The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2012

Problem Set #4
Issued Wednesday, 2/29/12; Due Wednesday, 3/14/12

Homework Information: Some of the problems are probably too time consuming to be done the night before the due date, so plan accordingly. Late homework will not be accepted. Feel free to get help from others, but the work you hand in should be your own.

Problem 1. "May Your Carries Overflowith"

In Lecture 9, overflows were defined as the case when either the sum two negative numbers gives a positive result, or the sum of two positive numbers gives negative result. This can be expressed as the following expression:

$$V = \overline{A}_{n-1}\overline{B}_{n-1}S_{n-1} + A_{n-1}B_{n-1}\overline{S}_{n+1}$$

where A_{n-1}, B_{n-1} are the inputs to most significant adder, and S_{n-1} is the most significant bit's sum output.

a) An alternative way of computing overflow considers only the carry inputs and outputs of the most significant bit's adder, and it can be expressed as: $V = \mathrm{XOR}(C_{out}, C_{in})$. Prove these two different implementations give identical results.

b) Overflows can also be generated by subtraction. In particular, when a negative number subtracted from a positive number yields a negative result, or when a positive number is subtracted from a negative one and a positive result is generated. Compare ***both*** of the two given approaches for computing overflows in these cases. Do they work? If so, under what assumptions. If not, explain why.

c) We can also define a notion of overflow when adding unsigned numbers. Basically, an overflow occurs when the correct result cannot be represented given the finite representation. What logical combination of the condition codes Z (zero), N (negative), C (carry), and V (overflow) detect this case?

d) The distinction between the MIPS' `add` and `addu` instructions is that the "unsigned" version of the instruction ignores overflows (i.e. it takes no special actions when an overflowed result is generated). Does this mean that the addu can't be used for signed 2's-complement arithmetic? Explain why or why not.

Problem 2. "Bits of Floating-Point"

Represent the following in single-precision IEEE floating point. Give your answers in hexadecimal:

a) 2007.0
b) -4014.0
c) -0.00390625 (Hint: -2^{-8})
d) 16777215.0 (Hint: $2^{24} -1$)
e) 16777216.0 (Hint: 2^{24})

Convert the following single-precision floating point numbers (given in hexadecimal) to decimal:

```
f) 0x3e800000
g) 0xbec00000
h) 0x46800000
i) 0xc67ffc00
j) 0x46800200
```

Problem 3. "Floating-Point Arithmetic"

Given the following two single-precision IEEE floating-point numbers:

x = 0x3c800000 and y = 0x48800000

Compute the following showing all work (sign, exponent, and mantissa) and give the result as a properly normalized IEEE floating-point number in hexadecimal:

a) $x + y$
b) $x \times y$
c) $x - y$

Problem 4. "Half-Precision"

Certain older graphics cards adopted a new floating format called "half precision". Half-precision numbers are stored in 16-bits (2-bytes), and they use the following format:

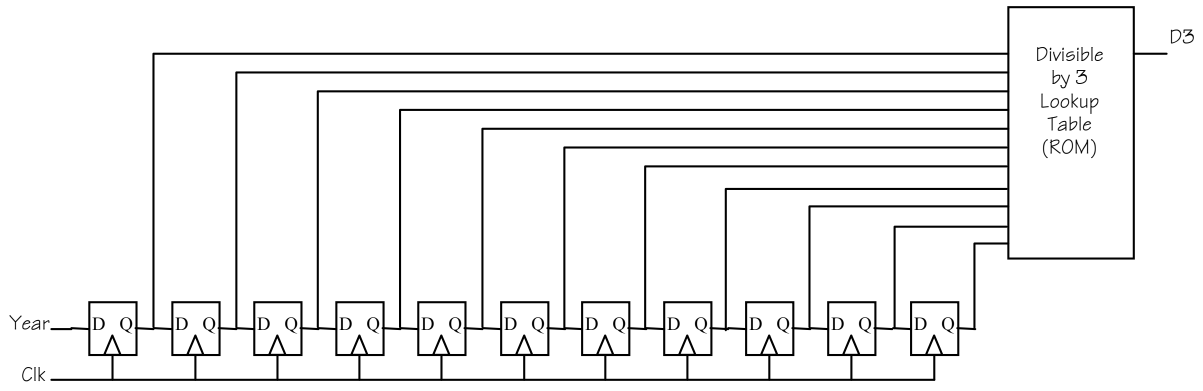
S EEEEE FFFFFFFFFF

Where S is the sign-bit, E indicates bits of the exponent, which are represented in bias-15, and F is a 10-bit fractional component of an 11-bit significand with a hidden "1". The exponent values of 00000_2 and 11111_2 are reserved, and the values 0x0000 and 0x8000 are interpreted as 0.0 and -0.0 respectively.

a) Represent the number -0.03125 in half-precision. Give your answer in hexadecimal.
b) What value does 0x2bad represent?
c) What is the largest positive number representable as a normalized number in half-precision format?
d) What is the smallest number greater than zero that is representable as a normalized number in this format?

Problem 5. The Real "Y2K"

Immediately following an unusually useful Comp 411 lecture, Lee Hart raced back to his part-time job at the upstart Mod-3 Calendar Corporation. Central to all products in the Mod-3's product line is a patented circuit that determines if a given calendar year (e.g. 2010) is divisible by 3. The year is entered into Mod-3's products as an unsigned integer, one bit at a time. In Mod-3's prototype system, this function was accomplished with a shift register and a look-up table using the circuit shown below.

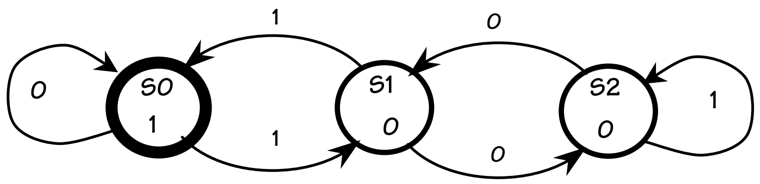


A) Lee has been told that the look-up table has been carefully determined and programmed into a ROM. Based on the circuit shown above, how many bits are in this ROM?

B) If each register has the following timing specs, $t_{pd} = 3$ ns, $t_s = 2$ ns, and $t_h = 1$ ns, and the ROM's specs are $t_{pd} = 11$ ns, what is the minimum value of t_{cd} that allows the circuit to operate properly?

C) What is the fastest rate which Mod-3's patented circuit can be clocked and still operate as intended?

Lee has warned the management at Mod-3 of a *real* looming Y2K problem associated with the current design, and in an effort to humor him they have assigned him the task of improving the circuit's design. Lee recalls a simple state machine specification from Comp 411. The state transition diagram of the circuit is shown below.



He recalls from lecture that this state machine also determines if a given unsigned integer input is divisible by 3, when entered least significant bit first.

D) Verify the operation the behavior of the state transition diagram given in class by determining its final state for each of the following input sequences following a reset {11111010011, 11111010001, 10111010100, 11011110000, 11110101010}

Convinced that the circuit performs as specified, Lee hastily sets out to implement the state machine. He cleverly assigns the following state encodings, S0 = 10, S1 = 00, and S2 = 01. This allows the most significant bit of the state encoding to serve double duty as the divisible by three output, D3.

E) Recreate Lee’s state machine using a ROM and a 2-bit state register. What is the size of the ROM. Write out a table showing the new ROM’s contents.

F) After completing his FSM design, Lee closely examines the behavior of the original circuit only to realize that the original circuit expects the year input to be entered *most significant bit* first. Feeling slightly demoralized, he decides to go ahead and substitute his version on the circuit into a working Mod-3 prototype. Much to his surprise, the prototype system operates just as before. Explain how this is possible.

G) Why is this new “divisible-by-3” circuit immune to the Y2K problem foreseen by Lee? Is it likely to be faster than the original design? Is it likely to be less expensive than the original (explain why)?