Exam for "Systèmes Digitaux" course

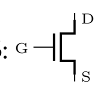
Tuesday January 14, 2020

Abstract

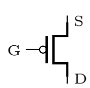
This exam is made up of two problems. It is better to answer to some of them in depth than all of them superficially.

The exam duration is 4 hours. The maximum number of pages is 6. You cannot use class material.

1 CMOS logic gates

We recall that a Negative MOS (NMOS: ) transistor is:

- *closed* if the gate G input is equal to one (that is, drain D and source S are connected), and
- *open* otherwise (that is, drain D and source S are not electrically connected).

A Positive (PMOS: ) transistor behaves the opposite way.

Q1

Comment on the gate depicted in Fig. 1.

Is it Complementary MOS (CMOS) logic?

If so, what is the Boolean function of y_1 and y_2 as a function of inputs a and b?

2 Arithmetic: Fast Addition in the Integers

Let $a = (a_{n-1}, \ldots, a_0)_2$ and $b = (b_{n-1}, \ldots, b_0)_2$ two n-bit integers, represented as a string of bits.

The bitwise operations for arithmetic addition are as follows:

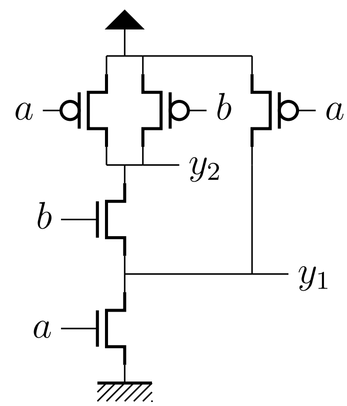


Figure 1: Gate whose functionality is to be assessed

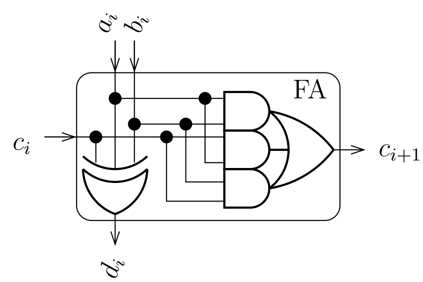


Figure 2: Netlist for the (one-bit) Full Adder

- $d_i = a_i \oplus b_i \oplus c_i$,
- $c_{i+1} = \mathsf{MAJ}(a_i, b_i, c_i)$,

where $\oplus$ represents the "exclusive-or" (or XOR) operation and where MAJ is the majority function, namely

$$\begin{aligned} \mathsf{MAJ}(a_i, b_i, c_i) &= (a_i \wedge b_i) \vee (b_i \wedge c_i) \vee (c_i \wedge a_i) \\ &= (a_i \wedge b_i) \oplus (b_i \wedge c_i) \oplus (c_i \wedge a_i). \end{aligned}$$

The FA (Full Adder) is the one-bit slice of an adder, depicted in Fig. 2.

Q2.1

Explain how to use n FAs to add up two n-bit integers $a = (a_{n-1}, \ldots, a_0)_2$ and $b = (b_{n-1}, \ldots, b_0)_2$.

Q2.2

Explain how the same structure can be leveraged to subtract two integers. Illustrate on the computation of $a - b$ or $a + b$ by the same netlist.

Q2.3

What is the critical path of the structures you proposed in Q2.1 and Q2.2.

Q2.4

In this section, we imagine a structure to reduce the critical path, in average. In this respect, notice that the existence of a carry anywhere in the adder can be predicted under some condition. Explicit this condition, and present a structure to accelerate the addition resorting to speculative execution.

3 Special operators: Butterfly Fourier transform in $\mathbb{F}_2^n$

Let n a positive integer. This problem consists in studying the butterfly algorithm for fast Fourier transform on $\mathbb{F}_2^n$. We start by a mathematical problem statement.

Let $\mathbb{F}_2^n$ be an n-dimensional vector space over the field $\mathbb{F}_2$, that we equip with the canonical scalar product $u \cdot x = \bigoplus_{i=0}^{n-1} u_i x_i$. A given $x \in \mathbb{F}_2^n$ is also written as $x_{n-1,\ldots,0}$ to highlight its coordinates $(x_i)_{0 \leq i \leq n-1}$. Slices are sets of coordinates, selected as per their indices. For example, the slice $x_{n-1,\ldots,i} \in \mathbb{F}_2^{n-i}$ (for $0 \leq i \leq n-1$) is the slice of the first (or leftmost) $n - i$ bits of x, and the slice $x_{i-1,\ldots,0} \in \mathbb{F}_2^i$ (for $1 \leq i \leq n$) is the slice of the last (or rightmost) i bits of x. One coordinate is also denoted as $x_{i,\ldots,i} = x_i$ (for $0 \leq i \leq n-1$), and we extend these notations with the empty slice, when upper and lower indices cross, that is $x_{i,\ldots,i+1}$ is considered empty. In the sequel, we shall also use the comma operator for the concatenation; for instance $x \in \mathbb{F}_2^n = (x_{n-1}, \ldots, x_0)$.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ a Boolean function. The Fourier transform $\hat{f}$ of f for all values $x \in \mathbb{F}_2^n$ is defined as:

$$\hat{f}(x) = \sum_{u \in \mathbb{F}_2^n} (-1)^{u \cdot x} f(u). \tag{1}$$

The Fourier transform takes its values in $\mathbb{Z}$.

Notice that the truth table of the Fourier transform $\hat{f}$ is a linear transformation of that of f, the transformation matrix being the Hadamard matrix. An illustration for $n = 4$ is provided hereafter:

$$\begin{pmatrix} \hat{f}(0000) \\ \hat{f}(0001) \\ \hat{f}(0010) \\ \hat{f}(0011) \\ \hat{f}(0100) \\ \hat{f}(0101) \\ \hat{f}(0110) \\ \hat{f}(0111) \\ \hat{f}(1000) \\ \hat{f}(1001) \\ \hat{f}(1010) \\ \hat{f}(1011) \\ \hat{f}(1100) \\ \hat{f}(1101) \\ \hat{f}(1110) \\ \hat{f}(1111) \end{pmatrix} = \begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 \end{pmatrix} \begin{pmatrix} f(0000) \\ f(0001) \\ f(0010) \\ f(0011) \\ f(0100) \\ f(0101) \\ f(0110) \\ f(0111) \\ f(1000) \\ f(1001) \\ f(1010) \\ f(1011) \\ f(1100) \\ f(1101) \\ f(1110) \\ f(1111) \end{pmatrix} \tag{2}$$

Proposition 1 (Butterfly Fourier transform)**.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Let us define recursively the functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$, (for $0 \leq i \leq n$), as follows: $f_0 = f$, and f_{i+1} in $y \in \mathbb{F}_2^n$ is equal to $f_{i+1}(y) = f_i(y_{n-1,\ldots,i+1}, 0, y_{i-1,\ldots,0}) + (-1)^{y_i} f_i(y_{n-1,\ldots,i+1}, 1, y_{i-1,\ldots,0})$, (for $0 \leq i \leq n-1$), or equivalently:*

$$f_{i+1}(y) = \begin{cases} f_i(y_{n-1,\ldots,i+1}, 0, y_{i-1,\ldots,0}) + f_i(y_{n-1,\ldots,i+1}, 1, y_{i-1,\ldots,0}) & \text{if } y_i = 1, \\ f_i(y_{n-1,\ldots,i+1}, 0, y_{i-1,\ldots,0}) - f_i(y_{n-1,\ldots,i+1}, 1, y_{i-1,\ldots,0}) & \text{if } y_i = 0. \end{cases}$$

The i-th coordinate in the argument of f_i is called the pivot. Then $\hat{f} = f_n$.

Q3.1

Prove Proposition 1.

Q3.2

What is the complexity of the Butterfly algorithm from Prop. 1.

Q3.3

Explain how to compute the Fourier transform using a combinatorial circuit.

Q3.4

Explain how to compute the Fourier transform using a sequential circuit, in n steps.