PS2 Mouse Interface Using FPGA

Submitted By:

Parth Mehta    200801209
# Index

<table>
<thead>
<tr>
<th></th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General Description</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Physical Interface</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Electrical Interface</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Modes of Operation</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Communication via PS2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Project Description</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>RTL Diagram</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Verilog HDL Code</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>HDL synthesis Report</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>Timing Report</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Map Report</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>Results of working code</td>
<td>29</td>
</tr>
</tbody>
</table>
1. General Description

There are many types of pointing devices available for the modern PC including mice, trackballs, touchpads, electronic whiteboards, etc. Virtually all of these devices communicate on one of two interfaces: Universal Serial Bus (USB) or the PS/2 mouse interface. Older pointing device interfaces include the Apple Desktop Bus (ADB), RS-232 serial port, and the bus mouse interface. These are obsolete and are not covered in this article.

The PS/2 mouse interface originally appeared in IBM's "Personal System/2" computers in the late 80's and it remains a widely-supported interface. The PS/2 mouse interface utilizes a bidirectional serial protocol to transmit movement and button-state. The controller, in turn, may send a number of commands to the mouse to set the report rate, resolution, reset the mouse, disable the mouse, etc. The host provides the mouse with a 5V ~100 mA power supply.

2. Physical Interface

The physical PS/2 port is one of two styles of connectors: The 5-pin DIN or the 6-pin mini-DIN. Both connectors are completely (electrically) similar; the only practical difference between the two is the arrangement of pins. The DIN standard was created by the German Standardization Organization (Deutsches Institut fuer Norm).

The pinouts for each connector are shown below:

<table>
<thead>
<tr>
<th>Male (Plug)</th>
<th>Female (Socket)</th>
<th>6-pin Mini-DIN (PS/2):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 - Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Not Implemented</td>
</tr>
<tr>
<td>3 - Ground</td>
<td></td>
<td>4 - Vcc (+5V)</td>
</tr>
<tr>
<td>5 - Clock</td>
<td></td>
<td>6 - Not Implemented</td>
</tr>
</tbody>
</table>

3. The Electrical Interface

Vcc/Ground provide power to the mouse. The mouse should not draw more than 275 mA from the host and care must be taken to avoid transient surges.
4. Modes of Operation

Data reporting is handled according to the mode in which the mouse is operating. There are four modes of operation:

- **Reset** - The initial mode, in which the mouse performs initialization and self-diagnostics.
- **Stream** - The default operating mode, in which the mouse issues movement data packets when movement occurs or button state changes.
- **Remote** - The host must poll for movement data packets.
- **Wrap** - A purely diagnostic mode where the mouse echoes every received packet back to the host.

4.1 Reset Mode

The mouse enters reset mode at power-on or in response to the "Reset" (0xFF) command. Upon entering this mode, the mouse performs a diagnostic self-test called **BAT** (Basic Assurance Test) and sets the following default values:

- **Sample Rate** = 100 samples/sec
- **Resolution** = 4 counts/mm

The mouse then sends a BAT completion code of either 0xAA (BAT successful) or 0xFC (Error). The host's response to a completion code other than 0xAA is undefined.

Following the BAT completion code (0xAA or 0xFC), the mouse sends its device ID of 0x00. This distinguishes it from a keyboard or nonstandard mouse. The host should not transmit any data until it receives a device ID.

Once the mouse has sent its device ID to the host, it enters stream mode.
4.2 Stream Mode

In stream mode the mouse sends movement data when it detects movement or a change in state of one or more mouse buttons. The maximum rate at which this data may be reported is known as the sample rate. This parameter ranges from 10-200 samples/sec, with a default value of 100 samples/sec. The host may set this value using the “Set Sample Rate” (0xF3) command.

The mouse will not actually issue any movement data packets until it receives the "Enable Data Reporting" (0xF4) command. The mouse acknowledges the reception of 0XF4 with 0xFA command.

Stream mode is the default operating mode, and is otherwise set using the "Set Stream Mode" (0xEA) command.

4.3 Remote Mode

In remote mode the mouse reads its inputs and updates its counters/flags at the current sample rate, but it does not automatically issue data packets when movement has occurred. Instead, the host polls the mouse using the "Read Data" (0xEB) command. Upon receiving this command the mouse will issue a single movement data packet and reset its movement counters.

The mouse enters remote mode upon receiving the "Set Remote Mode" (0xF0) command.

4.4 Wrap Mode

This is an "echoing" mode in which every byte received by the mouse is sent back to the host. Even if the byte represents a valid command, the mouse will not respond to that command—it will only echo that byte back to the host. There are two exceptions to this: the "Reset" (0xFF) and "Reset Wrap Mode" (0xEC) commands. The mouse treats these as valid commands and does not echo them back to the host.

In any mode, the datapacket sent by the mouse consists of three words and each word is 11 bits long. LSB of each word is a ‘Start bit - 0’. Data follows the start bit. Data is followed by a ‘Parity Bit’ and then the ‘Stop bit – 1’. The data of the three words is like:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Y overflow</td>
<td>X overflow</td>
<td>Y sign bit</td>
<td>X sign bit</td>
<td>Always 1</td>
<td>Middle Btn</td>
<td>Right Btn</td>
</tr>
<tr>
<td>Byte 2</td>
<td>X movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td>Y movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Communication via PS2

5.1 General Description

The PS/2 mouse implement a bidirectional synchronous serial protocol. The bus is "idle" when both lines are high (open-collector). This is the only state where the mouse is allowed begin transmitting data. The host has ultimate control over the bus and may inhibit communication at any time by pulling the Clock line low.

The device always generates the clock signal. If the host wants to send data, it must first inhibit communication from the device by pulling Clock low. The host then pulls Data low and releases Clock. This is the "Request-to-Send" state and signals the device to start generating clock pulses.

<table>
<thead>
<tr>
<th>Summary:</th>
<th>Bus States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data = high, Clock = high:</td>
<td>Idle state.</td>
</tr>
<tr>
<td>Data = high, Clock = low:</td>
<td>Communication Inhibited.</td>
</tr>
<tr>
<td>Data = low, Clock = high:</td>
<td>Host Request-to-Send</td>
</tr>
</tbody>
</table>

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11 bits. 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.

The parity bit is set if there is an even number of 1’s in the data bits and reset (0) if there is an odd number of 1’s in the data bits. The number of 1’s in the data bits plus the parity bit always add up to an odd number (odd parity.) This is used for error detection.

Data sent from the device to the host is read on the falling edge of the clock signal; data sent from the host to the device is read on the rising edge. The clock frequency must be in the range 10 - 16.7 kHz.
5.2 Device-to-Host Communication

When the keyboard or mouse wants to send information, it first checks the Clock line to make sure it's at a high logic level. If it's not, the host is inhibiting communication and the device must buffer any to-be-sent data until the host releases Clock. The Clock line must be continuously high for at least 50 microseconds before the device can begin to transmit its data.

As mentioned in the previous section, the keyboard and mouse use a serial protocol with 11-bit frames. 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.

The keyboard/mouse writes a bit on the Data line when Clock is high, and it is read by the host when Clock is low. Figures 2 and 3 illustrate this.

![Diagram of Device-to-host communication](image)

**Figure 1 : Device-to-host communication.** The Data line changes state when Clock is high and that data is valid when Clock is low.

The clock frequency is 10-16.7 kHz.

The host may inhibit communication at any time by pulling the Clock line low for at least 100 microseconds. If a transmission is inhibited before the 11th clock pulse, the device must abort the current transmission and prepare to retransmit the current "chunk" of data when host releases Clock. A "chunk" of data could be a make code, break code, device ID, mouse movement packet, etc. If the host pulls clock low before the first high-to-low clock transition, or after the falling edge of the last clock pulse, the keyboard/mouse does not need to retransmit any data. However, if new data is created that needs to be transmitted, it will have to be buffered until the host releases Clock. Mice only store the most current movement packet for transmission.
5.3 Host-to-Device Communication

The packet is sent a little differently in host-to-device communication.

First of all, the PS/2 device always generates the clock signal. If the host wants to send data, it must first put the Clock and Data lines in a "Request-to-send" state as follows:

- Inhibit communication by pulling Clock low for at least 100 microseconds.
- Apply "Request-to-send" by pulling Data low, then release Clock.

The device should check for this state at intervals not to exceed 10 milliseconds. When the device detects this state, it will begin generating Clock signals and clock in eight data bits and one stop bit. The host changes the Data line only when the Clock line is low, and data is read by the device when Clock is high.

After the stop bit is received, the device will acknowledge the received byte by bringing the Data line low and generating one last clock pulse. If the host does not release the Data line after the 11th clock pulse, the device will continue to generate clock pulses until the the Data line is released.

The host may abort transmission at time before the 11th clock pulse (acknowledge bit) by holding Clock low for at least 100 microseconds.

![Diagram](image)

Figure 2: Detailed host-to-device communication.
6. Project Description

We have implemented a PS2 mouse interface using FPGA on the Spartan 3E kit.

We have used the PS2 mouse in the stream mode. Initially the FPGA resets the mouse by pulling the ‘ps2c’ clock low for 100us and then gives control of ps2c to the mouse and also pulls the data line ‘ps2d’ low. The mouse thus starts generating the clock and the FPGA sends 0xF4 command serially to reset the mouse, on the negedge of ps2c.

After resetting the mouse, the FPGA gives the control of the data line to the mouse and the mouse enters the default mode i.e. stream mode. Then as the mouse moves, the FPGA decodes the signals from the mouse.

7. RTL Diagram
8. Verilog HDL Code

`timescale 1ns / 1ps

///////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date:  17:57:32 02/17/2011

// Design Name:

// Module Name:  ps2

// Project Name:

// Target Devices:

// Tool versions:
// Description:

// Dependencies:

// Revision:
// Revision 0.01 - File Created

// Additional Comments:

//

module ps2m(input clk,input reset, inout ps2c, inout ps2d,
           output dummyc, output dummyd, output reg x,output reg y);

    reg [12:0]counter = 0;
    reg flag = 0;
    reg psd = 0;
    reg selc = 0;
    reg seld = 1'b0;

    reg [8:0] dataout = 9'b011110100; // parity is put as one.
    reg negold = 0,negnew = 0;
    reg [3:0] dcount = 4'b0000 ;
reg [5:0] bcount=6'b000000;
reg initialised = 1'b0;
reg [32:0] data_in;
reg [5:0] init_count = 6'b000000;

always@(posedge clk)
begin

    negnew <= ps2c; // for detecting -ve edge
    negold <= negnew;

    counter = counter + 1;
    if((flag == 1) && (counter == 13'b1001110001000))
    begin

        flag = 0;
        selc = 0; //give clk control to mouse
        seld = 1'b1; // this makes the ps2d as zero and after this we start to tx the word

    end

    if(reset == 1'b1)// one or zero
    begin

        flag = 1;

    end

end
selc = 1;
counter = 0;
psd = 0;

end

if((selc == 0) && (negold == 1) && (negnew == 0))
begin

if(dcount < 9)
begin

    psd = dataout[dcount];

end

else
begin

    seld = 0;  //give data control to mouse

end


dcount = dcount + 1;

end
end // end always
initially give control to mouse. then make seld1 for fpga

// when psd is 0 give control to mouse

assign ps2c = selc ? 1'b0 : 1'bz;
assign ps2d = seld ? psd : 1'bz;
assign dummyc = ps2c;
assign dummyd = ps2d;

always @ (negedge ps2c)
begin

    if(init_count == 46)
    begin
        initialised = 1;
    end

    init_count = init_count + 1;

    if(initialised == 1'b1)
    begin
        data_in[bcount] = ps2d;
        bcount = bcount + 1;
        if(bcount == 33)
begin
bcount =0;

x = data_in[4];
y = data_in[5];

end

end
end // end always
// end
endmodule


9.1 HDL Synthesis Report

Macro Statistics

# ROMs : 1
4x1-bit ROM : 1

# Adders/Subtractors : 2
13-bit adder : 1
6-bit adder : 1

# Counters : 4
13-bit up counter : 1
4-bit up counter : 1
6-bit up counter : 2

# Registers : 42
1-bit register : 42

# Comparators : 2
4-bit comparator greatequal : 1
4-bit comparator less : 1

# Multiplexers : 2
1-bit 4-to-1 multiplexer : 2

# Tristates : 2
1-bit tristate buffer : 2

9.2 Advanced HDL Synthesis

Macro Statistics

# ROMs : 1
4x1-bit ROM : 1

# Adders/Subtractors : 2
13-bit adder : 1
6-bit adder : 1

# Counters : 4
13-bit up counter : 1
4-bit up counter : 1
6-bit up counter : 2
# Registers : 11
Flip-Flops : 11

# Comparators : 2
4-bit comparator greatequal : 1
4-bit comparator less : 1

# Multiplexers : 2
1-bit 4-to-1 multiplexer : 2

9.3 Final Register Report

Macro Statistics

# Registers : 40
Flip-Flops : 40

9.4 Final Results

Design Statistics

# IOs : 8

Cell Usage :

# BELS : 126
# GND : 1
# INV : 6
# LUT1 : 24
# LUT2 : 8
# LUT3 : 10
# LUT3_D : 1
# LUT4 : 22
# LUT4_D : 3
# LUT4_L : 1
# MUXCY : 24
# MUXF5 : 1
# VCC : 1
# XORCY : 24
# FlipFlops/Latches : 40
# FD : 3
# FD_1 : 6
# FDE : 6
# FDE_1 : 4
# FDR : 13
# FDR_1 : 1
# FDRE_1 : 6
# FDSE : 1
# Clock Buffers : 1
# BUFGP : 1
# IO Buffers : 7
# IBUF : 1
# IOBUF : 2
# OBUF : 4

9.5 Device utilization summary:
Selected Device : 3s500efg320-5
10. TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.

FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
--------------------

<table>
<thead>
<tr>
<th>Clock Signal</th>
<th>Clock buffer(FF name)</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>BUFGP</td>
<td>23</td>
</tr>
<tr>
<td>ps2c</td>
<td>IBUF</td>
<td>17</td>
</tr>
</tbody>
</table>
Asynchronous Control Signals Information:
----------------------------------------

No asynchronous control signals found in this design

Timing Summary:
----------------

Speed Grade: -5

Minimum period: 9.804ns (Maximum Frequency: 101.995MHz)
Minimum input arrival time before clock: 7.003ns
Maximum output required time after clock: 5.882ns
Maximum combinational path delay: 5.507ns

Timing Detail:
--------------

All values displayed in nanoseconds (ns)

=========================================================================

Timing constraint: Default period analysis for Clock 'clk'

Clock period: 9.804ns (frequency: 101.995MHz)
Total number of paths / destination ports: 1157 / 28

=========================================================================

Delay: 9.804ns (Levels of Logic = 7)
Source: counter_1 (FF)
Destination: seld (FF)
Source Clock: clk rising
Destination Clock: clk rising

Data Path: counter_1 to seld

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell: in-&gt;out</td>
<td>fanout  Delay  Delay  Logical Name (Net Name)</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FDR:C-&gt;Q</td>
<td>2</td>
<td>0.514</td>
<td>0.897</td>
<td>counter_1 (counter_1)</td>
</tr>
<tr>
<td>LUT1:I0-&gt;O</td>
<td>1</td>
<td>0.612</td>
<td>0.000</td>
<td>counter_1_rt (counter_1_rt)</td>
</tr>
<tr>
<td>MUXCY:S-&gt;O</td>
<td>1</td>
<td>0.404</td>
<td>0.000</td>
<td>Madd__old_counter_1_cy&lt;1&gt; (Madd__old_counter_1_cy&lt;1&gt;)</td>
</tr>
<tr>
<td>XORCY:CI-&gt;O</td>
<td>1</td>
<td>0.699</td>
<td>0.833</td>
<td>Madd__old_counter_1_xor&lt;2&gt; (_old_counter_1&lt;2&gt;)</td>
</tr>
<tr>
<td>LUT2:I0-&gt;O</td>
<td>3</td>
<td>0.612</td>
<td>0.801</td>
<td>_cmp_eq000138 (_cmp_eq0001_map72)</td>
</tr>
<tr>
<td>LUT4:I2-&gt;O</td>
<td>1</td>
<td>0.612</td>
<td>0.684</td>
<td>_and00001_SW1 (N106)</td>
</tr>
<tr>
<td>LUT4:I3-&gt;O</td>
<td>2</td>
<td>0.612</td>
<td>0.748</td>
<td>_old_selc_21 (_old_selc_2)</td>
</tr>
<tr>
<td>LUT4:I3-&gt;O</td>
<td>1</td>
<td>0.612</td>
<td>0.681</td>
<td>_not00032 (_not0003)</td>
</tr>
<tr>
<td>FDE:CE</td>
<td></td>
<td>0.483</td>
<td></td>
<td>seld</td>
</tr>
</tbody>
</table>

Total                        | 9.804ns (5.160ns logic, 4.644ns route) |
|                           | (52.6% logic, 47.4% route)              |

========================================================================

Timing constraint: Default period analysis for Clock 'ps2c'

Clock period: 7.203ns (frequency: 138.832MHz)

Total number of paths / destination ports: 335 / 31

========================================================================

Delay: 7.203ns (Levels of Logic = 4)

Source: bcount_1 (FF)
Destination: bcount_0 (FF)
Source Clock: ps2c falling
Destination Clock: ps2c falling

Data Path: bcount_1 to bcount_0

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cell:in-&gt;out fanout Delay Delay Logical Name (Net Name)</td>
</tr>
<tr>
<td></td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>FDRE_1:C-&gt;Q</td>
<td>7 0.514 0.936 bcount_1 (bcount_1)</td>
</tr>
<tr>
<td>LUT4_D:I2-&gt;O</td>
<td>1 0.612 0.711 Madd__add0000__cy&lt;3&gt;1 (Madd__add0000__cy&lt;3&gt;)</td>
</tr>
<tr>
<td>LUT3:I2-&gt;O</td>
<td>2 0.612 0.775 Result&lt;5&gt;1 (Result&lt;5&gt;)</td>
</tr>
<tr>
<td>LUT4_L:I2-&gt;LO</td>
<td>1 0.612 0.103 _and0002_SW0 (N31)</td>
</tr>
<tr>
<td>LUT4:I3-&gt;O</td>
<td>8 0.612 0.921 _and0002 (_and0002)</td>
</tr>
<tr>
<td>FDRE_1:R</td>
<td>0.795 bcount_0</td>
</tr>
</tbody>
</table>

Total 7.203ns (3.757ns logic, 3.446ns route)
(52.2% logic, 47.8% route)

Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'

Total number of paths / destination ports: 26 / 24

Offset: 7.003ns (Levels of Logic = 4)
Source: reset (PAD)
Destination: seld (FF)
Destination Clock: clk rising
Data Path: reset to seld

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell:in-&gt;out</td>
<td>fanout Delay Delay Logical Name (Net Name)</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>IBUF:1-&gt;O</td>
<td>20 1.106 1.465 reset_IBUF (reset_IBUF)</td>
</tr>
<tr>
<td>LUT4:I0-&gt;O</td>
<td>1 0.612 0.684 _and00001_SW (N106)</td>
</tr>
<tr>
<td>LUT4:I3-&gt;O</td>
<td>2 0.612 0.748 _old_selc_21 (_old_selc_2)</td>
</tr>
<tr>
<td>LUT4:I3-&gt;O</td>
<td>1 0.612 0.681 _not00032 (_not0003)</td>
</tr>
<tr>
<td>FDE:CE</td>
<td>0.483 seld</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>

Total 7.003ns (3.425ns logic, 3.578ns route)
(48.9% logic, 51.1% route)

========================================================================

Timing constraint: Default OFFSET IN BEFORE for Clock 'ps2c'

Total number of paths / destination ports: 2 / 2

========================================================================

Offset: 2.145ns (Levels of Logic = 1)
Source: ps2d (PAD)
Destination: data_in_4 (FF)
Destination Clock: ps2c falling

Data Path: ps2d to data_in_4

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell:in-&gt;out</td>
<td>fanout Delay Delay Logical Name (Net Name)</td>
</tr>
</tbody>
</table>
IOBUF:IO->O  3 1.106 0.771  ps2d_IOBUF (dummyd_OBUF)
FDE_1:D  0.268  data_in_5

-------------------------------
Total  2.145ns (1.374ns logic, 0.771ns route)
      (64.1% logic, 35.9% route)

-------------------------------
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
Total number of paths / destination ports: 3 / 2

-------------------------------
Offset:  5.882ns (Levels of Logic = 2)
Source:  selc (FF)
Destination:  ps2c (PAD)
Source Clock:  clk rising

Data Path: selc to ps2c

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell:in-&gt;out</td>
<td>fanout</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>FD:C-&gt;Q</td>
<td>7</td>
</tr>
<tr>
<td>INV:I-&gt;O</td>
<td>1</td>
</tr>
<tr>
<td>IOBUF:T-&gt;IO</td>
<td>3.169</td>
</tr>
</tbody>
</table>

-------------------------------
Total  5.882ns (4.295ns logic, 1.587ns route)
      (73.0% logic, 27.0% route)
Timing constraint: Default OFFSET OUT AFTER for Clock 'ps2c'  
Total number of paths / destination ports: 2 / 2

Offset: 4.364ns (Levels of Logic = 1)  
Source: x (FF)  
Destination: x (PAD)  
Source Clock: ps2c falling  

Data Path: x to x

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
<th>fanout</th>
<th>Delay (logic)</th>
<th>Delay (route)</th>
<th>Logical Name (Net Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDE_1:C-&gt;Q</td>
<td>1 0.514 0.681</td>
<td>x (x_OBUF)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBUF:I-&gt;O</td>
<td>3.169</td>
<td>x_OBUF (x)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total 4.364ns (3.683ns logic, 0.681ns route)  
(84.4% logic, 15.6% route)

Timing constraint: Default path analysis  
Total number of paths / destination ports: 2 / 2

Delay: 5.507ns (Levels of Logic = 2)  
Source: ps2c (PAD)
Destination: dummyc (PAD)

Data Path: ps2c to dummyc

<table>
<thead>
<tr>
<th>Gate</th>
<th>Net</th>
<th>Cell:in-&gt;out</th>
<th>fanout</th>
<th>Delay</th>
<th>Delay</th>
<th>Logical Name (Net Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBUF:IO-&gt;O</td>
<td>19</td>
<td>1.106</td>
<td>1.232</td>
<td>ps2c_IOBUF (dummyc_OBUF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBUF:I-&gt;O</td>
<td>3.169</td>
<td>dummyc_OBUF (dummyc)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total 5.507ns (4.275ns logic, 1.232ns route)
(77.6% logic, 22.4% route)

CPU : 20.90 / 21.32 s | Elapsed : 21.00 / 21.00 s

Total memory usage is 187220 kilobytes

Number of errors : 0 ( 0 filtered)
Number of warnings : 4 ( 0 filtered)
Number of infos : 2 ( 0 filtered)

11. Map Report
11.1 Design Information
C:/Xilinx/ps2mouse/ps2mouse.ise -intstyle ise -p xc3s500e-fg320-5 -cm area -pr b
-k 4 -c 100 -o ps2m_map.ncd ps2m.ngd ps2m.pcf
Target Device : xc3s500e
Target Package : fg320
Target Speed : -5
Mapper Version : spartan3e -- $Revision: 1.34.32.1 $
Mapped Date : Sun Feb 20 20:43:54 2011

11.2 Design Summary
Number of errors:  0
Number of warnings:  0
Logic Utilization:
Number of Slice Flip Flops:  35 out of  9,312  1%
Number of 4 input LUTs:  47 out of  9,312  1%
Logic Distribution:
Number of occupied Slices:  44 out of  4,656  1%
Number of Slices containing only related logic:  44 out of  44  100%
Number of Slices containing unrelated logic:  0 out of  44  0%
*See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:  71 out of  9,312  1%
Number used as logic:  47
Number used as a route-thru: 24
Number of bonded IOBs: 8 out of 232 3%
IOB Flip Flops: 5
Number of GCLKs: 1 out of 24 4%

Total equivalent gate count for design: 758
Additional JTAG gate count for IOBs: 384
Peak Memory Usage: 197 MB

### 11.3 IOB Properties

<table>
<thead>
<tr>
<th>IOB Name</th>
<th>IOB Type</th>
<th>Direction</th>
<th>IO Standard</th>
<th>Drive</th>
<th>Slew</th>
<th>Reg (s)</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>dummyc</td>
<td>IOB</td>
<td>OUTPUT</td>
<td>LVCMOS25</td>
<td>12</td>
<td>SLOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dummyyd</td>
<td>IOB</td>
<td>OUTPUT</td>
<td>LVCMOS25</td>
<td>12</td>
<td>SLOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ps2c</td>
<td>IOB</td>
<td>BIDIR</td>
<td>LVCMOS25</td>
<td>12</td>
<td>SLOW</td>
<td>IFF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
12. Results of the Working Code

Initial Transmission to reset the mouse
Signal generated on moving the mouse towards +ve X direction and +ve Y direction
Signal generated on moving the mouse towards -ve X direction and +ve Y direction

Signal generated on moving the mouse towards -ve X direction and -ve Y direction
Signal generated on moving the mouse towards +ve X direction and -ve Y direction