

ECE 366, Fall 2001, Instructor: Prof. Shantanu Dutt

Midterm Exam: Fri., Oct. 19, Time: 10-10:50 am
Exam Format: Closed Book, Total Points: 120

Important Note: You need to show all your work **clearly** in deriving the answers. Just writing down the final answers is not enough.

Suggestion: Begin by reading all questions and do those first that you think you know best.

1. **[2's complement arithmetic]** Let $-A$ be an n -bit negative number in 2's complement representation stored in a register R . Prove that an arithmetic shift right (ASR) of R , will result in R having the number $\lfloor -A/2 \rfloor$ stored in it, where $\lfloor X \rfloor$ is the largest integer that is $\leq X$ (e.g., $\lfloor 5/2 \rfloor = 2$, $\lfloor -5/2 \rfloor = -3$). You can use in your proof the property that a logical shift right (LSR) [in which 0 moves into the MSB of the number] of a positive number B results in the number $\lfloor B/2 \rfloor$. **20**

Hint: First prove for the case when A is even and then extend the proof to an odd A .

2. **[C.U. Design for Instruction Processing]** (a) Draw a Moore FSM for that part of the control unit of the `myth8` processor that performs the generic rotate (left or right) instruction given by:

`[ror r_i r_i 4-bit-constant X].`

Semantics: If $X \geq 0$ rotate r_i left by an amount of X bits, else ($X < 0$) rotate r_i right by an amount of $|X| = -X$ bits ($|X|$ is the magnitude of X).

You need to only provide the states after the `decode` state of the `myth8` control unit.

With each state, provide a description of what is taking place in it using the register-transfer language notation done in class (e.g., $r_i \leftarrow r_i + 1$, $mdr \leftarrow mem_data_bus$ for writing into registers r_i, mdr , respectively, and $r_j \rightarrow A_Bus, mar \rightarrow mem_addr_bus$ for reading/connecting r_j, mar , respectively, to the specified buses.).

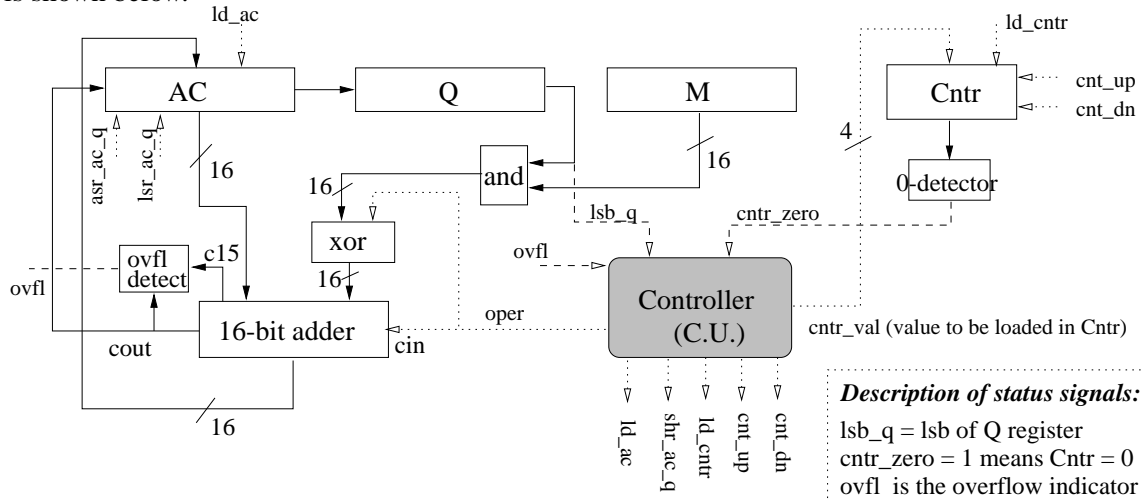
If appropriate, you can show transitions from the state(s) you design to a `fetch0` state without giving details of the `fetch0` state. The `myth8` processor organization is provided on the next page.

Grading criteria will be based on correctness, speed in cc's and number of states in decreasing order of importance. **30**

- (b) Not counting the fetch and decode times, determine the cc's needed for your FSM design to perform: (i) a rotate left of 3 bits; (ii) a rotate right of 4 bits. **10**

3. [Add-&-Shift Multiplication]

The hardware along with control signals from the controller and status signals to the controller for a 16-bit 2's complement A&S multiplier to perform $P = X \times Y$ (X is the multiplier, Y the multiplicand) is shown below.



Description of control signals:

ld_ac = 1 means load/write AC
 asr_ac_q=1 means asr of AC-Q
 lsr_ac_q=1 means lsr of AC-Q
 w/ cout into msb of AC
 oper = 0/1 means ADD/SUB resp.
 ld_cntr = 1 means load/write Cntr

cnt_up = 1 means Cntr <- Cntr +1
 cnt_dn = 1 means Cntr <- Cntr - 1
 cntr_val is the value to be stored
 in Cntr

Description of combinational logic units:

ovfl detect: c15 xor cout
 and: and's each bit of M w/ lsb of Q
 xor: xor's each bit of "and o/p Z" w/ "oper"
 => xor o/p is Z if oper=1 else it is Z
 0-detect: o/p = 1 of Cntr=0, else o/p = 0

NOTE: All write/shift/cnt up/cnt dn operations complete only at the +ve edge of the next cc (since they all load new values into regs)

NOTE: Signal lines without any specified width are 1-bit wide

The algorithm for n -bit 2's complement multiplication is specified below:

```

Initialize AC = 0; Q = Multiplier (X); M = Multiplicand (Y).
Do the following steps  $n - 1$  times
{
  If LSB of Q is 1 then AC = AC+M else AC = AC;
  If (no overflow in the addition) then
    Perform an arithmetic shift right (ASR) of AC-Q register combination by 1 bit
  else
    Shift right  $C_{out}$ -AC-Q register combination by 1 bit
}
If LSB of Q is 1 then AC = AC - M else AC = AC; /* the LSB of Q now is the sign bit of X */
If (no overflow in the subtraction) then
  Perform an arithmetic shift right (ASR) of AC-Q register combination by 1 bit
else
  Shift right  $C_{out}$ -AC-Q register combination by 1 bit;
Final product is in AC-Q register.
  
```

(a) As far as the controller FSM is concerned, assume that initially $AC = 0, Q = X, M = Y$ (you don't need to design a state to do this, but assume this is done in a LOAD state). **Also assume that addition/subtraction takes 3 cc's and shift right of any variety takes 1 cc.** Design the FSM of the controller in the above figure starting from after the LOAD state (just show this state transiting to the first state of your FSM design) for the 16-bit multiplication to be performed. Assume that this is a positive edge triggered system.

With each state, provide a description of what is taking place in it using the register-transfer language notation mentioned earlier.

Grading criteria will be based on correctness, speed in cc's and number of states in decreasing order of importance.

50

Correctness Hints:

- The controller does not need to determine if the input to the **xor** unit is Y or 0 based on the LSB of Q in a particular iteration. This is determined by the **And** combinational unit. However, the controller does need to figure out whether to perform an add or subtract when the LSB of Q is 1 (it can then do add or subtract by forcing the appropriate value on the `oper` control signal).
- The `cnt_up` or `cnt_dn` signal to the counter will result in the counter incrementing or decrementing, respectively, by 1 on the next positive edge of the clock cycle (`cc`), since incrementing or decrementing the counter is equivalent to writing to the counter register (i.e., to its FFs). Thus the controller can only check the `cntr_zero` status signal in a state **after** the state in which the `cnt_up` or `cnt_dn` control signal is set. Note also that it does not necessarily have to be the very next state in which the `cntr_zero` signal is checked; it can be checked in any state after the state in which the `cnt_up` or `cnt_dn` control signal is set, as long as there are no states in between in which the counter value is changed.
- Similarly, the $AC-Q$ register combination is shifted right only at the beginning of the next `cc` after the state in which either the `asr_ac_q` or `lsr_ac_q` control signal is set, again because a shift right (or any other register changing function) writes to the register. Thus, the controller can only check the LSB of the $AC-Q$ register (status signal `lsb_q`) in a state after the state in which either of the above two shift right control signals is set. Again, as in the counter case, the LSB checking state need not necessarily be the state right after the state in which either of the above two shift right control signals is set, but can be any state after it, as long as no intermediate states change the content of the Q register.

Speed Hint: In order to increase the speed of your design, in the controller FSM skip the 3cc add or subtract portion when the 1sb_q status signal is 0 (i.e., LSB of Q is 0).

(b) Analyze the number of cc's it will take to perform the 16-bit multiplication using your FSM design assuming that X has 8 1's and 8 0's. 10

