

Chapter 1

• Exercise 1:

Within the file *hard.c* used in lab sessions, there are, among others, the following functions:

```
#define PFE_INT    0xA2

void EnableBusAD( unsigned ALE )
{
    static union REGS inregs, outregs;

    inregs.h.ah = 0x80; /* Service for INT = 0x80 */
    inregs.h.al = ALE; /* 0: Disable ALE. 1: Enable ALE */
    inregs.x.dx = 0xFFFF; /* 0: Input. 1: Output */

    int86( PFE_INT, &inregs, &outregs );
}

unsigned char ReadBusAD( unsigned long direccio )
{
    unsigned char valor;

    valor = ( inportb( direccio ) );
    return valor;
}

void WriteBusAD( unsigned char valor, unsigned long direccio )
{
    outportb( direccio, ( unsigned char ) valor );
}
```

1. What does every one of these functions and how it does?
2. What is the mission of union REGS ?
3. Modify the function void EnableBusAD(unsigned ALE) to make it suitable for both read and write.

Chapter 2

• **Exercise 2:**

We have a digital system that works with a clock signal *clk* and receives external serial data through an input called *sdata*. These data are generated by means of an independent clock signal but with a frequency equal to that of signal *clk*. Internally, the system considered contains a subsystem that provides a signal *clk2* in phase with *clk* but of twice its frequency. This signal *clk2* is used to generate an internal signal called *clk_q*, according to figure 1.

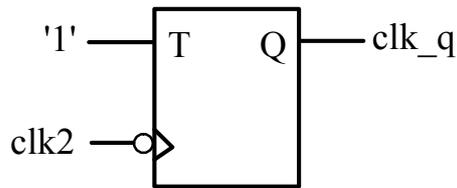


Figure 1.

Under these conditions, answer the following questions:

1. Which is the phase relation between signals *clk* i *clk_q*. Justify the answer.
2. Within the system considered we also have another subsystem as shown in figure 2:

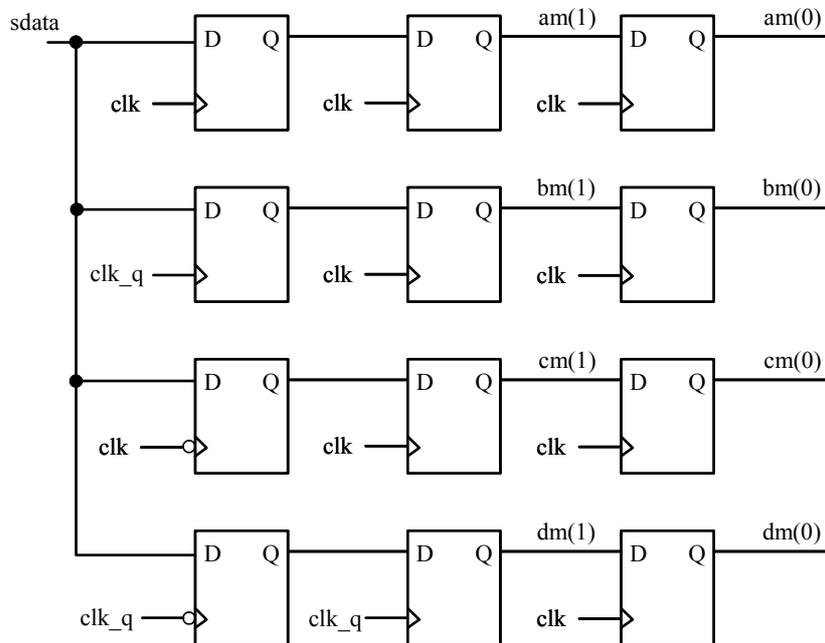


Figure 2.

Considering that the second register column is used just to solve any eventual metastability problem, hint a possible functionality for that system.

3. Using exclusively combinational logic functions, modify the system in figure 2 to make it able to produce a valid sample of input signal *sdata*. **Hint:** To answer this question it is convenient to consider the 4 different possible situations of signal *sdata* with respect to signal *clk* as shown in figure 3.

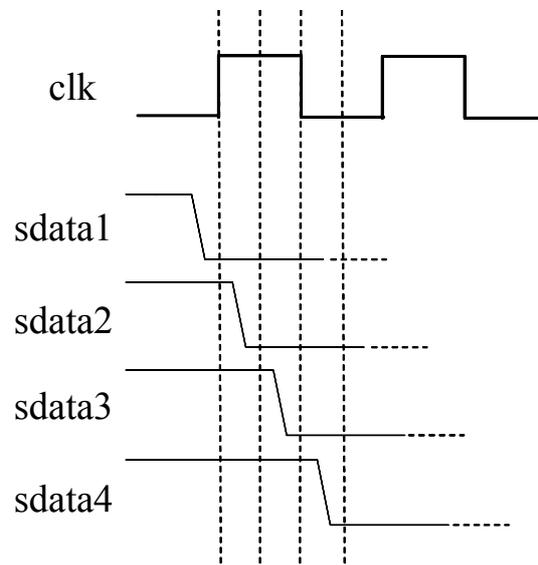


Figure 3.

• **Exercise 3:**

Given the following VHDL architecture description:

```
architecture comportament of examen is
signal q0, q1, q2, clearn_int: std_logic;
begin
    proces_1:process(clearn_int,extern)
    begin
        if (clearn_int='0') then
            q0 <= '0';
        elsif (extern'event and extern='1') then
            q0 <= '1';
        end if;
    end process;

    proces_2:process
    begin
        wait until (clk'event and clk='1');
        if(clearn_int='0') then
            q1 <= '0';
        else
            q1 <= q0;
        end if;
    end process;

    proces_3:process
    begin
        wait until (clk'event and clk='1');
        if (clearn_int='0') then
            q2 <= '0';
        else
```

```

                q2 <= q1;
            end if;
        end process;

        clearn_int <= '0' when ((clearn='0') or (q1='1' and q2='1'))
            else '1';

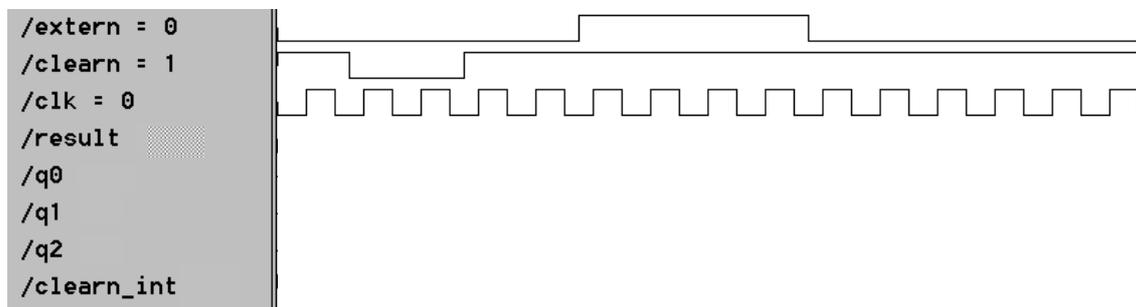
        result <= '1' when (q1='1' and q2='1')
            else '0';

    end comportament;

```

we ask:

- Explain in a reasoned way its functionality, with emphasis in the relationship between signals $q0$, $q1$, $q2$, $clearn_int$ and $result$ and to the signal $extern$.
- Based on the answer to question a) complete the following chronogram:



• Exercise 4:

Given the following VHDL architecture description:

```

architecture comportament of sistema is
    signal q0, clearn_int: std_logic;
    signal q1: std_logic_vector(2 downto 0);

begin
    proces_1:process
    begin
        wait until (clk'event and clk='1');
        if (clearn_int='0') then
            q0 <= '0';
        elsif (extern='1') then
            q0 <= '1';
        end if;
    end process;

    proces_2:process
    begin
        wait until (clk'event and clk='1');
        if(clearn_int='0') then
            q1 <= "000";
        elsif (q0='1') then
            q1 <= q1 + "1";
        end if;
    end process;

    clearn_int <= '0' when ((clearn='0') or (q1="110"))

```

```

        else '1';

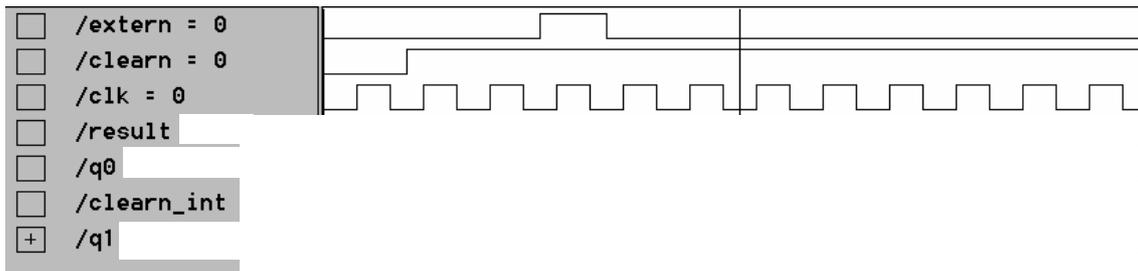
    result <= '1' when (q1="110")
        else '0';

end comportament;

```

we ask:

- a) Explain in a reasoned way its functionality, with emphasis in the relationship between signals *q0*, *q1*, *clearn_int* and *result* and to the signal *extern*.
- b) Based on the answer to question a) complete the following chronogram:



• **Exercise 20:**

In a CAN bus as shown in figure 1, where the line connecting the nodes has a delay of 5ns/m , we are considering the two **end** nodes.

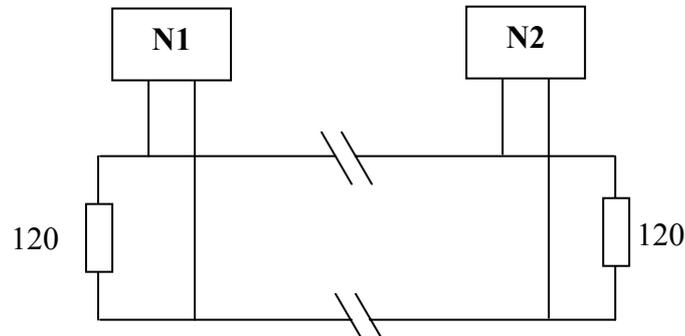


Figure 1

Between these two nodes is running a non-destructive arbitration process to get bus control, as is displayed in figure 2, where “S” means the start bit.

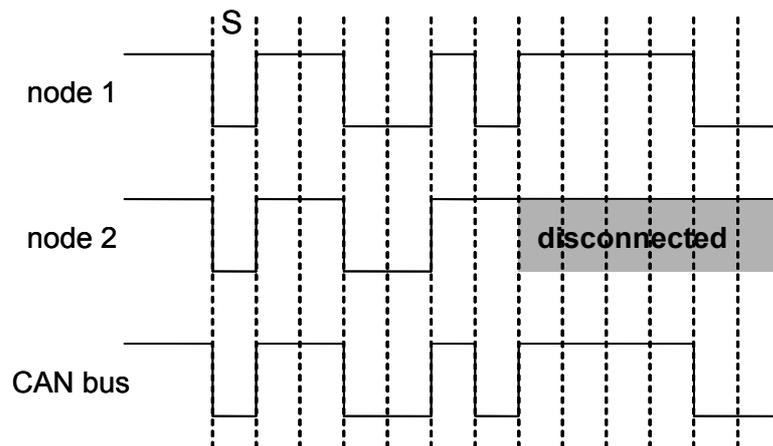


Figure 2

Looking at figure 2, we ask:

1. Explain how this arbitration process works and why node 2 disconnects from bus.
2. Explain the reason for a situation where two (or more) nodes try to get the bus, if every node signals with the start bit that he wants the bus and the rest should not attempt to transmit when they detect the bus busy.

In figure 3 the structure of a bit time for that particular bus is shown, where the figure under each segment name means the number of *quantum* it takes, and the **read time for that bit is right between TSEG1 and TSEG2**.

Syn c	TSEG1 6	TSEG 2
----------	------------	-----------

Figure 3

If the transmission velocity is **1Mb/s**, the line delay, as already mentioned, is **5ns/m**, the delay of node transceivers, both in transmission and reception is **30ns** and we assume that the frequency error between the clocks at different nodes is negligible, we ask:

3. Calculate the maximum length of bus line under these assumptions. Justify the answer.

Now, let's suppose that the maximum frequency error between any two nodes is $\pm 1.5\%$, that resynchronization takes place only at the *recessive-dominant* edge, and that the maximum number of bits without transition is five, as shown in figure 4.

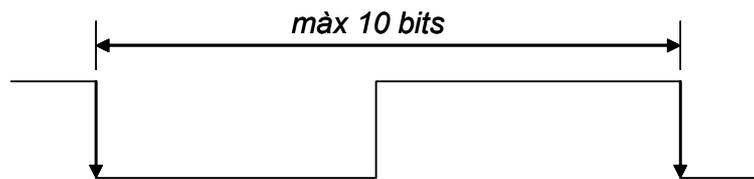


Figure 4

4. Modify, explaining the solution taken, the bit time structure of figure 3 to be able to resynchronize under the new conditions.
5. If the total number of *quantum* per bit is kept at 8, calculate the maximum length of bus line for the new conditions.

• Exercise 21:

We want to design a system to control a stepping motor in such way that its shaft can turn 90° between an initial position named I (angular position 0°) and a final position named F (angular position 90°). This zone has to be covered by means of angular increments (steps) of $0,9^\circ$.

The motor we intend to use has a basic step of $1,8^\circ$. The chosen driver is the L6219 from ST Microelectronics.

The system we have to design will have to generate four output signals, named *c1*, *c2*, *p1* and *p2*, that will be used as inputs to the L6219 driver. The time sequence to generate these signals is shown in figure 1.

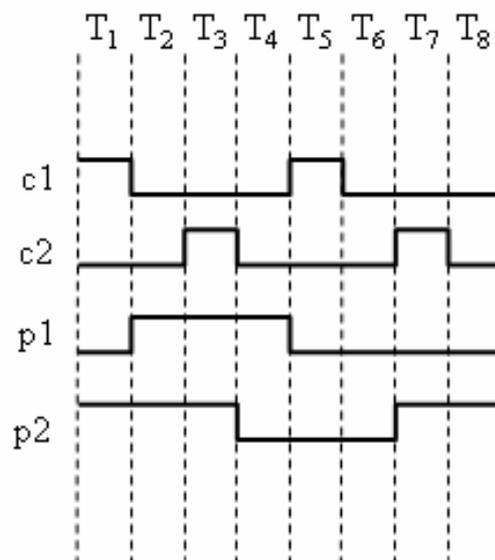


Figure 1.

The use of that particular time sequence (half step) produces an angular increment in shaft position, from T_1 to T_8 , of $0,9^\circ$. Accordingly, the 8 sequences in a row produce an angular movement of $7,2^\circ$. When the sequence is executed as $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5 \rightarrow T_6 \rightarrow T_7 \rightarrow T_8$, the motor moves from position I to position F, while if it is executed as $T_8 \rightarrow T_7 \rightarrow T_6 \rightarrow T_5 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2 \rightarrow T_1$ the motor moves from position F to position I.

Under these conditions, we ask:

1. If the motor chosen has a maximum angular speed of 300 r.p.m. and we want to sweep the mentioned 90° zone in 1 second, assuming that all the intervals shown in figure 1 are equal and of value T , find the value of T . (10%)
2. Design a system that has three inputs, *enable*, *reset* and *clk* and one output, *enable_out*. The signal *clk* is the system clock and runs at 10 MHz. This signal will act only at its rising edge on the system memory elements. The signal *reset* (active high) is the synchronous system reset. The signal *enable* is the system enable signal (active high). The signal *enable_out* must have duration at high level of one period of signal *clk*, and its own period will be the value T of question 1. This signal *enable_out* will only be active when *enable* input is active. For this design,

combinational or sequential functions of any type may be used. Every sequential function needs to have its reset signal clearly identified and connected. (20%)

3. Design a system with four inputs, *enable*, *reset*, *i2f* and *clk* and four outputs, *c1*, *c2*, *p1* and *p2*. The signal *clk* is the system clock and runs at 10 MHz. This signal will act only at its rising edge on the system memory elements. The signal *reset* (active high) is the synchronous system reset. The signal *i2f* indicates the movement direction of the motor, in such a way that a high level means from position I to position F, and a low level the opposite. The signal *enable* is the system enable signal (active high). This signal, while valid, makes the sequence of signals *c1*, *c2*, *p1* and *p2* advance one unit per clock cycle, depending on the value of input *i2f* (i.e., to move from, say, from T₄ to T₅ or from T₅ to T₄). Outputs *c1*, *c2*, *p1* and *p2* will be used as inputs to the driver L6219. For this design, combinational or sequential functions of any type may be used. Every sequential function needs to have its reset signal clearly identified and connected.

Note: For this design, the system designed in question 2 may be used as a single component. (30%)

4. Design a system with six inputs, *clk*, *reset*, *i2f*, *mode(1:0)*, *steps(6:0)* and *start* and five outputs, *c1*, *c2*, *p1*, *p2* and *fi_comanda*. Outputs *c1*, *c2*, *p1* and *p2* will be used as inputs to the driver L6219. Signals *clk* and *reset* have the same meaning as before. Input *mode* indicates three possible modes of movement, as explained in the following table:

mode(1:0)	Function
01	Irrespective of the present motor position, move up to position I.
10	Irrespective of the present motor position, move up to position F.
11	Move the number of steps (in the direction indicated by input <i>i2f</i>) equal to the value indicated by input <i>steps(6:0)</i> . Please note that motor position has to be between I and F at all times.

Input *steps(6:0)* indicate the number of steps that motor has to move in mode 11. Input *start*, active high, must have duration of one clock cycle (from rising edge to rising edge) and will be used to start the sequence corresponding to the mode selected. This signal will also be used to register internally the value of input *mode(1:0)*. Output *fi_comanda* (active high) has to become active when the system has completed the requested action initiated by input *start* and deactivate when input *start* becomes active. (40%)

Note: For this design, the system designed in question 3 may be used as a single component. Also consider that system is at position I at power-on.