**Chapter 5 Exercise Questions**

Name: **Victor Marrujo**

**1. What is the delay for the following types of 64-bit adders? Assume that each two-input gate delay is 150 ps and that a full adder delay is 450 ps. Show your work.**

**a. Ripple-Carry Adder (2 pts.):**

Each stage depends on the carry output from the previous stage. Therefore, the delay is the sum of the delays of all the full adders in series.

* **Delay per full adder =** 450 ps
* **Number of full adders =** 64
* ***Delay* ripple-carry​=** Delay per full adder × Number of full adders
* ***Delay* ripple-carry​=** 450ps×64
* ***Delay* ripple-carry​=** 28,800ps=28.8*μ*s

b. **Carry-Lookahead Adder with 4-bit Blocks** (4 pts.)**:**

The delay is reduced by generating carry signals for groups of bits in parallel. The propagation delay through each block is independent of the number of bits.

* **Number of blocks** = 64 / 4 = 16
* **Delay per 4-bit block =** 150 ps (two-input gate delay)
* ***Delay* carry-lookahead**​= Delay per 4-bit block × Number of blocks
* ***Delay* carry-lookahead**​=150ps×16
* ***Delay* carry-lookahead**​=2400ps=2.4*μ*s

c. **Prefix Adder** (4 pts.)**:**

A prefix adder further reduces the delay by computing carries in parallel across multiple levels.

* ***Delay* prefix**​=log2​(Number of bits)×Delay per two-input gate
* ***Delay* prefix**​=log2​(64)×150ps
* ***Delay* prefix**​=6×150ps
* ***Delay* prefix**​=900ps=0.9*μ*s

**2. Explain why a designer might choose to use a ripple-carry adder instead of a carry-lookahead adder. Cite your source(s) (3 pts.).**

A designer may opt for a ripple-carry adder over a carry-lookahead adder due to its simplicity (modularity and regularity) and reduced use of logic gates, resulting in cost-effectiveness and enhanced comprehensibility. This uncomplicated design renders the ripple-carry adder particularly well-suited for scenarios involving small bit-width binary addition.

Nonetheless, it is crucial to acknowledge that the ripple-carry adder exhibits slower operation compared to a carry-lookahead adder, especially when N is large due to bits waiting on the calculation of the carry bit from the preceding bit addition before determining its own result.

Although the slower operation may not be a critical factor in certain contexts, it is essential to consider the trade-off between simplicity and speed in the design decision.

Source:

Harris, David Money, and Sarah L. Harris. “CHAPTER FIVE Digital Building Blocks.” *Digital Design and Computer Architecture*, Morgan Kaufmann, Waltham, MA, 2013, pp. 239–243.

**3. Express the base 10 number -17.15625 in 16-bit fixed-point sign/magnitude format with eight integer bits and eight fraction bits. Express your answer in binary. Show your work (3 pts.).**

1. **Since the given number is negative:**

The sign bit is 1.

1. **Converting the integer part of 17 to binary :**

17/2 = 8.5 (bit 1)

8/2 = 4.0 (bit 0)

4/2 = 2.0 (bit 0)

2/2 = 1.0 (bit 0)

1/2 = 0.5 (bit 1)

So far, we have: 1 000100012

1. **Converting the fractional part 0.15625, repeatedly multiply by 2 and take the integer part of the result at each step:**

0.15625 \* 2 = 0.3125 (bit 0)

0.3125 \* 2 = 0.625 (bit 0)

0.625 \* 2 = 1.25 (bit 1)

0.25 \* 2 = 0.5 (bit 0)

0.5 \* 2 = 1.0 (bit 1)

Making the result .001010002

1. **Combine the sign bit, integer part, and fractional part:**

−17.1562510 ​= 1 00010001.001010002

4. Express the base 10 number -8.078125 in 12-bit fixed-point sign/magnitude format with six integer bits and six fraction bits. Express your answer in binary. Show your work (3 pts.).

1. **Since the given number is negative:**

The sign bit is 1.

1. **Converting the integer part of 8 to binary :**

8/2 = 4.0 (bit 0)

42 = 2 0 (bit 0)

2/2 = 1.0 (bit 0)

1/2 = 0.5 (bit 1)

So far, we have: 1 0010002

1. **Converting the fractional part 078125, repeatedly multiply by 2 and take the integer part of the result at each step:**

0.078125 \* 2 = 0.15625 (bit 0)

0.15625 \* 2 = 0.3125 (bit 0)

0.3125 \* 2 = 0.625 (bit 0)

0.625 \* 2 = 1.25 (bit 1)

0.25 \* 2 = 0.5 (bit 0)

0.5 \* 2 = 1.0 (bit 1)

Making the result .0001012

1. **Combine the sign bit, integer part, and fractional part:**

−8.078125 ​= 1 001000.0001012

**5. Express the base 10 number -30.5 in IEEE 754 single-precision floating-point format. Express your answer in binary. Show your work for each step (6 pts.).**

**Step 1: Convert decimal to binary.**

1. **Integer part:**

30/2= 15 (bit 0)

15/2 = 7.5 (bit 1)

7/2 = 3.5 (bit 1)

3/2 = 1.5 (bit 1)

½ = 0.5 (bit 1)

So far, we have: 1 111102

1. **Fractional part:**

0.5 \* 2 = 1.0 (bit 1)

Making the result .12

**Step 2: Write in scientific notation.**

1 11110.12 = −1.111012​×24

**Step 3: Fill in fields:**

1. **Sign bit:**

1 (Because negative)

1. **8 biased exponent bits:**

The exponent is 410+127bias=131biased​, and in binary, it is 100000112​.

Biased Exponent: 100000112​

1. **23 fraction bits:**

Take the fraction and add zeros to make it 23 bits long.

Fraction: 11101000000000000000000 2

**Step 4: Combine:**

1 10000011 111010000000000000000002

**6. Express the base 10 number 16.25 in IEEE 754 single-precision floating-point format. Express your answer in binary. Show your work for each step (6 pts.).**

**Step 1: Convert decimal to binary.**

1. **Integer part:**

16/2 = 8.0 (bit 0)

8/2 = 4.0 (bit 0)

4/2 = 2.0 (bit 0)

2/2 = 1.0 (bit 0)

1/2 = 0.5 (bit 1)

So far, we have 100002

1. **Fractional part:**

0.25 \* 2 = 0.5 (bit 0)

0.5 \* 2 = 1.0 (bit 1)

Making the result .012

**Step 2: Write in scientific notation.**

10000.012 = −1.0000012​×24

**Step 3: Fill in fields:**

1. **Sign bit:**

0 (Because positive)

1. **8 biased exponent bits:**

The exponent is 410+127bias=131biased​, and in binary, it is 100000112​.

Biased Exponent: 100000112​

1. **23 fraction bits:**

Take the fraction and add zeros to make it 23 bits long.

Fraction: 00000100000000000000000 2

**Step 4: Combine:**

0 10000011 00000100000000000000000 2

**7. Consider IEEE 754 single-precision floating-point numbers. Explain why positive infinity, negative infinity, and NaN are given special representations. Cite your source(s) (3 pts.).**

Infinity is indicated in IEEE floating-point representation by assigning all ones to the exponent and all zeros to the mantissa. The sign bit is utilized to distinguish between positive infinity and negative infinity. The ability to explicitly represent infinity proves beneficial as it facilitates the continuation of operations beyond overflow scenarios. IEEE standards ensure well-defined operations involving infinite values.

Indeterminate values are characterized by an exponent consisting of all ones, a mantissa with a leading one followed by zeros, and a sign bit set to one. This representation is employed for outcomes that are indeterminable, such as calculations involving expressions like (infinity - infinity) or (0 x infinity).

The value NaN (Not a Number) signifies an error and is represented with an exponent field of all ones and a zero sign bit or a mantissa that is not 1 followed by zeros. NaN serves as a special marker, often used to denote a variable that currently lacks a valid value.

Source:

Hollasch, Steve. “IEEE Standard 754 Floating Point Numbers.” IEEE Standard 754 Floating-Point, 8 Jan. 1997, courses.cs.washington.edu/courses/cse401/01au/details/fp.html.

**8. Flash EEPROM, simply called Flash memory, is a fairly recent invention that has revolutionized consumer electronics. Research and explain how Flash memory works. Describe how a bit in the memory is programmed. Cite your source(s) (6 pts.).**

Flash memory, classified as EEPROM (Electronically Erasable Programmable Read Only Memory), features a grid structure with intersecting columns and rows. Each cell at these intersections contains two transistors, namely the floating gate and the control gate, separated by a thin oxide layer. The floating gate's connection to the row, or wordline, is solely through the control gate, establishing a value of 1 as long as this link persists. Changing this value to 0 involves a unique process known as Fowler-Nordheim tunneling.

Flash memory operates by utilizing floating-gate transistors to store and retrieve data. During programming, a high voltage captures electrons in the floating gate to store information, and to erase data, the charge is released. Reading the stored data involves checking the charge on the floating gate.

Examining the components and processes of flash memory reveals its architecture, comprising a memory array with stacked flash cells. Each basic flash memory cell incorporates a storage transistor, a control gate, and a floating gate, insulated from the transistor by a thin dielectric material. The floating gate retains electrical charge and regulates the flow of electrical current.

Programming involves the addition or removal of electrons from the floating gate, altering the threshold voltage of the storage transistor. This change in voltage determines whether the cell is programmed as a zero or a one.

Despite its advantages, flash memory has drawbacks, including a wear-out mechanism and cell-to-cell interference as dies decrease in size. Excessive program/erase cycles can lead to the breakdown of the oxide layer, causing bits to fail and distorting the threshold value. Additionally, electrons escaping and becoming trapped in the oxide insulation layer may result in errors and bit rot.

Sources:

Tyson, Jeff. “How Flash Memory Works.” *HowStuffWorks*, HowStuffWorks, 30 Aug. 2000, computer.howstuffworks.com/flash-memory.htm.

Yasar, Kinza. “What Is Flash Memory?: Definition from TechTarget.” *TechTarget*, 28 June 2023, www.techtarget.com/searchstorage/definition/flash-memory.