

GMU ECE 545

Fall 2007 Midterm

Name: _____

G Number: _____

Score: _____

Problem 1 (5 points)

Design a combinational logic block SelL. The circuit takes in two N-bit signed two's complement inputs A and B, and outputs the larger of the two: Specifically,

$$Z = A \text{ if } A \geq B, \text{ else } Z = B$$



Restriction: You are not allowed to use the comparison operators (<, <=, >, >=) in your block diagram or code. Use another method. You can use any other operators.

Note: This is a simple combinational logic block (no clock, reset, etc.)

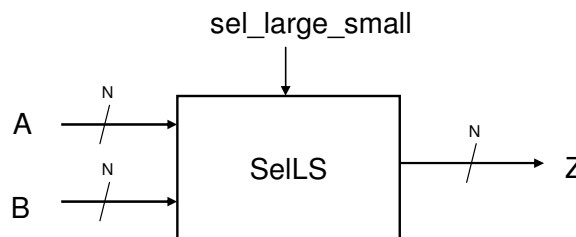
- (a) Draw the datapath block diagram of your circuit which shows the contents of SelL. Mark clearly the sizes of all buses. You can use basic building blocks such as logic gates, muxes, demuxes, decoders, priority encoders, adders, subtractors, multipliers, etc. (2 points)
- (b) Write the synthesizable VHDL circuit for the SelL entity, including library declaration, entity declaration, and architecture. N should be a generic integer with a default value of 8. (2 points)
- (c) Now we will enhance the design by adding an additional input: the input sel_large_small selects whether to output the larger or smaller of the two inputs:

If sel_large_small = 1 (i.e. put the largest input on Z)

$$Z = A \text{ if } A \geq B, \text{ else } Z = B$$

If sel_large_small = 0 (i.e. put the smallest input on Z)

$$Z = A \text{ if } A < B, \text{ else } Z = B$$



Draw the datapath block diagram of your circuit which shows the contents of SelLS. Mark clearly the sizes of all buses. Design for minimum area/complexity. You do not need to code the VHDL. (1 point)

Problem 2 (10 points)

Draw a **datapath block diagram (5 points)** and **detailed controller ASM (5 points)** for the digital circuit capable of executing the function described below. In your block diagram mark clearly the sizes of all buses. Design your circuit for minimum area and hardware. In particular the circuit must not store input data before processing.

The circuit accepts samples from four A/D converters, provided at the inputs A, B, C, and D, respectively, as N-bit signed numbers in two's complement format. The samples are bursted by each A/D converter in groups of three, separated by 10 ns within each group, as shown in the figure below.

Two consecutive "bursts" of the samples are separated by 40 ns. The first burst is preceded by an active value of the input **start**.

For each burst:

When the circuit receives the first group (marked A00, B00, C00, D00 below) of a burst it should compute the largest value of A, B, C, and D and save this value.

When the circuit receives the second group (marked A01, B01, C01, D01 below) of a burst it should compute the smallest value of A, B, C, and D and save this value.

When the circuit receives the third group (marked A02, B02, C02, D02 below) of a burst it should compute larger of the following two values: largest(A,B) and smallest(C,D). In other words it should compute largest(largest(A,B), smallest(C,D)) and save this value.

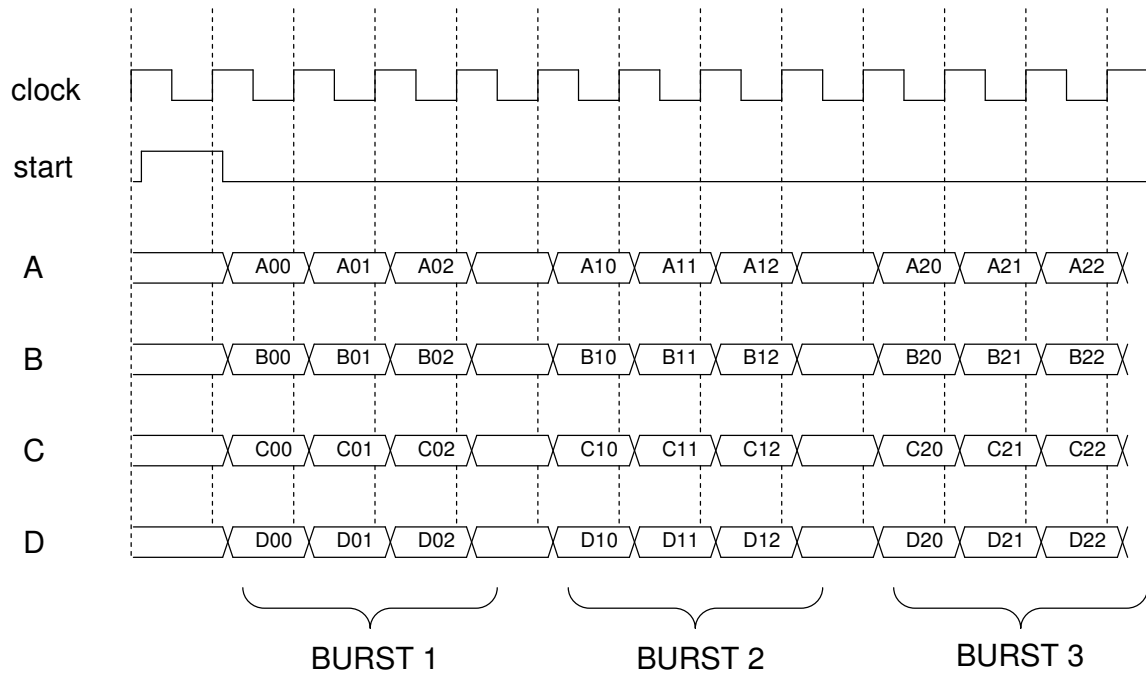
After the third group it should then add these three saved values together to create a burst_sum. This burst_sum should be added with the burst_sum of the previous bursts and stored in an accumulator, i.e. $\text{accumulator} \leq \text{accumulator} + \text{burst_sum}$. If the burst_sum is equal to zero, a burst_zero counter should also be incremented. The accumulator and burst_zero counter should be initialized to zero before the first burst.

After 32 bursts:

After there have been 32 bursts the circuit asserts the **done** output indicating the accumulator output and burst_zero counter output is valid. The circuit holds the done output at 1 indefinitely (i.e. until the circuit is reset).

Note: You can use the selL and/or selS blocks from the previous problem as building blocks (even if you did not complete the previous problem). You only need one ASM for the entire circuit.

Timing Diagram:

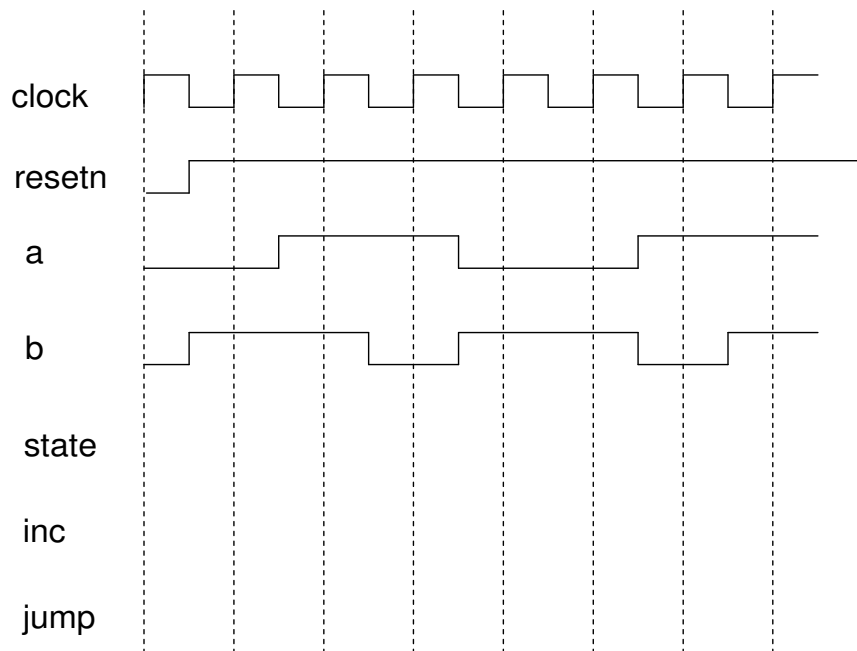
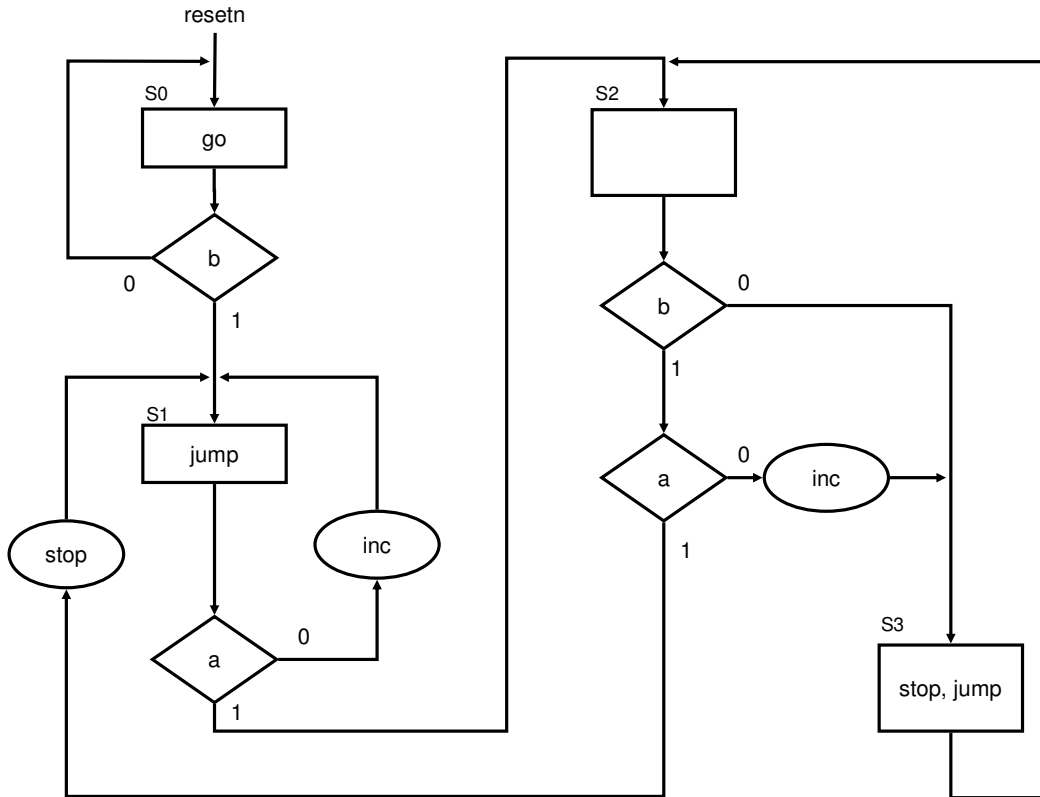


Interface:

Port	Direction	Bits	Explanation
clock	in	1	System clock.
resetsn	in	1	Active-low asynchronous system reset.
A, B, C, D	in	N	N-bit inputs from the A/D converters.
start	in	1	Signal set to high one cycle clock to indicate the start of 32 consecutive bursts.
accumulator_out	out	N+7	Accumulator output.
burst_zero	out	6	Burst-zero counter output.
done	out	1	Set high when 32 bursts have been accumulated.

Problem 3 (5 points)

- (a) Translate the ASM chart given below into the corresponding synthesizable VHDL code, **including library declaration, entity declaration, and architecture**. Assume resetn is an asynchronous active-low signal. (3 points)
- (b) Supplement the given waveform below with the values of state, inc, and jump. (2 points)



Problem 4 (5 points)

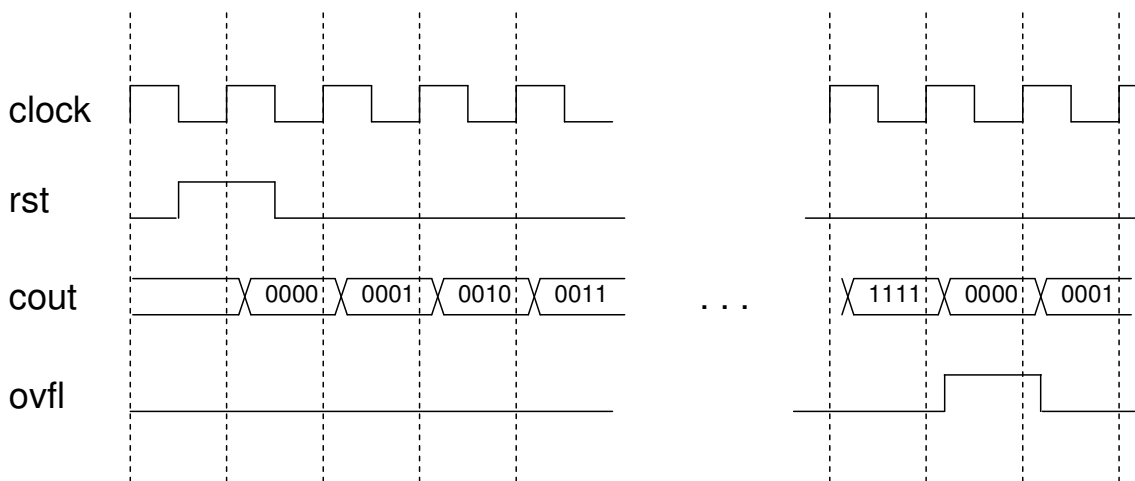
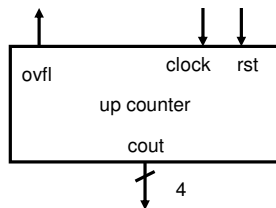
- (a) Translate the counter shown below to the corresponding synthesizable VHDL code, **including library declaration and architecture**. You do not need to code the entity declaration. (2 points)

Reset: Reset is an active-high synchronous signal. When reset is asserted high, the counter is reset to the value “0000” on the next rising clock edge.

Overflow: The counter has an overflow flag (ovfl) which goes high for one cycle when you transition from a “1111” output to a “0000” output, as shown in the timing diagram. **Note that ovfl should not go high when the counter is reset, only when it overflows.** Before the counter is reset for the first time, it is okay if ovfl and cout are unknown.

Entity (you do not need to include this in your code):

```
entity upcount is
  port (clock: in std_logic;
        rst: in std_logic;
        ovfl: out std_logic;
        cout: out std_logic_vector(3 downto 0));
end upcount;
```



(b) Translate the given block diagram to the corresponding synthesizable VHDL code, **including library declaration and architecture.** You do not need to code the entity declaration. (3 points)

Reset: Reset is an active-high synchronous signal. When reset is asserted high, the R register is reset to the value “11001” for R[4] down to R[0] respectively, and the counter is reset to the value “0000” (as described earlier) on the next rising clock edge.

Enable: Enable is an active-high synchronous signal. When enable is high, the R register takes in new data in normal operation. When enable is low, the R register holds its value. **Reset has a higher priority than enable** (i.e. if reset is high the register resets regardless of the value of enable).

Entity (you do not need to include this in your code):

```
entity prob4 is
  port (clock: in std_logic;
        reset: in std_logic;
        enable: in std_logic;
        din: in std_logic;
        dout: out std_logic);
end prob4;
```

Note: Assume upcount is a component declared in the package mypack in the work library.

