Universitatea Tehnica Cluj-Napoca
Facultatea de Automatica si Calculatoare
Sectia Calculatoare
Anul 2015-2016
Semestrul II

DIGITAL SYSTEM DESIGN PROJECT: PS2 Controller

Coordinator: Spatariu Diana
Students: Dan Mitrea
Andrei Tudorica
Grupa: 30411

CONTENTS

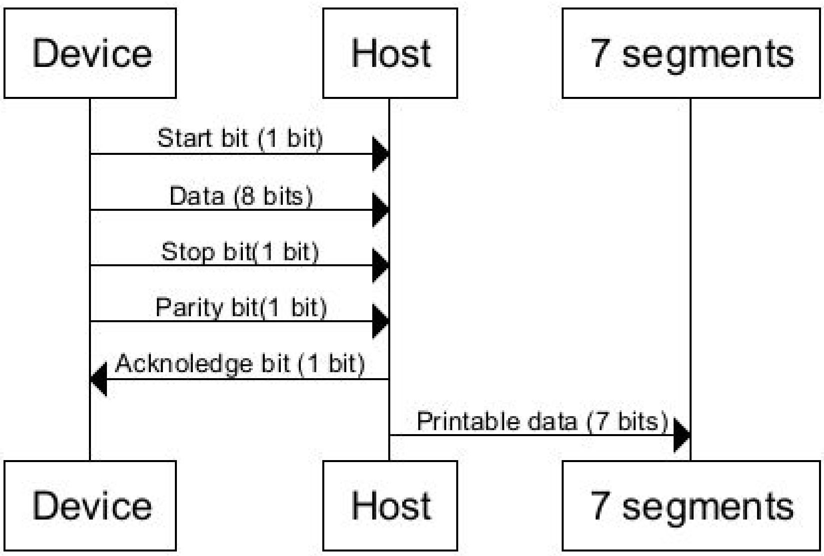
1.Problem statement

Design a PS2 Keyboard controller. You should read the keys and display the corresponding characters on the 7-segments display.

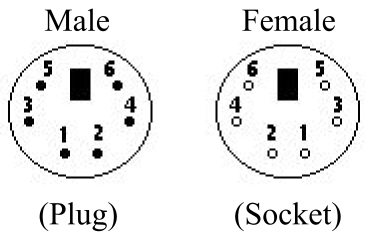
2.Controller description

The PS/2 controller can be used to communicate with a PS/2 keyboard . It provides an interface to the PS/2 protocol, handling the data transmission and the timing control. The PS/2 controller interface consists of the PS/2 clock and the PS/2 data input(11 bits). The 11-bit data port is for receiving data from the PS/2 device . All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11.
These bits are:
-1 start bit. This is always 0;
-8 data bits, least significant bit first;
-1 parity bit (odd parity);
-1 stop bit. This is always 1;

Data transfer protocol

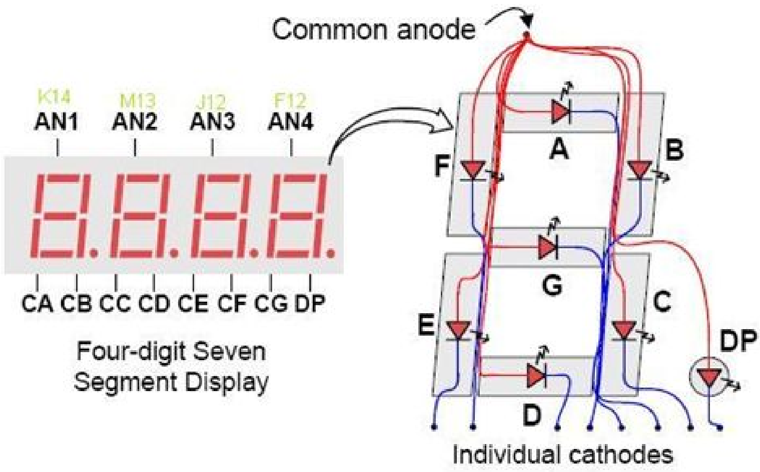


The keyboard is connected to the FPGA board through the PS/2 port.



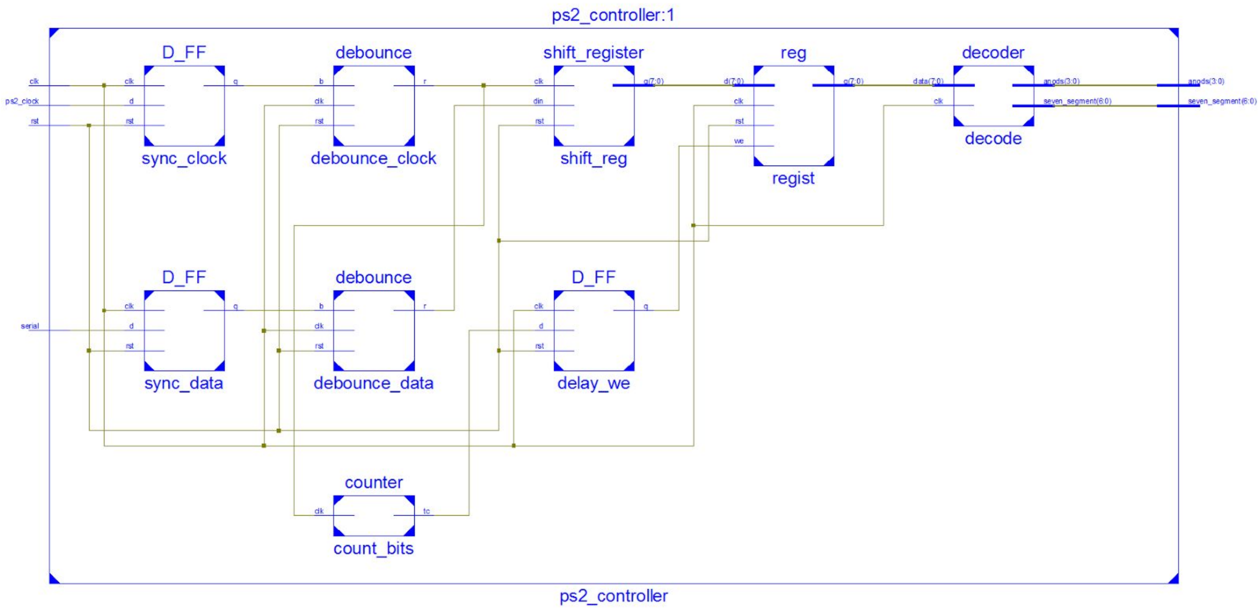
6-pin Mini-DIN (PS/2):
1 - Data
2 - Not Implemented
3 - Ground
4 - Vcc (+5V)
5 - Clock
6 - Not Implemented

When a button is pressed on the keyboard, it will generate a 11-bit code from which the 8 data bits are decoded and sent to the 7-segment LED display.The character pressed on the keyboard will then appear on the LED display.
The 7-segment display is 0-active.



For example when the key ’A’ is pressed the keyboard will send the following code ”00011100” along with the start bit,parity bit and stop bit.The code is then decoded and the letter ’A’ is shown on the LED display.

3.Block scheme



(larger scheme in)

4.Design

3.1. Component list:

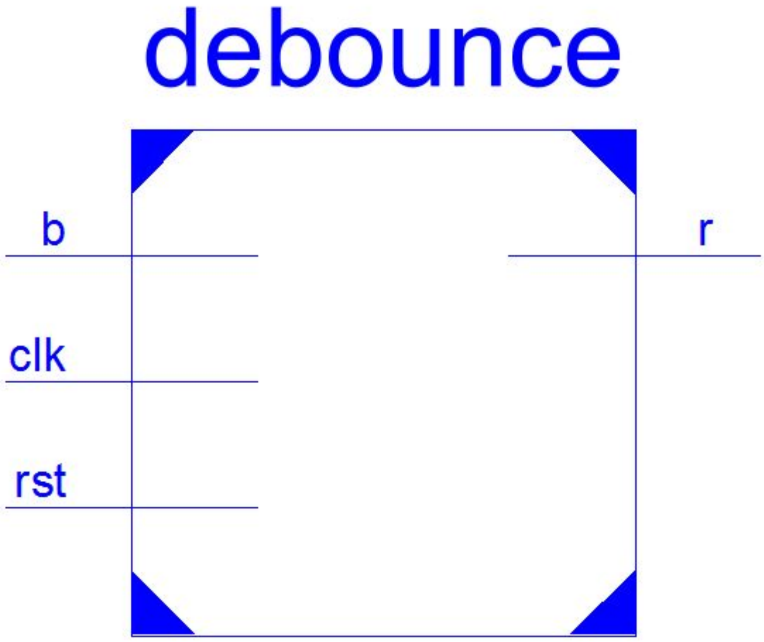
- Debouncer
- D Flip-Flop
- Counter
- Shift Register
- Register
- Decoder

3.2 Codes for the components + descriptions

We have used the IEEE library and packets 1164 and unsigned before each entity.

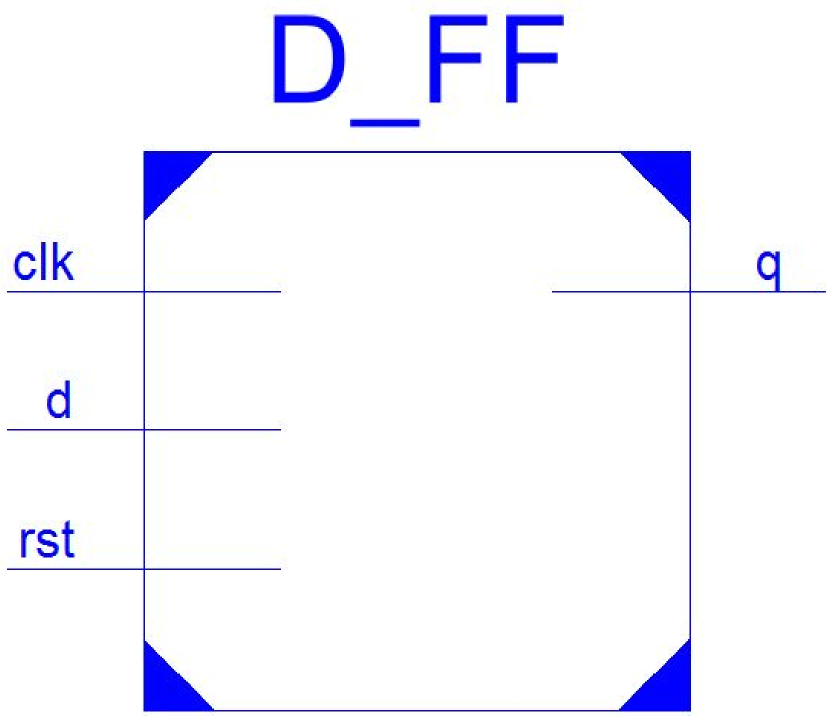
```
library  IEEE;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_1164.all;
```

1.Debouncer



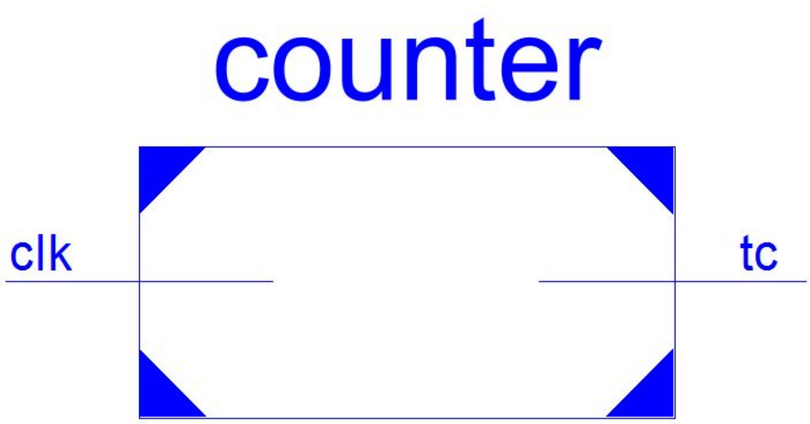
This component is used twice in the project, to debounce The serial data and the clock received from the keyboard. It is used to get rid of the noise that might appear during the transmission of the data. If a certain bit received is stable for 2^10 clock impulses, than no noise was encountered, otherwise the counter is restarted.

2. D Flip-Flop



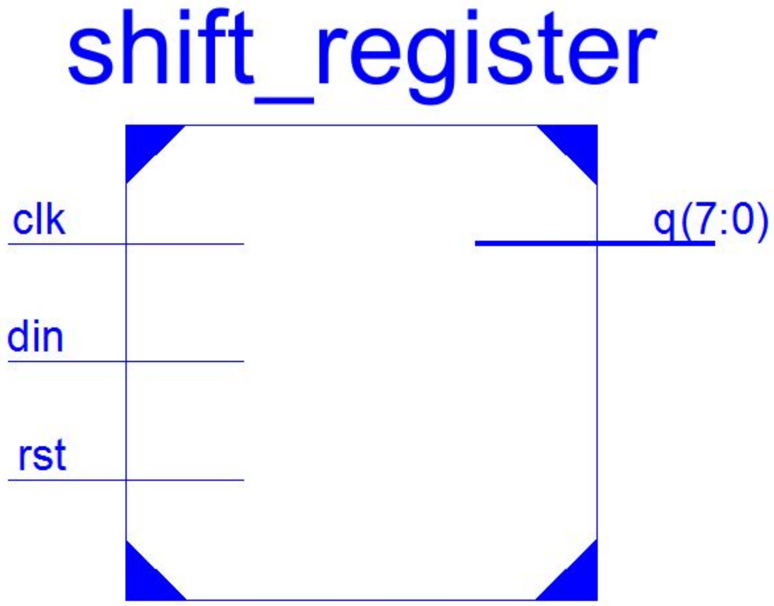
This component is used three times in the project. First two of them are are used to synchronize the data and the clock received from the keyboard, before the debouncing and the third time it is used to delay with one clock signal the ‘rst_we’ output of the counter that signals a word was completed.

3. Counter



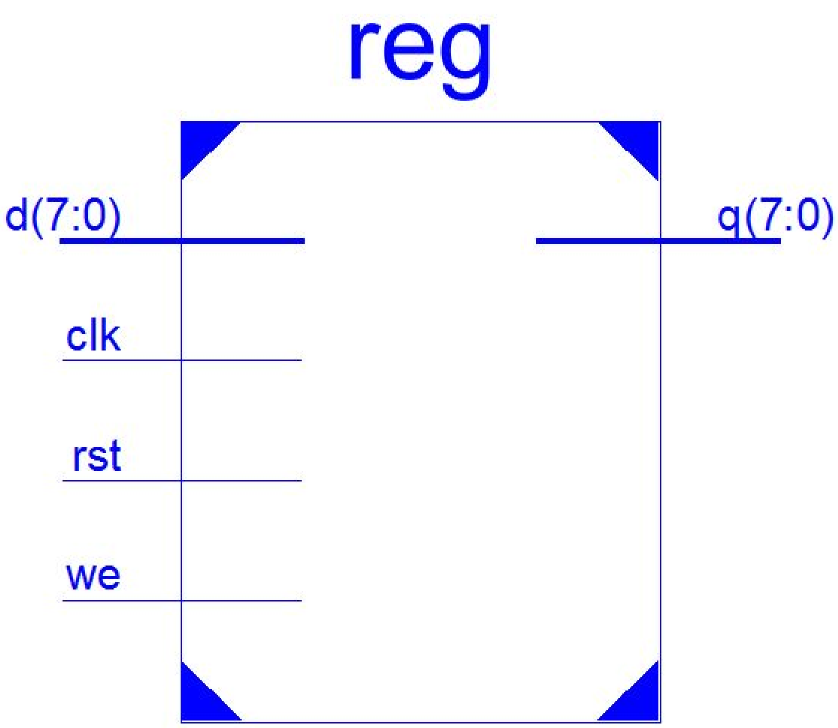
This component is used only once in the project to count the synchronized and debounced clock signals received from the keyboard. It has one output that is activated when the counter reaches 11 to signal that we completed a word.

4. Shift Register



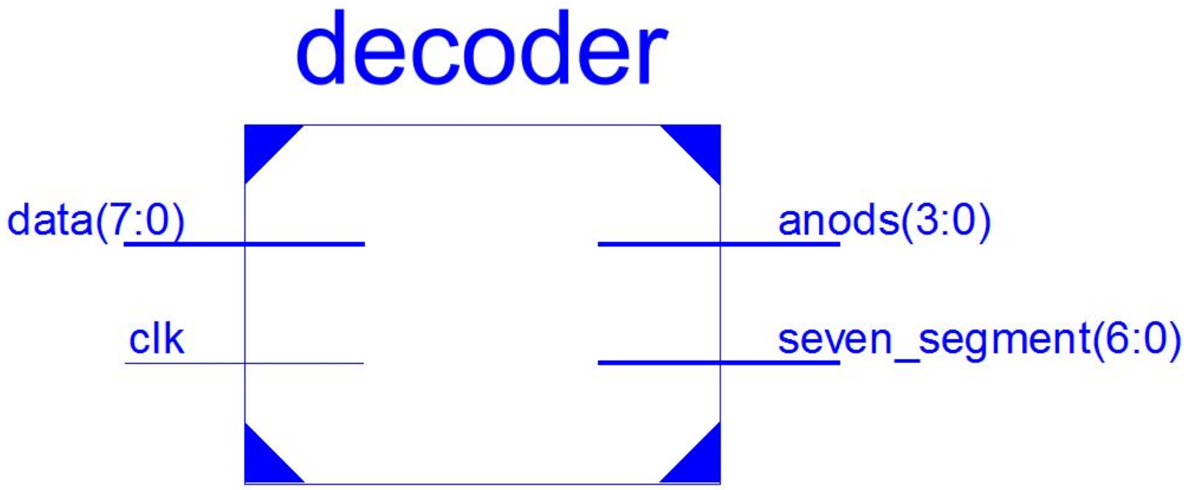
This component is used once in the project to transform the serial bits into parallel words containing only the data bits sent by the keyboard.

5. Register



This component is used nce in the project to memorize the parallel data sent by the shift register and deliver it to the decoder.

6.Decoder



This component is used to convert (decode) the 8-bit words from the register into 7-bit words used for the 7-segment display.

5.List of inputs, outputs and intermediary signals

Inputs:

Serial - the serial data received from the keyboard
Ps2_clock - the keyboard clock
Rst - reset button

Outputs:

Anods - 4-bit vector used to activate the anods of the 34 7-segment displays
Seven_segment - 7-bit vector to be sent to display

Internal signals list:

Ps2_clock_sync - ps2_clock after the synchronization
Serial_data - serial after the synchronization
Ps2_clock_debounced - the ps2_clock_synk after debounce
Serial_data_debounced - the serial after the debouncing
rst_we - output of the counter, activates only at the 11th clock pulse
int_shift_mem - output of the shift register
Int_mem_dec - output of the register to the decoder
Int_reset - interior reset for when a word ends

6.Application notes and uses

Xilins and FPGA implementation

In Xilinx you can use the project like this::

1.Open the Xilinx ISE program.

2.Create a new Workspace

3.Click on ”User constraints” 

, then select “Edit constraints” 

,where we declare the connection between the buttons and LEDs to the signals in the code.

In our case :

```
NET "clk" LOC=B8;
NET "anods<3>" LOC=K14;
NET "anods<2>" LOC=M13;
NET "anods<1>" LOC=J12;
NET "anods<0>" LOC=F12;
NET "seven_segment<6>" LOC=L14;
NET "seven_segment<5>" LOC=H12;
NET "seven_segment<4>" LOC=N14;
NET "seven_segment<3>" LOC=N11;
NET "seven_segment<2>" LOC=P12;
NET "seven_segment<1>" LOC=L13;
NET "seven_segment<0>" LOC=M12;
NET "ps2_clock" LOC=B1;
NET "serial" LOC=C3;
NET "rst" LOC = A7;
```

4.Clik on “Generate Programming File”

5.Open the program „Adept”, used to implement the FPGA project.

6.Click on „Initialize Chain”

7.Look for the program clicking on „Browse” and then look for the file with the **.bit** extension.

8.Click on „Program”.

9.Press a key and watch as it is displayed on the 7-segments LED. Below you have a list of the characters that will be displayed: ‘0’, ‘A’, ‘C’, ‘E’, ‘F’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘b’, ‘d ’, ‘g’, ‘h’, ‘i’, ‘J’, ‘L’, ‘P’, ‘t’, ‘U’, ‘o’.

10.We also have a “reset” button, which can be pressed in order to reset the whole system, and the display will show a ‘-’ character until another key is pressed.

7. Further development

In what concerns the further development of the project certain elements can be improved.

A shift register can be implemented on the 7-segment displays, such that the display always shows the last 4 entered characters.

A VGA controller can be implemented, such that the symbols can be printed more accurately. This way the entire keyset can be added to the internal memory. Even upper and lower case can be printed, so the “Shift” and “Caps Lock” keys can be also implemented;

8. Bibliography

The ps2 protocol - http://www.computer-engineering.org/ps2protocol/
The PS/2 Keyboard Interface - http://www.computer-engineering.org/ps2keyboard/