Advanced Digital Systems

Lab Assignment 5

Design of a data acquisition and representation system

0. Goals

The goal of this laboratory assignment consists the design of a data acquisition and representation system with monitoring capabilities. The design will contain the hardware controllers for the ADCS7476 A/D converter and the VGA display developed in previous laboratory assignments. In order to facilitate the development, it is recommended to start first with the design of the hardware controller in charge of representing signals on the display and then complete it with the monitoring application. As an optional work, several improvements of the functionality of the system will be proposed.

1. Data acquisition and representation unit

The block diagram of the system to be designed is presented in Figure 1. For the sake of clarity the clock signal has been omitted in all the system components.

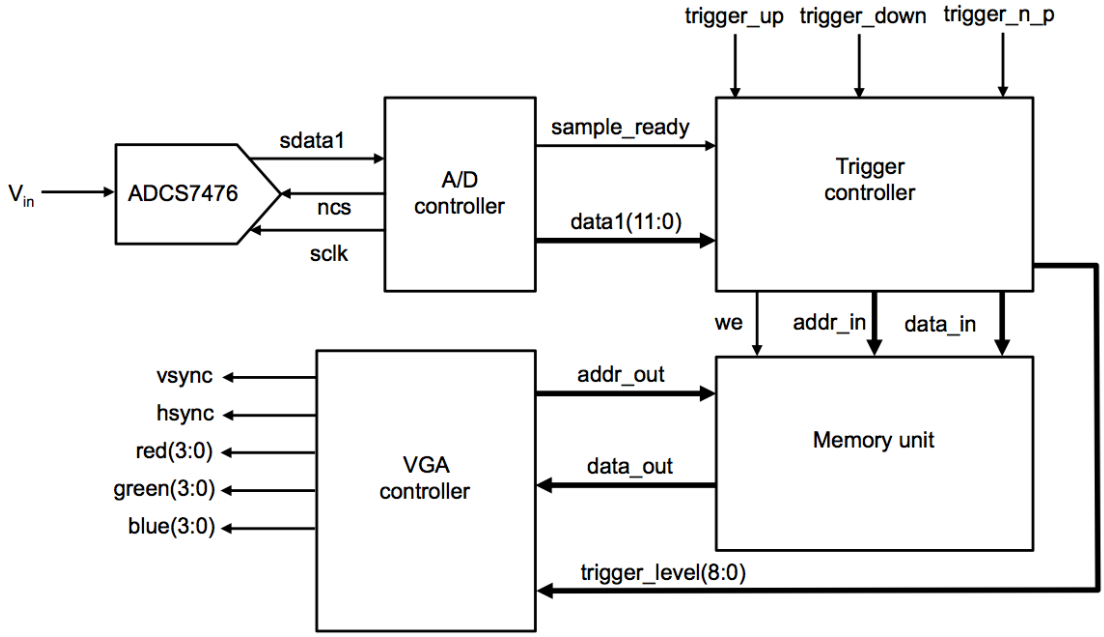

Figure 1. Block diagram of the data acquisition and representation system.

In this block diagram de A/D controller and the VGA controller correspond to the systems designed in previous laboratory assignments. The next sections will describe the functionality of the remaining components.

1.1. Trigger controller

This block is responsible for determining the time at which the acquisition of the data provided by the A / D converter should start. To do so it has an internal signal used to compare the value provided by the A / D converter. To make the comparison only the 9 most significant bits of the value provided by the converter will be used, to minimize the effects of noise. This internal comparison level will have an initial value of 256 (100000000 in binary), and the input signals *trigger_up* and *trigger_down* will increase or decrease the value, respectively, as explained later.

To ensure that the input signal is always recorded from the same point, in addition to comparing the current value provided by the converter is necessary to consider what was its previous value. For this reason, this subsystem must contain an additional register that stores the value provided by the converter in each conversion. Thus, if the current value of the converter is equal to the reference level and its previous value was below this value it will be possible to capture the signal when it has a positive slope, whereas if the previous value was greater than the internal reference level the acquisition is performed when the signal has a negative slope. To make the comparison with the previous value provided by the converter only the four most significant bits will be used in order to minimize the effects of noise in the input signal. The input signal *trigger_n_p* can choose if the signal is captured from a positive or negative slope. This signal will come from a button located on the ZedBoard (button labelled BTNC, see later) so that whenever acting on this button will change the mode of trigger (positive/negative slope). Since it is an external signal it will be necessary to synchronize it before it can be used by the system. When the system is in its initial state (after a reset) acquisition must be done with a positive slope of the input signal.

As mentioned above, inputs *trigger_up* and *trigger_down* permit to control the level used to compare the value provided by the A/D converter. These signals come from pushbuttons located on the development board, and therefore their values have to be synchronized before they can be used by the system. The signal *trigger_up* comes from the button labelled BTNU in the ZedBoard, and each time this button is pressed the level of internal comparison should be increased by 16 units. The signal *trigger_down* comes from the button labelled BTND in the ZedBoard, and each time this button is pressed the level of internal comparison should be decreased by 16 units.

The buttons used to control the application are placed in the bottom right corner of the ZedBoard. Figure 2 represents these buttons and their corresponding labels.

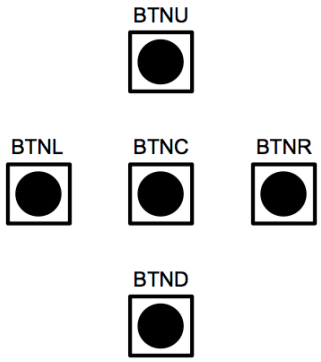

Figure 2. Organisation and labels of the buttons used to control the application.

In order not to interfere with the representation of the data on the TFT screen the acquisition of the input signal must occur once for each frame, specifically at the moment no lines on screen are represented. That is, the data acquisition must start when the *vsync* signal is at level 0.

This subsystem will also provide data and control signals to the writing section of the memory unit that will store one capture of the input signal. This memory unit, the operation of which will be explained later, is a dual-port memory, i.e., has separate read and write ports. The trigger controller will handle the write address bus (*addr_in* signal

in Figure 1), the write data bus (signal *data_in* Figure 1) and the write enable signal (signal *we* figure 1). The signal *we,* active at high level, must have a duration of one clock cycle of the system, and must be activated when the write address and data are stable.

Since the display used to represent the data has a horizontal resolution of 1280 pixels this will imply to store in the memory unit 1280 samples of the input signal. Therefore, it is important that the write address bus comes back to its initial value once the acquisition of these samples has been completed.

1.2. Memory unit

To facilitate the implementation of the proposed system a dual port memory (i.e., with independent read and write ports) will be used that is available in the dedicated internal memory (BlokRAM) of the Zynq device. Thus, it will be possible to separate the write process that is carried out by the trigger controller from the read processes that is performed by the VGA controller. The description of this memory unit, which is contained in the file *sync_ram_dualport.vhd* provided in the laboratory, is as follows:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity sync_ram_dualport is
generic (
            data_width : natural := 32;
a           ddr_width : natural := 5);
port(
            clk_in : in std_logic;
            clk_out : in std_logic;
            we : in std_logic;
            addr_in : in std_logic_vector(addr_width - 1 downto 0);
            addr_out : in std_logic_vector(addr_width - 1 downto 0);
            data_in : in std_logic_vector(data_width - 1 downto 0);
            data_out : out std_logic_vector(data_width - 1 downto 0)
);
end sync_ram_dualport ;

architecture rtl of sync_ram_dualport is

type    mem_type    is    array    (2**addr_width    downto    0)    of
std_logic_vector(data_width - 1 downto 0) ;

signal mem : mem_type:=(others=>(others =>'0'));

begin

write : process (clk_in)
begin
      if (clk_in'event and clk_in = '1') then
            if (we = '1') then
                  mem(conv_integer(addr_in)) <= data_in ;
            end if ;
      end if ;
end process write ;

read : process (clk_out)
begin
```

```
      if (clk_out'event and clk_out = '1') then
            data_out <= mem(conv_integer(addr_out)) ;
      end if ;
end process read ;

end rtl ;
```

As it can be deduced, it is a parameterizable description, from which it is possible to define memory units of any depth and width. In the system to be designed the data width is 12 bits and the address width is 11 bits (since it is necessary to store 1280 samples of a signal).

entenc que ho hem de canviar no?

This memory unit is comprised of two separate processes, one for reading and one for writing, they can even work with independent clock signals. In the system to be designed both clocks must be driven by the global clock signal of the system.

aixo també s'hauria de canviar perque pel que entenc ara els clocks son clk_in and clk_out...?

encara que aixo potser esta bé pero com no puc veure a on estan conectades les entrades... :(

1.3. VGA controller

The core of this subsystem has been already developed in a previous laboratory assignment. Now it will be necessary to modify it in order to provide the functionality required by the application.

In this way, the information to be represented in the display is not pre-fixed, but depends on the values acquired from the external signal. Therefore, this controller must generate the read address bus of the memory unit.

The display used has a vertical resolution of 1024 lines. To facilitate the process of representation the first 512 lines will be used to represent the input signal. While a line is being displayed the VGA controller will scan the memory unit, increasing the read address bus for each new pixel necessary to represent on the screen. The content of a memory location corresponds to the vertical position of the pixel that should represent its value. Therefore, comparing the data read from memory (in fact, its most significant 9 bits, since the screen will only display 512 different levels) with the line number can easily determine whether a given pixel should be represented or not. To facilitate the visualisation, it is recommended that active pixels have a yellow colour (red and green colour channels with maximum value and blue colour channel with minimum value).

Additionally, the VGA controller has to represent the trigger level on the display. This will be done by means of a blue horizontal line with a length of 20 pixels.

Therefore, the data acquisition and representation system will have an input/output interface encompassing the following signals:

- **clk:** Global clock input of the system. It will have a frequency of 108 MHz and acts on its rising edge on the storage components of the system.
- **resetn:** Global synchronous reset of the system, active low.
- **trigger_n_p:** This input indicates if the acquisition of the external signal is performed with a positive or negative slope.
- **trigger_up:** The activation of this signal produces an increment of 16 units in the internal reference level against which the external signal is compared.

- **trigger_down:** The activation of this signal produces an decrement of 16 units in the internal reference level against which the external signal is compared.
- **vsync:** Frame (vertical) synchronisation signal for the display.
- **hsync:** Line (horizontal) synchronisation signal for the display.
- **red(3:0):** 4-bit output signal that indicates the intensity of the red colour channel for a pixel to be displayed.
- **green(3:0):** 4-bit output signal that indicates the intensity of the green colour channel for a pixel to be displayed.
- **blue(3:0):** 4-bit output signal that indicates the intensity of the blue colour channel for a pixel to be displayed.
- **sdata1:** Serial data input corresponding to the first A/D converter.
- **sdata2:** Serial data input corresponding to the second A/D converter (not used in the design).
- **sclk:** Reference clock signal for the converters. It has a frequency of 108 MHz.
- **ncs:** Chip select signal for the converters, active low.

2. Monitoring application

The design presented in the previous section will be augmented in order to provide temperature monitoring capabilities. For this purpose the temperature sensor included in the XADC component included in the Zynq device will be used.

The hardware design specified in the previous section should be modified so as to represent in the display the current temperature of the Zynq device and a threshold for this temperature. The temperature will be displayed by means of a solid horizontal bar of green colour. This horizontal bar has a width of 30 lines and is separated from the signal represented in the display by 30 lines. A vertical line of blue colour will display the threshold for the temperature. This line has a width of 35 lines.

Dins del if de estem a la columna que toca, en faria un altre adins per fer un check de la horitzontal i el color

Faria una série de if amb la vertical

The temperature range to be displayed is [0º C, 80º C], and the resolution in the display should be 1º C. If the temperature in the Zynq device is higher than the threshold the hardware design should stop displaying the captured signal and the temperature bar has to be displayed in red colour.

again, tot amb if

The hardware design specified in the previous section has to be modified in order to include three new inputs to its interface:

- **alarm:** This input indicates if the temperature in the Zynq device is higher ('1') or lower ('0') than the threshold.
- **temperature(10:0):** This input indicates the right limit on the display for the bar representing the temperature.
- **t_temperature(10:0):** This input indicates the position in the display for the line representing the temperature threshold.

These three inputs will be provided by a software application running in the processing system of the Zynq device. The software application is supported by the FreeRTOS operating system, and can be constructed as it was described in laboratory assignment 2.

The application will read the values of the BTNR and BTNL buttons (Figure 2) in order to modify the temperature threshold through an *AXI GPIO* peripheral, as it was described in laboratory assignment 1. Each time the BTNR button is pressed the threshold has to be augmented by 1º C, while each time the BTNL button is pressed the threshold has to be decremented by 1º C. The management of the GPIO peripheral has to be done by means of a polling technique (i.e., its interrupt has to be disabled). The *AXI GPIO* peripheral should be defined just with an input channel of two bits. Since the component will have a 2-bit input bus and the block diagram will have two independent input ports, it is recommended to use the *Concat* component of the Vivado standard IP libraries. This component will merge on a single 2-bit bus two independent signals coming from two input ports.

The software application to be developed will consists of two tasks:

- **User interface task:** This task reads the current value of the BTNR and BTNL buttons and updates the temperature, threshold and alarm values to be sent to the hardware controller.
- **Monitoring task:** This task reads the current temperature of the Zynq device and updates the temperature, threshold and alarm values to be sent to the hardware controller.

The functions to be used in order to read the temperature of the Zynq device will be provided in the laboratory in a file called *app.c*.

In order to provide a correct name to the external ports of the system, it is important to bear in mind the contents of the *lab5.xdc* file that will be provided in the laboratory:

```
set_property PACKAGE_PIN AB11 [get_ports {ncs}];           # "JA7"
set_property PACKAGE_PIN AB10 [get_ports {sdata1}];        # "JA8"
set_property PACKAGE_PIN AA8  [get_ports {sclk}];          # "JA10"
set_property PACKAGE_PIN Y21  [get_ports {blue[0]}];       # "VGA-B1"
set_property PACKAGE_PIN Y20  [get_ports {blue[1]}];       # "VGA-B2"
set_property PACKAGE_PIN AB20 [get_ports {blue[2]}];       # "VGA-B3"
set_property PACKAGE_PIN AB19 [get_ports {blue[3]}];       # "VGA-B4"
set_property PACKAGE_PIN AB22 [get_ports {green[0]}];      # "VGA-G1"
set_property PACKAGE_PIN AA22 [get_ports {green[1]}];      # "VGA-G2"
set_property PACKAGE_PIN AB21 [get_ports {green[2]}];      # "VGA-G3"
set_property PACKAGE_PIN AA21 [get_ports {green[3]}];      # "VGA-G4"
set_property PACKAGE_PIN AA19 [get_ports {hsync}];         # "VGA-HS"
set_property PACKAGE_PIN V20  [get_ports {red[0]}];        # "VGA-R1"
set_property PACKAGE_PIN U20  [get_ports {red[1]}];        # "VGA-R2"
set_property PACKAGE_PIN V19  [get_ports {red[2]}];        # "VGA-R3"
set_property PACKAGE_PIN V18  [get_ports {red[3]}];        # "VGA-R4"
set_property PACKAGE_PIN Y19  [get_ports {vsync}];         # "VGA-VS"
set_property PACKAGE_PIN P16 [get_ports {trigger_n_p}];    # "BTNC"
set_property PACKAGE_PIN R16 [get_ports {trigger_down}];   # "BTND"
set_property PACKAGE_PIN N15 [get_ports {temp_down}];      # "BTNL"
set_property PACKAGE_PIN R18 [get_ports {temp_up}];        # "BTNR"
set_property PACKAGE_PIN T18 [get_ports {trigger_up}];     # "BTNU"
```

3. Optional improvement of the design

It is possible to improve the design in order to add new capabilities. Some possible improvements could be:

- Represent the signal centred in the display.
- Modify the vertical scale of the display.
- Modify the horizontal scale of the display.
- Provide a measurement for the frequency/period of the signal.