Midterm Solutions

CSEE W3827 - Fundamentals of Computer Systems | Mar. 28, 2019
Spring 2019 | Prof. Rubenstein

This midterm contains 9 pages: a question 0 and 3 other questions, and totals 100 points. To get full credit you must answer all questions. **NO BOOKS, NOTES OR ELECTRONIC DEVICES PERMITTED!** The time allowed is 75 minutes.

Please answer all questions in the blue book, using a **separate** page for each question. **Show all work! We are not just looking for the right answer, but also how you reached the right answer. Right answers without work get no credit!!!**

SECTION 1 students: Please put your UNI on your exam copy and submit it at the end. Do not take with you!

YOUR UNI: ______

SECTION 2 students: You may keep your copy of the exam.

Some advice:

- Be sure to leave some time to work on each problem. The right answer to each problem does not require a very long answer.
- Be sure to start every problem. And take some time to think about how to set the problem up before you start writing.

0. (5 pts) Do the following in the blue book:
   (a) CLEARLY write your
   - name **and UNI**
   - **lecture section number** (1 for 10:10 class, 2 for 11:40 class) on the front cover.
   - P-credit TA and section number (or day/time)

   (b) start each numbered question's solution on a new page. So question 2 should start on a new page, question 3 on a new page, etc.

   (c) label solutions (e.g., 2a, 2b, 2c or 2a, b, c)

1. (35 pts) Given two unsigned n-bit integers, $A = A_{n-1}A_{n-2}\cdots A_0$ and $B = B_{n-1}B_{n-2}\cdots B_0$, suppose you wish to do the subtraction $A - B$, returning both the n-bit unsigned result $R = R_{n-1}R_{n-2}\cdots R_0$ and an additional bit V indicating overflow.

   (a) (7 pts) For humans, it is easy to see whether $V = 1$, even before doing the subtraction. Give the simplest pseudocode[1] which, given A and B as inputs, computes V. [No boolean algebra or circuits needed.]

   Answer: $V = (A < B)$. i.e., if A is a smaller unsigned value than B, then there will be overflow. Otherwise, there won't be: if $A \geq B$, then $A - B$ is positive and is smaller than A, so if A is representable by n bits, then so is R.

   (b) (7 pts) R can be computed by treating A and B as signed 2's-complement values, and performing subtraction (i.e., let $-B = \overline{B} + 1$ then apply the unsigned binary add algorithm to inputs A and $-B$ to generate R), and then re-interpreting R as an unsigned value. R will be correct modulo 2^n, but overflow needs to be determined. For the case where $n = 3$ and $A = 6$, $B = 3$, show that when using the process described above to compute $R = A - B$, neither the traditional unsigned or 2's-complement overflow detectors correctly identify overflow for this unsigned subtraction.

   Answer:
   $A = 110$, $B = 011$, $-B = 101$.

$$\begin{array}{rccccc}
\text{carry} & 1 & 0 & 0 & \\
A: & & 1 & 1 & 0 \\
-B: & + & 1 & 0 & 1 \\
\hline
 & & 0 & 1 & 1
\end{array}$$

   The sum results in 011, which is 3. Hence the correct answer can be represented such that there is no overflow, yet highest-order carry = 1 (unsigned overflow) and 2 highest-order carries differ (2's complement overflow).

   (c) (7 pts) If $A_{n-1} = B_{n-1}$, then overflow occurs when and only when $R_{n-1} = 1$ (take this as a given). How about when $A_{n-1} \neq B_{n-1}$? Give a boolean expression for overflow (V) when it is given that $A_{n-1} \neq B_{n-1}$, and a brief (1 sentence) explanation that explains why this is the right expression. The boolean expression can utilize the following variables:

   - the input or output bits (i.e., any or all bits from A, B, and R),
   - carry bits from the adder that added A and $-B$, where C_i is the resulting carry of the ith most significant bits.

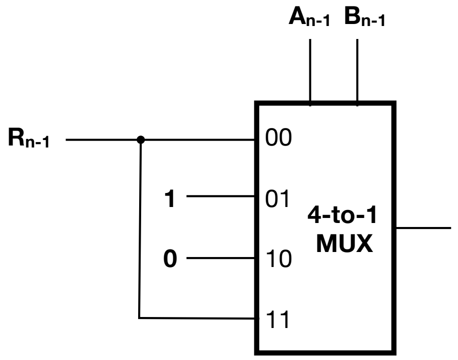
   Answer: When $A_{n-1} = 1$ and $B_{n-1} = 0$, then $A > B$, so the subtraction will be a positive value less than A, and hence will not cause an overflow. When $A_{n-1} = 0$ and $B_{n-1} = 1$, then $A < B$, hence the result of the subtraction will be negative and therefore there must be overflow. Thus, $V = \overline{A_{n-1}}B_{n-1}$. Of course since we know that $A_{n-1} \neq B_{n-1}$, this can be further simplified to just $V = B_{n-1}$ (we'd give full credit for the 2-variable solution as well). It also turns out that you can use the complement of the final carry to determine this case, so we will accept that as a solution as well.

[1] You may assume the pseudocode interpreter recognizes that A and B are unsigned integers.

(d) (7 pts) Use the observations from part 1c to build, using a 4-to-1 MUX, the circuit that computes V for any possible values of A_{n-1} and B_{n-1}. You may assume that the inputs to the circuit can be any of the following:

- the input or output bits (i.e., any or all bits from A, B, and R),
- constants.
- carry bits from the adder that added A and $-B$.
- Basic gates (AND, OR, NOT, XOR).

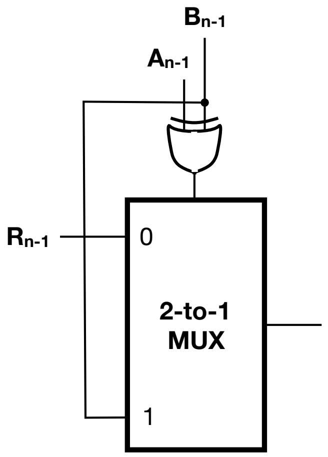
Answer:



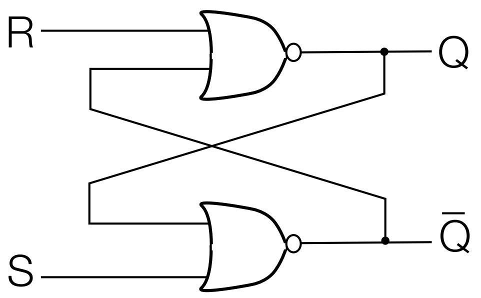
The 4 cases we wish to consider are the values of A_{n-1} and B_{n-1} so feed these in as the selectors. For the two cases where $A_{n-1} = B_{n-1}$, then $V = R_{n-1}$ (we were given this fact in part 1c), and when $A_{n-1} = 1, B_{n-1} = 0$, then $V = 1$, and when $A_{n-1} = 0, B_{n-1} = 1$, then $V = 0$ (solved in 1c). Note that if you simply fed B in for the cases where $A_{n-1} \neq B_{n-1}$, that is also correct (and helpful for the next part).

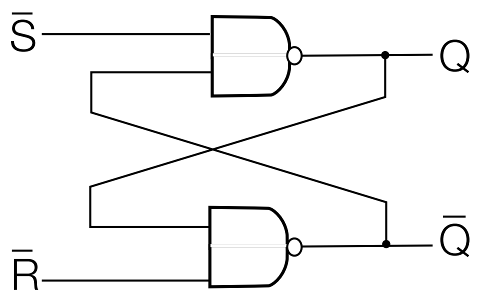
(e) (7 pts) Redo part 1d, but use a 2-1 MUX instead of a 4-1 MUX (the inputs to the circuit can come from the same place as specified in part 1d).

Answer:



The single selector bit indicates whether $A_{n-1} = B_{n-1}$. When they are equal then $V = R_{n-1}$, and when they differ then note $V = B_{n-1}$.



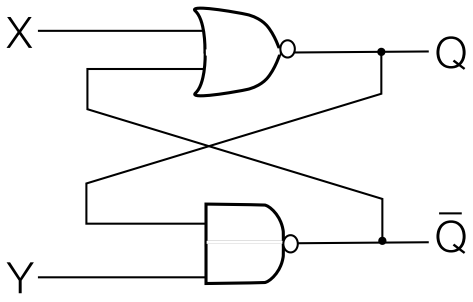


2. (20 pts) Above we have pictured an SR-Latch (on the left) and an $\bar{S}\bar{R}$-Latch (on the right). They can both be set, reset, or made to hold the current value. If it helps, remember these are the characteristic tables for these latches:

SR-latch			
R	S	Q	$\bar{Q}$
0	0	stay same	
0	1	1	0
1	0	0	1
1	1	don't use	

$\bar{S}\bar{R}$-latch			
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
0	0	don't use	
0	1	1	0
1	0	0	1
1	1	stay same	

Below is a circuit that is some weird hybrid of these 2 latches.



Is this hybrid sequential circuit also a latch? Show what effect different inputs (values assigned to X and Y) have on what this circuit outputs. Don't forget to explain why it is or isn't the case that this circuit is a latch. Show all work!

Answer: This circuit cannot be used as a latch because there is no way to set the circuit to holding (and outputting) $Q = 1$.

It is enough to show that with the 4 input combinations, at least one of a latch's necessary functions (set, reset, hold value) cannot be implemented.
To see this, observe that when $X = 1$, the NOR gate will certainly output 0 (hence $Q = 0$). With this 0 feeding into the NAND gate, the NAND gate will certainly output 1 (hence $\bar{Q} = 1$). So having $X = 1$ **resets** the circuit. Similarly, when $Y = 0$, the NAND gate will certainly output 1 (hence $\bar{Q} = 1$). With this 1 feeding into the NOR gate, the NOR gate will certainly output 0 (hence $Q = 0$). So having $Y = 0$ also **resets** the circuit.
So three of the four input combinations ($XY = 00$,$XY = 10$,$XY = 11$) reset the circuit. With only one input combination remaining ($XY = 01$), it cannot be possible to have an input that **sets** the circuit and another one that **holds** the value. A "latch" that either can't be set to 1, or can't hold a value is useless.
For the sake of completeness, let's see how $XY = 01$ affects the output. We can even consider the 4 possible combinations of what might exit from the NAND and NOR gate (even though some of these combinations may not be attainable).

- NAND produces 0, NOR produces 0: The 0 out of the NOR would feed into the NAND changing its output to 1, which would feed into the NOR, keeping its output as 0, so this would change to $Q = 0, \bar{Q} = 1$: **reset**
- NAND produces 0, NOR produces 1: $X = 0$ and the 0 from the NAND feed into the NOR, so its output stays at 1, which feeds into the NAND along with $Y = 1$ so that the NAND stays at 0, keeping $Q = 1, \bar{Q} = 0$: **stay same**.
- NAND produces 1, NOR produces 0: the NAND output of 1 feeding into the NOR with $X = 0$ keeps the NOR output at 0, which feeds into the NAND keeping its output as 1, so $Q = 0, \bar{Q} = 1$: **stay same** or **reset**.
- NAND produces 1, NOR produces 1: the 1 from the NAND feeds into the NOR causing its output to chance to 0, which feeds into the NAND, keeping its output at 1, so $Q = 0, \bar{Q} = 1$: **reset**.

So we see that while it would be possible to keep this circuit held at 1, we have no way to set the circuit to storing a 1.

(Page intentionally left blank)

You may use this for scrap, but it will not be graded.

Question 3 is on the next page.

3. (40 pts) This semester is the first time Professor Rubenstein has ever taught 2 sections of the same class, and he's been trying hard to keep them moving at the same pace. To help him maintain consistency, you are to build a sequential circuit that Professor Rubenstein can use to make sure that neither section falls more than 1 topic behind the other. Each lecture, he teaches some standard set of material and then has the option to either cover or skip one additional topic.

   - The circuit runs on a clock, where he teaches one section's lecture within each clock cycle[2], where the section taught in the current clock cycle differs from the section taught in the previous (and next) clock cycle.
   - At the start of lecture, Professor Rubenstein enters into the circuit a 1-bit input indicating whether he is willing to teach an additional topic near the end of lecture.
   - If he is not willing (input of 0), or the current section is ahead of the other section, the circuit outputs 0, telling him to not teach the additional topic.[3]
   - If he indicates he is willing (input of 1) and the current section is not ahead of the other section, the circuit outputs 1, telling him to teach the additional topic, which he does during the current lecture (current clock cycle).[4]

   The table below is an example with the two sections respectively labeled A and B and shows the number of topics by which A leads. The amount A leads by is shown in an alternate form in the next row as the lead of the current active section (same as the previous row's value when the active section is A, the negation when B). Note that each time the output is 1 when A is being taught, A's lead grows by 1, and each time the output is 1 when B is being taught, A's lead shrinks by 1.

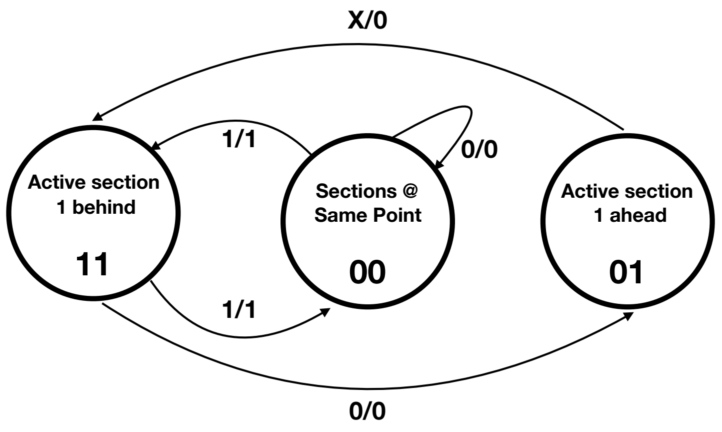
Clock Cycle	0	1	2	3	4	5	6	7	8	9	10
Active Section	A	B	A	B	A	B	A	B	A	B	A
A ahead by	0	0	-1	-1	-1	0	0	1	0	1	1
Active Section ahead by	0	0	-1	1	-1	0	0	-1	0	-1	1
Input	0	1	0	1	1	0	1	1	1	0	1
Output	0	1	0	0	1	0	1	1	1	0	0

   (a) (20 pts) Design a state machine that describes the above system (i.e., takes a 1-bit "random" input each clock cycle that indicates whether there is anything Prof. Rubenstein wants to teach, and outputs whether or not he should teach during the current lecture). Each state should include **a short text description that indicates what the state represents**, as well as the **binary labeling** you choose to use.

   HINT: why, in the example above, did we show the lead from two perspectives (one row from just A's perspective, the other from the active section's perspective)? One perspective might be more efficient in terms of the number of states than the other.

[2]Giving new meaning to feeling the clock is running slow when sitting in his class
[3]He instead entertains students with great jokes and fascinating factoids during the remaining time.
[4]He still works great jokes and fascinating factoids into his remaining lecture.

Answer:
An observation to be made is that the state only needs to track **how many topics the active section is ahead by**. It can be ahead by -1, 0 or 1, and hence our state machine has only 3 states. Note that in the current clock cycle, if the current section finishes off being 1 topic ahead, then in the next clock cycle, when the other section becomes the current section, it will be 1 lecture behind. Similarly, if the current section finishes off 1 lecture behind, then in the next clock cycle, the new "current" section will be 1 lecture ahead. The binary labeling of states is arbitrary, but we will use (2-bit) signed binary to indicate how far ahead the current section is with respect to the other section.



Note: if you used 6 states, you did alot of extra work, but can still receive full credit despite the non-optimality of your solution.

(b) (20 pts) Provide **simplified** algebraic equations when using T flip-flops describing a sequential circuit implementing your state machine. The characteristic table for the T flip-flop is given below, with $T(t)$ being the input to the T flip-flop during clock cycle t.

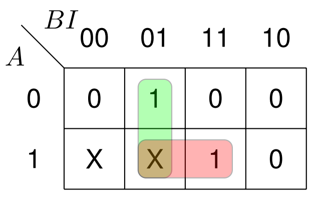
$T(t)$	$Q(t+1)$
0	$Q(t)$
1	$\overline{Q(t)}$

Answer:

$A(t)$	$B(t)$	$I(t)$	$O(t)$	$A(t+1)$	$T_A(t)$	$B(t+1)$	$T_B(t)$
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1
0	1	0	0	1	1	1	0
0	1	1	0	1	1	1	0
1	0	0	X	X	X	X	X
1	0	1	X	X	X	X	X
1	1	0	0	0	1	1	0
1	1	1	1	0	1	0	1

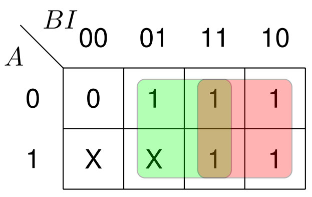
Note that columns for $O(t)$ and $T_B(t)$ are identical and hence are described by the same boolean expressions.

$O(t) = T_B(t)$:



$A \backslash BI$	00	01	11	10
0	0	1	0	0
1	X	X	1	0

$T_A(t)$:



$A \backslash BI$	00	01	11	10
0	0	1	1	1
1	X	X	1	1

$O(t) = T_B(t) = AI + \bar{B}I$.

$T_A(t) = B + I$.