Timing Analysis in Vivado

An example used in this tutorial is the circuit generated during “Exercise 4A: Creating IP in HDL” from the The Zynq Book Tutorials. The block diagram of this circuit is shown in Fig. 1.

Fig. 1: Block diagram for Exercise 4A

As a reminder, the generation of this circuit included the following steps:

2. Creating and packaging a new IP called led_controller.
3. Creating a new block design, including Zynq PS and led_controller_v1_0.
4. Connecting IPs together.
5. Validating the design.
6. Creating the HDL wrapper.
7. Generating block design.
8. Synthesizing and implementing the design.

Please note that similar operations may be necessary before performing the remaining steps described in this tutorial for an arbitrary circuit of your choice.
A new window will pop up that shows all the option for timing summary. Do not change anything and click OK to select the default options.
A new timing summary will appear that will show you timing information. Click on the “Worst Negative Slack (WNS)”

Clicking on the value of WNS will open the information for the 10 paths (Path 1 to 10) that have maximum delay in the design. Path 1 has the largest delay in the design, and thus is called the critical path.

Double click on Path 1 to show you all the information for Path 1.
This information includes

1. Summary
2. Source Clock Path
3. Data Path
4. Destination Clock Path

### Summary

<table>
<thead>
<tr>
<th>Data</th>
<th>Src Path</th>
<th>Clock</th>
<th>Requirement</th>
<th>Source</th>
<th>Destination</th>
<th>Path Group</th>
<th>Path Type</th>
<th>Source Clock Path</th>
<th>Data Path</th>
<th>Destination Clock Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>12.50ns</td>
<td></td>
<td></td>
<td></td>
<td>clk_fpga_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source Clock Path

<table>
<thead>
<tr>
<th>Delay Type</th>
<th>Ins (ns)</th>
<th>Path (ns)</th>
<th>Location</th>
<th>Netlist Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock (clk_fpga_0 rise edge)</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS7</td>
<td>0.000</td>
<td>0.000 Sites: PS7_X0Y0</td>
<td>led_controller_j processing_system?_0/inst/PS7_JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=1, routed)</td>
<td>1.207</td>
<td>1.207</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Data Path

<table>
<thead>
<tr>
<th>Delay Type</th>
<th>Ins (ns)</th>
<th>Path (ns)</th>
<th>Location</th>
<th>Netlist Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS7</td>
<td>1.334</td>
<td>4.457 Sites: PS7_X0Y0</td>
<td>led_controller_j processing_system?_0/inst/PS7_JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=1, routed)</td>
<td>1.285</td>
<td>5.692</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LTC1</td>
<td>0.124</td>
<td>5.815 Sites: SLICE_X3Y8</td>
<td>led_controller_j processing_system?_0/inst/JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=133, routed)</td>
<td>3.215</td>
<td>9.030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LTC1</td>
<td>0.124</td>
<td>9.154 Sites: SLICE_X3Y8</td>
<td>led_controller_j processing_system?_0/inst/JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=1, routed)</td>
<td>0.000</td>
<td>9.354</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Arrival Time

9.254

### Destination Clock Path

<table>
<thead>
<tr>
<th>Delay Type</th>
<th>Ins (ns)</th>
<th>Path (ns)</th>
<th>Location</th>
<th>Netlist Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock (clk_fpga_0 rise edge)</td>
<td>20.000</td>
<td>20.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS7</td>
<td>0.000</td>
<td>0.000 Sites: PS7_X0Y0</td>
<td>led_controller_j processing_system?_0/inst/PS7_JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=1, routed)</td>
<td>1.101</td>
<td>21.121</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LTC1</td>
<td>0.091</td>
<td>21.212 Sites: SLICE_X3Y8</td>
<td>led_controller_j processing_system?_0/inst/JMA0X000JK</td>
<td></td>
</tr>
<tr>
<td>net (f=133, routed)</td>
<td>1.501</td>
<td>22.693</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| clock pessimism | 0.251 | 22.944 Sites: SLICE_X3Y8 | led_controller_j processing_system?_0/inst/JMA0X000JK | |
| clock uncertainty | -0.302 | 22.622 | | |

### Required Time

22.644
Select all the information in the “Data Path” tab, and then click on the “Device X” tab.

This will show you which slices and LUTs utilized in the chip are a part of critical path.
Select all the information in the “Data Path” tab again, right click and then click on “Schematic (F4)”

The schematic will display the critical path information

**Critical Path:** $\text{PS} \rightarrow \text{AXI interconnect} \rightarrow \text{led_controller} \rightarrow \text{slv_reg3}$

Click on all the individual components in the critical path to highlight them and then press **F7** to go to their source code. This way you can find the critical path in your VHDL code and then fix the critical path if possible.
Components and corresponding code:

1. Processign system (PS):

```
PS7 instance

MAXIGP0WVALID

```

2. AXI Interconnect:

```
processing_system7_0_axis_periph

s00_couplers

s00_AXI_wvalid

s0i_wvalid

s01_axis_periph

m00_AXI_wvalid

led_controller_auto_pc

s00_couplers_mm mandates

led_controller_processing_system7_0_axis_periph_0
```

3. LED controller:
auto_pc: component led_controller_auto_pc_0

process (S_AXI_ACLK)
begin
    if rising_edge(S_AXI_ACLK) then
        if S_AXI_ARESETN = '0' then
            axi_awready <= '0';
        else
            if (axi_awready = '0' and S_AXI_AWVALID = '1' and S_AXI_WVALID = '1') then

process (S_AXI_ACLK)
begin
    if rising_edge(S_AXI_ACLK) then
        if S_AXI_ARESETN = '0' then
            slv_reg0 <= (others => '0');
        else
            slv_reg0 <= (others => '0');
slv_reg1 <= (others => '0');
slv_reg2 <= (others => '0');
slv_reg3 <= (others => '0');