NTNU
Department of Electronic Systems

TFE4141 Design of Digital Systems 1

Assignment 5: Datapath design

Deadline: October 6 at 23:59

Mega adder requirements

In this assignment we will design a large adder circuit. This adder circuit must satisfy a set of requirements as described in this section.

Functional requirements

The design takes two 128-bit numbers `a` and `b`, add these together and produces a 128-bit result `y`.

Interface requirements

The design has two interfaces. One interface for sending data in to the design and one interface that is used for sending data out of the circuit.

The input interface is used for sending in the two 128-bit numbers `a` and `b` in chunks of 32-bits at a time.

These numbers are added together and produces a 128-bit number `y` that is sent out of the circuit also in 32-bit chunks.

The 128-bit numbers are chopped up in 32-bit words as illustrated in Table 1.

127..96	95..64	63 ..32	31 ..0
a3	a2	a1	a0
b3	b2	b1	b0
y3	y2	y1	y0

Table 1. Partitioning the 128-bit operands into four 32-bit words.

Both the input and the output interfaces are valid/ready based handshake interfaces.

- `data_<in/out>` must have valid data when `data_<in/out>_valid` is high.
- Data is transferred on the interface when `data_<in/out>_valid` and `data_<in/out>_ready` are high both in the same clock cycle.
- If `data_<in/out>_valid` is high and the corresponding `ready` signal is low in a particular clock cycle, then `data_<in/out>_valid` must be high also in the following clock cycle and the `data_<in/out>` signal must retain its previous value. This situation is illustrated for the data transfer of `a3` and `y3` in Figure 1 and Figure 2.

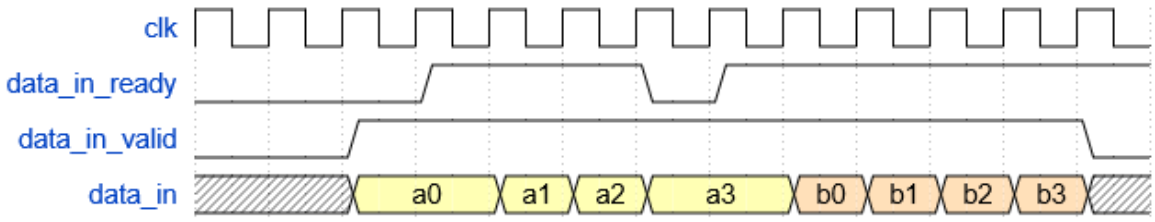

Figure 1. Input interface.

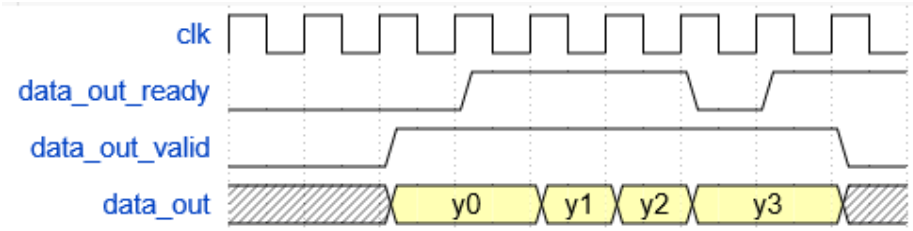

Figure 2. Output interface.

Performance requirements

The design must be able to run at minimum 200MHz.

Task 1: Draw microarchitecture

Examine the code for the mega_adder. Draw the microarchitecture for the data path part of the design. Examine the code of the control part to understand how it controls the data path.

Task 2: Simulate the circuit

Run the simple testbench that has been made for the design. Check that the design produces the correct result.

Task 3: Count the number of flip flops that should be inferred during synthesis

Count the number of flip-flops that will be inferred during synthesis. Do this by counting the number of flip-flops that will be inferred by each process in the design and add up the numbers.

Run synthesis and check that the number of flip flops you counted matches that inferred during synthesis.

Task 4: Synthesize the design and report the max clock frequency.

Synthesize the design by pressing “Run Synthesis”. Constraints have been set up so that the target frequency is 200MHz. The synthesis tool will not be able to meet those constraints. See Appendix A at the end of this assignment to see how to set and find timing constraints.

Task 5: Find the critical path in the design and improve the timing.

Locate the critical path in the design and improve the timing. This will most likely require a change in the microarchitecture. Draw a new version of the microarchitecture for the design that will have a shorter critical path.

Task 6: Write RTL code, test and synthesize the new design

Implement the design you suggested in Task 5. Check that it works as expected and that the timing constrains are met.

Appendix A: Reporting timing and editing constraints

The screenshot below shows that the clock period is set up to be 5ns. This corresponds to a clock frequency of 200MHz. When you press the "Report Timing Summary" to the left, the "Design Timing Summary" window will appear.

The "Design Timing Summary" window below shows that we have a negative slack of 1ns. This means that the max frequency is around 166MHz instead of the required 200MHz.

If you want to change the constrains, you can just press "Edit Timing Constraints" and change the period of the clock from 5ns to e.g. 10ns. Hopefully the design will be able to meet those relaxed constraints.