# Computer Organization Test 1

Question 1. (20 points) For the Boolean function F specified in the following truth table (a "d" indicates a "don't care"), use a K-map to write the simplified sum-of-products (SOP) Boolean function.

| A | B | C | D | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | d |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | d |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | d |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | d |
| 1 | 1 | 1 | 1 | d |

b) Implement your above simplified Boolean function for F using AND, OR, and NOT gates. (Alternatively, if you had problems with part (a), you may implement the *unsimplified* Boolean function F. Treat the "don't cares" as "0"s).

c) Ignoring NOT gates, how many gate delays are in your above circuit?

d) What is the complexity (# of gates + # of inputs to those gates) of your circuit?

Question 2. (25 points)

a) Convert $174_{10}$ to a binary (base 2) value.

b) Convert $174_{10}$ to a hexadecimal (base 16) value.

c) Covert $-174_{10}$ to a two's complement value.

d) Perform the following arithmetic operations:

| | | | |
|---|---|---|---|
| $1010011_2$ | $1100110_2$ | $A\ 8\ 7\ D\ 4_{16}$ | $E\ 9\ D\ 3\ A_{16}$ |
| $+\underline{1100110_2}$ | $-\ \underline{0101011_2}$ | $+\ \underline{7\ 3\ A\ 1\ 8_{16}}$ | $-\ \underline{B\ 9\ C\ 4\ B_{16}}$ |

Question 3. (10 points)

a) Convert the value $55.375_{10}$ to its binary representation.

64 32 16 8 4 2 1 .5 .25 .125 .0625

[ ] . [ ]

b) Normalize the above value so that the most significant 1 is immediately to the left of the radix point. Include the corresponding exponent value to indicate the motion of the radix point.

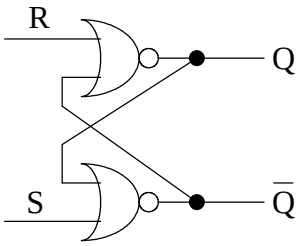
1. [ ] x 2[ ]

c) Write the corresponding 32-bit IEEE 754 floating point representation for $55.375_{10}$.

| Sign bit | 8-bit Exponent (bias 127) | 23-bit Mantissa (for normalized values, leading 1 not stored) |
|---|---|---|
| | | |

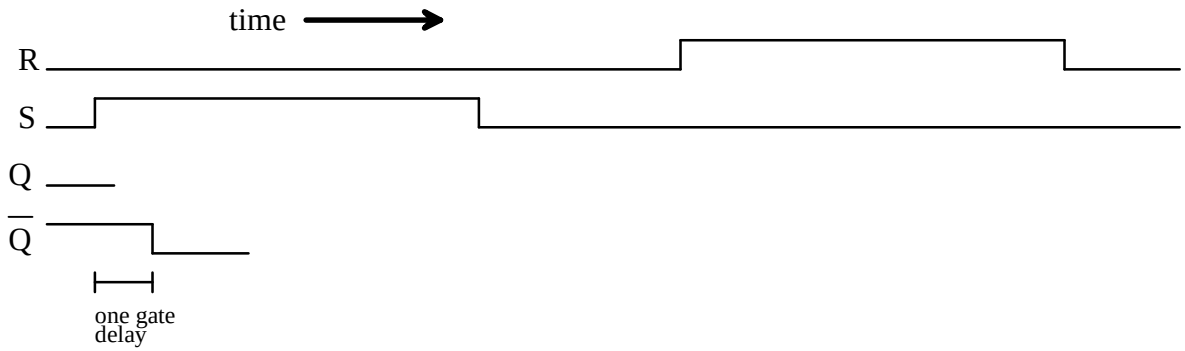
d) Suppose A, B and C are 32-bit IEEE 754 floating point variables with A having a normalized value of $1.11_2 \times 2^{75}$ and B having a normalized value of $1.101_2 \times 2^{30}$. After the assignment statement "C = A+B, " why is C's value equal to A's value?

Question 4. (10 points) a) If R = 0 and S = 1, then what will be the output on Q and $\overline{Q}$?

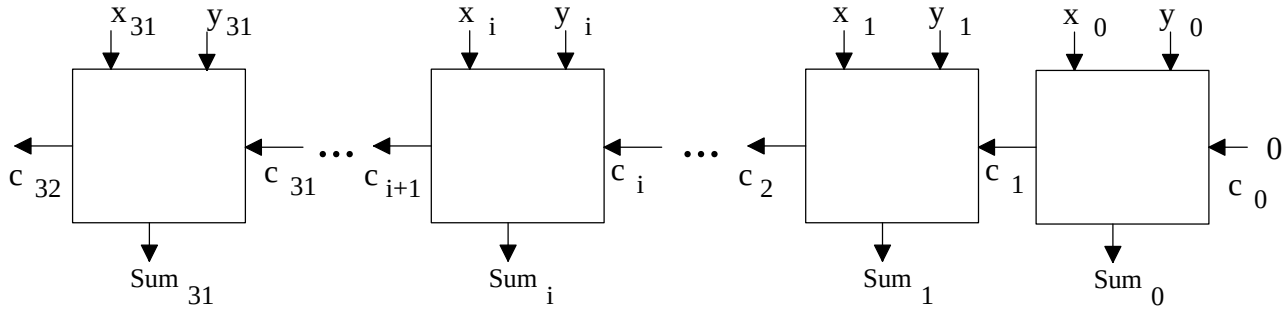



b) Now, if S goes to a 0 value, what happens to the output on Q and $\overline{Q}$?

c) Complete the following timing diagram for the SR latch. Include gate delays in the diagram.




Question 5. (5 points) A 32-bit, ripple-adder is made up of a collection of single-bit Full-Adders connected together as shown below:

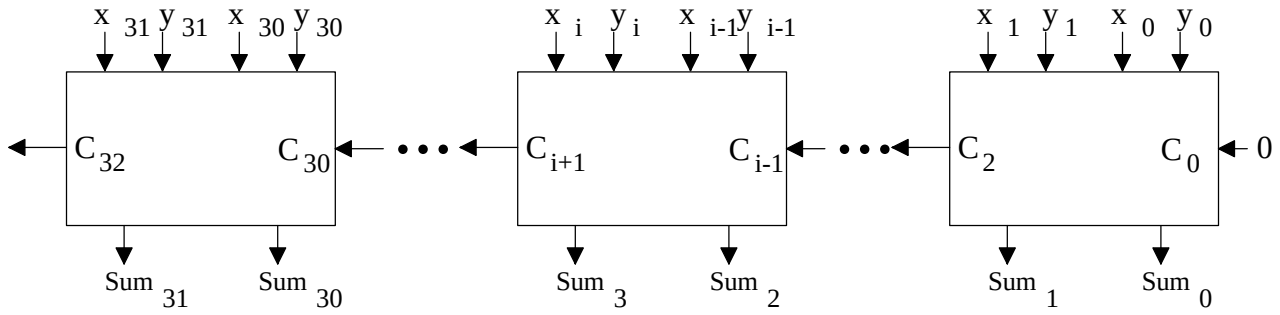



The carry-out ($c_{i+1}$) signal of a Full-Adder is calculated as the minimized, sum-of-products expression, $c_{i+1} = x_i y_i + x_i c_i + y_i c_i$.

a) For a single Full-Adder, how many gate delays are needed before the carry-out signal is correct?

b) For the whole 32-bit ripple-adder, how many gate delays are needed before $c_{32}$ is correct?

Question 6. (10 points) To speed up the calculation of the carry-out ($c_{i+1}$) signals, consider constructing a 32-bit adder using two-bit, carry-lookahead adders as shown in:




a) If $c_{i+1}$ is calculated directly from the inputs as $c_{i+1} = x_i y_i + x_i x_{i-1} y_{i-1} + x_i x_{i-1} c_{i-1} + x_i y_{i-1} c_{i-1} + y_i x_{i-1} y_{i-1} + y_i x_{i-1} c_{i-1} + y_i y_{i-1} c_{i-1}$, then how many gate delays would be needed to calculate the carry-out ($c_{i+1}$) signal in a single two-bit adder?

b) For the whole 32-bit adder using two-bit adders, how many gate delays are needed before $c_{32}$ is correct?

Question 7. (15 points) Suppose we have a register file with the following specifications:

- 32 registers numbered from R0 to R31
- each register has 32-bits
- one write ports
- two read ports

a) How many bits(/"wires") would be need for each of the following?

- specifying the register number for either a read or write port
- data output read from a read port

b) How many decoders would be needed in the implementation of the whole register file?

c) What type of decoders (i.e., number of inputs and number of outputs for each decoder) are needed?

d) How many multiplexers would be needed in the implementation of the whole register file?

e) What type of multiplexers are needed?

Question 8. (5 points) The register-file implementation does not scale well for large memories because the number of gates in the address decoder (and MUXs) grows exponentially with the number of bits in the address. Large memories use a square-array of bits and decode the address in two parts (row number then column number). For example, a 1 M x 4-bit memory would use two 10-to$2^{10}$ decoders instead of one 20-to-$2^{20}$ decoder needed for a register-file implementation. How many gates are saved using two 10-to$2^{10}$ decoders instead of one 20-to-$2^{20}$ decoder?