FPGA and mouse application

DSD first year project

3.06.2025

Modi Raul & Rotea Dragoș

PROJECT SUPERVISOR: Ing. Ioana Virna

Table of Contents

FPGA and mouse application

1. Specifications

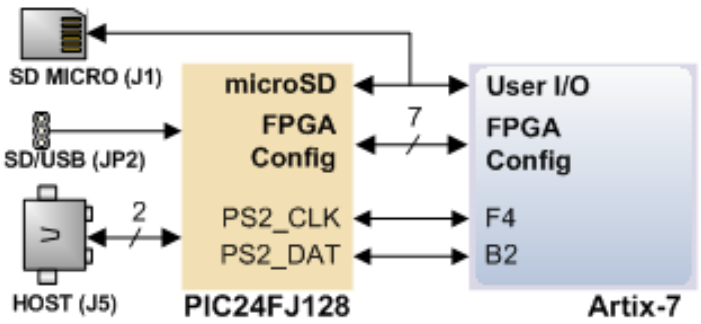
Implement an application that allows the user to count the number of mouse clicks.

Functional requirements:

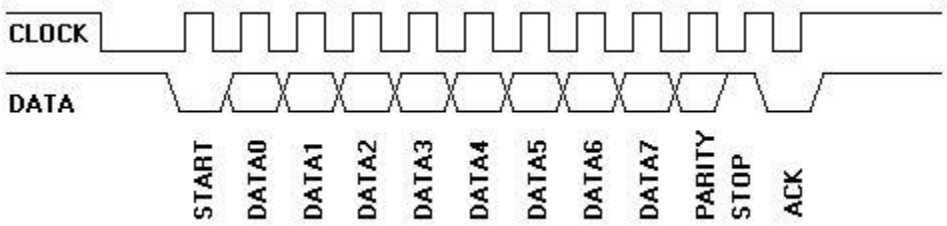
1. Existing RESET button that will clear the SSD display (status “0000”)
2. The current status is displayed on the SSD
3. Pressing the left mouse button will increase the current status
4. Pressing the right mouse button will decrease the current status
5. An IS_LEFT LED is lit, marking the fact that the left click increases and the right click decreases
6. A REVERSE button / switch is used to reverse the mouse functions, IS_LEFT will turn off when pressed

2. Problem Analysis

The Nexys A7 board has an Auxiliary Function microcontroller (Microchip PIC24FJ128) that provides the Nexys A7 with USB Embedded HID host capability. It hides the USB HID protocol from the FPGA and emulates and old-style PS/2 bus when connected to a keyboard or mouse.



The PS/2 protocol has 2 serial buses, one for clock and one for data. This is used to transmit 11-bit words that include a start and stop bit, odd parity bit and a data byte. The clock and data signals are only driven when data transfers occur; otherwise, they are held in the idle state at high-impedance (open-drain drivers). The Data line changes state when Clock is high and that data is valid when Clock is low.

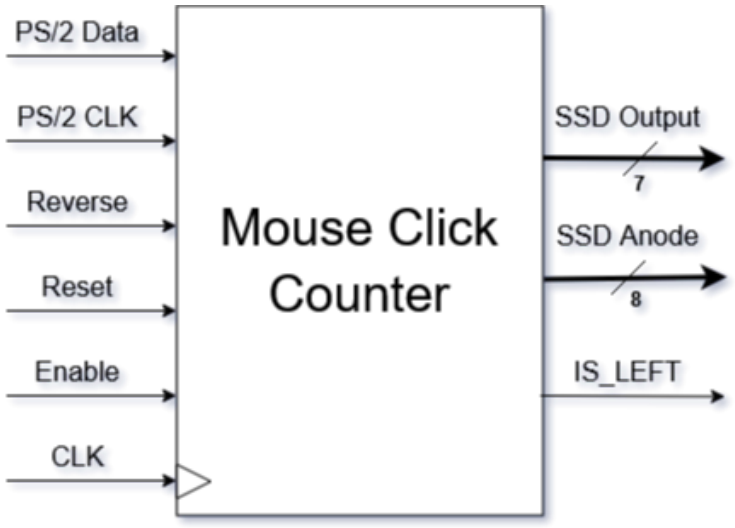


Once entered in stream mode and data reporting is enabled, the mouse outputs a clock and data signal when it is moved; otherwise, these signals remain at logic '1.' Each time the mouse is moved, three 11-bit words are sent from the mouse to the host device. The 1st byte contains data regarding the status of the mouse buttons, as well as the sign and overflow status of the x and y axis. The 2nd and 3rd (4th if the mouse has a scroll wheel) bytes store the movement delta on the X,Y (and Z) axis. Microsoft® IntelliMouse®-type extensions for reporting back a third axis representing the mouse wheel are also supported, where different command bytes represent different signals sent to and from the mouse.

3. Design

3.1. Black Box

This first step can be a complicated one, there are inputs and outputs that *may not be specified in the text* but which the designer must add in order to implement the functionality on the board (eg: *enable* switch that permits the memorisation of the last recorded counter value when it is set to zero. The enable is not explicitly specified in the text but it helps at implementation).



3.2. Resources

Resources can be **simple circuits**, which can be implemented directly (counter, register, etc.) or **complex resources** (remainder algorithm, multiplication algorithm, etc.). These complex resources may appear in the first breakdown with black boxes to which we must establish inputs and outputs, but later they must be further broken down until we reach known circuits.

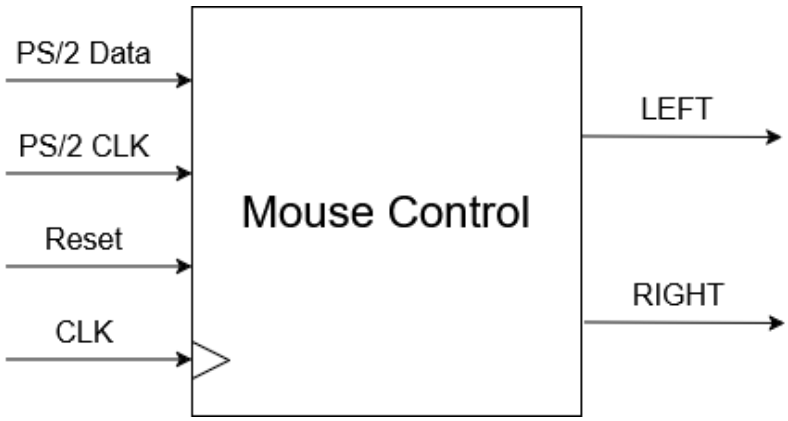
1. 8-bit to Seven Segment Display decoder

The decoder takes an 8-bit input from the counter and converts it into a 3 digit integer. It then takes each separate digit of the number and outputs it to a different display digit. This is achieved by using a selection signal that cycles through MODULO 3 values on the ticks of a clock-divider that takes the system clock as input. For every value of this signal it sends the necessary values to display each associated digit on the cathode and the signal needed to activate the right display-digit on the anode.

2. Mouse Controller

The Mouse Controller contains an internal component that (Ps2Interface) that:

- Debounces the PS/2 data and clock signals and cleans them up
- Uses a shift register to store the 11-bit data packets transmitted through the PS/2 interface
- Memorizes the current state of the port (which is needed when outputting signals write operations, error management, and handshake protocols)
- Checks for errors by using the parity bit and ROM. (For 8-bit data packets, the correct parity bit is at stored in the ROM at the address with the value of the 8-bit number analyzed)
- Sends "busy", "reading" or "error" signals to mouse controller



The mouse controller takes the 11-bit packets from the PS/2 interface. By sending a command to the mouse to get the device ID it checks if the mouse has a scroll wheel or not. If the mouse has a scroll wheel, it sends 4 data-bytes per cycle instead of the usual 3.

It periodically checks if the mouse has any activity and places it in idle mode if not.

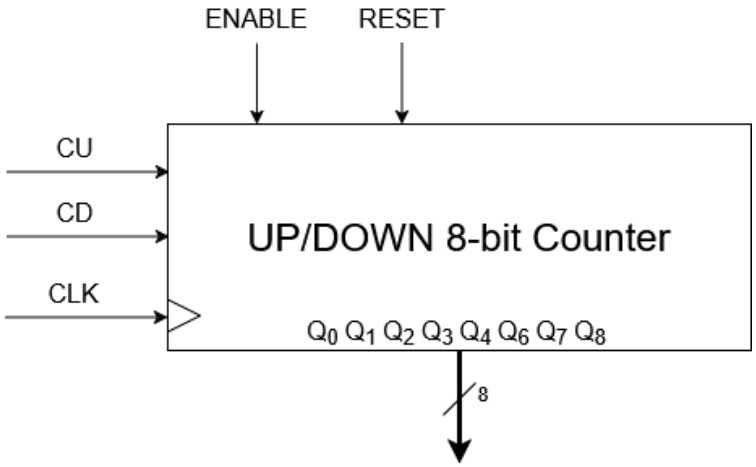
When encountering an error, it returns to the idle state and sends a reset signal to the Ps2Interface component.

It cycles through every data byte sent by the Ps2Interface and for the first byte of the set analyses the bits that signal if the left or right mouse buttons are pressed.

3. 8-bit counter

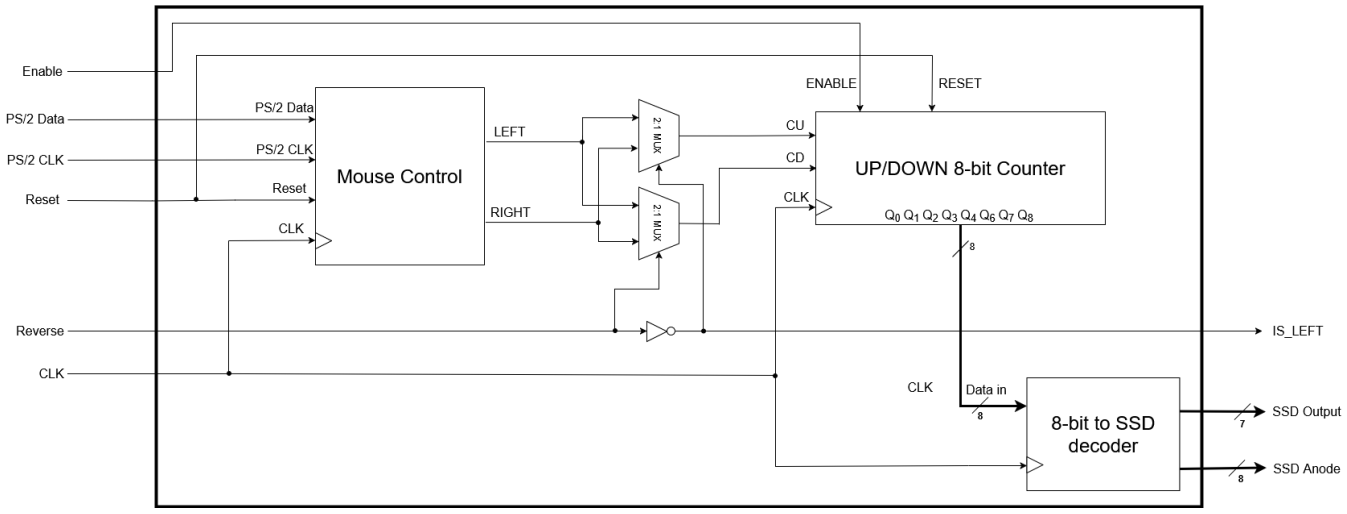
The 8-bit counter takes 2 inputs (CU, CD) that are internally debounced which increment and decrement the number of clicks. It stores values of up to 255. The enable input allows the

memorization of the current value. There is also an asynchronous reset button that sets the number stored to 0.



3.3. Detailed diagram

For simplicity, the IS_LEFT Led takes the direct inverted input of the REVERSE input. The two 2:1 multiplexors choose whether the left mouse button is the CU input of the counter and the right mouse button is the CD input or the other way around. The system clock and reset button is shared by all the components that need it. The enable switch is connected directly to the counter.



4. User manual

The user should connect a USB mouse to the port on the FPGA board. After making sure the enable switch (rightmost one) is in the ‘1’ position, clicking the buttons of the mouse should increment and decrement the 3-digit number displayed on the 7-segment display. Otherwise the current value will be stored regardless of other inputs.

If the user tries to increment past 255 or decrement below 0, the counter loops to the other side. During this, the IS_LEFT LED (rightmost one) is turned on.

In order to reverse the inputs of the mouse, the REVERSE switch (second from the right) should be activated. This will cause the IS_LEFT LED to turn off, and now the left click will decrement the counter while the right click increments.

Pressing the reset button (the one in the center) should cause the 7-segment display to return to the “000” state. After this the user can increment and decrement the number as before.

5. Technical justifications for the design

a. *8-bit counter:*

We have opted to use a singular 8-bit counter instead of multiple decimal counters for efficiency and simplicity of implementation. Using a single counter centralizes the outputted data into a single source. This also allows for the future implementation of generic parameters to make the maximum number displayed greater or smaller as needed.

b. *Using ROM to check the parity of the data packets:*

Using ROM eliminates the need to count the bits and calculate the parity needed, increasing the speed and efficiency of the PS/2 interface component.

6. Future developments

a. *Implementing generic parameters:*

Because of the way the counter and SSD decoder work, generic parameters storing a maximum and minimum number for the counter can be easily added. This can be used to control the number of bits the counter holds, the number of digits used by the 7-Segment Display, etc.

b. Implementing signed output:

Instead of the counter looping to the maximum when trying to decrement below 0, a sign bit can be added to the counter and be used to display a minus on the SSD when the counter goes into negative numbers.

7. References

1. http://burtonsys.com/ps2_chapweske.htm
2. https://github.com/Digilent/Nexys-A7-50T-OOB/blob/master/src/hdl/MouseCtl.vhd
3. https://www.fpga4student.com/2017/12/how-to-interface-mouse-with-FPGA.html?m=1
4. https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual