Exam for "Systèmes Digitaux" course

Tuesday January 16, 2018

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 4 hours. There are 5 pages of questions in this exam.

1 CMOS logic gates

We recall that a Negative MOS (NMOS — 

) transistor is:

- *passing* if the gate input G is equal to one (that is, the source S is connected to the drain D), and
- *blocking* otherwise.

A Positive (PMOS — 

) transistor behaves the opposite way.

1.1 Q1.1

What is the logic function of the two gates depicted in Fig. 1.

1.2 Q1.2

We recall that the majority function is a three-input Boolean function. Let us denote the inputs as a, b, c and the output by y. Then $y = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$. Give a structure in transistors for this gate. Optimize it to minimize the number of transistors.

1.3 Q1.3

Consider the exclusive-or function $y = a \oplus b$. Explain whether or not this function can be implemented as a single CMOS gate, and give a netlist.

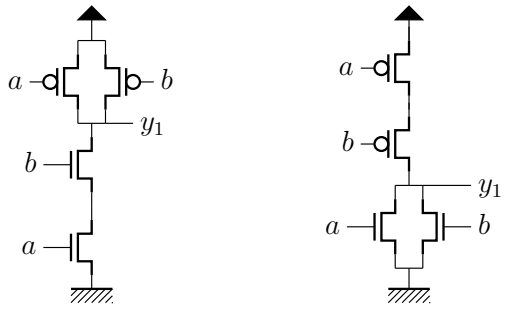

Figure 1: Two CMOS gates whose functionality is to be derived

Table 1: Look-up-table of PRESENT sbox

input	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
output	0xc	0x5	0x6	0xb	0x9	0x0	0xa	0xd
input	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
output	0x3	0xe	0xf	0x8	0x4	0x7	0x1	0x2

2 Boolean functions

PRESENT is a block cipher which makes use of 4×4 substitution boxes. Its function is given in Table 1, using hexadecimal notation.

2.1 Q2.1

Verify that the netlist depicted in Fig. 2 implements the PRESENT substitution box. What is your proposed verification method?

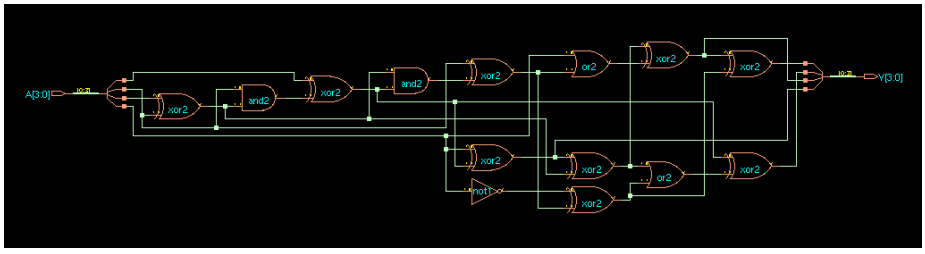

Figure 2: Netlist of the PRESENT substitution box considered in this exercise

3 Hardware arithmetic

Sensitive computations must be secured against non-invasive attacks, which attempt to correlate the leakage of some operations with a hypothetical model.

A protection against this threat is called *masking*. Masking consists in changing the unprotected intermediate variables of the computation into randomized versions, which are thus decorrelated from the unprotected variables, each being a potential target for a side-channel attack.

In particular, in the field of symmetrical encryption, the mainstream approach consists in the use of purely Boolean structures. This is the case of widely used (and standardized) ciphers, such as DES and AES. In both these examples, the operations (except for simple data move, which does not leak by itself) simply consist in XORs and in look-up-tables (LUTs). It is therefore natural to restrict to so-called *Boolean masking*, where the only operation in terms of masking is the XOR. This is innately compatible with the functional XOR operations, and LUTs are also easy to protect, e.g., using recomputation.

However, some other symmetrical encryption algorithms, such as IDEA or RC5, mix *Boolean* and *arithmetic* operations. Arithmetic operations are usually additions on n-bits, modulo 2^n (to handle overflows nicely). When n is large ($n \gg 8$, such as $n = 16$ or 32), implementing the addition in a table has a prohibitive cost. We intend in this exercise to present a masking scheme which is compatible with both Boolean and arithmetic operations.

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

The bitwise operations for arithmetic addition are as follows:

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

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. 3.

Now we have the property that, for all $m \in \{0, 1\}$,

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

3.1 Q3.1

Prove the last two expressions.

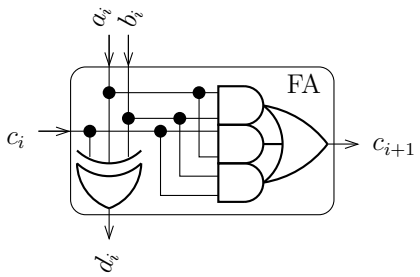

Figure 3: Schematic for the Full Adder

3.2 Q3.2

Show how to implement a masked addition using the same mask $(m, m, \ldots, m)_2$ for both operands.

4 Finite State Machines

Let n be a strictly positive integer. The Hamming weight of an n-bit word is equal to the sum of its bits.

4.1 Q4.1

Propose a combinational architecture for the computation of the Hamming weight of an n-bit word, when n is a power of two.

4.2 Q4.2

Consider an incoming sequence of bits, and imagine a minimal operator which computes the Hamming weight on the fly. Specifically, explain what logic shall be used into the white oval box in Fig. 4.

4.3 Q4.3

The connections in bold lines are buses, that is vectors of bits. What is the minimum bit width of the bold lines in Fig. 4?

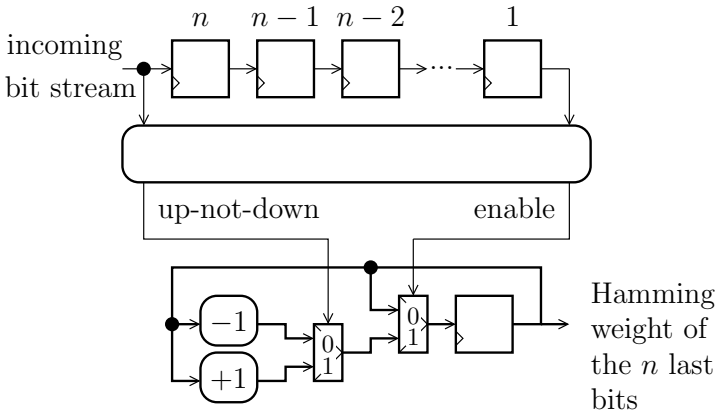

Figure 4: On-the-fly serial computation of Hamming weight