GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL AND COMPUTER ENGINEERING

Lab 7: Feedforward Processor Pipeline with Rounding, Overflow/Underflow and Saturation

---

Due Date : Thursday October 26 (11:55 pm)

---

1. Introduction

One of the biggest problems with Customizable DSP ASICs is the fixed-point approximation of floating point operations. In General Purpose GPUs/CPUs, this is not an issue since we have separated data-paths for the integer and fractional part. Since we have limited #bits on an ASIC, in many situations, we must turn to fixed point operations like Rounding, Overflow/Underflow detection and Saturation. This obviously comes at the cost of decrease in the resolution. The rationale behind how the fixed-point resolution is chosen is determined by a lot of factors namely resolution of ADCs, Tolerable Limit of the SNR which are determined by System Engineers. We as hardware engineers implement our datapath operations in accordance with these specifications. This lab is the same as Lab4 where we designed a processor pipeline. The only difference is that we have added a sequential stage during which the overhead fixed point operations are executed. We could have merged these operations in the second stage itself which would not have change the latency. But it would have affected the maximum frequency at which the design could have been run. This is of course a tradeoff. For this lab, we are going to sacrifice latency. Again, all operations are **signed.** The template for the processor is provided in "processor_v2_top.vhd" in /src directory.

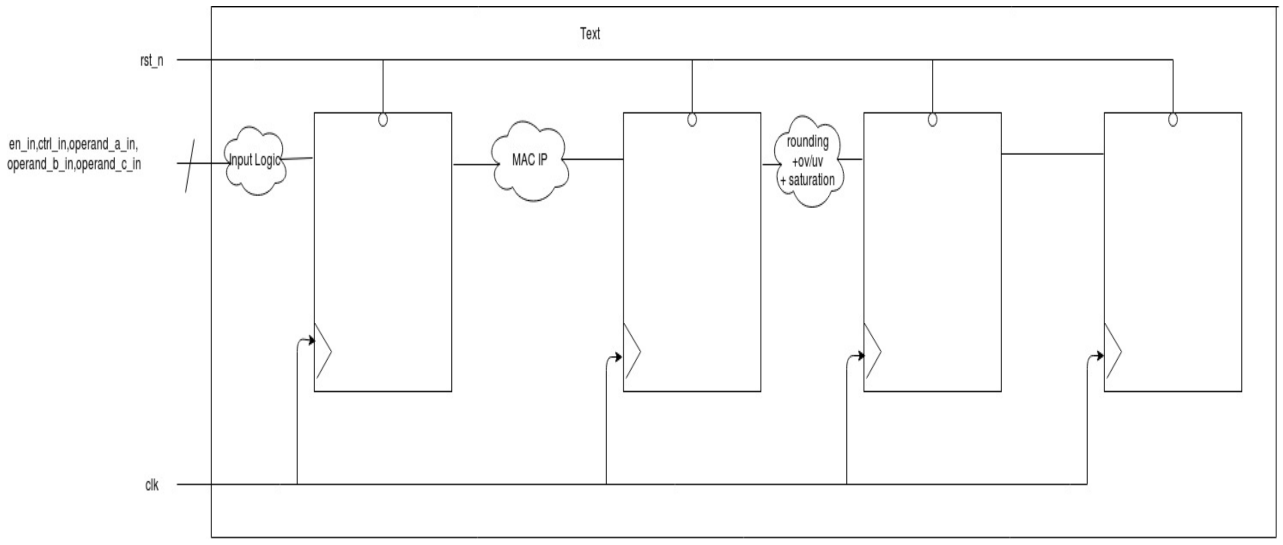
The explanations of the input and output ports are given below.

i. clk => Clock signal
ii. rst_n => Reset signal. Note that the sequential elements that are used in your design should have asynchronous active low reset. i.e. if the reset is pulled low, the flops should be reset regardless of the state of the clock

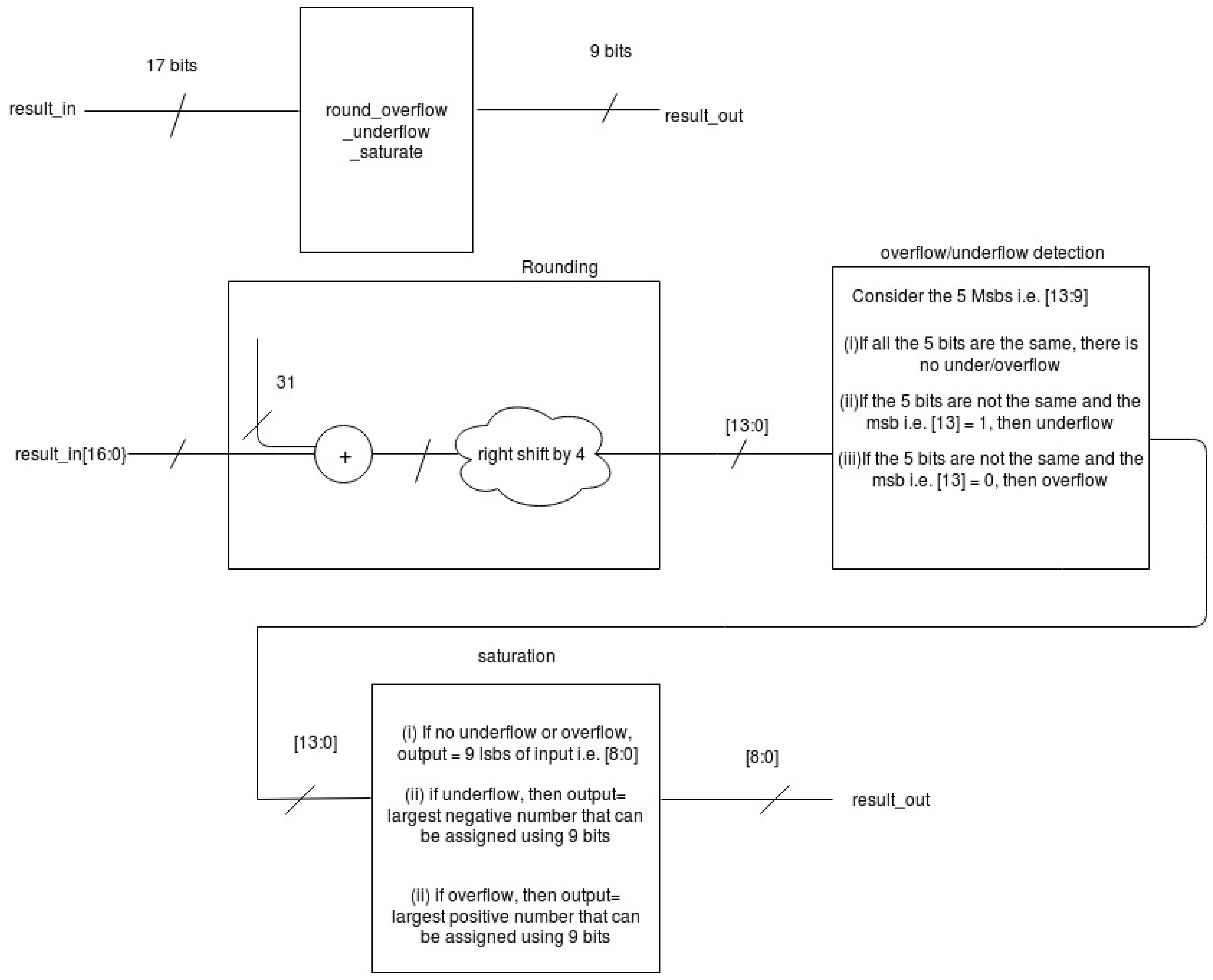
iii. en_in => Input enable signal. It qualifies all the inputs. If en_in =’1’, you should capture all the inputs (i.e. ctrl and operands). Just like in Lab4, we may or may not consider this for latching the input data.
iv. ctrl_in=> The control signal indicates what operation is to be performed. Use the same table given in Lab_3 for decoding the operation
v. operand_a_in,operand_b_in,operand_c_in => These are the data inputs
vi. out_valid => Ctrl signal for the output. When ctrl =’1’, the output is valid
vii. result_out=> the output of the arithmetic operations.

ctrl_in	Operation for mac_unit_out
0	a_in + c_in
1	a_in*b_in
2	(a_in*b_in) + c_in
3	0

The controls remain the same.



After the execute stage, we have the fixed-point overhead operations.


17 bits
result_in
round_overflow _underflow _saturate
9 bits
result_out
Rounding
31
result_in[16:0]
+
right shift by 4
[13:0]
overflow/underflow detection
Consider the 5 Msbs i.e. [13:9]
(i)If all the 5 bits are the same, there is no under/overflow
(ii)If the 5 bits are not the same and the msb i.e. [13] = 1, then underflow
(iii)If the 5 bits are not the same and the msb i.e. [13] = 0, then overflow
saturation
[13:0]
(i) If no underflow or overflow, output = 9 lsbs of input i.e. [8:0]
(ii) if underflow, then output= largest negative number that can be assigned using 9 bits
(ii) if overflow, then output= largest positive number that can be assigned using 9 bits
[8:0]
result_out

The logic for the fixed-point overhead operations is described in the diagram above.
There is a sample input file from which the data is read into the testbench . The testbench dumps out an output file. Compare this output with the reference output file provided with the assignment. The grading will take into consideration the functional correctness, utilization (area), and timing of the design. Note that you must create a clock for the Vivado Process Flow. This can be done by adding “constraints_lab7.xdc”to the constraints while creating the project. The constraints file contains the name and period of the clock which is created. 3 re-submissions are allowed for this lab. The resources information is obtained from “processor_v2_top_utilization_placed.rpt”. The timing information is obtained from “processor_v2_top_timing_summary_routed.rpt”. The power information is obtained from “processor_v2_top_power_routed.rpt”.

2. Instructions

i. Write the design in vhdl(processor_v2_top.vhd) per the block diagram. The template for the design has been provided in processor_v2_top.vhd in /src folder. Verify the design by comparing it with the reference output. (70% of grade) (**Note that if the design is not verified, the block diagram and implementation will not be considered**)
ii. Perform Synthesis and Implementation for the design. List the area (#Slice LUTs, #Bonded IO Buffers) and Timing (Worst Negative slack) (10 % of grade).
iii. Answer the following question – "In this lab, we have the done the fixed-point overhead operations considering signed arithmetic. If the system just had positive numbers (i.e. unsigned operations), what changes would you do to 3 stages of the overhead system i.e. (i)Rounding(ii)Overflow/Underflow detection(iii)Saturation?" (20% of grade)

3. Deliverables

i. Create a PDF which lists #resources (Slice Logic Distribution, IO Information, #Primitives) and Worst Negative Slack of "processor_v2_top.vhd". The title of the document should be of the form "lab7_firstname. lastname.pdf"
ii. The PDF should also have the answer to the question
iii. The design top file "processor_v2_top.vhd"
iv. Other modules (if applicable)
v. Output file for the design.

I have been facing grading issues because students are not following the naming convention of the files and folders. **Please do not move these files into a folder. Upload all the attachments directly on t-square.**

Note: Late submissions are not accepted. In case of extraordinary circumstances, written permission must be obtained from Dr.Madisetti.