| | |
|---|---|
| Cairo University | Computer Arithmetic |
| Electronics and Communications Department | MSc midterm exam |
| Dr. Hossam A. H. Fahmy | November 2010 |

This is an open-book examination. Total time allowed for this exam is **90 min**.

---

1. Indicate which of the following expressions is valid (i.e. left hand side equals right hand side) under all conditions within the IEEE standard for floating point arithmetic. You must justify your answer whether you think the expression is valid or invalid.

(10 marks)

$$
\begin{aligned}
x/2 &= x * 0.5 && (1)\\
1 * x &= x/1 = x && (2)\\
x/x &= 1.0 && (3)\\
x - y &= x + (-y) = (-y) + x && (4)\\
x - y &= -(y - x) && (5)\\
x - x &= 0 && (6)\\
0 * x &= 0 && (7)\\
x + 0 &= x && (8)\\
x - 0 &= x && (9)\\
-x &= 0 - x && (10)
\end{aligned}
$$

**Answer:** Note: these expressions were taken from the C-language standard where they were given as guidelines for compiler writers on what can be optimized without problem and what cannot.

- $x/2 = x * 0.5$

  Although similar transformations involving inexact constants generally do not yield numerically equivalent expressions, if the constants are exact then such transformations can be made on IEEE 754 compliant machines and others that round perfectly.

- $1 * x = x/1 = x$

  The expressions 1 * x, x / 1, and x are equivalent for the same reasons as above.

- $x/x \neq 1.0$

  The expressions $x/x$ and 1.0 are not equivalent if $x$ can be zero, infinite, or NaN.

**Answer:**

- $x - y = x + (-y) = (-y) + x$

  The expressions $x-y$, $x+(-y)$, and $(-y)+x$ are equivalent since the subtraction is treated as a negation followed by addition and those operations are still commutative.

  Note that commutativity is valid for two operands only. If three terms should be summed then commutativity is not always valid for example $1.0 \times 10^{20} - 1.0 \times 10^{20} + 1.0 \times 10^{2} \neq 1.0 \times 10^{20} + 1.0 \times 10^{2} - 1.0 \times 10^{20}$.

- $x - y \neq -(y - x)$

  The expressions $x - y$ and $-(y - x)$ are not equivalent because $1-1$ is $+0$ but $-(1-1)$ is $-0$ (in the default rounding direction).

- $x - x \neq 0$

  The expressions $x - x$ and 0.0 are not equivalent if $x$ is a NaN or infinite.

- $0 * x \neq 0$

  The expressions $0 * x$ and 0.0 are not equivalent if $x$ is a NaN, infinite, or -0.

- $x + 0 \neq x$

  The expressions $x+0$ and $x$ are not equivalent if $x$ is -0, because $(-0) + (+0)$ yields $+0$ (in the default rounding direction), not $-0$.

- $x - 0 \neq x$

  $(+0) - (+0)$ yields $-0$ when rounding is downward (toward $-\infty$), but $+0$ otherwise, and $(-0) - (+0)$ always yields $-0$. Hence, the expression is not always valid under all rounding directions and the compiler should not make this optimization unless the rounding direction is guaranteed not to be towards minus infinity.

- $-x \neq 0 - x$

  The expressions $-x$ and $0 - x$ are not equivalent if $x$ is $+0$, because $-(+0)$ yields $-0$, but $0-(+0)$ yields $+0$ (unless rounding is towards minus infinity).

2. Several authors suggested the use of the residue number system in low power applications. An embedded system has an RNS adder for the calculation of the hours, minutes and seconds.

(a) If any integer modulus is permitted, choose the best set to represent each of the three fields (hours, minutes and seconds). Please state the reasons for your choice.

(3 marks)

**Answer:** The required range to represent the hours is 24 while that for the minutes and seconds is 60. If any integer modulus is permitted then the sets are chosen as:

| | | |
|---|---|---|
| Hours | $8 \times 3$ | $= 24$ |
| Minutes | $5 \times 4 \times 3$ | $= 60$ |
| Seconds | $5 \times 4 \times 3$ | $= 60$ |

These sets represent the required range exactly, use the minimum total number of bits for each range, and minimize the longest carry propagation delay.

(b) Show how the following operation is performed:

| | hours | minutes | seconds |
|---|---|---|---|
| | 13 | 10 | 55 |
| + | 10 | 12 | 04 |

(3 marks)

**Answer:** The operation

| | hours | minutes | seconds |
|---|---|---|---|
| | 13 | 10 | 55 |
| + | 10 | 12 | 04 |
| | 23 | 22 | 59 |

is performed as:

| | hours | minutes | seconds |
|---|---|---|---|
| | $(5, 1)$ | $(0, 2, 1)$ | $(0, 3, 1)$ |
| + | $(2, 1)$ | $(2, 0, 0)$ | $(4, 0, 1)$ |
| | $(7, 2)$ | $(2, 2, 1)$ | $(4, 3, 2)$ |

The results of the RNS operation correspond to $23h\ 22m\ 59s$.

(c) Can you think of a problem that exists due to the use of RNS in such a system? You do not need to solve it, just state it. (Hint: add big numbers.)

(2 marks)

**Answer:** If large numbers are added then an overflow may occur and a carry from the seconds to the minutes or the minutes to the hours might be needed. This overflow is not easily detected in residue number systems.

(d) Why do you think those authors suggested RNS for low power applications?

(2 marks)

**Answer:** In CMOS circuits, any node that switches its value consumes power. The limited carry propagation in RNS reduces the amount of node switching and hence is suitable for low power applications. From another point of view, a system that is inherently faster than another one (RNS faster than binary) can be run at a lower clock frequency to achieve the same results while consuming less power.
Note: Some of you suggested the use of $7 \times 4 = 28$ for the hours. This set uses the minimum total number of bits and minimizes the longest carry propagation delay. However, it does not represent the required range exactly. This complicates the circuits slightly since we must implement a block for modulo M=24. Otherwise, $15 + 10 = 25$ and not 1 as expected in the hours of a clock. I accepted this solution from those who wrote it but I would like to remind you that the exact representation of the range has its own merit.

3. You are given the following arrangement of partial product bits where $\bar{x}_i = 1 - x_i$ and $\bar{a}_i = 1 - a_i$.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
| | | | | $\times$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| | | | | | $a_4\bar{x}_0$ | $a_3x_0$ | $a_2x_0$ | $a_1x_0$ | $a_0x_0$ |
| | | | | $a_4\bar{x}_1$ | $a_3x_1$ | $a_2x_1$ | $a_1x_1$ | $a_0x_1$ | |
| | | | $a_4\bar{x}_2$ | $a_3x_2$ | $a_2x_2$ | $a_1x_2$ | $a_0x_2$ | | |
| | | $a_4\bar{x}_3$ | $a_3x_3$ | $a_2x_3$ | $a_1x_3$ | $a_0x_3$ | | | |
| | $a_4x_4$ | $\bar{a}_3x_4$ | $\bar{a}_2x_4$ | $\bar{a}_1x_4$ | $\bar{a}_0x_4$ | | | | |
| | $\bar{a}_4$ | 0 | 0 | 0 | $a_4$ | | | | |
| 1 | $\bar{x}_4$ | 0 | 0 | 0 | $x_4$ | | | | |
| $p_9$ | $p_8$ | $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

(a) Prove that this arrangement produces the correct product of two five bits two's complement operands.

(4 marks)

**Answer:** Several proofs are possible. Any correct proof gets the full mark. Here is one solution.
Since $\bar{x}_i = 1 - x_i$ then the top left corner of the array is equal to $a_4(1 - x_0) = a_4 - a_4x_0$. Similarly, the one the term on a diagonal below it and to the left is equal to $a_4 - a_4x_1$. We can continue to rewrite all the elements containing the complement of a bit in this manner to get

| | | | | | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $\times$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| | | | | | $a_4 - a_4x_0$ | $a_3x_0$ | $a_2x_0$ | $a_1x_0$ | $a_0x_0$ |
| | | | | $a_4 - a_4x_1$ | $a_3x_1$ | $a_2x_1$ | $a_1x_1$ | $a_0x_1$ | |
| | | | $a_4 - a_4x_2$ | $a_3x_2$ | $a_2x_2$ | $a_1x_2$ | $a_0x_2$ | | |
| | | $a_4 - a_4x_3$ | $a_3x_3$ | $a_2x_3$ | $a_1x_3$ | $a_0x_3$ | | | |
| | $a_4x_4$ | $x_4 - a_3x_4$ | $x_4 - a_2x_4$ | $x_4 - a_1x_4$ | $x_4 - a_0x_4$ | | | | |
| | $1 - a_4$ | 0 | 0 | 0 | $a_4$ | | | | |
| 1 | $1 - x_4$ | 0 | 0 | 0 | $x_4$ | | | | |
| $p_9$ | $p_8$ | $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

which is then rearranged as

| | | | | | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $\times$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| | | | | | $-a_4x_0$ | $a_3x_0$ | $a_2x_0$ | $a_1x_0$ | $a_0x_0$ |
| | | | | $-a_4x_1$ | $a_3x_1$ | $a_2x_1$ | $a_1x_1$ | $a_0x_1$ | |
| | | | $-a_4x_2$ | $a_3x_2$ | $a_2x_2$ | $a_1x_2$ | $a_0x_2$ | | |
| | | $-a_4x_3$ | $a_3x_3$ | $a_2x_3$ | $a_1x_3$ | $a_0x_3$ | | | |
| | $a_4x_4$ | $-a_3x_4$ | $-a_2x_4$ | $-a_1x_4$ | $-a_0x_4$ | | | | |
| | | $a_4$ | $a_4$ | $a_4$ | $a_4$ | | | | |
| | $1 - a_4$ | 0 | 0 | 0 | $a_4$ | | | | |
| | | $x_4$ | $x_4$ | $x_4$ | $x_4$ | | | | |
| 1 | $1 - x_4$ | 0 | 0 | 0 | $x_4$ | | | | |
| $p_9$ | $p_8$ | $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

If $x_4$ is equal to either zero or one the last two rows reduce to 1 1 0 0 $\cdots$. Similarly, the previous two rows always reduce to 0 1 0 0 $\cdots$. However, the position of those resulting ones is such that they add up to overflow as a carry that is neglected and produce a zero in the range of bits representing the product. Hence, the array is equal to:

| | | | | | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $\times$ | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| | | | | | $(-a_4)x_0$ | $a_3x_0$ | $a_2x_0$ | $a_1x_0$ | $a_0x_0$ |
| | | | | $(-a_4)x_1$ | $a_3x_1$ | $a_2x_1$ | $a_1x_1$ | $a_0x_1$ | |
| | | | $(-a_4)x_2$ | $a_3x_2$ | $a_2x_2$ | $a_1x_2$ | $a_0x_2$ | | |
| | | $(-a_4)x_3$ | $a_3x_3$ | $a_2x_3$ | $a_1x_3$ | $a_0x_3$ | | | |
| | $(-a_4)(-x_4)$ | $a_3(-x_4)$ | $a_2(-x_4)$ | $a_1(-x_4)$ | $a_0(-x_4)$ | | | | |
| $p_9$ | $p_8$ | $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

where we used the fact that $a_4x_4 = (-a_4)(-x_4)$.
Since in the two's complement format the value of the most significant bit is negatively valued, this resulting array effectively produces the product of two five bits operands.

(b) If the operands were $n$ bits long each, how many rows and columns does such a multiplier have?

(1 mark)

**Answer:** It has $n + 2$ rows and $2n$ columns.

(c) What are the formulas for obtaining the partial product bit at row $i$ and column $j$?

(4 marks)

**Answer:** The values of $i$ go from 0 to $n + 1$ while the values of $j$ go from 0 to $2n - 1$. The bit $pp_{i,j}$ at row $i$ and column $j$ is given by

| Range of $i$ | Range of $j$ | $pp_{i,j}$ |
|---|---|---|
| $i < n - 1$ | $0 \leq j < i$ | 0 |
| | $i \leq j < i + n - 1$ | $a_{j-i}x_i$ |
| | $j = i + n - 1$ | $a_{j-i}\bar{x}_i$ |
| | $i + n - 1 < j$ | 0 |
| $i = n - 1$ | $0 \leq j < i$ | 0 |
| | $i \leq j < i + n - 1$ | $\bar{a}_{j-i}x_i$ |
| | $j = i + n - 1$ | $a_{j-i}x_i$ |
| | $i + n - 1 < j$ | 0 |
| $i = n$ | $j = n - 1$ | $a_{n-1}$ |
| | $j = 2n - 2$ | $\bar{a}_{n-1}$ |
| | all other $j$ | 0 |
| $i = n + 1$ | $j = n - 1$ | $x_{n-1}$ |
| | $j = 2n - 2$ | $\bar{x}_{n-1}$ |
| | $j = 2n - 1$ | 1 |
| | all other $j$ | 0 |

(d) Briefly compare this multiplier from the area and speed points of view to the parallel multipliers that you studied.

(1 mark)

**Answer:** This multiplier requires more area since it has more rows than the standard multipliers. The additional rows also increase the time delay of this multiplier making it probably slower than the standard multipliers.