Eun Jung Kim

# Computer Organization
# Homework Set # 1 - Solution Note

**1.** Suppose we enhance a machine to make all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 20 seconds, what will speedup be if half of the 20 seconds is spent executing floating-point instructions? (5 pts)

**Solution: (5' or 3' for Amdahl's law)**

$Speedup = \frac{20}{10+10/5} = 1.67$

**2.** Suppose you have a machine which executes a program consisting of 60% floating point multiply, 20% floating point divide, and the remaining 20% are from other instructions.

(a) Management wants the machine to run 5 times faster. You can make the divide run at most 3 times faster and the multiply run at most 8 times faster. Can you meet management's goal by making only one improvement, and if so, which one? (5 pts)

**Solution: (2.5' each)**

When we only make improvement for divide, we can achieve speedup:
$speedup_{div} = \frac{1}{80\%+20\%/3} = 1.15$

When we only make improvement for multiply, we can achieve speedup:
$speedup_{mult} = \frac{1}{40\%+60\%/8} = 2.11$

We can't meet management's goal by making only one improvement.

**Alternative Solution: (5' or 3' for Amdahl's law)**

There is 20% other instructions cannot be improved, even totally improve multiply and divide, the upper bound for speedup will be $\frac{1}{20\%} = 5$, which is impossible.
So only make one improvement cannot meet the goal.

(b) EJ has now taken over the company removing all the previous managers. If you make both the multiply and divide improvements, what is the speed of the improved machine relative to the original machine? (5 pts)

**Solution: (5' or 3' for Amdahl's law)**

$speedup = \frac{1}{20\%+20\%/3+60\%/8} = 2.93$

**1.8 Solution: (1' each)**

a) 4
b) 11
c) 1
d) 63
e) 42

**1.15 Solution: (1' each)**

a) 10011
b) 11110
c) 1000000
d) 10000000

**1.18 Solution: (1' each)**

a) F0
b) FF
c) 5A
d) 136D

**1.22 Solution: (1' each)**

a) 0100 1111 0101 1110
b) 0011 1111 1010 1101
c) 0011 1110 0010 1010
d) 1101 1110 1110 1101

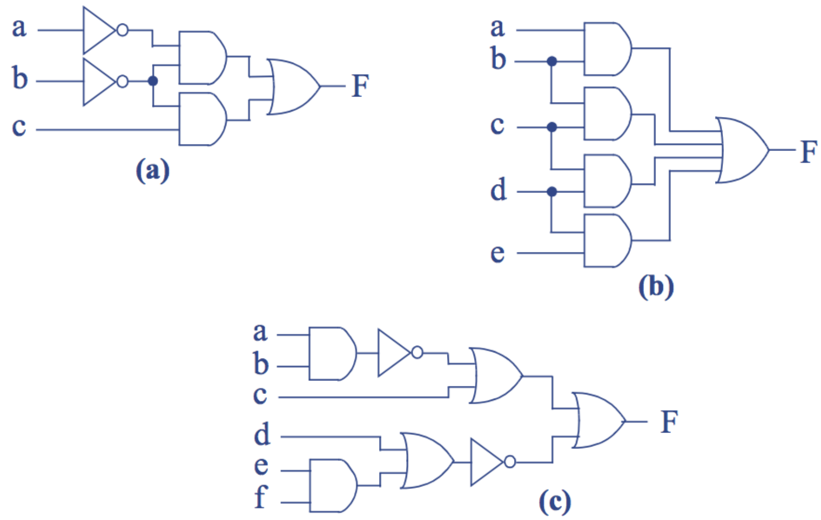
**1.32 Solution: (2': time/trasaction, 3' for either solution)**

40 transactions / second means that decryption should occur at a rate of 1 second / 40 transactions = 0.025 seconds / transaction, or 25ms/transaction. Implementing all three tasks on the microprocessor would result in 50+20+20 = 90 ms/transaction, which is too slow. Implementing any one task as a digital circuit is still too slow. Implementing A as a digital circuit would reduce the time to 1+20+20 = 41 ms. Implementing A and B as a digital circuit would reduce the time to 1+2+20 = 23 ms. Implementing A and C as a digital circuit would reduce the time to 1+20+1 = 22 ms. Thus, either solution suffices. Implementing B and C as a digital circuit would not suffice, as the time would be 50+2+1 = 53 ms. Implementing all three as a digital circuit would result in 1+2+1 = 4 ms/transaction, which is plenty fast but uses extra digital circuitry. Thus, one solution is A and B as digital circuits, C on the micropro- cessor. Another solution is A and C as digital circuits, B on the microprocessor.

**2.12 Solution: (1' each)**

a) F = (1 AND 1) OR 1 OR 0 = 1 OR 1 OR 0 = 1
b) F = (0 AND 1) OR 1 OR 0 = 0 OR 1 OR 0 = 1
c) F = (1 AND 1) OR 0 OR 0 = 1 OR 0 OR 0 = 1
d) F = (1 AND 0) OR 0 OR 0 = 0 OR 0 OR 0 = 0

**2.18 Solution: (5' each)**




**2.24 Solution: (1' for each, total 17')**

a) a, b, c, d
b) a', d', a', c, b', c, d', c, d
c) a'd', a'c, b'cd', cd

**2.28 Solution: (5')**

F = a'bc + a'bd' + ab' + ac + (ab + ad)c
F = a'bc + a'bd' + ab' + ac + abc + acd
F = a'bc + a'bd' + ab' + ac

**2.33 Solution: (5')**

F = (ab' + b) + a'c

**2.39 Solution: (1' each term, 5' in total)**

F = a'b'c' + a'bc' + ab'c' + ab'c + abc'

**2.52:**

a) **Solution: (5' each F and G, 10' in total)**

F=ab+cd and G=(1*((ab)' * (cd)')')'
In canonical sum-of-minterms form, F = a'b'cd + a'bcd + ab'cd + abc'd' + abc'd + abcd' + abcd and G = a'b'c'd' + a'b'c'd + a'b'cd'+ a'bc'd' + a'bc'd + a'bcd' + ab'c'd' + ab'c'd + ab'cd'. F and G are not equivalent (F = G')

b) **Solution: (1' each True term, 7'+9'=16' in total)**

| Inputs | | | | Outputs |
|---|---|---|---|---|
| a | b | c | d | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(a)

| Inputs | | | | Outputs |
|---|---|---|---|---|
| a | b | c | d | F |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

(b)

**2.58 Solution:**

**Step 1 - Capture the function (5' for the table, should be all correct)**

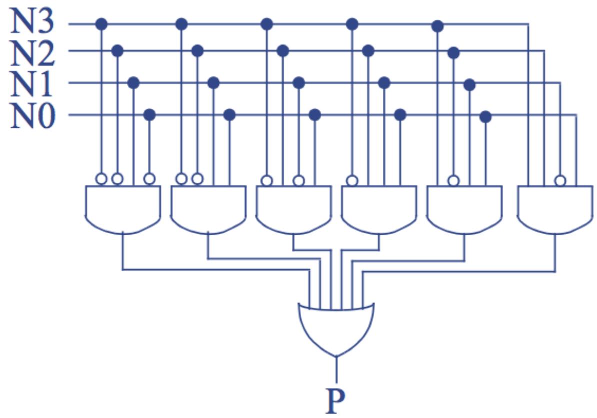
The prime numbers in the range 0-15 are 2, 3, 5, 7, 11, and 13. Rows whose input binary number correspond to those numbers have P set to a 1; the other rows get 0.

| Inputs | | | | Outputs |
|---|---|---|---|---|
| N3 | N2 | N1 | N0 | P |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

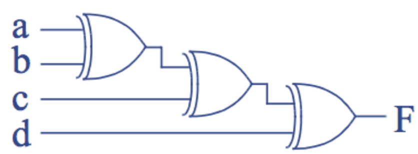
**Step 2A - Create equation (5')**

P = N3'N2'N1N0' + N3'N2'N1N0 + N3'N2N1'N0 + N3'N2N1N0 + N3N2'N1N0 + N3N2N1'N0

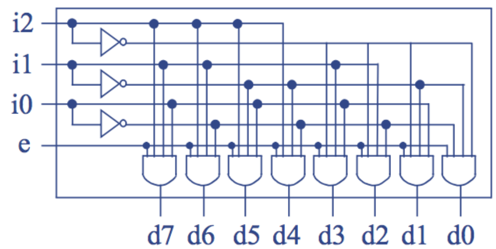
**Step 2B- Implement as a gate-based circuit (5')**



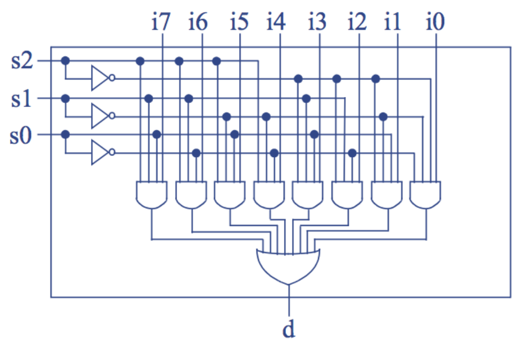

**2.66 Solution: (5')**



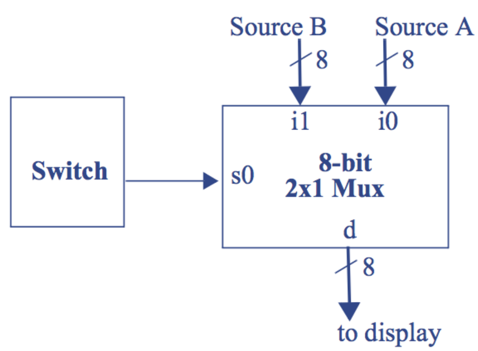
**2.70 Solution: (10', should be all correct, careless mistakes -2')**




**2.71 Solution: (10', should be all correct, careless mistakes -2')**




**2.75 Solution: (10', should be all correct, careless mistakes -2')**



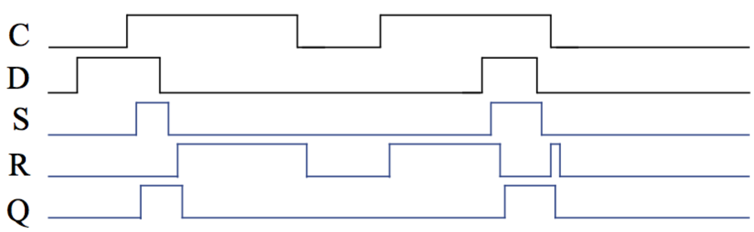
**3.2 Solution: (5' all correct, 0 all wrong, -1' per wrong answer)**

a) 1/32768 = 30.5 us
b) 1/100,000,000 = 10 ns
c) 1/1,500,000,000 = 0.66 ns = 667 ps
d) 1/ 2,400,000,000 = 0.416 ns = 416 ps

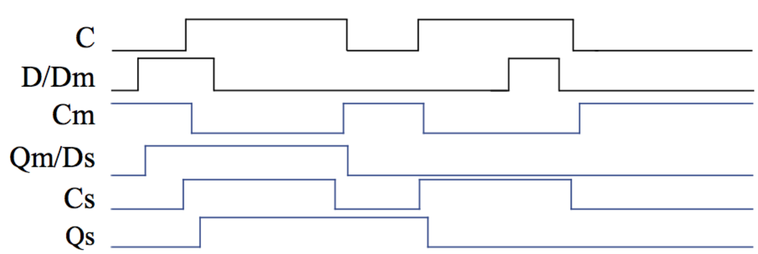
**3.4 Solution: (5' all correct, 0 all wrong, -1' per wrong answer)**

a) 1/500ms = 2 Hz
b) 1/400 ns = 2,500,000 Hz = 2.5 MHz
c) 1/4ns = 250,000,000 Hz = 250 MHz
d) 1/20ps = 50,000,000,000 Hz = 50 GHz

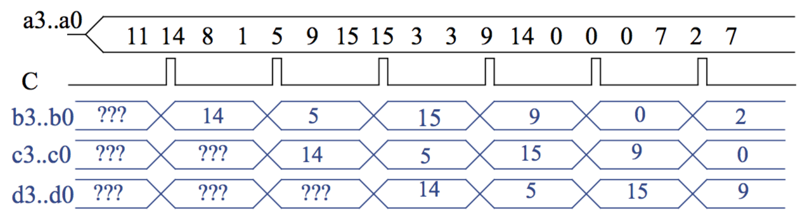
**3.10 Solution: (20' all correct, 0 all wrong, -1' per wrong On/Off (17 On/Offs))**



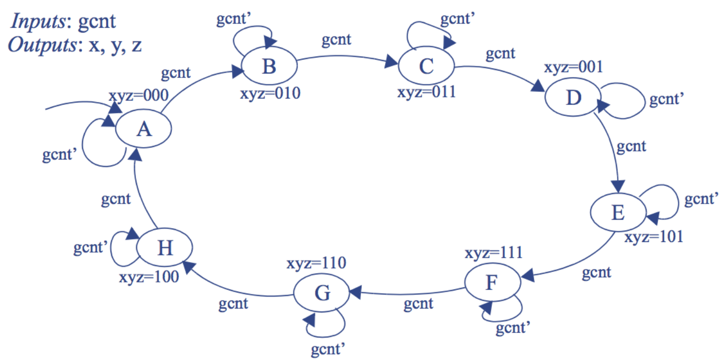

**3.12 Solution: (20' all correct, 0 all wrong, -1' per wrong On/Off (16 On/Offs))**




**3.21 Solution: (10': 6' for 6 values (1' each), 4' for the sequence of b c d)**



**3.30 Solution: (10' all correct, 0' all wrong, -1' per wrong state transitions)**

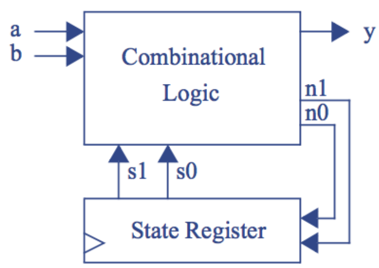



**3.40 Solution:**

**Step 1 - Capture the FSM**

FSM as given by the problem.

**Step 2A - Set up the architecture (5')**




**Step 2B- Encode the states (4')**

A straightforward encoding is A=00, B=01, C=10, D=11.

**Step 2C - Fill in the truth table (16', 1' each entry)**

| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| s1 | s0 | a | b | n1 | n0 | y |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

**Step 2D - Implement the combinational logic (5' all correct, 0 all wrong, -1' per wrong answer)**

n1 = s1's0'a'b' + s1's0a + s1s0'

n0 = s1's0'a'b + s1's0a' + s1s0'b

y = s1's0 + s1s0'

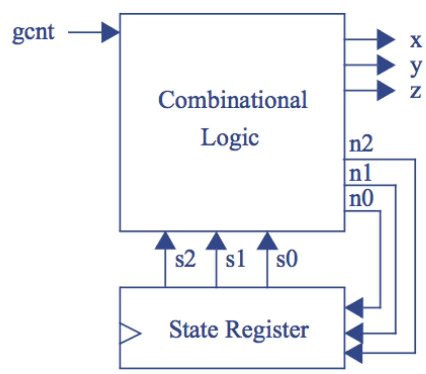
Note: The above equations can be minimized further.

**3.43 Solution:**

**Step 1 - Capture the FSM**

The FSM was created during Exercise 3.30.

**Step 2A - Set up the architecture (5')**




**Step 2B - Encode the states (8')**

A straightforward encoding is A=000, B=001, C=010, D=011, E=100, F=101, G=110, H=111.r

**Step 2C - Fill in the truth table (16', 1 each row)**

| | Inputs | | | | Outputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | s2 | s1 | s0 | gcnt | n2 | n1 | n0 | x | y | z |
| *A* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| *B* | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| *C* | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| *D* | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| *E* | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| *F* | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| *G* | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| *H* | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

**Step 2D - Implement the combinational logic (6', 1' each)**

n2 = s2's1s0gcnt + s2s1' + s2s1s0' + s2s1s0gcnt'

n1 = s2's1's0gcnt + s2's1s0' + s2's1s0gcnt' + s2s1's0gcnt + s2s1s0' + s2s1s0gcnt'

n0 = s2's1's0'gcnt + s2's1's0gcnt' + s2's1s0'gcnt + s2's1s0gcnt' + s2s1's0'gcnt + s2s1's0gcnt' + s2s1s0'gcnt + s2s1s0gcnt'

x = s2

y = s2's1's0 + s2's1s0' + s2s1's0 + s2s1s0'

z = s2's1 + s2s1'