ESE 555 Final Project

Project report is due December 17 on Brightspace
Project live demonstration will take place on December 18 at 10AM
(more details will be available soon)
Project can be completed with a group of at most 2 students

Dr. Emre Salman
Electrical and Computer Engineering Department
Stony Brook University

The final project for the course is to design and verify a pipelined synchronous 8-bit carry select adder. The primary goal is to minimize the power consumption while ensuring that the design can run at a clock frequency of 4 GHz.

In a multiple bit adder circuit, the carry generation is typically the critical path. Various architectures have been proposed to alleviate this issue, trading power and area with the delay of the carry generation path.

Carry select adder is a popular architecture that exploits this tradeoff. The carry-out is computed for both values of the carry-in before the actual carry-in is generated. Once the correct carry-in is generated, the corresponding carry-out is chosen utilizing a multiplexor. The primary inputs of your circuit will be Cin, clock, reset, A0, A1,…., A7, and B0, B1, …., and B7. The primary outputs will be S0, S1,….., S7, and Cout. Each of these outputs should be able to drive an external load of 5 fF.

For this architecture, you will need to design the following primary building blocks:

Flip-flop
1-bit full adder
2-to-1 multiplexor

Since you will design an 8-bit adder, you should design a linear carry select adder with two blocks. Each block will have 4-bit adders where the carry ripples.

Number of pipeline stages is also a design parameter that you need to decide. Objective is to minimize power consumption while ensuring 4 GHz clock frequency, as mentioned above. This result will be the primary parameter of your project. However, reliable operation and accurate functionality are the most important characteristics. The input data pattern and simulation time for verification are indicated at the end of this document. The frequency of the data signals dictates a minimum clock frequency. In other words, if the input data (A0---A7 and B0…B7) change at a frequency of *f*, the clock frequency used to sample this data should be sufficiently higher than *f*. Specifically, ensure that the design can correctly run at a clock frequency of 4 GHz. Assume that rise/fall times for clock and data signals are 25 ps.

Remember that, at the minimum, you have to place flip-flops at the primary inputs and primary outputs. In this case, the first output signals should be valid after the second rising edge of the

clock signal (similar to your DFF assignment). After the first outputs emerge, every clock edge should produce the next outputs. Additional pipelining flip-flops within the circuit are upon your decision. For example, if you have 3 DFFs between the inputs and outputs, first outputs should emerge after the third rising edge. Then, every rising edge will produce an output. More pipelining will increase clock frequency (and throughput) at the expense of power and area.

PS: You are not asked to complete the layout of this design. Only schematic-level design is requested. However, you will receive 20 points bonus if you can demonstrate clean DRC, LVS, and functional (fully accurate) post-layout simulation results. There will be no partial bonus.

Your report should include the following:

Abstract – Summarize the project with several sentences mentioning your final design parameters such as speed and power.
Introduction – Discuss various adder topologies.
Carry Select Adder Design – This section should have four subsections

Block Level Architecture: Show the high-level picture of the adder, describe its operation, describe your pipelining methodology, whether you placed flip-flops only at the primary inputs and outputs, or you used additional flip-flops within the circuit. Identify the critical path that determines your clock speed.
Flip-Flop: Describe the flip-flop you used.
1-Bit Adder: Describe the 1-bit adder that you used as a building block.
2-to-1 Multiplexor: Describe the mux that you used.

Simulation Results: Show correct operation of the adder by converting the signals into decimal form in ADE (S0 has to be the least significant bit and Cout should be the most significant bit). You can do this conversion in ADE simulation window. Look at the pages at the end of this document from Virtuoso User Guide.

You have to use the input pattern described below for verification because I will check your SUM results assuming that you use these inputs provided below. Specifically illustrate decimal results (inputs A and B, and SUM) at the following time instances: 5ns, 11ns, 16ns, 19ns, 23ns, 28ns, 33ns, 36ns, 42ns, 49ns. For example, if we are adding 56 and 144 at 5ns, your figure should show these inputs in decimal form as well as the sum, which would be 200. Make sure you have the correct sum in each of the time instances.
Discussion: Discuss any points (about your design experience) that you believe is worth mentioning.
Conclusion: Conclude the project with a brief summary. Elaborate on possible improvements.

Remember once again that correct functionality is number 1 priority. This is true even if only one of the bits is incorrect at a particular time instance. All output bits should be correct at all times.

PS: +20 bonus points for those who demonstrate DRC/LVS clean layout and accurate post-layout simulation

Input pattern for testing

All signals have rise/fall times of 25 ps. You can convert the two 8-bit inputs and output signals into decimal representation to ease the verification. **Make sure your circuit works correctly.**

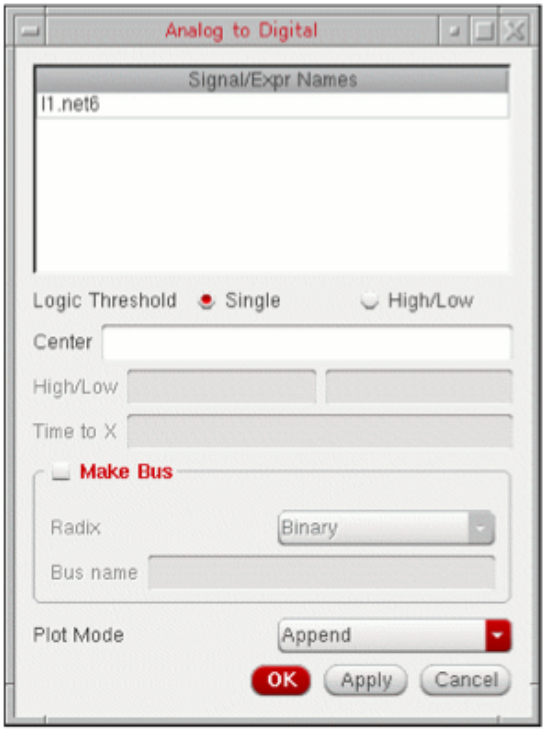
A0: Period=4 ns, delay time=1 ns
A1: Period= 3 ns, delay time=0
A2: Period= 6 ns, delay time=5 ns
A3: Period= 2.5 ns, delay time=2 ns
A4: Period=12 ns, delay time=0
A5: Always 1
A6: Period=5 ns, delay time=12 ns
A7: Period=2.5ns, delay time= 3 ns
B0: Period=1 ns, delay time= 0
B1: Period= 9 ns, delay time=0
B2: Period= 14 ns, delay time=30 ns
B3: Period=3 ns, delay time=2 ns
B4: Period=2.5 ns, delay time=4 ns
B5: Period=5 ns, delay time=2 ns
B6: Always zero
B7: Period=4 ns, delay time=0
Cin: Always zero
Reset: High for 1ns, then always low

Simulate the circuit for 50 ns for functional verification and average power analysis.

To convert an analog signal into a digital signal, do the following:

1. In the graph, select a trace and choose *Measurements – Analog To Digital*. You can select more than one analog trace at a time.

   The *Analog to Digital* conversion form appears. This form displays the name of the selected analog signals. The signals are displayed on the basis of their selection order; however, you can rearrange the order either by clicking the column header or by using the drag operation. The assistant has the following fields



2. In the *Logic Threshold* field, select *Single* or *High/Low*.

   - If you select *Single*, you need to specify a *Center* value. Analog values equal to or greater than the specified center value are mapped to a digital value of 1. Analog values less than the center value are mapped to a logical value of 0.
   - If you select *High/Low*, you need to specify a high and a low threshold value. All analog values equal to or greater than the high threshold value are mapped to a digital value of 1. All analog values equal to or less than the low threshold value are mapped to a logical value of 0.

The value *Time to X* puts a time limit on the interval that the signal may remain between the high and low threshold values before the signal is assigned a value of *X*.

3. If you want to make a bus of digital signals from the analog signal, select the *Make Bus* check box.
4. Then, select the radix type in the drop-down list and provide a bus name.
5. In the *Plot Mode* drop-down list box, select whether you want to *append* the digital trace to an existing graph, *replace* an existing graph with the digital graph, or add the digital trace to a *new subwindow* or *new window*.
6. Click *OK*.

Converting a Digital Signal to an Analog Signal

In the stand-alone SKILL mode, you can also create an analog representation of a digital signal.

To convert a digital signal into an analog signal, do the following:

1. In the window, select a digital trace and choose *Measurements – Digital To Analog*. You can select more than one digital trace at a time.

   The *Digital to Analog* Conversion form appears. This form displays the digital signals you select. The signals are displayed on the basis of their selection order; however, you