Counter design, Verilog coding: In your midterm, you arrived at the following design of a special 3-bit counter which repeats 3, 4 one more time (0, 1, 2, 3, 4, 3, 4, 5, 6, 7).

1.1 Now add an EN (enable) control to the above counter reproduced below. Two recirculating muxes were added below for you to complete. Also rewrite the state diagram to include the EN control.

1.1.1 Was your state diagram revision to incorporate an EN input in question above more like the difficult way or the easier way shown in the divider lab with single-stepping? difficult way / easier way
1.1.2 Verilog code for your design on previous page with EN control using the easier way: Complete the clocked always block on the side *treating F as a flag* (rather than as a state variable for a 2-state machine).

Why EN or *posedge EN* is not added to the event list as in

```
always @(posedge CLK, negedge RESET, EN)  
```

or

```
always @(posedge CLK, negedge RESET, posedge EN)  
```

Why *negedge RESET* was added? ________________

1.1.3 The EN (enable) control signal is coming from an external system (external to the counter). If it changes quite late in the clock cycle, we may have ________________ (set-up / hold) time violation, whereas if it changes quite early in the clock cycle, we may have ________________ (set-up / hold) time violation. In ________________ (set-up / hold) time violation check we will consider the *minimum* delay through the mux and we may like to add brute-force delay (using pairs of inverters) in (select one of the two choices, and state why that is the better choice).

(a) inverters between the mux and the register
(b) inverters between the EN input and the mux select line.

1.2 Complete the state diagram below for a similar 3-bit special *down* counter with no enable, but it repeats 5, 4, 3, two more times as shown in (7, 6, 5, 4, 3, 5, 4, 3, 5, 4, 3, 2, 1, 0). Reset asynchronously initializes I to 7 and F1F0 to 00.
2  ( 4 + 18 + 18 = 40 points) 25 min.  RTL design:

2.1  Array divider: Perform 10 divisions $C[I] \leftarrow A[I] / B[I]$ for $I$ going from 0 to 9. Here the dividends are exactly (evenly) divisible by the divisors. So you do not record any remainders.

The array access time is about 80% of the clock period. So you cannot access and simultaneously perform a comparison or subtraction operation on the accessed items in the same clock. So, do not write "if $A[I] > B[I]$" or "if $A[I] == B[I]$" or "$A[I] - B[I]$".


In state LF we load first set $A[0]$, $B[0]$ into $X$ and $Y$. We spend only one clock in LF state.

Do you want to write in the DIV state $C[I] <= Q$; or $C[IP] <= Q$; _____________________

Which of these 4 conditions, you want to use as part of the state transition condition to move out of the DIV state? (circle all your choices)  (I == 9) / (IP == 9) / (I == 10) / (IP == 10)

Complete the state diagram by completing the DIV state’s RTL and state transition conditions.

2.1.1  This is similar to the above problem except that here you are given two sets of dividend-divisor arrays. Besides producing quotient $C[I]$ by dividing $A[I]$ by $B[I]$, you are also producing $F[J]$ by dividing $D[J]$ by $E[J]$. $X$, $Y$, $Q$, and $I$ are used with the $A$, $B$, and $C$ arrays. $U$, $V$, $W$, and $J$ are used with the $D$, $E$, and $F$ arrays. Both divisions go side by side in the DIV state.

Since the number of clocks taken for a division is data-dependent, $I$ and $J$ may not increment simultaneously. For some time $I$ may be ahead and for some time $J$ may be ahead. And as you expect, the last clock of the last division of each set of arrays may coincide or may occur one after another. Complete the DIV state.
You do not need to fill-in the two rectangles labeled A and B in the DIV state.
Add additional lines as needed to see that whoever finishes early will idle while other is working.
3 \quad (4+16+4+22+7+6+19 = 78 \text{ points}) \quad 60 \text{ min.}

Memory and FIFO:

3.1 The 20 address lines $A[19:0]$, on an 8-bit data processor go through their $2^{20}$ combinations and produce an address space of size ________ ________ (specify in GBytes or MBytes or KBytes).

The 24 address lines $A[23:0]$, on another 8-bit data processor go through their $2^{24}$ combinations and produce an address space of size ________ ________ (specify in GBytes or MBytes or KBytes).

3.2 In each of the following 4 copies of the 64KB memory maps (of a 16-bit address $A[15:0]$ and 8-bit data $D[7:0]$ processor) we have shown two equal sized shaded rectangles. They may be in similar slots (similar = differ only in one higher order address bit in identifying them) or in dissimilar slots (dissimilar = differ in more than one higher order address bit in identifying them). See the example on the right side. For each map

(a) specify the size of one shaded rectangle ($2^{11} = 2\text{KB}$ in the example),
(b) the higher order bits and their combination identifying each of the two rectangles

(in the example: you note that one 2KB rectangle is in the bottom-most 2KB area of the 64KB map, it is a 1/32 part and is identified by 5 zeros on the upper 5 bits (16-11=5) $A[15:11] = 00000$; and you also note that the other equal size (2KB) rectangle is at the top of the address space identified by 5 ones on the upper 5 bits (16-11=5) $A[15:11] = 11111$;)

(c) the rectangles occupy similar or dissimilar slots (in the example, they occupy dissimilar slots as 00000 and 11111 differ in more than one bit).
3.2.1 For the lower shaded rectangle in Map#3 above, please state the range of addresses both in binary and hexadecimal.

Range in Binary:       to
Range in Hex:         to

3.3 You need to add 128KB memory staring from C0000H to a 8088 processor based system. You know 8088 is a 20-bit address 8-bit data processor. You are given 4 memory chips to build this memory. Two 64Kx4 and two 32Kx8. Please write addresses in the six boxes in the memory map on the side and complete the memory interface.

3.4 Single Clock FIFO discussed in class:

3.4.1 The FIFO could not have become empty without becoming almost __________ (full / empty) most recently. Similarly, it could not have become full without becoming almost __________ (full / empty) most recently.

We used DIFF[2:0] = 11X = 6 or 7 for the ________ (RAF/RAE) where as DIFF[2:0] = 01X = ________ for the ________ (RAF/RAE)

DIFF[2:0] = 000 ___________ (A / B / C)
A. shall not be part of either RAF or RAE
B. can be part of RAF        C. can be part of RAE
3.4.2 Miss Stanford used DIFF[2:0] = 001 for RAE and DIFF[2:0] = 010 for RAF. Your comment?

_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________

3.5 In the case of a 2-clock FIFO of 8-locations, we used ____________ (2-bit / 3-bit / 4-bit) pointers for WP and RP, ____________ (along with / without any) JK Flip-flop to record AE/AF (AE = Almost Empty; AF = Almost Full). The counters shall use ____________ (Gray code / Binary).

Here, we do Mod subtraction to calculate depth. We do ____________ Mod_4 / Mod_8 / Mod_16. While potentially this mod subtrouter's output can produce ____________ (4/8/16 / other) values for depth, only ____________ (state a number) are legal values and the remaining ____________ (state a number) are illegal values. The illegal values for depth ____________ (will never occur / sometimes occur and are harmful / sometimes occur but are harmless).

DCCW = Depth Calculation Circuit in the WCLK domain
DCCR = Depth Calculation Circuit in the RCLK domain

It is ____________ (safe/unsafe) if the DCCW receives late WP update (late by 1 WCLK clock).
It is ____________ (safe/unsafe) if the DCCW receives late RP update (late by 1 WCLK clock).
It is ____________ (safe/unsafe) if the DCCR receives late WP update (late by 1 RCLK clock).
It is ____________ (safe/unsafe) if the DCCR receives late RP update (late by 1 RCLK clock).

4 ( 8 + 10 = 18 points) 10 min.

Microprogrammed Control Unit Design: In a special application needing 64-steps to be performed repetitively to control 128 Christmas lights, there are no branches (except for natural rolling over from step 63 to step 0). So Mr. Bruin understood that the Branch Address field is not needed. Help him by crossing out all that is not needed in the design on the right. Sizes of the following two items:
(a) Size of the CM (Control Memory):

______ locations x ______ bits per location

(b) Size of the uPC (microprogram counter):

___-bit ____________ (loadable/non-loadable) counter

4.1 Compare the following four methods of designing a general control unit (not the above special CU). Consider only performance by clocks and ignore cost of implementation as well as time-to-market advantage in the case of late design modifications.
(a) Encoded State Assignment method
(b) One-Hot State Assignment method
(c) Output coded State Assignment method
(d) Microprogrammed Control Unit method
5 (8 + 6 + 15 + 12 + 8 = 49 points) 25 min.

Miscellaneous:

5.1 Asynchronous inputs: In an elevator control for a 3-floor building, if multiple requests (among R0, R1, and R2) are true simultaneously (as indicated by MR true (Multiple Requests true)), the elevator should ignore all requests. Compare/comment on the two designs below differing in synchronization.

5.2 At a 4-way stop, if we enter the intersection without looking at the other three cars, we can cause collision. Similarly, if we drive a bus line without getting a grant from the bus arbiter, we can cause bus contention. Is the later statement true in the case of tristate-output devices driving the bus or open-collector-output devices driving the bus or either or none? ____________________________

In the diagram on the right above, the bar is held up by the spring by default. Similarly the SDA line held up (close to VDD) by ________________

If the 4-way handshake is not implemented properly (completely), even though the sender is sending one character "A", the receiver receives several of them if receiver is much ________________ (faster/slower) compared to the sender. Given a choice between (a) a 4-way handshake, (b) a 2-way handshake, and (c) a 2-clock FIFO to provide data transfer between two separately clocked systems, the best choice is ________________________ and the lowest preferred choice is ________________________

Rough work area
5.3 Reproduced below is the function table for the 4-bit magnitude comparison unit 74LS85 from TI (Texas Instruments).

<table>
<thead>
<tr>
<th>COMPARING</th>
<th>CASCADING</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3, B3</td>
<td>A2, B2</td>
<td>A1, B1</td>
</tr>
<tr>
<td>A3 &gt; B3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A3 &lt; B3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>A3 = B3</td>
<td>A2 &gt; B2</td>
<td>X</td>
</tr>
<tr>
<td>A3 = B3</td>
<td>A2 &lt; B2</td>
<td>A1 &gt; B1</td>
</tr>
<tr>
<td>A3 = B3</td>
<td>A2 = B2</td>
<td>A1 &lt; B1</td>
</tr>
<tr>
<td>A2 &gt; B2</td>
<td>A1 = B1</td>
<td>A0 &gt; B0</td>
</tr>
<tr>
<td>A2 &lt; B2</td>
<td>A1 = B1</td>
<td>A0 &lt; B0</td>
</tr>
<tr>
<td>A2 = B2</td>
<td>A1 = B1</td>
<td>A0 = B0</td>
</tr>
<tr>
<td>A2 = B2</td>
<td>A1 = B1</td>
<td>A0 = B0</td>
</tr>
<tr>
<td>A2 = B2</td>
<td>A1 = B1</td>
<td>A0 = B0</td>
</tr>
<tr>
<td>A2 = B2</td>
<td>A1 = B1</td>
<td>A0 = B0</td>
</tr>
<tr>
<td>A2 = B2</td>
<td>A1 = B1</td>
<td>A0 = B0</td>
</tr>
</tbody>
</table>

Suppose TI decided to make a 3-bit magnitude comparator similar to the above. So there are no A3, B3 inputs. Revise the above function table by crossing out unneeded columns and rows.

Using the "tree" cascading below, TI demonstrated how a 24-bit comparator can be built using 6 of their 4-bit units. Fill-in the 4 rows of the tables below.

Base element: 4-bit magnitude comparison unit

<table>
<thead>
<tr>
<th># of level in the tree</th>
<th>Total # of comparison units used</th>
<th>Size of resulting comparison unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>124</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Base element: 3-bit magnitude comparison unit

<table>
<thead>
<tr>
<th># of level in the tree</th>
<th>Total # of comparison units used</th>
<th>Size of resulting comparison unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.4 Readable Logic (Bubble-to-Bubble logic): If at least two out of the three systems on the left is ready one of the right systems is to be told to go ahead (= no need to wait). Each of the right systems have an output called M (M for Maintenance = Out of Service). Both the right systems are never taken out of service simultaneously. If both right systems are available, System_0 shall be told to go ahead.
5.5 Mr. Bruin (our favorite character) was asked to build a 4-input 4-output barrel shifter using 2-to-1 muxes only. He was not taught the design of a logarithmic barrel shifter. He thought of converting the encoded two bit control lines $S_1 S_0$ into four value control lines $V_3 V_2 V_1 V_0$ as shown in the table below, and placing 4 columns of 2-to-1 muxes as shown below under the control of the value bits. But he did not know how to proceed further. Please correct and complete his design.

### Conversion table

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>$V_3$</th>
<th>$V_2$</th>
<th>$V_1$</th>
<th>$V_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The value of $S_1 S_0 = V_3 + V_2 + V_1 + V_0$

The TAs, Graders, and I have enjoyed teaching this course. Hope you also liked it. Hope to see you again in EE457 and perhaps some of you in EE560. - Gandhi