Real-Time Imaging and Control
Practice

MSCV/ESIREM

Antoine Lavault
antoine.lavault@u-bourgogne.fr

VHDL 2 - Digital System Design in VHDL

The goal of this tutorial is to show how to describe more advanced components and systems.

Problem 1

FIR Filter
Let's consider an FIR filter implementing the filter whose coefficients are $(-8, -5, -5, -1, 1, 2, 2, 3, 5, 7, 7)$ (these coefficients are purely illustrative). 4-bit signed integers can represent these values. 4-bit signed integers will also represent the input x.

1. If we multiply two n-bit integers, what should be the minimum number of bits required to handle the product without overflow?
2. What should be the number of bits necessary to handle a sum of m n-bits ($m \geq 2$) integers?
3. Recall the equation verified by an FIR filter with an impulse response h, an input x, and an output y.
4. Implement a FIR filter in VHDL. The filter can be separated into a set of 2D shift registers, a set of adders, and a set of multipliers.

   To store the coefficients, we suggest you use the following construction:

```
type int_array is array (0 to scoffs-1) of integer range
 ↪  -2**(nbits-1) to 2**(nbits-1)-1;
constant coef: int_array := (-8,-5,-5,-1,1,2,2,3,5,7,7);
```

   To describe 2D shift registers, we suggest the following construction:

```
type signed_array is array (natural range <>) of signed;
signal example: signed_array(0 to ncoeffs-1)(nbits-1 downto 0);
```

   Finally, we suggest you use the `for...generate` statement to create the adders/multipliers.

Problem 2

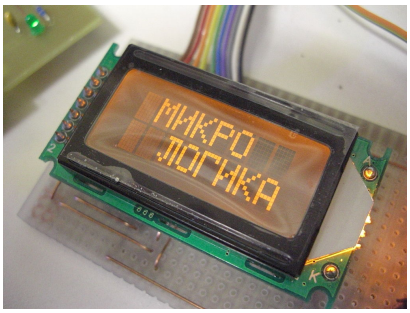

Figure 1: Example of HD44780-based display

Digital Watch with LCD Display
We will design two different circuits in this exercise:

- the timer module, outputing seconds (`secU, secT`), minutes (`minU, minT`) and hours (`hourU, hourT`). Its inputs are a clock and a reset. We will also add set buttons for seconds, minutes, and hours later.
- the LCD interface, whose inputs are the time values from the timer module, and its outputs are controls for the LCD.

1. Propose a component architecture for the timer module (in terms of basic functions, no VHDL required)
2. LCD has a special set of sequential instructions to follow to be initialized and ready to function. What kind of structure could we use to model this behavior?
3. The LCD can't work with high clock speeds. We will use a 500Hz clock obtained from the original 50MHz clock. Create a component dividing the clock frequency down to 500Hz.
4. To implement the setting buttons, we will use a different set of limits for the main 1s counter. In particular, we can set the hours by dividing the counter target by 8192, the hours by 256, and the seconds by 8. Why can we use divisions in this particular case?
5. Propose a VHDL description for the timer module. *Get some paper. It's going to be looooonnnngggg.*
6. Most low-cost applications will use an LCD controlled by a Hitachi HD44780U (or similar), as shown in fig. 1. Propose a finite state machine implementing the LCD interface, especially the initialization, with the 500Hz clock. You can assume it exists a `int_to_lcd` function converting an integer (between 0 and 9) into LCD compatible data and outputs ":" otherwise. *This question is extra hard and requires access to the documentation of the HD44780. We suggest you begin with schematics and diagrams before writing the VHDL implementation.*

Problem 3

Serial Interface - I2C ADC
Most modern ADC, like the ADS1115, are controlled via the I2C bus(Inter-Integrated Circuit, pronounced I-squared-C). The general structure of an I2C bus is depicted in fig. 2. Its two wires are called SCL (serial clock) and SDA (serial data), which interconnect one or more master units to several slave units. A standard ground wire (not shown) is also needed for the system to function.

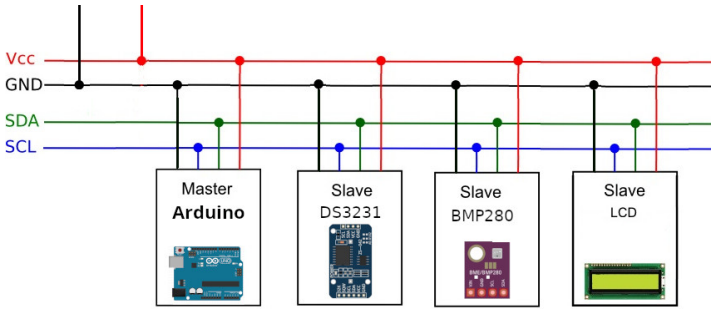

Figure 2: I2C Interface

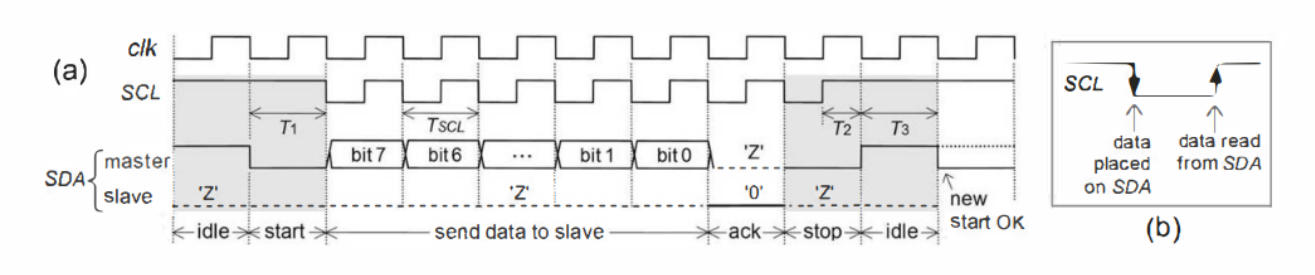

Figure 3: I2C Communication principle

As indicated in fig. 2, the clock (SCL), always generated by the master, is unidirectional, whereas the data (SDA) wire is bidirectional. Because SCL and SDA are open-drain lines (the 5Mbps version allows push-pull logic), external pull-up resistors must be connected between these wires and VCC/VDD. The number of devices sharing the same bus can be up to 128 (7-bit address) or 1024 (1O-bit address, rarer). Advanced features of the 12C protocol include bus arbitration, clock stretching, general calls, reset by software, and so on. Typical values for VCC/VDD are 3.3 V and 5 V, but devices with lower voltages, like 1.8 V, are also available.

Data transfers are done in groups of 8 bits, after which an acknowledgment bit is issued by the receiving end. The general principle is depicted in fig. 3, showing data transmission from master to slave. The start sequence consists of lowering SDA with SCL high (gray area), and the stop sequence (the other gray area) consists of raising SDA with SCL high. This means that during data transmission, SDA must remain stable while SCL is high; otherwise, start/stop commands might occur.

While the master is transmitting (always the most significant bit [MSB] first), the slave remains in high-impedance mode ('Z'), so the master has control over the SDA wire. After the eighth bit is sent, the master goes to the 'Z' state so that the slave can transmit its acknowledgment bit (= 0). In reading procedures, it is the master who emits the ack bit (= 0) after each byte received; however, if the reading includes multiple bytes, then the master sends a no-ack bit (= 1) after the last byte that it wants to retrieve. As depicted in fig. 3, data is always placed on the SDA wire (by either end) at falling clock edges and is read from it at rising clock edges (just like SPI, by the way).

It is necessary to respect the minimum values of the time parameters T_1 to T_3 and TSCL as shown in figure 3 (these are just the main time parameters). T_1 to T_3 are usually smaller than $T_{SCL}/2$, so $T_{SCL}/2$ or TSCL can be employed for those time windows (see fig. 3). As actual examples, the values of these parameters in the Maxim 11647 A/D converter device are T_1 = T_2 = 0.6us, T_3 = 1.3us, and TSCL = 2.5us; and in the NXP PCF8593 real-time clock device they are T_1 = T_3 =4.7us, T_2= 4 s, and TSCL = 10us.

The read/write procedure for the 11647 is shown in fig. 4.

1. We will use a 400kHz clock. Propose an architecture producing such a clock, with an input clock

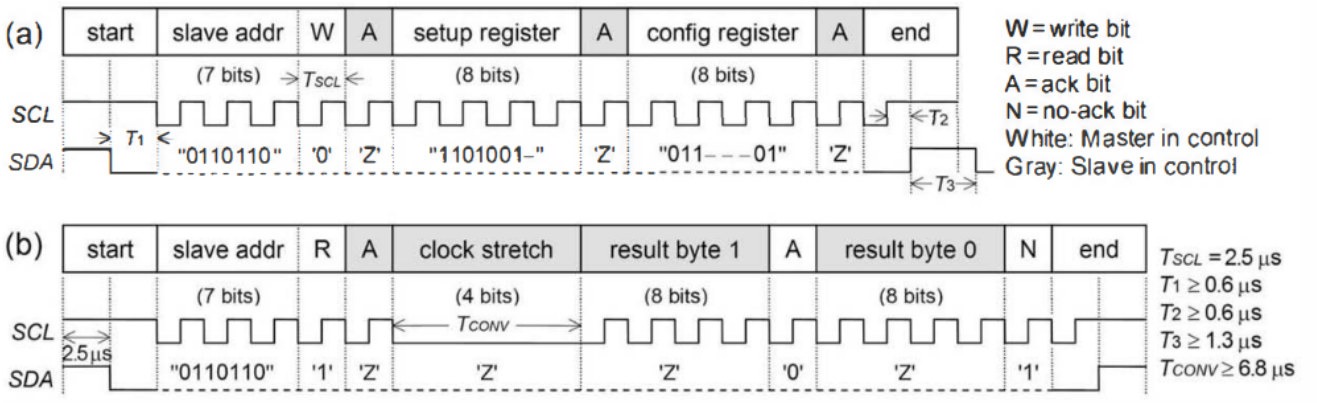

Figure 4: write and read procedures for the MAX1l647 ADC

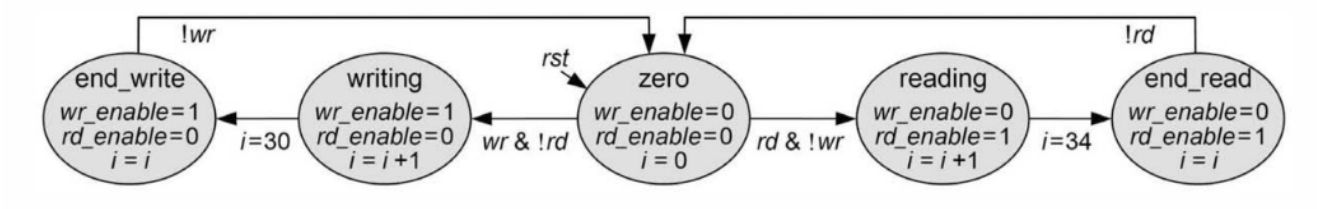

Figure 5: Pointer i implemented using a conventional recursive FSM

frequency being the same as on the Nexys 4 board. Is this clock speed compatible with the timing requirements of the I2C protocol?

2. Slots in gray indicate that it is the slave who is in control of SDA, so the master must retreat to the high-impedance condition ('Z'). How to implement this in VHDL? Same question with the do not care.

3. We suggest the following entity for the I2C interface:

```
entity I2C_interface is
generic (
    SLAVE_ADDRESS: std_logic_vector(6 downto 0):="fill me";
    SETUP_REGISTER: std_logic_vector(7 downto 0):="please";
    CONFIG_REGISTER: std_logic_vector(7 downto 0):="awkward.");
port (
    clk, rst: in std_logic;
    wr, rd: in std_logic;
    SCL: out std_logic;
    SDA: inout std_logic;
    received_data: out std_logic_vector(9 downto 0));
end entity;
```

Complete the generic value initializations with the right value.

4. To implement this kind of protocol, a good way is to use a pointer, which in this case means it is the "step" in the read/write process. And to generate this pointer, we can use an FSM like fig. 5. Choose your weapon, and describe a VHDL component creating the interface to the Maxim ADC.

Hint: don't jump into VHDL first. Make some drawings first. And annotate your exercise sheet.

Problem 4

Transition Minimized Differential Signaling
Transition minimized differential signaling (TMDS) is a line code for serial transmission of video signals introduced in 1999 by Silicon Image. It is used as part of the DVI (Digital Visual Interface) video standard for interconnecting desktop computers to LCD monitors and also as part of the high-definition multimedia interface (HDMI) standard for connecting high-definition video game consoles, set-top boxes, and so on to high definition TV (HDTV) monitors and video projectors.
The general architecture of TMDS links is shown in fig. 6. The transmitter consists of an 8B/10B encoder, a serializer, and current-mode logic(CML) input and output pins. The 8B/10B encoder converts an 8-bit word into a 10-bit word with fewer internal transitions, thereby reducing high-frequency emissions. It also provides near-perfect DC balance (number of ones close to the number of zeros) on the communication wires, improving noise margin. Be aware, however, that this 8B/1OB code is not the same as the original 8B/10B code introduced by IBM in the 1980s.

1. The *dena* input is also known as "display enable." What kind of basic combinational circuit can switch the output depending on the state of *dena*?
2. Write a VHDL function counting the number of ones in a `std_logic_vector`.
3. We would like to have *dout* to be a registered output. How will you do this in VHDL?
4. Describe a component implementing the algorithm shown in fig. 6. Does your circuit behave correctly against the examples of fig. 7? Your circuit can use the following entity description:

```
entity tmds_encoder is
port (
    tmds_clk: in std_logic;
    dena: in std_logic;
    din : in std_logic_vector(7 downto 0);
    control: in std_logic_vector(1 downto 0);
    dout: out std_logic_vector(9 down to 0));
end entity;
```

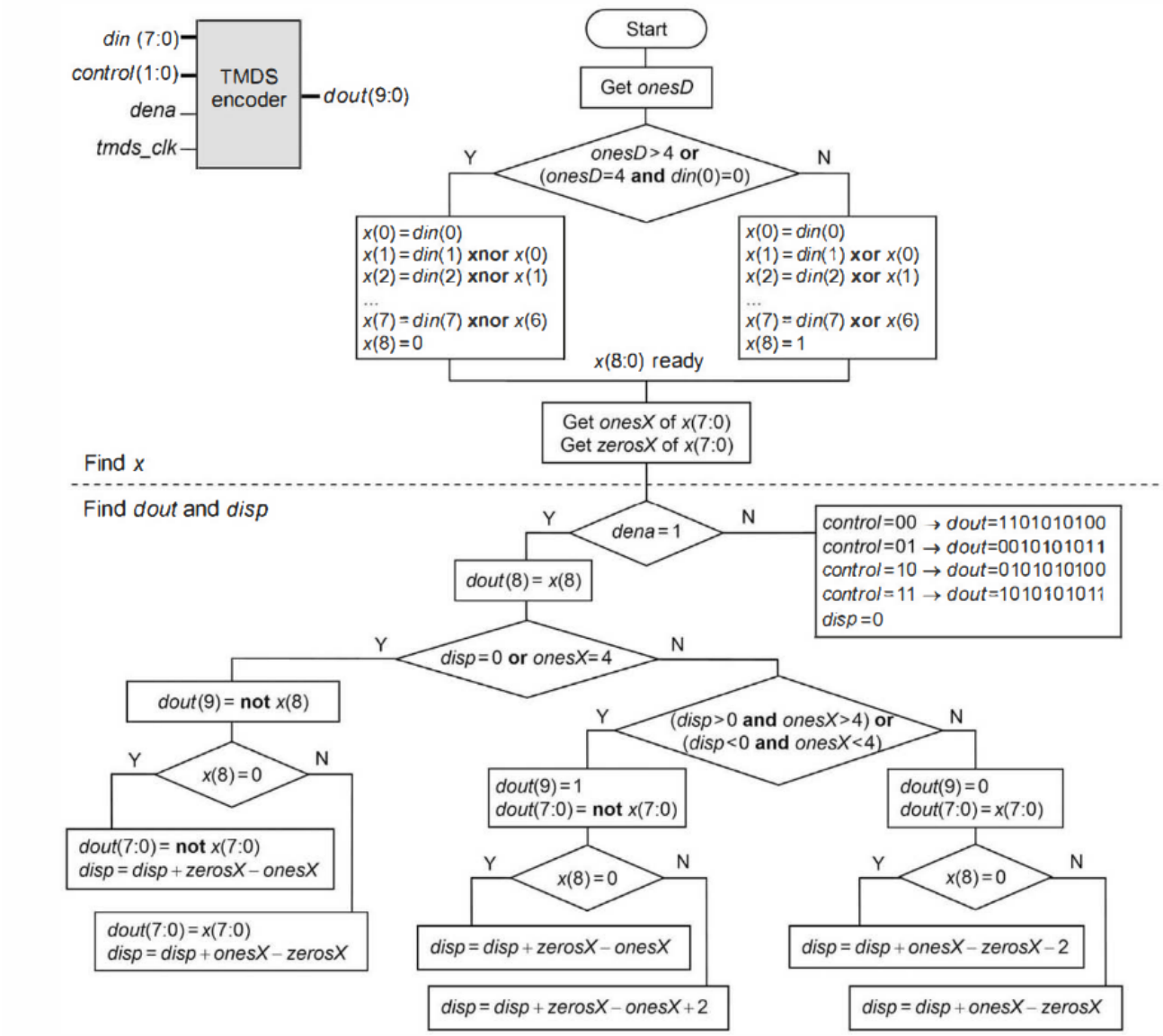

Figure 6: TMDS Algorithm

Input *din*(7:0)	*onesD*	Function	*dout*(8)	*x*(7:0)	*onesX*	Assumed disparity	Output *dout*(9:0)	New disparity
0000 0000	0	XOR	1	00000000	0	+2	01 0000 0000	−6
						0	01 0000 0000	−8
						−2	11 1111 1111	+8
1111 1111	8	XNOR	0	11111111	8	+2	10 0000 0000	−6
						0	10 0000 0000	−8
						−2	00 1111 1111	+4
0101 0101	4	XOR	1	00110011	4	+2	01 0011 0011	+2
						0	01 0011 0011	0
1010 1111	6	XNOR	0	11001111	6	+2	10 0011 0000	−2
						−2	00 1100 1111	0

Figure 7: TMDS Examples