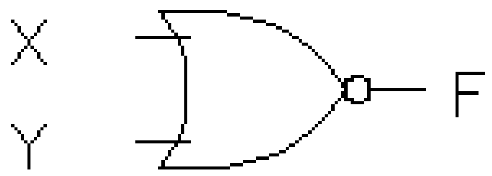
- NAND gate



X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

Additional Gates

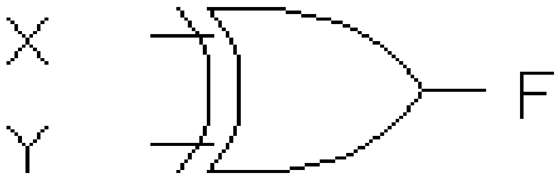
- NOR gate



X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

Additional gates

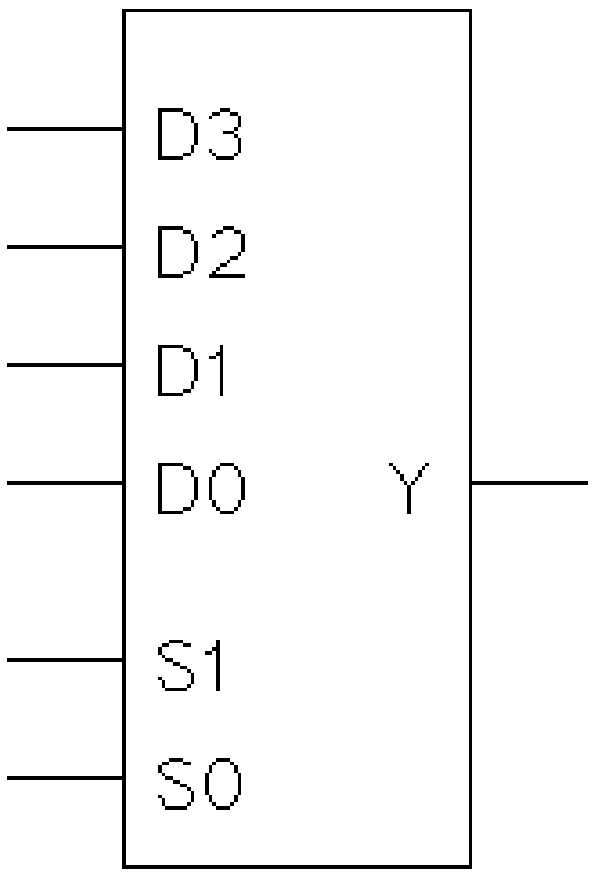
- XOR gate (odd function)



X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Building Blocks

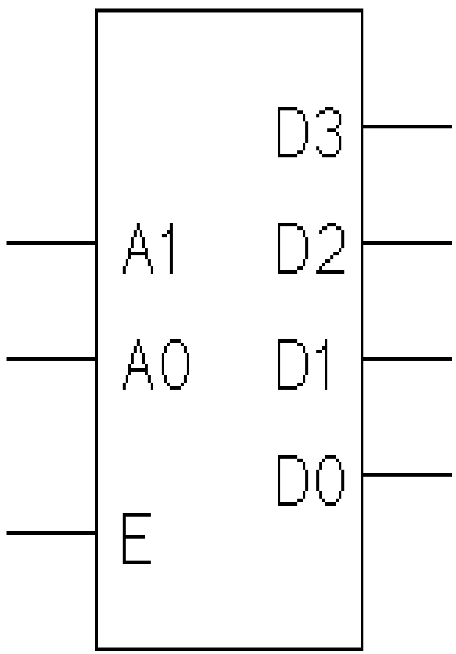
- Multiplexer



S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Building Blocks

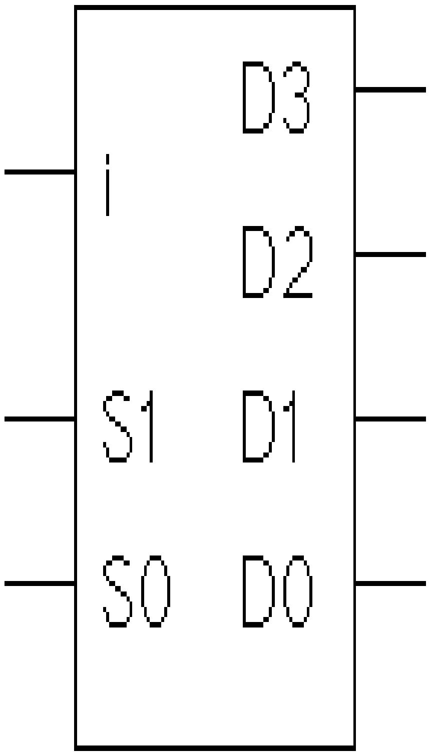
- Decoder



E	A1	A0	D3	D2	D1	D0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Building Blocks

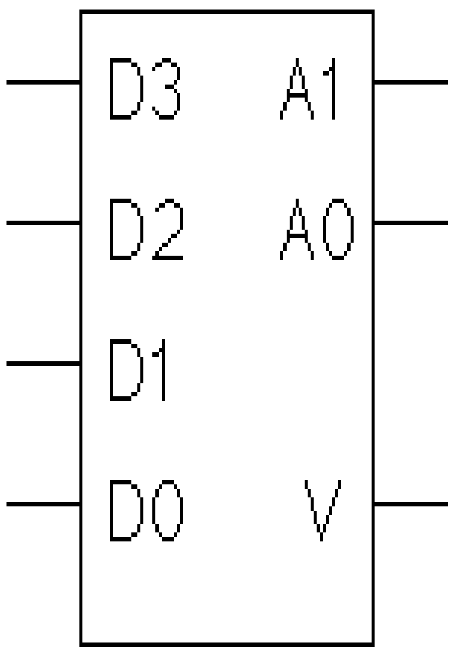
- Demultiplexer



S1	S0	D3	D2	D1	D0
0	0	0	0	0	i
0	1	0	0	i	0
1	0	0	i	0	0
1	1	i	0	0	0

Building Blocks

- Priority Encoder



D3	D2	D1	D0	V	A1	A0
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	X	1	0	1
0	1	X	X	1	1	0
1	X	X	X	1	1	1

Timing

- Real logic gates take time to react to an input change.
- The delay is called the propagation delay.
- A complicated circuit may have “glitches” in its output signals when the inputs change from one state to another.
- Reducing “glitches” often requires extra logic gates.