# Lab 4: Timing, Pipelining

## Objective

Explore some of the timing issues such as register-to-register delay, clock-to-out, etc. discussed in class. Create a design that means a performance criteria where pipelining must be used in order to meet the timing requirements.

All files referred to in this lab are available in this ZIP archive listed on the WWW page for this lab. To unzip on Unix machines, do '*unzip zipfile.zip*'.

Map all designs produced in this lab to the Flex 10K family, EPF10K20RC240-4 (for the Professional edition of the software, you will need to click on the 'show all speed grades' button on the device menu in order to get the "-4" speed grade). Use the FAST synthesis option.

This lab is worth 150 points.

## Background:

In this lab exercise you will make use of all analysis modes available in the Timing Analyzer tool. These different analysis modes are available from the *Analysis* menu once the timing analysis tool is active. The different analysis modes are:

1. *Delay Matrix* : this is the mode that you have used in previous exercises. It will give you combinational delays from input pins to output pins of the mapped device. These delays are known as *pin-to-pin* delays. Note that if all outputs are registered, then the only combinational paths from inputs to outputs is the clock to each output. Sometimes this type of delay is known as *clock-to-out* delay.
2. *Setup Matrix*: this mode reports the setup/hold times for all registered inputs (inputs that go the data pin of a latch or flip flop).
3. *Registered Performance*: this examines all of the register to register delay paths, finds the longest path, and uses the inverse of this value to report a maximum clock speed for the design. Individual delay paths can be listed from within this mode.

## To Do (part 1, basic timing):

Create a SINGLE schematic from which you can use the Timing Analyzer to measure all of the following:

1. Minimum register to register delay. This is produced using a minimum combinational logic block (any 2-input gate) whose inputs and output are connected to DFFs. The input DFFs should be fed from input pins, and the output DFF should be tied to an output pin.. Use the “Registered Peformance” mode of the Timing analyzer. Use “list paths” to show the complete path delay.

2. Minimum *clock-to-out* delay – use the "Delay Matrix" mode of the timing analyzer to get this information from the above logic.
3. Minimum *setup* time – use the "Setup/Hold Matrix" mode of the timing analyzer to get this information from the above logic.
4. Minimum *pin-to-pin* delay for combinational logic.Use a simple 2-input gate (any function) whose inputs/outputs are directly connected to pins to the above schematic to get this measurement. The inputs and outputs are unregistered.

## To Do (part 2, pipelined multiplier):

1. Use the LPM_MULT parameterized module, and configure it to do an 8 bit x 8 bit multiply with a 8 bit product. Place DFFs on the inputs, DFFs on the outputs, and determine the maximum clock frequency using the same device, synthesis mode used in part #1.
2. Use the previous schematic, and change the LPM_MULT parameters such that it is now pipelined with a latency of '1'. This means that you will need to mark the 'clock' input as 'used', and set the value of LPM_PIPELINE to 1. You will need to connect the clock input of the multiplier to the clock input. Determine the maximum clock frequency using the same device, synthesis mode used in part #1. Repeat this procedure for LPM_PIPELINE =2, and LPM_PIPELINE = 4, and determine maximum clock frequencies for each.
3. For the schematic with LPM_PIPELINE=4, create a waveform file that exercises the multiplier and verify that this multiplier is operating correctly with a latency of 4. Your waveform file MUST feed at least 50 vectors through the multiplier. DEMONSTRATE this simulation to the TA.

## To Do (part 3, pipelined BLEND operation):

Modify the blend implementation that you did in Lab #3 so that it will operate at a clock rate of at least 50 Mhz. Place DFFs on the inputs, and DFFs on the outputs, and use any number of pipeline stages necessary to reach 50 Mhz. You must use the Flex 10K family, device EPF10K20RC240-4. Use the FAST synthesis option. If you are using the professional edition, you will use the "show all speed grades" in order to get the "-4" speed grade. You CANNOT use a speed grade other than "-4".

Call your design "pblend". Look at the testbench schematic (tbpblend.gdf) included in the ZIP archive to get the required port names. It is your "pblend" that must meet the clock rate specification above, NOT the testbench. This means that to check your clock rate specification, do a 'set current project' to your pblend schematic, compile, and run the timing analyzer. If you include the testbench, then your design may be slower than necessary. The testbench schematic is included only to make sure that your design isstill functional after you add pipeline stages.

## Other Files:

The ZIP archive contains other files that you may or may not find interesting:

- onemalt.gdf - this implements the "1-F" function using LPMs. You might want to use this instead of the VHDL that you did for Lab #3 because the Altera synthesis software can sometimes produce faster logic from LPMs than from VHDL.
- dff8.vhd, dff9.vhd – 8 and 9 bit registers specified in VHDL. You could also use the LPM_DFF

## Testing the Design:

You can use the schematic *tbblend.gdf* as the testbench. This a *design* problem, which means that there are multiple solutions that will meet the specifications. I have provided 'golden' waveforms files that have differing amounts of latency – the lat15gold.scf waveform file below gives the maximum amount of latency that I will allow.

1. lat1gold.scf - output latency = 1 (testbench)
2. lat2gold.scf - output latency = 2 (testbench)
3. lat3gold.scf - output latency = 3 (testbench)
4. lat4gold.scf - output latency = 4 (testbench)
5. lat5gold.scf - output latency = 5 (testbench)
6. lat6gold.scf - output latency = 6 (testbench)
7. lat7gold.scf - output latency = 7 (testbench)
8. lat8gold.scf - output latency = 8 (testbench)
9. lat9gold.scf - output latency = 9 (testbench)
10. l10gold.scf - output latency = 10 (testbench)
11. l11gold.scf - output latency = 11 (testbench)
12. l12gold.scf - output latency = 12 (testbench)
13. l13gold.scf - output latency = 13 (testbench)
14. lat14gold.scf - output latency=14 (testbench)
15. lat15gold.scf - output latency=15 (testbench)

When you are checking your design, particular trouble spots might be where the F value transitions from 0 to a non-zero value or from 1.0 to a value not equal to 1.0. One cause for failure at these junctures would be if the amount of latency in the paths that bypass the multiplier do not equal to the latency within the multiplier. Be sure that the select line going to the mux on the output the multiplier has the same amount of latency as the data paths to the mux.

## Checkoff:

You must DEMONSTRATE that your design meets the timing specification using speed grade "-4" and also produces the correct output vectors.

## Report

You must hand in plots of all schematics, and report the timings from parts 1,2, and the final maximum frequency in part 3. Hand in a partial plot of the multiplier simulation and show the latency between a set of input values and the correct output value. EXPLAIN how you modified the original blend implementation to include pipelining.