# Digital Design and Computer Architecture (252-0028-00L), Spring 2023
# Optional HW 1: Combinational Logic

Instructor: Prof. Onur Mutlu
TAs: Juan Gomez Luna, Mohammad Sadrosadati, Mohammed Alser, Ataberk Olgun, Giray Yaglikci, Can Firtina, Geraldo De Oliveira Junior, Rahul Bera, Konstantinos Kanellopoulos, Nika Mansouri Ghiasi, Nisa Bostancı, Rakesh Nadig, Joel Lindegger, İsmail Emir Yüksel, Haocong Luo, Yahya Can Tuğrul, Julien Eudine

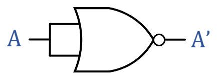
Released: Thursday, Mar. 16, 2023

## 1 Logical Completeness

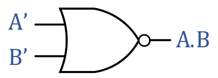
The set of {AND, OR, and NOT} gates is logically complete. We can build a circuit to carry out the specification of any truth table we wish, without using any other kind of gate. From Lecture 4, you know that the NOR gate by itself is also logically complete. Prove that you can build a circuit to carry out the specification of any truth table, by using only NOR gates.

**Solution.**
To prove that the NOR gate is logically complete, it would be sufficient to show that it is possible to build the logically complete set of {AND, OR, and NOT} gates using only NOR gates.

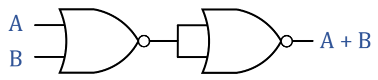
The figures below show how each of these gates can be implemented using NOR gates.



**NOT**



**AND**



**OR**

## 2 Boolean Algebra

(a) Find the simplest sum-of-products representation of the following Boolean equation. Show your work step-by-step.

$F = (\overline{A} + B + C).(A + B + \overline{C}).C + A$

$F = B.C + A$

**Explanation:**

$F = (\overline{A}.A + \overline{A}.B + \overline{A}.\overline{C} + B.A + B.B + B.\overline{C} + C.A + C.B + C.\overline{C}).C + A$

$F = (0 + B.(\overline{A} + A) + \overline{A}.\overline{C} + B + B.(\overline{C} + C) + C.A + 0).C + A$

$F = (B + \overline{A}.\overline{C} + B + B + C.A).C + A$

$F = (B.C + \overline{A}.\overline{C}.C + B.C + C.A.C) + A$

$F = (B.C + 0 + C.A) + A$

$F = B.C + A.(C + 1)$

$F = B.C + A$

(b) Using Boolean algebra, simplify the following min-terms:
$\sum(1111, 1110, 1000, 1001, 1011, 1010, 0000)$. Show your work step-by-step.

$F = (\overline{B}.\overline{C}.\overline{D}) + (A.C) + (A.\overline{B})$

**Explanation:**

$F = (A.B.C.D) + (A.B.C.\overline{D}) + (A.\overline{B}.\overline{C}.\overline{D}) + (A.\overline{B}.\overline{C}.D) + (A.\overline{B}.C.D) + (A.\overline{B}.C.\overline{D}) + (\overline{A}.\overline{B}.\overline{C}.\overline{D})$

$F = (\overline{B}.\overline{C}.\overline{D}).(A + \overline{A}) + (A.C).(B.D + B.\overline{D} + \overline{B}.D + \overline{B}.\overline{D}) + (A.\overline{B}).(\overline{C}.D + \overline{C}.\overline{D} + C.D + C.\overline{D})$

$F = (\overline{B}.\overline{C}.\overline{D}) + (A.C) + (A.\overline{B})$

(c) Convert the following Boolean equation so that it only contains NAND operations. Show your work step-by-step.

$F = \overline{A} + \overline{(B.C + \overline{A.C})}$

$F = \overline{(A.\overline{(\overline{B.C}.\overline{\overline{A.C}.\overline{A.C}})})}$

**Explanation:**

$F = \overline{(\overline{(\overline{A} + \overline{(B.C + \overline{A.C})})})}$
$F = \overline{(A.(B.C + \overline{A.C}))}$
$F = \overline{(A.\overline{\overline{(B.C + \overline{A.C})}})}$
$F = \overline{(A.\overline{(\overline{B.C}.\overline{\overline{A.C}})})}$
$F = \overline{(A.\overline{(\overline{B.C}.\overline{\overline{A.C}.\overline{A.C}})})}$

(d) Convert the same Boolean equation given in part (c) so that it only contains NOR operations. Show your work step-by-step.

$F = \overline{A} + \overline{(B.C + \overline{A.C})}$

$F = \overline{(\overline{(\overline{A+A} + \overline{(B.C + \overline{A.C})})} + \overline{(\overline{A+A} + \overline{(B.C + \overline{A.C})})})}$
$B.C = \overline{\overline{B+B} + \overline{C+C}}$
$\overline{A.C} = \overline{(\overline{(\overline{(A+A)} + \overline{(C+C)})} + \overline{(\overline{(A+A)} + \overline{(C+C)})})}$

**Explanation:**

$F = \overline{(\overline{(\overline{A} + \overline{(B.C + \overline{A.C})})})}$
$F = \overline{(\overline{(\overline{A} + \overline{(B.C + \overline{A.C})})} + \overline{(\overline{A} + \overline{(B.C + \overline{A.C})})})}$
$F = \overline{(\overline{(\overline{A+A} + \overline{(B.C + \overline{A.C})})} + \overline{(\overline{A+A} + \overline{(B.C + \overline{A.C})})})}$
$B.C = \overline{\overline{B+B} + \overline{C+C}}$

$\overline{A.C} = \overline{(\overline{(\overline{A.C})})}$
$\overline{A.C} = \overline{(\overline{(\overline{A} + \overline{C})})}$
$\overline{A.C} = \overline{(\overline{(\overline{(A+A)} + \overline{(C+C)})})}$
$\overline{A.C} = \overline{(\overline{(\overline{(A+A)} + \overline{(C+C)})} + \overline{(\overline{(A+A)} + \overline{(C+C)})})}$

## 3 Boolean Algebra and Combinational Logic Design

In this question we ask you to derive the boolean equations for two 4-input logic functions, $X$ and $Y$. Please use the truth table below to answer the following three questions.

| **Inputs** | | | | **Outputs** | |
|---|---|---|---|---|---|
| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $X$ | $Y$ |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

(a) The output $X$ is *one* when the input does **not** contain two consecutive 1's in the word $A_3, A_2, A_1, A_0$. **Fill in the truth table above** and write the corresponding Boolean equation for $X$, using *only* **2-input *NAND* gates**. Derive the expression using Boolean algebra laws.

X is 0 if there is two consecutive 1s in the 4-bit word. Therefore, $X = 0$ if at least one of the following statements is true:

- $A_3 \cdot A_2 = 1$
- $A_2 \cdot A_1 = 1$
- $A_1 \cdot A_0 = 1$

Therefore, we can write the equation of $\overline{X}$ as $\overline{X} = A_3A_2 + A_2A_1 + A_1A_0$.
Then, $X = \overline{(A_3A_2 + A_2A_1 + A_1A_0)}$

Applying DeMorgan's rule, we can form this expression as: $X = \overline{A_3A_2} \cdot \overline{A_2A_1} \cdot \overline{A_1A_0}$

This expression includes three 2-input NAND gates, whose outputs are provided to a 3-input AND gate. As the next step, we need to implement the 3-input AND gate using only 2-input NAND gates.
Let $X_A = \overline{A_3A_2}$, $X_B = \overline{A_2A_1}$, and $X_C = \overline{A_1A_0}$. Then, $X = X_AX_BX_C$.

Using associative law, $X = (X_AX_B)X_C$.
Let $X_D = X_AX_B$. Then, $X = X_DX_C$.

Using involution and idempotent laws, we can derive $X_D$ using only 2-input NAND gates.

Applying the involution law:
$X_D = \overline{\overline{X_D}} = \overline{\overline{X_AX_B}}$

$X_D = X_AX_B = \overline{\overline{(X_AX_B)} \cdot \overline{(X_AX_B)}}$

Applying the idempotent law:
Let $X_E = \overline{X_D} = \overline{X_AX_B}$. Then, $X_D = \overline{X_E} = \overline{X_EX_E}$.

We can apply the involution and idempotent laws for $X$ as well:
Let $X_F = \overline{X_DX_C}$, Then $X = \overline{X_FX_F}$.

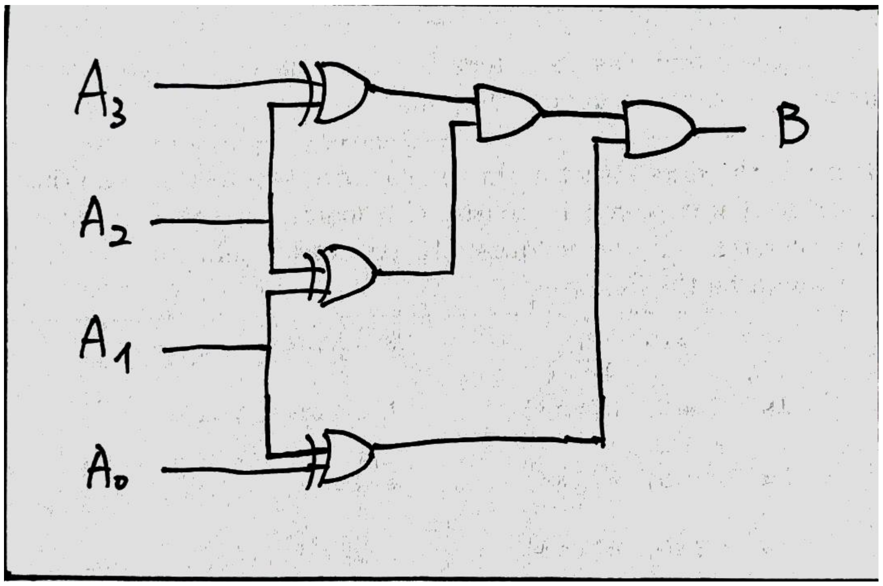
Putting all together:

- $X_A = \overline{A_3A_2}$
- $X_B = \overline{A_2A_1}$
- $X_C = \overline{A_1A_0}$
- $X_E = \overline{X_AX_B}$
- $X_D = \overline{X_EX_E}$
- $X_F = \overline{X_DX_C}$
- $X = \overline{X_FX_F}$

(b) The output $Y$ is *one* when no two adjacent bits in the word $A_3, A_2, A_1, A_0$ are the same (e.g., if $A_2$ is 0 then $A_3$ and $A_1$ cannot be 0). The output $Y$ is *zero*, otherwise (e.g., 0000). **Fill in the truth table above** and use the *sum of products* form to **write the corresponding boolean equation** for $Y$. (*No simplification needed.*)

$Y = \overline{A_3}A_2\overline{A_1}A_0 + A_3\overline{A_2}A_1\overline{A_0}$

(c) Please represent the circuit of $Y$ using *only* 2-input XOR and AND gates.

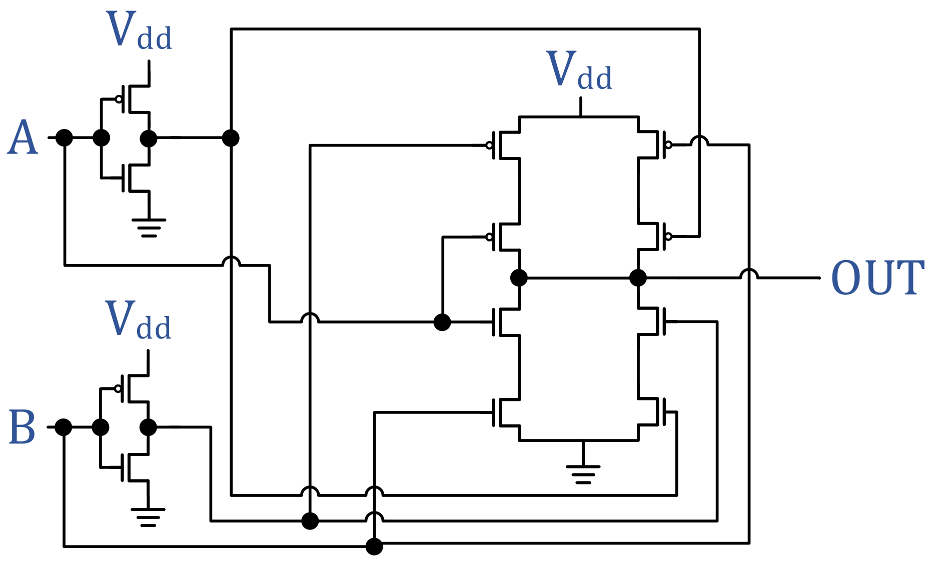


## 4 Transistor-Level Circuit Design

In Lecture 4, we learned how to implement digital circuits using the CMOS technology (i.e., p-type and n-type MOS transistors). In this assignment, we ask you to schematically design circuits using CMOS transistors for the following logic gates:

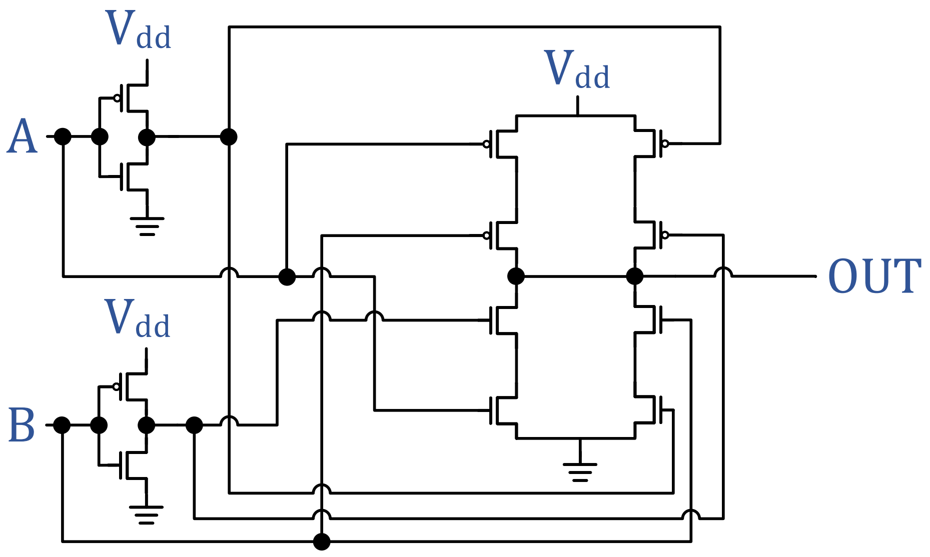
- Exclusive OR Gate (XOR)

**Solution.**



- Exclusive NOT OR Gate (XNOR)

**Solution.**

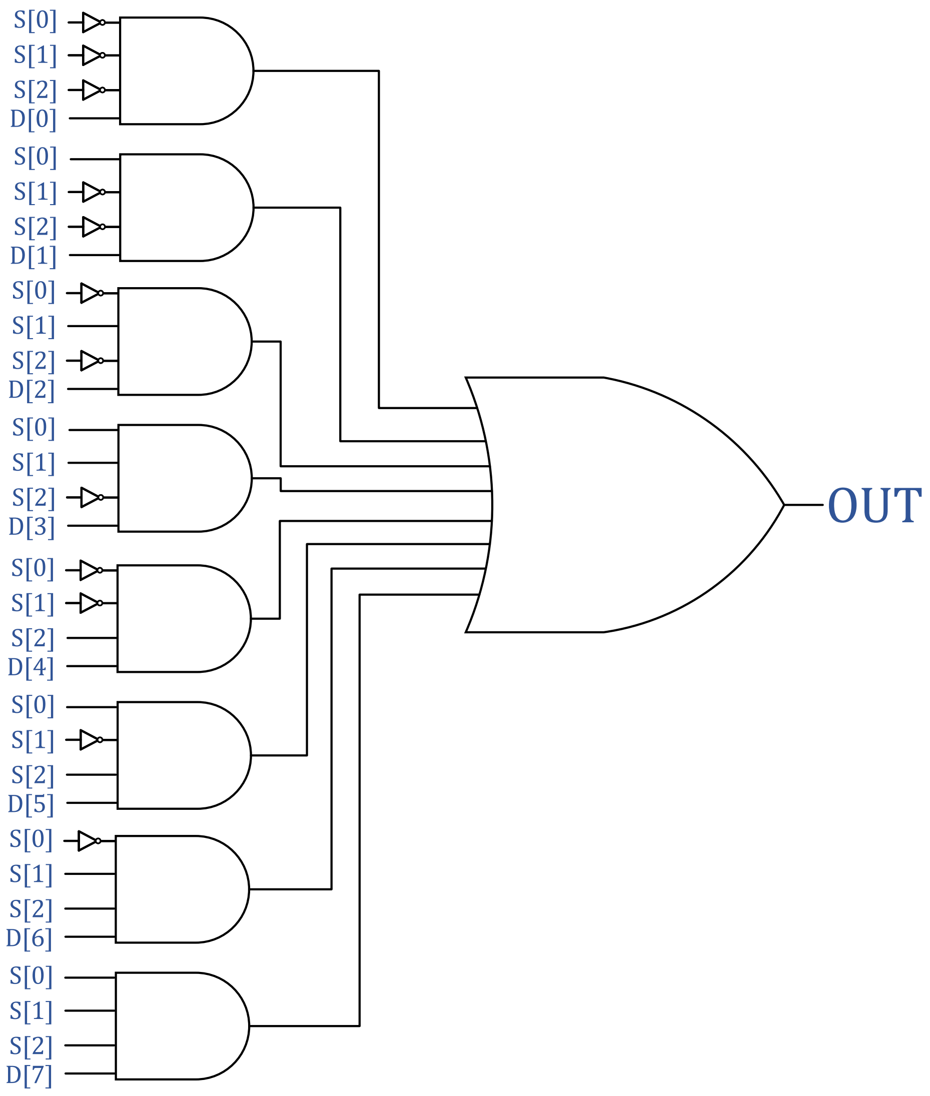


## 5 Multiplexer (MUX)

Draw the following schematics for an 8-input (8:1) MUX.

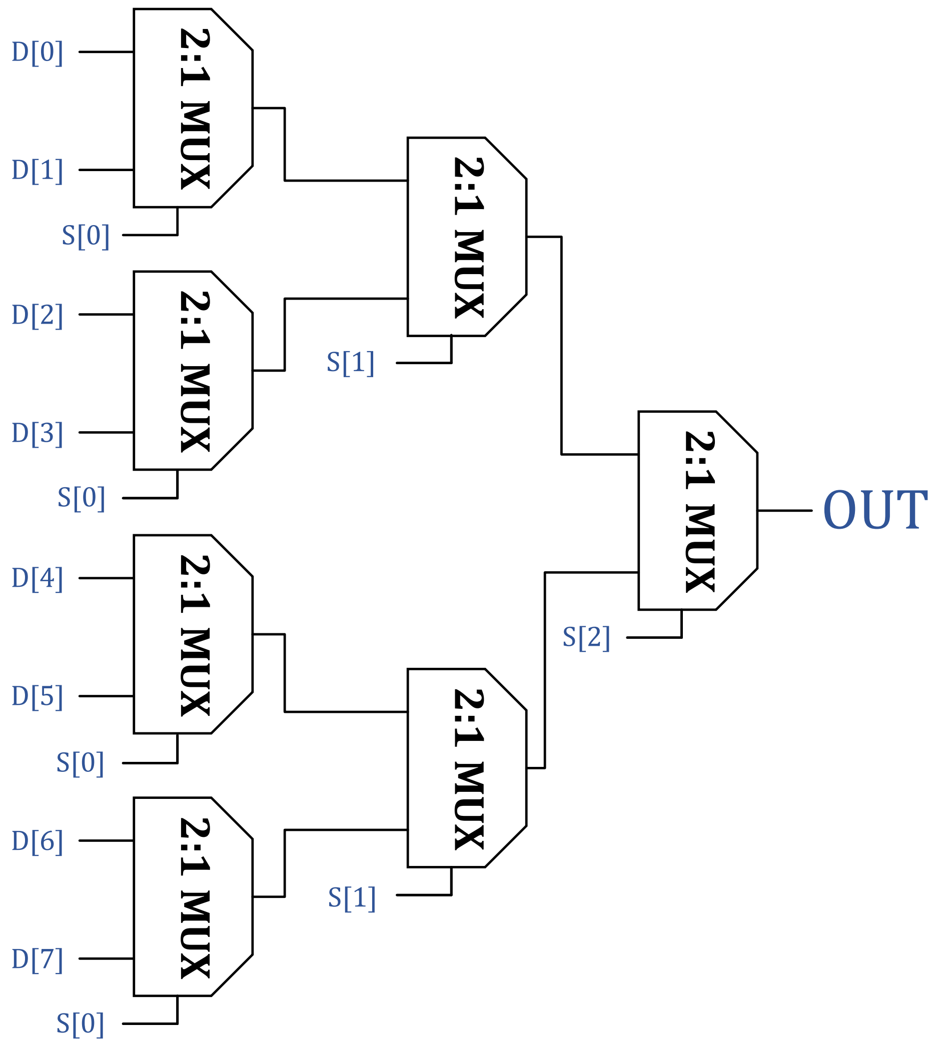
- Gate level: as a combination of basic AND, OR, NOT gates. Use as few gates as possible.

**Solution.**



- Module level: as a combination of 2-input (2:1) MUXes. Use as few 2-input MUXes as possible.

**Solution.**



D[0]
D[1]
S[0]
D[2]
D[3]
S[0]
D[4]
D[5]
S[0]
D[6]
D[7]
S[0]
S[1]
S[1]
S[2]
2:1 MUX
OUT