EECS 151/251A Homework 11

Due Friday, May 3rd, 2024

Introduction

This homework is meant to test your understanding adders, multipliers and shifters. This is the last homework! Rejoice :-)!
Problem 1: CLA Latency

Given the gate delays presented below, trace out and calculate the delay on the critical path for the 8-bit CLA presented in slide 22 from the week 14 Adder lecture.

<table>
<thead>
<tr>
<th>GATE</th>
<th>Delay(ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>25</td>
</tr>
<tr>
<td>AND</td>
<td>19</td>
</tr>
<tr>
<td>XOR</td>
<td>37</td>
</tr>
</tbody>
</table>

Problem 2: CSA Latency

Suppose you want to design a fast 64-bit Wallace Tree multiplier using CSAs for the partial product reduction.

1. How many CSAs are needed?

2. Suppose the delay for a CSA and a CPA, are 1ns and 4ns respectively. What is the minimal delay for product reduction to final result (summing together the partial products)?
Problem 3: Log-Shifter

Design and draw a 4-bit arithmetic log-shifter capable of performing both logical left shifts and right shifts. In addition to the shift-amount input (\textit{shamnt}), there is an input named \textit{dir} which is set to 1 for right shift and 0 for left shift.

Problem 4: Practice Base 2 Multiplication

Compute the products for the following base 10 numbers as 4-bit 2’s-complement binary numbers: (a) \(-7 \times 3\), (b) \(7 \times -3\), and (c) \(-7 \times -3\). Show your work for each step in the boxes below.
Problem 5: Ripple Borrow Subtractor

In lecture, we discussed the design for a ripple carry adder. Your task is to design a ripple borrow subtractor which has the same structure of the adder except it performs subtraction.

(a) Draw a single stage at the logic-gate level.

(b) Draw a diagram for a ripple borrow subtractor for 4-bit 2’s complement inputs at the stage level.
**Problem 6: Cross-bar Switch**

Based on the cross-bar switch presented in lecture:

For the scenarios below, provide the hexadecimal value to represent the decoder input for a 4-bit inputs. For example, given an implementation for a 16-bit input the first 4-bits of the hex value would control the decoder of the output’s LSB, and the next consecutive 4-bits control the decoder for the output’s second to last bit.

1. Bit-Reversal (ex. $a_0a_1a_2a_3 \rightarrow a_3a_2a_1a_0$)
2. Pairwise Reversal (ex. $a_0a_1a_2a_3 \rightarrow a_1a_0a_3a_2$)
3. Random Permutation: $a_0a_1a_2a_3 \rightarrow a_1a_3a_2a_0$

**Problem 7: 64-bit Adder**

An application you are designing hardware for requires a 64-bit adder. You could: (1) design and build a 64-bit adder, or (2) construct a 64-bit using very fast 32-bit adders you’ve already designed and built. You decide to save time and construct a 64-bit adder using 32-bit adders.

Your 32-bit adder has the following specifications:

- **Inputs:** 1-bit $carry_{in}$, two 32-bit operands
- **Outputs:** 1-bit $carry_{out}$, a 33-bit sum
- **Latency:** 64ps from $carry_{in}$ to $carry_{out}$

Perform an analysis for two design points: (1) carry-select architecture, and (2) carry-lookahead architecture. Design each architecture and compare the total delay and HW cost (number of each adder, gates, multiplexors, etc) between the architectures.
Problem 8: Baugh-Wooley Annotation

We would like to prove that the Baugh-Wooley approach to signed multiplication is correct. Annotate the corresponding circuit on slide 22 from the “Multipliers & Shifters” lecture for \( X = -3, Y = -5 \), by writing down the inputs and outputs for each FA and HA blocks. Verify that the results match what you expect as a result \( (Z = 15) \).

A labelled diagram of the multiplier is provided below. The naming convention is \( ELEMENT_{[ROW])[INDEX]} \) (\( ELEMENT \) can be HA, FA, NAND or AND, \( ROW \) is zero-indexed, and \( INDEX \) is zero-indexed and is agnostic to the \( ELEMENT \)). Fill in the tables below for the output of each element. **Only the values in the table will be graded.** However, labelling may be instructive and easier to visualize.

<table>
<thead>
<tr>
<th>NAND_03</th>
<th>AND_02</th>
<th>AND_01</th>
<th>AND_00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAND_13</th>
<th>AND_12</th>
<th>AND_11</th>
<th>AND_10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAND_23</th>
<th>AND_22</th>
<th>AND_21</th>
<th>AND_20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND_33</th>
<th>NAND_32</th>
<th>NAND_31</th>
<th>NAND_30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 9: Constant Multiplication

Using the CSD representation, design a multiplier for \( C \cdot X \), where \( C \) is a constant and \( X \) is a signed 4-bit input to the circuit. For your circuit, let \( C = 29 \). Show your circuit to the FA level of detail.

Problem 10: CSA + CPA Annotation

Using CSAs and a final CPA step, design and draw a circuit to calculate the following: \( 1 + 2 + 5 + 2 + 4 + 1 \). You must annotate your circuit showing all intermediate values.

Problem 11: Parallel Prefix Adder

Draw out the circuit diagram for a 7-bit parallel prefix adder. Drawn in the style of the example on slide 26 from the Adder lecture.