Exam for “Systèmes Numériques” course

Tuesday January 17, 2023

Abstract

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

The exam duration is 3 hours and 45 minutes. 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:

- *open* if the gate input is equal to one, and
- *closed* otherwise.

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

1.1 Q1.1

Comment on gate depicted in Fig. 1. Is it Complementary MOS (CMOS) logic?

1.2 Q1.2

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

2 Boolean functions

Let $n \geq 1$ be a strictly positive number. We call f a Boolean function if it maps a vector of n bits to one bit.

Let “$\oplus$” (resp. “$\wedge$”) denote the *addition* (resp. *multiplication*) of two bits, 0 and 1, as depicted below:

- $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$,
- $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$ and $1 \wedge 1 = 1$.

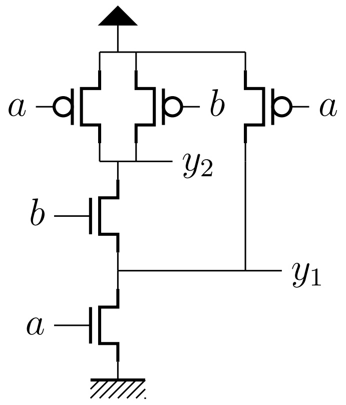

Figure 1: Gate whose functionality is to be assessed

2.1 Q2.1

What is the structure of $\mathbb{F}_2 = (\{0,1\}, \oplus, \wedge)$?

Let $\mathcal{P}(N)$ denote the power set of $N = \{1, \dots, n\}$, that is:

$$\mathcal{P}(N) = \{\emptyset, \{1\}, \dots, \{n\}, \{1,2\}, \dots, \{1,n\}, \{2,3\}, \dots, \{2,n\}, \dots, \{1,2,\dots,n\}\}.$$

The *algebraic normal form* (ANF) of a Boolean-function is the expression:

$$f(x) = \bigoplus_{I \in \mathcal{P}(N)} a_I \left(\prod_{i \in I} x_i \right). \tag{1}$$

In this equation:

- a_I are bits (there are 2^n of them), and
- $\prod_{i \in I} x_i = x_2 \wedge x_3$ if $I = \{2,3\}$, $\prod_{i \in I} x_i = 1$ if $I = \emptyset$, etc.

2.2 Q2.2

Explain why the expression (1) is unique. As a hint, notice that the set of Boolean functions is a space-vector, and find a basis suitable for the ANF.

2.3 Q2.3

As an example, write the ANF of the Boolean function given in Fig. 2.

x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figure 2: Truth table of one Boolean function with $n = 3$ inputs.

2.4 Q2.4

The product $x^I = \prod_{i \in I} x_i$ is nonzero if and only if x_i is nonzero (i.e. equals 1) for every $i \in I$, that is, if I is included in the support of x (the subset of N corresponding to entries i for which $x_i \neq 0$); hence, the Boolean function $f(x) = \bigoplus_{I \in \mathcal{P}(N)} a_I x^I$ takes value

$$f(x) = \bigoplus_{I \subseteq \mathsf{supp}(x)} a_I, \tag{2}$$

where $\mathsf{supp}(x)$ denotes the support of x.

If we use the notation $f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u x^u$, with $\mathbb{F}_2^n = \mathbb{F}_2 \times \ldots \mathbb{F}_2$, we obtain the relation $f(x) = \bigoplus_{u \preceq x} a_u$, where $u \preceq x$ means that $\mathsf{supp}(u) \subseteq \mathsf{supp}(x)$ (we say that u is *covered* by x). A Boolean function f° can be associated to the ANF of f: for every $x \in \mathbb{F}_2^n$, we set $f^\circ(x) = a_{\mathsf{supp}(x)}$, that is, with the notation $f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u x^u$: $f^\circ(u) = a_u$. Relation (2) shows that f is the image of f° by the so-called binary Möbius transform.

Comment on the OCaml code in Listing 1, especially *why* it works and *how* efficient it is.

Code Listing 1: Algorithm to turn a Boolean function into its ANF

```
let n = 3
let mobius f =
  for i = n - 1 downto 0 do
    for j = 0 to (1 lsl n) - 1 do
      if (j lsr i) land 1 <> 0
      then f.(j) <- f.(j) lxor f.(j - (1 lsl i))
      done
    done;
  f
```

In the Listing 1, we use the definitions given in Listing 2.

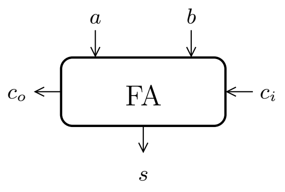

Figure 3: Schematic for the FA gate

Code Listing 2: A few definitions of Boolean operators in OCaml

```
x lsl n = x * 2^n
x lsr n = x / 2^n      (* result is an integer *)
land = bitwise and
lxor = bitwise xor
```

2.5 Q2.5

Show that the converse is also true, namely: let f be a Boolean function on $\mathbb{F}_2^n$ and let $\bigoplus_{I\in\mathcal{P}(N)} a_I x^I$ be its ANF, and we have:

$$\forall I \in \mathcal{P}(N), \qquad a_i = \bigoplus_{x\in\mathbb{F}_2^n/\mathsf{supp}(x)\subseteq I} f(x).$$

3 Hardware arithmetic

3.1 Q3.1

Let a, b and c_i be three Boolean variables. The Full Adder (FA) is the gate that computes:

- $s = a \oplus b \oplus c_i$, and
- $c_o = (a \wedge b) \vee (b \wedge c_i) \vee (c_i \wedge a)$.

Given $n > 1$ and two integers $A = \sum_{i=0}^{n-1} a_i 2^i$ and $B = \sum_{i=0}^{n-1} b_i 2^i$, where a_i and b_i $(0 \le i < n)$ are bits. Describe how to compute the sum $C = A + B$ of A and B using some FA gates. How many of them are required? Draw the netlist of this adder, using the symbol of Fig. 3 to represent the FA gate.

Give the Boolean equations of the gate which computes the Least Significant Bit (LSB) of C, that is $c_0 = C \bmod 2$. Indeed, the value c_0 can be produced by a simplified gate called "half-adder" (HA).

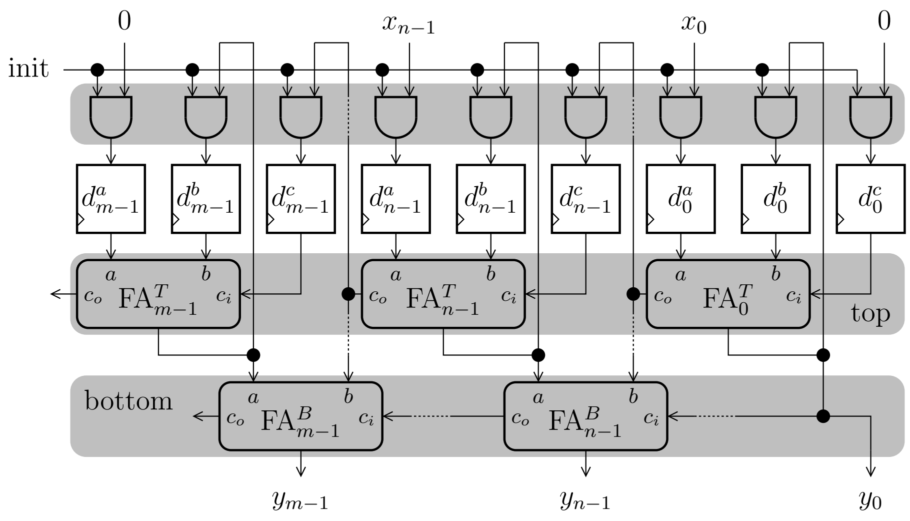

Figure 4: CSA accumulator

3.2 Q3.2

Imagine that n grows larger and larger (e.g., $n = 128$ bit). Can you explain the risk of the straightforward architecture you proposed in Q3.1?

In this part, we optimize the addition, in the case that more than two numbers are added. We consider a sequential circuit, which adds many numbers $X = \sum_{i=0}^{n-1} x_i 2^i$.

Let $p > 0$ and $m = n + p$. We intend to devise an accumulator, able to add 2^p such n-bit numbers X, presented sequentially (one after the other). For this purpose, we use the datapath presented in Fig. 4. The inputs are the consecutive values of $X \in \{0, \ldots, 2^n - 1\}$, and one control signal named "init". As the circuit is sequential, some values are memorized into flip-flops (one-bit registers), represented as in Fig. 5 (*left*). The clock signal (denoted as "clk") will be implicit in the sequel.

In the datapath depicted in Fig. 4, there are three layers of combinational gates:

1. some AND gates,
2. some FA gates termed "top" and
3. others termed "bottom".

Can you account for their role, starting with the AND gates?

Besides, as a clue to understand the role of the "top" FA, consider only their critical path. Also notice that inputs a, b and c_i of a FA gate are equivalent. What then does the "top" layer of FA gates compute? After $0 < j < 2^p$

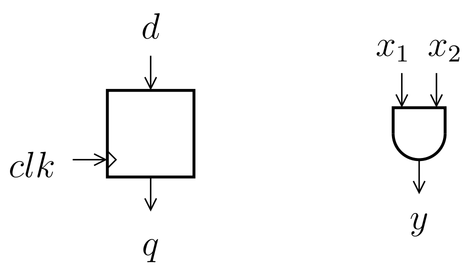

Figure 5: Schematic of a flip-flop (*left*) and of an AND gate (*right*).

inputs $X = \sum_{j'=0}^{j-1} X_{j'}$ have been presented as input (with init $= 1$), can you explain how the partial sum can be retrieved from the contents of flip-flops d_i^a, d_i^b and d_i^c (for $0 \leq i < 2^p$)? Explain why this architecture is referred to as "Carry-Save Adders" (CSA). Also comment why the output carries of MSB (Most Significant Bits) gates FA_{m-1}^T and FA_{m-1}^B are unused.

After at most 2^p n-bit words X have been presented, how should the computation be finalized?

Explain the role of the "bottom" layer, what does it do?

Assume that m clock cycles are needed to propagate the carries from FA_0^B to FA_{m-1}^B. Propose a method to keep the register values unchanged during the m clock cycles required for the "bottom" addition to finalize.

4 Combinational and Sequential circuits

4.1 Q4.1

Let p be a prime number, which can be represented in n bits. Specify, either with a netlist or some pseudo-code, a combinational circuit to multiply two elements of $\mathbb{Z}_p$ (the ring p elements $\{0, 1, \ldots, p-1\}$).

4.2 Q4.2

Assume you have at hand one such multiplier in $\mathbb{Z}_p$. Describe a sequential circuit (speed is not a concern) which instantiates only one single multiplier and is able to find the inverse of a nonzero element of $\mathbb{Z}_p$.

4.3 Q4.3

Discuss in what respect the time taken to compute the inverse (*unintentionally*) leaks information about the data to be inversed. How much information is available to an attacker?

Assume the input of the inverse is $x \oplus k$, where x is known but not k (k as *secret* $\underline{k}$*ey*). How many timing measurements are needed to recover the value k? (the attacker knows the value of x).