

Comp 120 Computer Organization
Spring 2005

Solutions to Problem Set #4

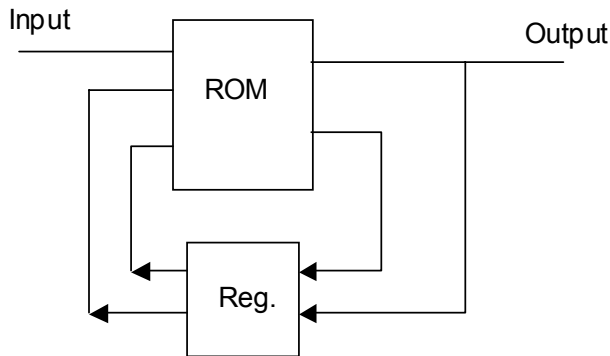
Problem 1. “Beware of the Real Y2K” [35 points]

(A) $2^{11} * 1 = 2048$ bits.

This design is a finite machine, because its output is not strictly a function of the current input. The hold time should be no bigger than the contamination delay. That is, $t_{hold} < t_{cd}$.

(B) S_2, S_0, S_1, S_0, S_0

(C)



| Input | Current State($C_1 C_0$) | Next State($N_1 N_0$) |
|-------|----------------------------|-------------------------|
| 0 | 10 | 10 |
| 0 | 00 | 01 |
| 0 | 01 | 00 |
| 1 | 10 | 00 |
| 1 | 00 | 10 |
| 1 | 01 | 01 |

The ROM now has 3 input lines and two output lines, so its size is $2^3 * 2 = 16$ bits.

(D) The old design requires the input no bigger than 2047, year 2048 will be interpreted as year 0. The new design has no limitation on the length of input, so it is immune to Y2K problem.

The new design may be slower than original one, because it loops output to input through registers. It is also less expensive because the new design uses much less registers and a smaller ROM.

Problem 2. “A Next Step in Ant Evolution” [25 points]

| ; | now | L | R | S | next | T | T | | | | |
|-------|-----|-----|-----|-----|-------|---|---|---|---|---|--|
| ; | --- | --- | --- | --- | --- | L | R | F | M | E | |
| lost | 0 | 0 | - | | lost | 0 | 0 | 1 | 0 | 0 | |
| lost | 1 | - | - | | eorg | 0 | 0 | 1 | 0 | 0 | |
| lost | 0 | 1 | - | | eorg | 0 | 0 | 1 | 0 | 0 | |
| eorg | 0 | 0 | - | | borc1 | 1 | 0 | 0 | 0 | 0 | |
| eorg | 1 | - | - | | eorg | 1 | 0 | 0 | 0 | 0 | |
| eorg | 0 | 1 | - | | eorg | 1 | 0 | 0 | 0 | 0 | |
| borc1 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc1 | 0 | 0 | - | | borc2 | 0 | 1 | 1 | 0 | 0 | |
| borc1 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc2 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc2 | 0 | 0 | - | | borc3 | 0 | 1 | 1 | 0 | 0 | |
| borc2 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc3 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc3 | 0 | 0 | - | | borc4 | 0 | 1 | 1 | 0 | 0 | |
| borc3 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc4 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc4 | 0 | 0 | - | | borc5 | 0 | 1 | 1 | 0 | 0 | |
| borc4 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc5 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc5 | 0 | 0 | - | | borc6 | 0 | 1 | 1 | 0 | 0 | |
| borc5 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc6 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc6 | 0 | 0 | - | | borc7 | 0 | 1 | 1 | 0 | 0 | |
| borc6 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc7 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc7 | 0 | 0 | - | | borc8 | 0 | 1 | 1 | 0 | 0 | |
| borc7 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borc8 | 1 | - | - | | eorg | 0 | 1 | 1 | 0 | 0 | |
| borc8 | 0 | 0 | - | | borcx | 0 | 1 | 1 | 0 | 0 | |
| borc8 | 0 | 1 | - | | a | 0 | 1 | 1 | 0 | 0 | |
| borcx | - | 0 | - | | borcx | 0 | 1 | 1 | 0 | 0 | |
| borcx | - | 1 | 0 | | a | 0 | 1 | 1 | 1 | 0 | |
| borcx | - | 1 | 1 | | alost | 0 | 1 | 1 | 1 | 0 | |
| a | 1 | - | - | | eorg | 1 | 0 | 1 | 0 | 0 | |
| a | 0 | 1 | - | | a | 1 | 0 | 1 | 0 | 0 | |
| a | 0 | 0 | - | | borc1 | 1 | 0 | 1 | 0 | 0 | |
| alost | 1 | - | - | | lost | 0 | 0 | 1 | 0 | 1 | |
| alost | 0 | 1 | - | | alost | 1 | 0 | 1 | 0 | 1 | |
| alost | 0 | 0 | - | | lost | 0 | 0 | 1 | 0 | 1 | |

Problem 3. “To Compute A Pipelined Pixel, LEEly” [40 points]

(A) The propagation delay is $(40+10+10)=60\text{ns}$, and the throughput is $1/60\text{ns}$.

(B) The minimum clock period we can use is determined by the slowest component, which is a multiplier. Let us set our clock period to be

$t_{pd}(\text{reg}) + t_{pd}(\text{multiplier}) + t_{su}(\text{reg}) = 50\text{ns}$. We can perform both multiplications in parallel. There is no need to put the two adders in separate clock stages; we can perform both additions in one clock period. Thus, it will take only two clock cycles to perform an LEE evaluation. Our latency is therefore 100ns. Since we produce a value at every clock cycle, our throughput is 1/50ns.

(C) Again, the minimum clock period we can use is determined by the slowest component, which is the 25ns multiplier. Let us set our clock period to be $t_{pd}(\text{reg}) + t_{pd}(\text{multiplier}) + t_{su}(\text{reg}) = 35\text{ns}$. We can combine the two adders internal to the multiplier also, since their combined t_{pd} 's (15) is less than $t_{pd}(\text{multiplier}) = 25$. We'll need one more pipeline stage to compute the final two additions. Thus, the maximum throughput is 1/35ns, and the latency is $3 * 35\text{ns} = 105\text{ns}$.

(D) Each triangle will take $30 * 7 * 35\text{ns} = 7350\text{ns}$. With one LEE unit, we will be able to render $(1 \text{ triangle}) / 7350\text{ns} = 13605.4 \text{ triangles/second}$.