CME 341: Midterm 1 Review

Brian Berscheid

Department of Electrical and Computer Engineering
University of Saskatchewan

Question 1

Re-design the student circuit to implement a combinational circuit which detects whether the value held within `cct_input` is a Fibonacci number. When `cct_input` is a Fibonacci number, `cct_output` should be set to `8'HF7`. Otherwise, `cct_output` should be set to `8'H00`. The clear input should be ignored for this question.

For reference, the first few Fibonacci numbers are:
0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...

Question 1: My answer

My answer:

```
always @ *
  case (cct_input)
    8’d0, 8’d1, 8’d2, 8’d3, 8’d5, 8’d8, 8’d13,
    8’d21, 8’d34, 8’d55, 8’d89, 8’d144, 8’d233: cct_output = 8’HF7;
    default: cct_output = 8’d0;
  endcase
```

Question 2

Re-design prototype student_circuit to describe a synchronous logic circuit with the following behavior:

- The circuit includes an 8-bit wide bank of flip flops which will be referred to here as temp.
- temp should be asynchronously cleared when the clear input is high.
- On the positive edges of the clock, if cct_input is greater than 112, temp should be loaded with cct_input. Otherwise, temp should retain its previous value.
- The student circuit output, cct_output, should be generated combinationally by performing a bitwise XOR between temp and the constant 8'HAB.
- The clear input should not directly connect to cct_output. Of course, it will indirectly influence the output through its connection to temp.

Question 2: My answer

```
reg [7:0] temp;

always @ (posedge clk or posedge clear)
  if (clear == 1’b1)
    temp = 8’d0;
  else if (cct_input > 8’d112)
    temp = cct_input;
  else
    temp = temp;

always @ (*)
  cct_output = temp ^ 8’HAB;
```

Question 3

Re-design prototype `student_circuit` to describe a combinational logic circuit with the following behavior:

- If the clear input is high, `cct_output` will be set to zero.
- Otherwise, `cct_output` should be a reordered version of the bits of `cct_input`. In particular, the ordering of the bits in each of the 4-bit nibbles of `cct_input` should be reversed. If the individual bits of circuit input are referred to (from MSB to LSB) as $X_7X_6X_5X_4\_\_X_3X_2X_1X_0$, the desired ordering is $X_4X_5X_6X_7\_\_X_0X_1X_2X_3$. For example, if the input is `8'd43`, the output should be `8'd77`.
- Note that the circuit does not make use of the clock signal.

Question 3: My answer

```
always @ (*)
  if (clear)
    cct_output  = 8’H0;
  else
    cct_output  = {cct_input[4], cct_input[5], cct_input[6], cct_input[7],
                   cct_input[0], cct_input[1], cct_input[2], cct_input[3]};
```

Question 4

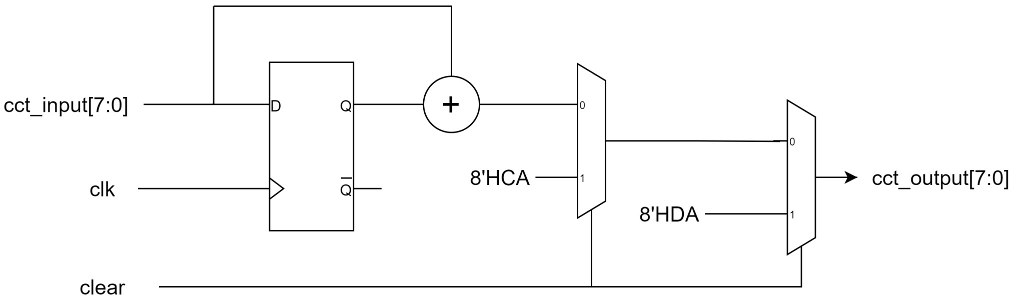
Re-design prototype `student_circuit` to implement a synchronous circuit according to the following specifications:

- On each positive edge of the clock input, `cct_output` is set to the sum of `{4'd0, cct_input[3:0]}` and the previous value of `cct_output`.
- The clear input is used in an active-high synchronous fashion to reset `cct_output` to 0.

Question 4: My answer

```
always @ (posedge clk)
  if (clear == 1’b1)
    cct_output = 8’d0;
  else
    cct_output = cct_output + cct_input[3:0];
```

Question 5



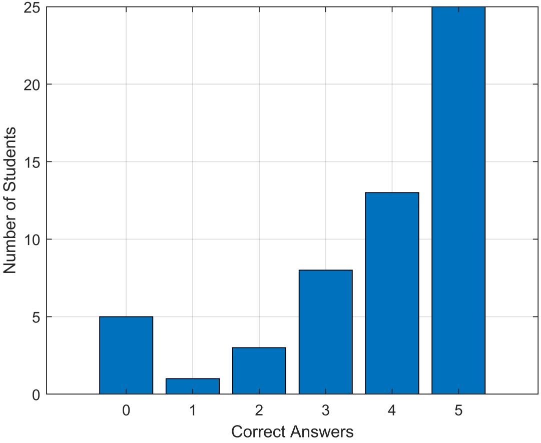
Question 5: My answer

```
reg [7:0] temp;
always @ (posedge clk) // build flip flop
  temp = cct_input;


reg [7:0] s;
always @ (*) // build adder
  s = temp + cct_input;

always @ (*)   // omit first mux, build second mux only
  if (clear == 1’b1)
    cct_output = 8’HDA;
  else
    cct_output = s;
```

Midterm 1 Marks

# Correct	# students	% students
0	5	9%
1	1	2%
2	3	5%
3	8	15%
4	13	24%
5	25	45%

Class average $\approx 3.75/5 = 75\%$



Common Mistakes on the Exam

- Mix-ups with combinational vs synchronous logic:
  - ▷ Combinational logic uses `always @ (*)`
  - ▷ Synchronous logic uses `always @ (posedge clk)`
- Mix-ups between synchronous and asynchronous resets/clears
  - ▷ Asynchronous resets/clears go in the sensitivity list
  - ▷ Synchronous resets/clears don't go in the sensitivity list
- "Multiple drivers" - each signal may be controlled by only one procedure!
- Not using the Modelsim wave window to your advantage - look at the waveforms to see what is not behaving properly if your answer doesn't match the given value.
- A few students used the wrong seed or wrote down the wrong values; I was generous and checked their code in many cases to confirm that it was correct
  - ▷ Don't count on this going forward! Professor Salt will likely not do this!

Final thoughts

- After this midterm, CME 341 becomes more practical (and interesting!). You will be working on some larger designs:
  - ▷ Traffic light controller logic for an intersection
  - ▷ Keycard locking system for hotel doors
  - ▷ An actual microprocessor
- Professor Salt will be teaching the remainder of the course, starting with 'Part 2' on FSMs
- His lectures will be critical to completing the designs successfully, so pay close attention.
- Don't forget: the vast majority of the marks in CME 341 are still to be allocated
  - ▷ If you did well on the midterm, don't relax! Things get more complex from here...
  - ▷ If you didn't do well on the midterm, don't despair! There are opportunities to recover these marks! You can still do well in CME 341!
- Best of luck with the rest of CME 341!

Thank you!
Have a great day!