1. (20+8+8 = 36 points) 20 min. Verilog Coding

1.1 We said this a few times, "Later assignments override earlier assignments in HDL coding".

```
always @(*) begin
  D <= B-C;
  A <= 7;
  B <= A;
  A <= 1;
  C <= A;
  E <= B-C;
end
```

```
always @(*) begin
  D <= B-C;
  A = 7;
  B <= A;
  A = 1;
  C <= A;
  E <= B-C;
end
```

```
always @(posedge clk) begin
  D <= B-C;
  A = 7;
  B <= A;
  A = 1;
  C <= A;
  E <= B-C;
end
```

```
always @(posedge clk) begin
  D <= B-C;
  A <= 7;
  B <= A;
  A <= 1;
  C <= A;
  E <= B-C;
end
```

```
assign D = B-C;
assign A = sel ? 7 : 4'bZ;
assign B = A;
assign A = sel ? 4'bZ : 1;
assign C = A;
assign E = B-C;
```

In the 4 codes above the statement `D <= B-C;` is at the beginning of the 6-line code and the statement `E <= B-C;` is at the end of the code. Please comment on whether this made any difference in the procedural block where statements are executed sequentially between `begin` and `end`.

```
In the 4 codes above the statement D <= B-C; is at the beginning of the 6-line code and the statement E <= B-C; is at the end of the code. Please comment on whether this made any difference in the procedural block where statements are executed sequentially between begin..end. Since B and C are assigned using non-blocking assignments, B and C do not change immediately. however D and E are always identical.
```

What would B and C become in each of the 5 separate codes?

| #1 | B = ___; C = ___; D = ___; E = ___ |
| #2 | B = ___; C = ___; D = ___; E = ___ |
| #3 | B = ___; C = ___; D = ___; E = ___ |
| #4 | B = ___; C = ___; D = ___; E = ___ |
| #5 | Assume select is true. B = ___; C = ___; D = ___; E = ___ |
| #5 | Assume select is false. B = ___; C = ___; D = ___; E = ___ |

1.2 Four students described a 2-bit wide 2-to-1 mux in slightly different 4 ways as shown below.

```
always @(*) begin
  X <= X0;
  Y <= Y0;
if (sel)
begin
  X <= X1;
  Y <= Y1;
end
end
```

```
always @(*) begin
  Y <= Y0;
if (sel)
begin
  X <= X1;
  Y <= Y1;
end
end
```

```
always @(*) begin
  if (sel)
begin
  X <= X1;
  Y <= Y1;
end
else
begin
  X <= X0;
  Y <= Y0;
end
end
```

```
always @(*) begin
  if (sel)
begin
  X <= X1;
  Y <= Y1;
end
else
begin
  X <= X0;
  Y <= Y0;
end
end
```

```
always @(*) begin
  #3 Works Y/N
  X <= X1;
  Y <= Y1;
end
```

```
always @(*) begin
  #2 Works Y/N
  X <= X1;
  Y <= Y1;
end
```

```
always @(*) begin
  #3 Works Y/N
  X <= X1;
  Y <= Y1;
end
```

```
always @(*) begin
  #4 Works Y/N
  X <= X1;
  Y <= Y1;
end
```

May 1, 2016 9:08 pm
Explain why if you marked any code(s) as non-working. Code #3 doesn't work, as the last 2 statements are unconditional and will prevail over the previous conditional assignments. So X and Y will permanently be X and Y0.

For the working codes, can we change all non-blocking assignments (<=) to blocking (=)? Y/N

Short explanation together with your general recommendation of using non-blocking or blocking assignments. Consider deep combinational logic such as a tree of OR gates. In a properly designed combinational logic, there shouldn't be any RACE. So there should be no problem if we change all assignment operators to blocking. In fact, that is a preferred practice in Verilog combinational logic coding, because deep combinational logic coding you often produce intermediate results for further processing. These intermediate variables are assigned using blocking assignment operator. That is our general recommendation.

2 (35 points) 25 min. Counters

2.1 Build a 12-bit synchronous down-counter Q[11:0], by cascading the three 4-bit down-counters below. Please provide an overall enable EN12.

2.1.1 Trace the longest path on this 12-bit counter (from any source register(s) to any destination register). Number the items (decrementer, mux, inverter, AND gate...) as #1, #2, etc. in the diagram above after you complete it. Narrate the path below.

1 From Q[5:0], NOR #1, AND #1, AND #2, Mux M1C, Mux M2C
2 Any of the three paths in the 3 counters via the decrementer. If EN2 arrives before the decrementer D3 is done, the critical path is Q[11:8], D3, MIC, M2C.

2.2 Build a 4-bit synchronous free-running (= with no enables) up counter using the 4 single bit counters (toggle flip-flops, some with enable control (the enable control is either active-low or active-high)). Label the outputs Q3, Q2, Q1, Q0 (Q0 is the LSB). Label the overall clock as SCLK (for synchronous counter clock).

What did you do to make it an up counter? Toggle Qi when all the lower FFs are at 1

© Copyright 2015 Gandhi Puvvada
3 (36 points) 25 min. Fixed Priority Resolver, Barrel shifter, active levels

3.1 At the end of the chapter on the above topic, we had the "4 low-active inputs 4 low active output Fixed Priority Resolver" shown on the left. In another design, the bottom two requests are high-active and the top two grants are needed to be high-active. Complete the FPR design on the right using the least number of inverters and 3 standard gates (any mix of AND/OR/NAND/NOR). Name all 3 standard gates except inverters as AND/OR/NAND/NOR.

3.2 We designed a 4-input 4-output right-shifting barrel shifter assuming that the S1 S0 represent right-shifting by 0, 1, 2, or 3 depending on S1S0 = 00, 01, 10, and 11. We assumed that everyone uses unsigned binary numbers when it comes to S1S0. Our design is shown on the far left below. Three more copies of the same design are given for you to modify to suit our special customers from the planet MARS who treated the S1S0 as (a) Gray Code (b) Signed numbers represented in 2's- Complement system (c) Signed numbers represented in Sign-Magnitude system. Negative 1 means rotate left by 1 step which is same as rotate right by 3. Negative 2 means rotate left by 2 steps which is same as rotate right by 2. Modify our original design to the least amount by adding one or two gates as necessary between the external select pins and internal select lines. Relabeling of inputs and/or outputs is allowed if needed. Recall that our right-shifting barrel shifter can be used as a left-shifting barrel shifter by just relabeling!
4 \quad (20 + 10 + 12 = \text{42 points}) \quad 25 \text{ min.}

Memory width and depth expansion: Build a 16Kx8 ROM memory (with an overall \text{CE} chip-enable) using the 7 smaller ROM chips as shown in the map on the side. The 16Kx8 ROM has 14 address pins and 8 data pins. Complete the design below. Add missing labels, wires, and gates.

4.1 Given below is the address map of a 8088 processor in a PC-XT computer (of 1980’s). You know that the 8088 processor has 1 Megabyte address space (address range: 00000 H - FFFFF H). Find the size and address range of the shadowed area in the map below.

| Size: \(16\) KB = \(16\) Bytes |
| Address range: \(A8000 - ABFFFH\) |

The shaded area is designated by \(A_{19-14} = 1010\_10\). So the range is those upper 6 bits appended with 14 zeros to those appended with 14 ones. So it is a 16KB \((2^{10})\) from \(A8000\) to \(ABFFFH\).
4 ( 20 + 10 + 12 = 42 points) 25 min.

Memory width and depth expansion: Build a 16Kx8 ROM memory (with an overall CE chip-enable) using the 7 smaller ROM chips as shown in the map on the side. The 16Kx8 ROM has ___ address pins and ___ data pins.

Complete the design below. Add missing labels, wires, and gates.

4.1 Given below is the address map of a 8088 processor in a PC-XT computer (of 1980’s). You know that the 8088 processor has 1 Megabyte address space (address range: 00000 H - FFFFF H). Find the size and address range of the shadowed area in the map below.

<table>
<thead>
<tr>
<th>Size: ____ KB = 2^12 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>=&gt; Address pins A[12:0] change from 0’s to 1’s</td>
</tr>
</tbody>
</table>

Address range: A8000 - ABFFH

The shaded area is designated by A[19-14] = 1010_10. So the range is those upper 6 bits appended with 14 zeros to those appended with 14 ones. So it is a 16KB (2^12) from A8000 to ABFFH.

---

C Copyright 2015 Gandhi Puvvada
4.2 A RAM (=RWM) has one __________ control pin compared to a ROM because __________
Data lines on a RAM are __________ (unidirectional/bidirectional). Address lines are
bidirectional on __________ (a ROM/ a RAM/both/neither). Microprocessor is _______
(the master/a slave) where as the memory is _______ (the master/a slave).

5.1 You are familiar with the TREE structure OR gates to reduce the delay.
To OR 8 variables X0 to X7 we needed 7 2-input OR gates in 3 levels as shown on the side.
(A) To OR 1024 variables A1023 to A0, using 4-input OR gates, you need ________ 4-input OR
gates arranged in ______ levels. 1024 256 gates 256 64 gates 16 4 gates 4 1 gate
(B) If each of the 2-input OR gates has 1ns delay specification, it takes _______ ns to OR X0 to
X7 in the TREE on the far left below.
(C) Now suppose we need to OR only 6 items X0 through X5. X0 and X1 arrive at 0ns. X2
arrives at 3ns, X3, X4, and X5 arrive at 1ns. Show the best design in the middle so far as total
delay is concerned. Now exchange two input of the middle design to create the worst design in
the far right.

5.2 Topic: Readable logic

Using only 2-input totem-pole gates (AND, OR, NAND, NOR) and in addition as few inverters as needed, complete
the logic to allow the right two systems to go ahead (meaning that there is no need to wait) if both
of the left two systems are ready and at least one the boiler is not too hot (i.e. if either H1 or H2 is a 0). There is a overall enable EN.
Only if the overall enable is active (EN = 0) the two right systems are permitted to go ahead if other three signals

5.3 A pico second is equal to __________ (10^-3 / 10^-6 / 10^-9 / 10^-12 / 10^-15) seconds
MTBF stands for __________ a term used to express reliability of electronic equipment
Double synchronization (as compared to single synchronization) reduces the probability of the signal going into __________. Asynchronous signals are synchronized to receiver's
time instead of sender's clock. Reset signal ______ (does not require also requires) synchronization. It is best to perform CDC (Clock domain crossing) using ________
[A/B/C/D] where A = 4-way Handshake, B = 2-way Handshake, C = single clock FIFO, D = 2-clock FIFO.
6.1 A change in depth of the FIFO ________ (will / will not) cause a change in the pin-out of the FIFO because the WP and RP pointers which change with the fifo depth are internal to the FIFO.

6.2 If WP = 5 and RP = 1, the depth is ________ if the FIFO is a 16-location FIFO, and is ________ if the FIFO is a 64-location FIFO.
If WP = 1 and RP = 5, the depth is ________ if the FIFO is a 16-location FIFO, and is ________ if the FIFO is a 64-location FIFO.

6.3 Consider a 32-location 2-clock FIFO using 6-bit counters (in place of 5-bit counters) for the WP and RP pointers. If the producer is much faster, then the FIFO will be running FULL most of the time. So WP may be 100000 (=32) and RP may be 000000 (=0) for quite some time and then finally the consumer consumes one item and in no time, the producer replenishes (fills the FIFO). Then the WP will be 100001 (=33) and RP will be 000001 (=1) for some time. However sometimes the producer may be busy and it is possible that WP is 111110 (=62) and RP is 111101 (=61).

It is ________ (possible / not possible) to have WP = 000001 (=1) and RP = 111111 (=63).
It is ________ (possible / not possible) to have WP = 111111 (=63) and RP = 000001 (=1).
For each of the above 2, if you said "possible", state the depth and if you said "not possible" state your reason. Also state whether you do MOD_32 or MOD_64 subtraction here.

WP = 000001 (=1) and RP = 111111 (=63)
WP = 111111 (=63) and RP = 000001 (=1)

6.4 Never ever synchronize by sampling and holding (circle all applicable):
(a) a single-bit data item
(b) a multi-bit data item where multiple bits could be changing simultaneously
(c) a multi-bit data item where one bit at most changes at any time
(d) none of the above

6.5 It is ________ (necessary / not necessary) that the WP and RP are initialized to zero-zero only and nothing else. It ________ (is / isn't) fine if they are both initialized to say two-two.

6.6 Deadlock can be created if the FIFOs are ________ (shallow / deep) and if the frequencies of the producer and consumer are ________ (nearly the same / orders of magnitude different) and used ________ [n-bit pointers for 2^n-location FIFO / (n+1)-bit pointers for 2^n-location FIFO]. Deadlock will never occur irrespective of how deep is the FIFO, how different are the frequencies if we use ________ [n-bit / (n+1)-bit] pointers for a 2^n-location FIFO.

6.7 Gray code counters are wasteful but not harmful in ________ (single-clock / two-clock / both / neither) FIFO.
The 16-bit Gray code 1010_1010_0110_1110 represents an ________ (odd / even) decimal number.
The 16-bit Binary code 1010_1010_0110_1110 represents an ________ (odd / even) decimal number.
7 (60 points) 50 min.

7.1 Given below is the state diagram of the merge-sort lab

We are not interested in the microprogrammed control unit method or a Moore machine here. Let us design a Mealy machine. Merge the three states CMP, SP, and SQ into one state called CMST (compare and store state) below. Complete the state diagram. Assume that P and Q have 4 element each like in your lab. Do not use MCI and such signals. Write $P[I] < Q[J]$ and $I = 3$, etc.

Mr. Trojan (Trojan Bruin) says that the total number of clocks spent in the three states, CMST, RQ, and RP is a constant and is not data dependent. The minimum number of clocks spent in CMST is $\text{___}$ and the maximum number of clocks spent in CMST is $\text{___}$.

Minimum number of clock spent in either RQ or RP is $\text{___}$. The maximum number of clocks spent in RQ or RP is $\text{___}$. Note: (I-3) and (J-3) condition can occur simultaneously. Consider $P[I] = 4, 5, 6, B$ and $Q[J] = 7, 8, 9, A$. Then the {IJ} sequence is 00, 10, 20, 30, 31, 32, 33, [30 or 34]. The 30 or 34 occurs in RP state. The 0 or 4 for J happens for J depending on whether J is implemented as a 2-bit or a 3-bit counter.

7.2 In a variation of the this problem, we need to merge three arrays A[I], B[J], and C[K] into M[L]. The A, B, and C arrays are in ascending order and result array M also shall be in the ascending order. We have only one comparison unit. So compare A with B in CAB state (Compare A with B) first and then move to CACS (Compare A with C and Store) or CBCS (Compare B with C and Store). If we store C[K] in these states, we remain there and process the next C[K]. Otherwise we return to CAB for comparing A with B. Eventually one of the three arrays will be over and then to process the remaining two arrays, we have CABS2, CBCS2, CACS2. Then we have RA, RB, RC state for processing the reaming elements of the last array and the DONE state. The RTL is complete. Add the missing state transitions and transition conditions to complete the state diagram.
If we consume $C[K]$ here because that is smaller than $A[I]$, then there is no need to go back to CAB state as $A[I]$ is still the smaller of $A[I]$ and $B[J]$. So we continue in the CACS1 state unless we are consuming the last $C[K]$ in which case we will be moving to the CABS2 state.

These three states are similar to the CMST state on the previous page, merging the elements of two arrays until one of the two arrays depletes completely.

We enjoyed teaching this course! Hope you liked it! Hope to see some of you in EE457. Grades will be out in a week. Enjoy your winter break!

Happy Holidays!!! - Gandhi, Shreyas, Gunjae, Minnu

HAPPY NEW YEAR 2015!