EE1D1: Digital Systems A

BSc. EE, year 1, 2025-2026, lecture 2

Boolean Circuits

Computer Engineering Lab

Faculty of Electrical Engineering, Mathematics & Computer Science

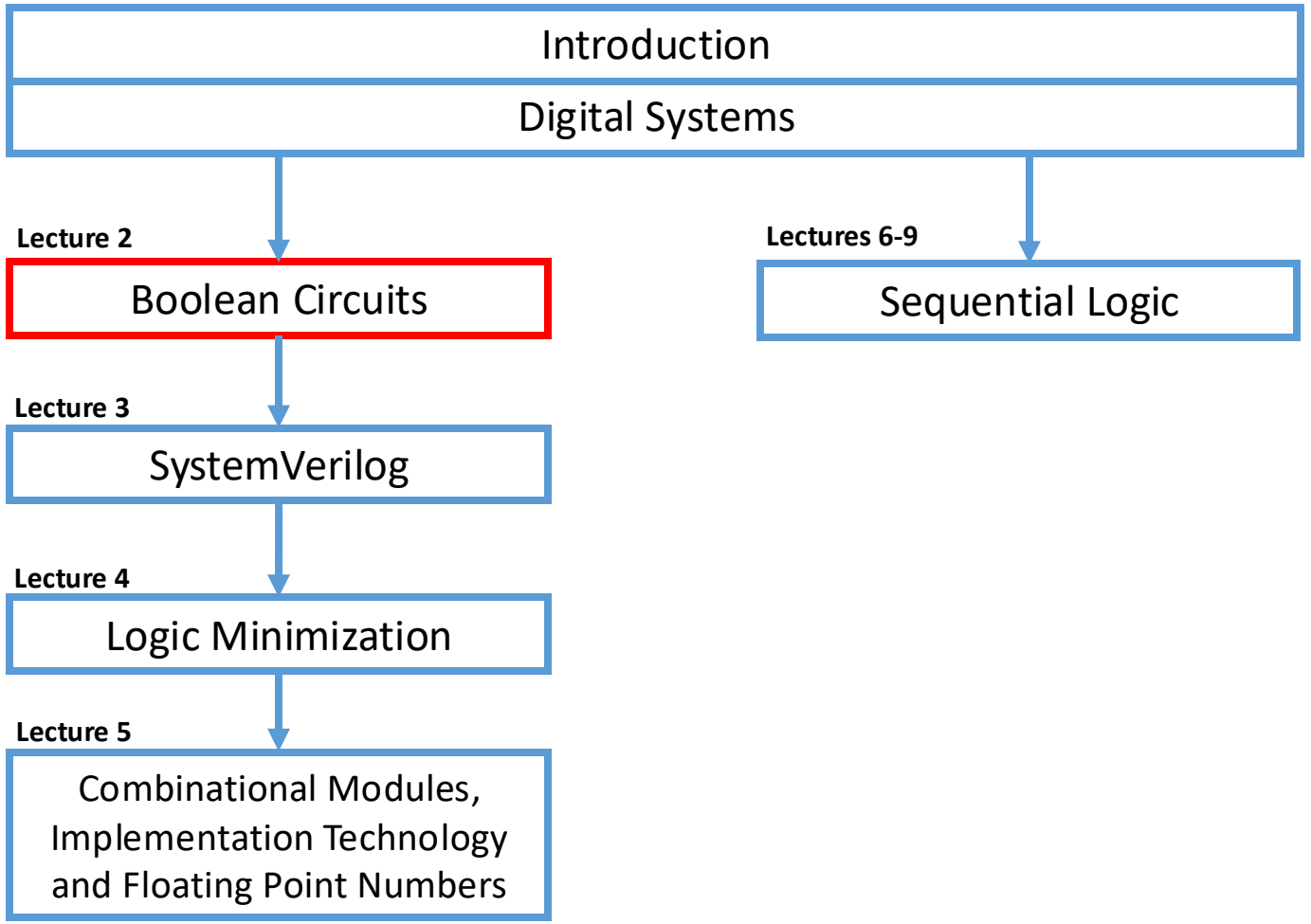
TU Delft

Delft University of Technology

Recap

- Introduction to Digital Systems
  - Hardware
  - Software
- Digital versus Analog
  - Digital needs more resources
  - Digital requires less accuracy
- Binary Systems and Boolean Algebra
  - Logic/Boolean operations
- Data Representation
  - Positive/negative numbers
  - Binary, decimal, hexadecimal
  - Fixed point numbers

Recap


Introduction
Digital Systems
Lecture 2
Boolean Circuits
Lecture 3
SystemVerilog
Lecture 4
Logic Minimization
Lecture 5
Combinational Modules, Implementation Technology and Floating Point Numbers
Lectures 6-9
Sequential Logic

Recap

Week	Lecture 1 (Mo)	Lecture 2 (Tue)	Assignments	Mock-Up/Exam
1.1	Intro	Boolean Circuits	GP-lec1, GP-lec2	
1.2	SystemVerilog	Logic minim.	GP-lec3, GP-lec4	
1.3	Combinational Modules		GP-lec5 Course Lab Part 1	
1.4			Course Lab Part 2	Mock Exam (Tuesday) Discuss Mock Exam (Friday)
1.5				Partial Exam 1 (Friday)

Week	Date	Lecture	Topics	Material
1.1	04/09	Lec 1	Introduction to Digital System	Sections 1 – 1.5.4
	05/09	Lec 2	Boolean Circuits	Sections 2.1, 2.3 (not yet 2.3.5) and 2.4
1.2	11/09	Lec 3	SystemVerilog	Sections 4.1, 4.2 (not 4.2.3, 4.2.6, 4.2.9), 4.3 and 4.9
	12/09	Lec 4	Logic Minimization	Sections 2.2, 2.3.5, 2.5, 2.7 and 2.9
1.3	18/09	Lec 5	Combinational Modules, Implementation Technology and Floating Point Numbers	Sections 1.6, 1.7.4-1.7.7, 2.6.2, 2.8, 5.3.1, and 5.3.2 (no rounding and addition)

Outline

System Descriptions

- Switches, truth table, gates, timing diagram, ...

System Types

- Combinatorial systems and Sequential systems

Boolean Algebra

- Laws and theorems
- Simplification, mapping to a circuit

Summary

Learning Objectives

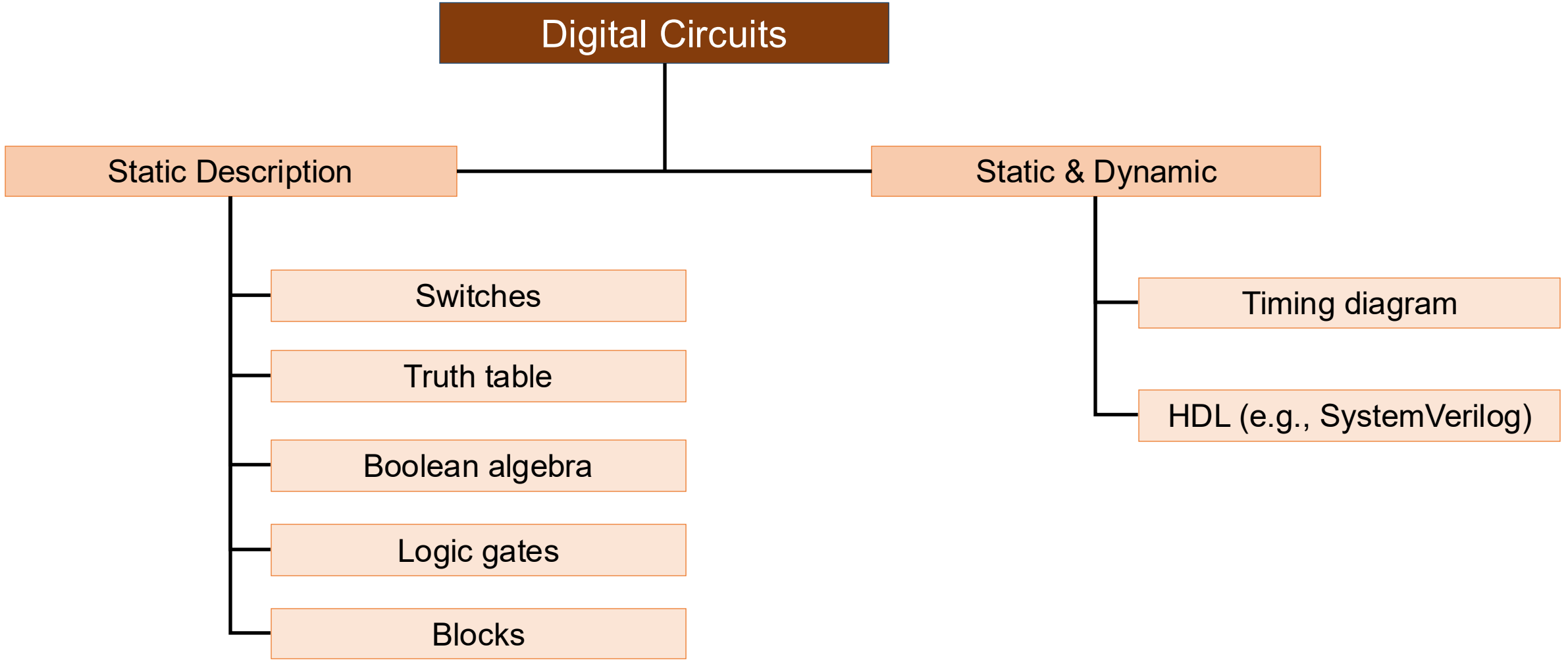
As student you should be able to:

- describe digital systems using different formats (e.g. truth table, switches, etc.) and convert them from one format to the other
- explain how combinational and sequential circuits work at a high abstraction level
- recognize the symbols of the Boolean gates and draw Boolean circuits from expressions
- manipulate Boolean expressions using Boolean theorems and DeMorgan's law
- convert Boolean circuits to NAND, NOR, NOT equivalents

EE1D1: Digital Systems A

System Descriptions

System Descriptions

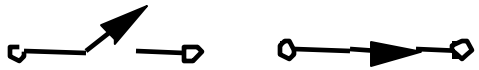


System Description — Static

- Switches

Types:

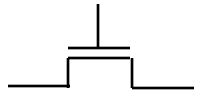
Normally Open:



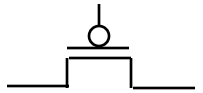
Control: False True

Normally Closed:

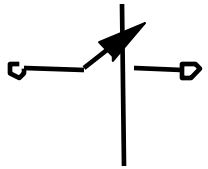
Control: False True



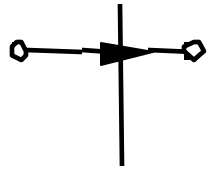
nmos transistor



pmos transistor



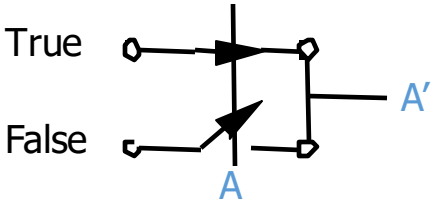
True: close
False: open



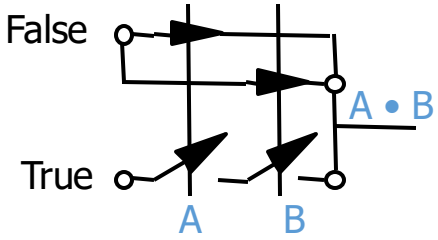
True: open
False: closed

Implementation NOT, AND, OR:

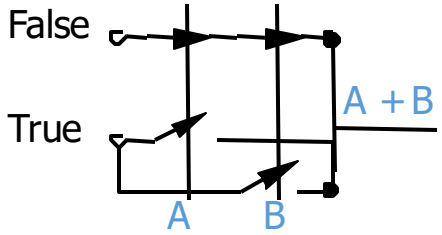
NOT:



AND:

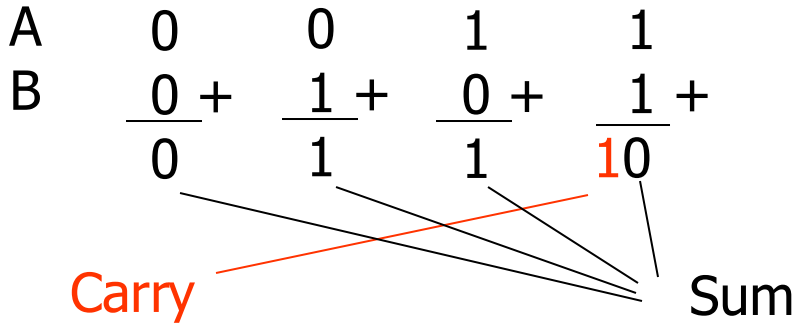


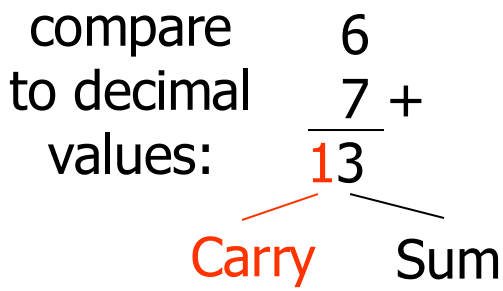
OR:



System Description — Static

- Truth Tables
  - List the output values for all possible input value combinations
  - *Example:* Half Adder (add 2 binary values A and B)





Truth table Half Adder:

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

System Description — Static

- Boolean algebra:
  - *Values:* 0 (false), 1 (true)
  - *Operations:* NOT, AND, OR, …

NOT X ≡ X′ X AND Y ≡ X • Y ≡ X Y X OR Y ≡ X + Y

- Derivation of Boolean description for Half Adder:

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Sum = A′ B + A B′

Carry = A B

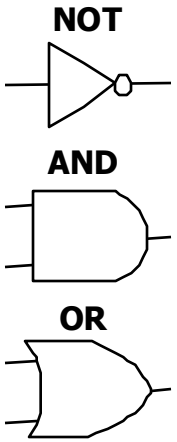
Note:

A	B	A•B	A′	B	A′•B
0	0	0	1	0	0
0	1	0	1	1	1
1	0	0	0	0	0
1	1	1	0	1	0

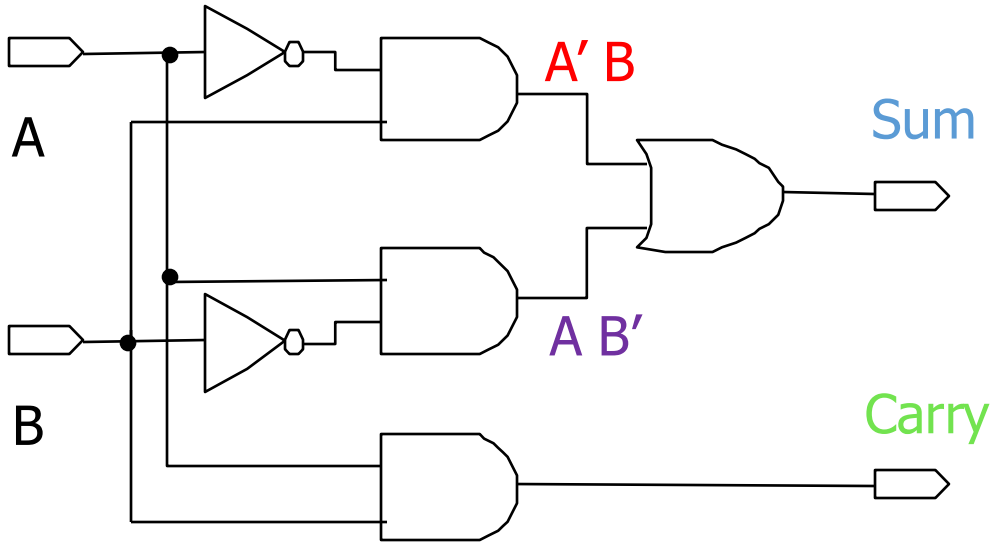
System Description — Static

- Logic gates:
  - building blocks for which an electronic implementation exists

Basic logic gates:



Circuit for a Half Adder:



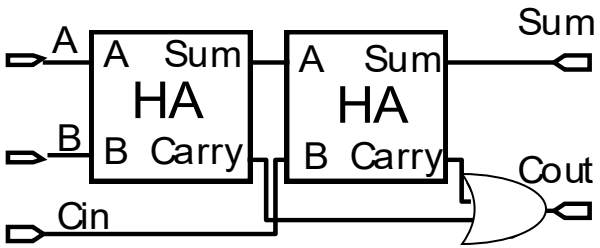
Sum = A′ B + A B′

Carry = A B

System Description — Static

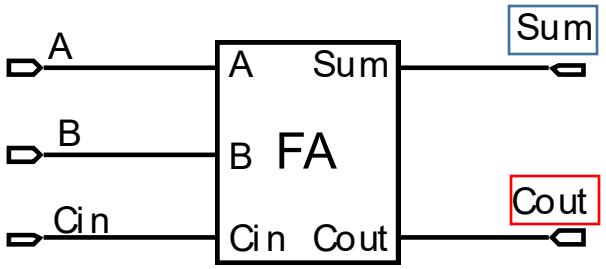
- Blocks
  - emphasis on *structure* of the system instead of *behaviour*
  - possibility for hierarchical design

Full Adder made of
2 Half Adder blocks:



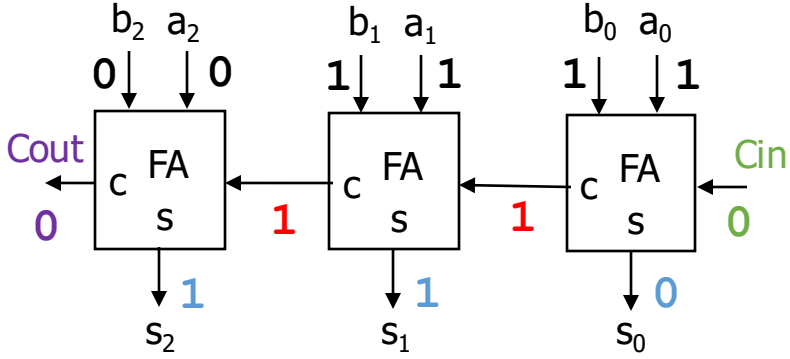
B.T.W. a Full Adder adds 3 bits: A, B en Cin

Block representation
for the Full Adder



A	0	1	1	1
B	0	0	1	1
Cin	0+	0+	0+	1+
	00	01	10	11

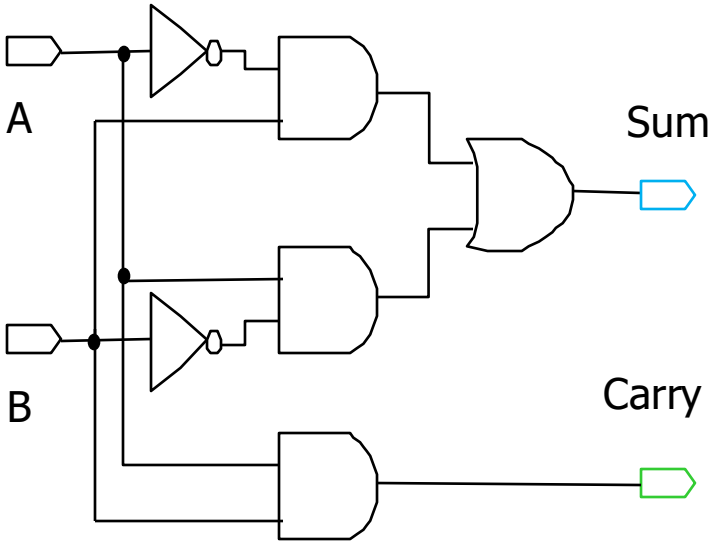
3-bit Full Adder
Computes s = a+b, where
$a=a_2a_1a_0$, $b=b_2b_1b_0$

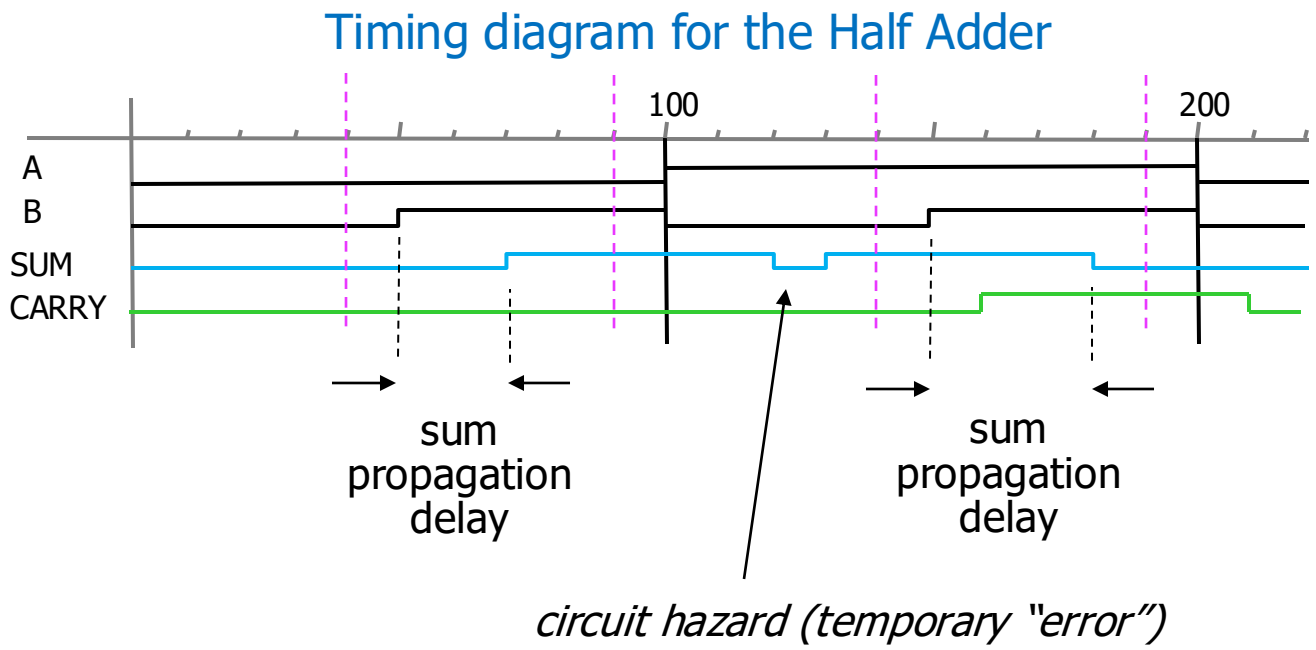



```
0110  (C)
 011  (A)
 011  (B)+
-----------
 110  (S)
```

System Description – Dynamic

- Using Timing Diagrams
  - Contains structural information and dynamic behaviour (delay/time!)





System Description - Overview

- Different Half Adder Descriptions
  - Static:

Truth Table

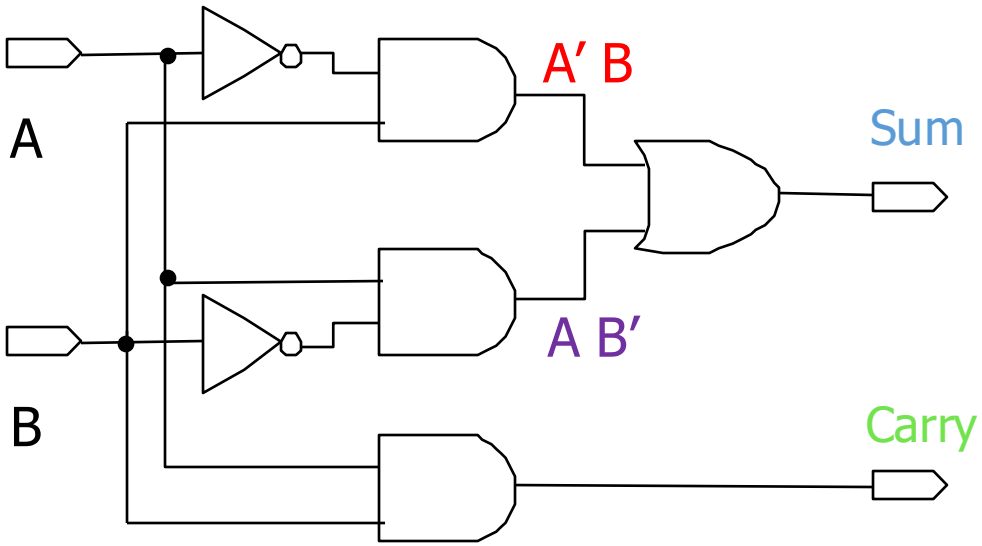
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Boolean Algebra

Sum = A' B + A B'

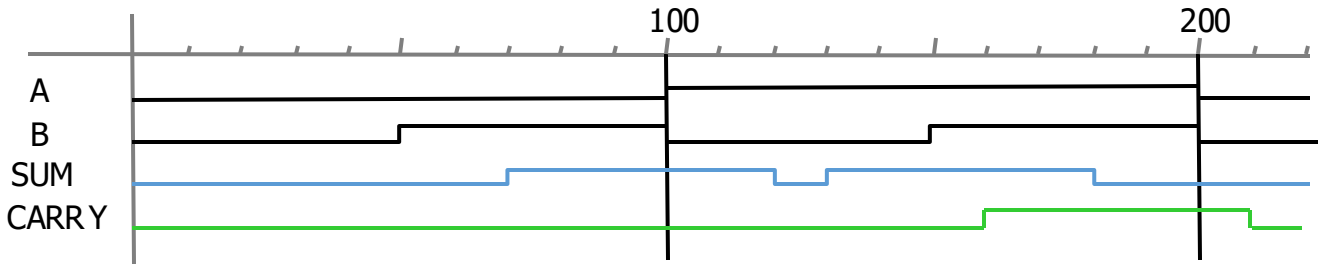
Carry = A B

Logic Gates

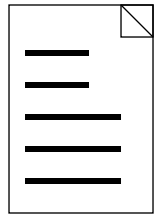


  - Static + Dynamic:

Timing Diagram



SystemVerilog



EE1D1: Digital Systems A

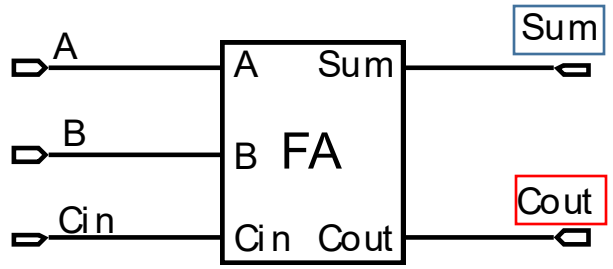
System Types

System Types - Combinatorial Systems

- Constructed from logic gates
- No feedback from outputs to inputs
- Outputs are only directly dependent on current input values
- No memory (no internal state)

Example: Full Adder

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

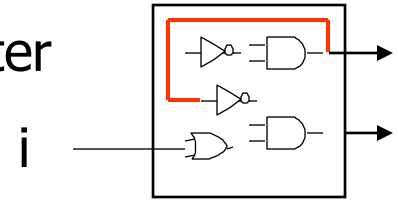


A	0	1	1	1
B	0	0	1	1
Cin	0+	0+	0+	1+
	00	01	10	11

System Types – Sequential Systems

- System has internal state space (set of possible states)
- Feedback van from outputs to inputs
- Outputs dependent on inputs and past (memory)

Example: counter



o1 (*0 1 0 1 0* …)

o2 (*0 0 1 1 0* …)

Synchronous systems

- State transitions triggered by special signal: clock
- E.g.: i = clock, counter goes to next state when i goes from 0 to 1

- Asynchronous systems
  - State transitions at arbitrary moments
  - E.g.: counter starts when i = 1

EE1D1: Digital Systems A

Boolean Algebra

Boolean Algebra

- Boolean algebra is used for specification desired behaviour

- Theorems/properties are used to rewrite Boolean expression:
  - *minimization* of the number of operations (*simplification*)
  - rewriting in terms of *available* operations

- In the coming lectures you will learn:
  - a *systematic procedure* to simplify Boolean expressions.
  - how to use SystemVerilog, and its synthesis tools, for this.

Boolean Algebra

- Boolean algebra consists of:
  - set of numbers $B = \{0, 1\}$ ({false, true})
  - set of operations
    - unary operation $'$, $\bar{}$ (NOT)
    - binary operations $\bullet$, $+$ (AND, OR)

Axioms:

$0' = 1$
$1' = 0$

$0 \bullet 0 = 0$
$0 \bullet 1 = 0$
$1 \bullet 0 = 0$
$1 \bullet 1 = 1$

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 1$

Boolean Algebra

- Important Theorems

1. Involution property:	(i) $(a')' = a$	
2. Idempotent property:	(i) $a + a = a$	(ii) $a \cdot a = a$
3. Identity property:	(i) $a + 0 = a$	(ii) $a \cdot 1 = a$
4. Null element property	(i) $a + 1 = 1$	(ii) $a \cdot 0 = 0$
5. Complement property:	(i) $a + a' = 1$	(ii) $a \cdot a' = 0$
6. Commutative property:	(i) $a + b = b + a$	(ii) $a \cdot b = b \cdot a$
7. Associative property:	(i) $(a + b) + c = a + (b + c) = a + b + c$	
	(ii) $(a \cdot b) \cdot c = a \cdot (b \cdot c) = a \cdot b \cdot c$	
8. Distributive property:	(i) $a + (b \cdot c) = (a + b) \cdot (a + c)$	
	(ii) $a \cdot (b + c) = a \cdot b + a \cdot c$	

All simply derived from:

$0' = 1$	$0 \cdot 0 = 0$	$0 + 0 = 0$
$1' = 0$	$0 \cdot 1 = 0$	$0 + 1 = 1$
	$1 \cdot 0 = 0$	$1 + 0 = 1$
	$1 \cdot 1 = 1$	$1 + 1 = 1$

Boolean Algebra

- Theorems (continued)

Also we often use the following properties:

De Morgan's laws:

(i) $(a + b)' = a' \bullet b'$
(ii) $(a \bullet b)' = a' + b'$

Absorption properties:

(i) $a \bullet b + a \bullet b' = a$
(ii) $(a + b) \bullet (a + b') = a$

(iii) $a + a \bullet b = a$
(iv) $a \bullet (a + b) = a$

(v) $a + a' \bullet b = a + b$
(vi) $a \bullet (a' + b) = a \bullet b$

Boolean Algebra – Some Proofs

Absorption property (i): $X \bullet Y + X \bullet Y' = X$

distributive property (8)	$X \bullet Y + X \bullet Y'$	$= X \bullet (Y + Y')$
complement p. (5)	$X \bullet (Y + Y')$	$= X \bullet 1$
identity p. (3)	$X \bullet 1$	$= X$

Absorption property (iii): $X + X \bullet Y = X$

identity p. (3)	$X + X \bullet Y$	$= X \bullet 1 + X \bullet Y$
distributive p. (8)	$X \bullet 1 + X \bullet Y$	$= X \bullet (1 + Y)$
null element p. (4)	$X \bullet (1 + Y)$	$= X \bullet 1$
identity p. (3)	$X \bullet 1$	$= X$

Absorption property (v): $X + X' \bullet Y = X + Y$

null element p. (4)	$X + X' \bullet Y$	$= X \bullet (1 + Y) + X' \bullet Y$
distributive p. (8)	$X \bullet (1 + Y) + X' \bullet Y$	$= X + X \bullet Y + X' \bullet Y$
distributive p. (8)	$X + X \bullet Y + X' \bullet Y$	$= X + (X + X') \bullet Y$
complement p. (5)	$X + (X + X') \bullet Y$	$= X + Y$

1. Involution property:	(i) $(a')' = a$	
2. Idempotent property:	(i) $a + a = a$	(ii) $a \bullet a = a$
3. Identity property:	(i) $a + 0 = a$	(ii) $a \bullet 1 = a$
4. Null element property	(i) $a + 1 = 1$	(ii) $a \bullet 0 = 0$
5. Complement property:	(i) $a + a' = 1$	(ii) $a \bullet a' = 0$
6. Commutative property:	(i) $a + b = b + a$	(ii) $a \bullet b = b \bullet a$
7. Associative property:	(i) $(a + b) + c = a + (b + c) = a + b + c$ (ii) $(a \bullet b) \bullet c = a \bullet (b \bullet c) = a \bullet b \bullet c$	
8. Distributive property:	(i) $a + (b \bullet c) = (a + b) \bullet (a + c)$ (ii) $a \bullet (b + c) = a \bullet b + a \bullet c$	

Boolean Algebra – Example: Cout of Full Adder

Cout = A' B Cin + A B' Cin + A B Cin' + A B Cin

= A' B Cin + A B' Cin + A B Cin' + A B Cin

= A' B Cin + A B' Cin + A B Cin' + A B Cin + A B Cin

= A' B Cin + A B Cin + A B' Cin + A B Cin' + A B Cin

= (A' + A) B Cin + A B' Cin + A B Cin' + A B Cin

= (1) B Cin + A B' Cin + A B Cin' + A B Cin

= B Cin + A B' Cin + A B Cin' + A B Cin + A B Cin

= B Cin + A B' Cin + A B Cin + A B Cin' + A B Cin

= B Cin + A (B' + B) Cin + A B Cin' + A B Cin

= B Cin + A (1) Cin + A B Cin' + A B Cin — *distributive*

= B Cin + A Cin + A B (Cin' + Cin) — *complement*

= B Cin + A Cin + A B (1) — *identity*

= B Cin + A Cin + A B

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

idempotent

Boolean Algebra – Question 1

What is a simplification of F = A B C + A B C′ + A′ B C′ ?

a. F = A′ B + A C′

b. F = A B + A C′

c. F = A′ B + B C′

d. F = A B + B C′

F = A B C + A B C′ + A′ B C′
= A B (C + C′) + A′ B C′
= A B + A′ B C′
= B (A + A′ C′)
= B (A + C′)
= A B + B C′
Hence answer d

F = A B C + A B C′ + A′ B C′
= A B C + **A B C′** + A′ B C′ + **A B C′**
= A B (C + C′) + B C′(A′+A)
= A B + B C′
Hence answer d

Boolean Algebra – De Morgan's law

$(X + Y)' = X' \cdot Y'$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X} \cdot \overline{Y}$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

$(X \cdot Y)' = X' + Y'$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X \cdot Y}$	$\overline{X} + \overline{Y}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

Also valid for more terms, e.g. $(X + Y + Z)' = X' \cdot Y' \cdot Z'$ as $(X + Y + Z)' = X' \cdot (Y+Z)' = X' \cdot Y' \cdot Z'$

Example application: find the complement of an AND/OR expression

$Z = A'\,B'\,C + A'\,B\,C + A\,B'\,C + A\,B\,C'$

$Z' = (A'\,B'\,C + A'\,B\,C + A\,B'\,C + A\,B\,C')'$ inverting both sides

$Z' = (A'\,B'\,C)' \cdot (A'\,B\,C)' \cdot (A\,B'\,C)' \cdot (A\,B\,C')'$ using De Morgan

$Z' = (A + B + C') \cdot (A + B' + C') \cdot (A' + B + C') \cdot (A' + B' + C)$ using De Morgan

$$\overline{Z} = \overline{\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C}}$$

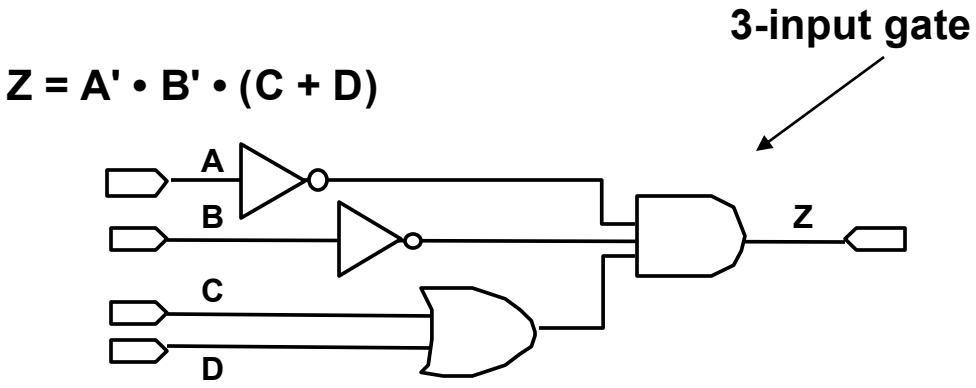
$$\overline{Z} = \overline{\bar{A}\bar{B}C} \cdot \overline{\bar{A}BC} \cdot \overline{A\bar{B}C} \cdot \overline{AB\bar{C}}$$

$$\overline{Z} = (A+B+\bar{C}) \cdot (A+\bar{B}+\bar{C}) \cdot (\bar{A}+B+\bar{C}) \cdot (\bar{A}+\bar{B}+C)$$

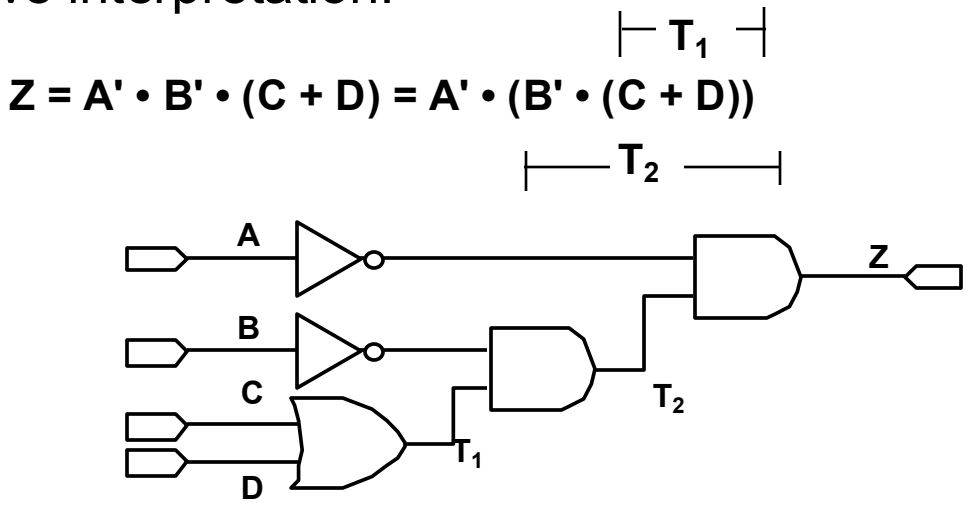
Boolean Algebra

- Application: from expression to circuit

Example:



Alternative interpretation:



Rewriting an expression implies another circuit implementation

Boolean Algebra – Application

- A circuit for a practical problem using Boolean algebra

Example: "When will I go carpooling ?" (Y):
- if Alex is going (A) and Ben is not going (B)
- for sure if Casper is going (C)"

Create a binary systems that decides when I will go

Y = ?

Corresponding Boolean expression:

Y = A B' + C

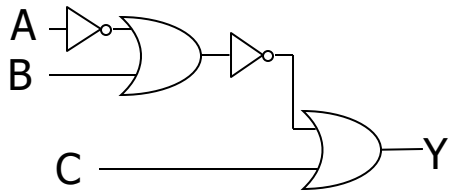
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

When only inverters (NOT) and OR gates are available:

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

Hence Y = A B' + C = (A' + B)' + C

2 inverters (') and 2 OR gates (+)



Boolean Algebra – Application

- Simplification of expressions

Carpool example: Suppose we start with the truth table.

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

Simplify this to a Boolean expression:

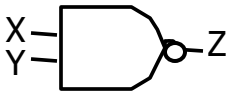
Y = A′ B′ C + A′ B C + A B′ C′ + A B′ C + A B C

Minimization using Boolean properties:

Y = BC(A′ + A) + B′C(A′ + A) + AB′C′
= BC + B′C + AB′C′
= C(B + B′) + AB′C′
= C + AB′C′
= C + AB′ (because C + C′X= C + X, with X = AB′)

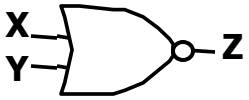
Y = A′ B′ C + A′ B C + A B′ C′ + **A B′ C** + A B′ C + A B C
= BC(A′ + A) + B′C(A′ + A) + AB′(C+C′)
= BC + B′C + AB′
= C(B + B′) + AB′
= C + AB′

Boolean Algebra – Compound operations

NAND 

A NAND B = NOT (A AND B) = (A . B)'

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR 

A NOR B = NOT (A OR B) = (A + B)'

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

- As we will show later: in current technologies, NAND, NOR gates have better properties than AND, OR gates

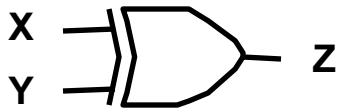
- Every Boolean expression can be expressed in NAND, NOR, NOT using De Morgan:
  - A + B = ((A + B)')' = (A'.B')'
    OR is equivalent to NAND
    with *inverted* inputs
  - A.B = ((A.B)')' = (A' + B')'
    AND is equivalent to NOR
    with *inverted* inputs

- BTW: NOT = NAND or NOR with both inputs connected to each other (proof this using Boolean algebra!)

Boolean Algebra – Compound operations

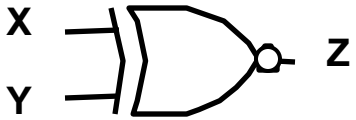
XOR (eXclusive OR): X XOR Y = (NOT X) Y OR X (NOT Y)

$X \oplus Y = X' Y + X Y'$



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

XNOR: $(X \oplus Y)' = (X' Y + X Y')' =$ ... $= X' Y' + X Y$

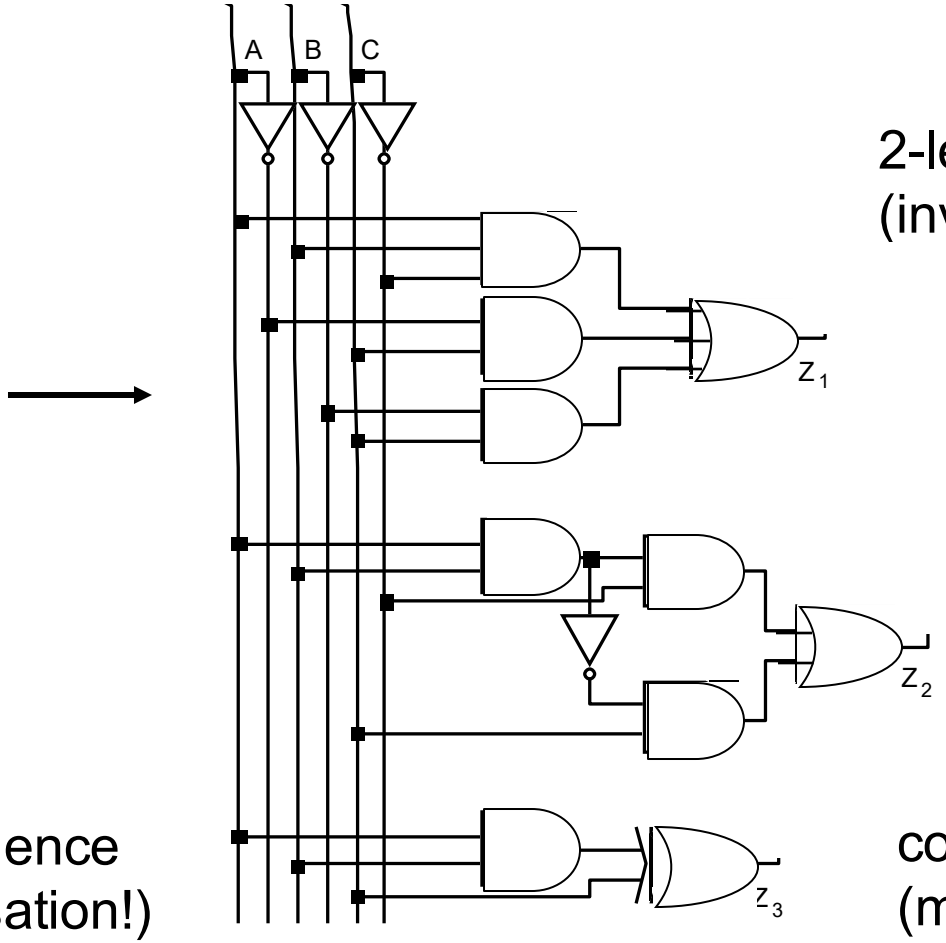


X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra

Example: alternative circuit realisations

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



2-level realisation
(inverters don't count)

multi-level realisation
(using simple gates)

(proof the equivalence
of the three realisation!)

compound gates
(min. # gates)

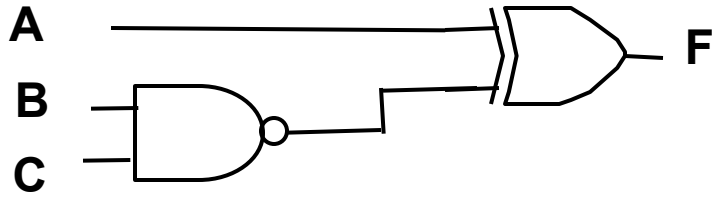
Boolean Algebra – Simplification of Expressions

- Aim: reduce complexity of resulting circuit
  - number operations/variables = number gates and number gate inputs
  - number bracket levels = number gate levels

- Background:
  - less gate inputs → faster gates
  - number of gate inputs in any case limited
  - less gates → lower costs / less power usage
  - less gate *levels* → less delay times

Question 2

What is a logic expression for the circuit below:



a. F = A′ B C + A B′ + A C′

b. F = A B C + A′ B′ + A′ C′

c. F = A B′ C′ + A B′ + A′ C

d. None of the above answers.

Let the output of the NAND be X = (B C)′. Then

F = A X′ + A′ X

= A ((B C)′)′ + A′ (B C)′

= A B C + A′ (B′ + C′)

= A B C + A′ B′ + A′ C′

Hence answer b

Summary

- System Descriptions
  - Switches, truth table, Boolean algebra, logic gates, timing diagram, HDL
- System Types
  - Combinational circuits
  - Sequential circuits
- Boolean algebra
  - Logic/Boolean operations
  - Boolean simplification/minimization
  - Prove system equivalence

To do list

- Reading Material book "Digital Design":
  - Sections 2.1 and 2.3 (not yet 2.3.5) and 2.4

- Assignments for this lecture:
  - Gated Practise Lecture 2

- Reading material for next lecture: Logic Minimization
  - Sections 2.2, 2.3.5, 2.5, 2.7 and 2.9

Thank you