SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Aufgabenblatt 5

Sie können Ihre Lösungen bis **Mittwoch, dem 26.05.2021, um 10:00 Uhr** im CMS abgeben.
Geben Sie auf Ihrer Lösung Ihr Tutorium sowie die Namen und Matrikelnummern aller Gruppenmitglieder an.

Aufgabe 5.1: Verbesserter Volladdierer

Im Folgenden betrachten wir eine Menge von Gattern AND, OR, XOR, NAND, NOR, NOT, sowie einen 1-bit Multiplexer, jeweils mit Kosten 1. In der Vorlesung haben wir einen Volladdierer kennen gelernt, der Kosten 5 hat, da er 5 Gatter verwendet. Finden Sie eine Variante, die nur Kosten 3 hat.

Aufgabe 5.2: Carry-Lookahead Addierer

In der Vorlesung haben wir den Carry-Lookahead Addierer kennen gelernt. Er hat zwei Arten von Überträgen unterschieden: p und g. Zur Berechnung dieser Überträge verwenden wir eine parallele Präfixberechnung der folgenden Operation

$$(g_1, p_1) \circ (g_2, p_2) = (g_1 \vee p_1 \wedge g_2, p_1 \wedge p_2)$$

Die Operation muss assoziativ sein, um sie in unserer schnellen Implementierung des Parallel-Prefix verwenden zu dürfen. Zeigen Sie, dass die obige Operation assoziativ ist. Geben Sie bei jeder Umformung die verwendete Regel an. Geben Sie *alle* Zwischenschritte an und verwenden Sie nur eine Regel pro Schritt.

Aufgabe 5.3: Multiplikation von Zahlen in Zweier-Komplement-Darstellung

In der Vorlesung haben wir einen Multiplikationsschaltkreis für vorzeichenlose Binärzahlen kennen gelernt. In dieser Aufgabe geht es um die Multiplikation zweier vorzeichenbehafteter Binärzahlen in Zweier-Komplement-Darstellung. Gegeben zwei Zahlen in Zweier-Komplement-Darstellung $[a] = [a_{n-1}a_{n-2}\dots a_0]$, $[b] = [b_{n-1}b_{n-2}\dots b_0]$.

Zuerst möchten wir $[a] \cdot [b]$ als Summe von Partialprodukten darstellen. Es ergibt sich folgendes:

$$[a] \cdot [b] = -2^{n-1} b_{n-1} [a] + \sum_{i=0}^{n-2} [a] b_i 2^i$$

Wir haben ein negatives Partialprodukt. Wir wissen vom letzten Übungsblatt: $-[a] = [\overline{a}] + 1$.

$$[a] \cdot [b] = ([\overline{a}] + 1)\, b_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} [a] b_i 2^i$$

(a) Wir können nun die Multiplikationsmatrix aufbauen, die aus n Reihen mit jeweils $2n$ Stellen besteht. Die Multiplikation mit 2^i wird durch Verschieben um i Stellen und Auffüllen mit 0 realisiert. Einige Summanden, z.B. $[a]b_0$, müssen auf $2n$ Zeichen aufgefüllt werden ohne den repräsentierten Wert zu verändern. Wie müssen die Reihen aufgefüllt werden? Denken Sie hierbei an die Rechenregeln des letzten Blattes. Bedenken Sie ebenfalls, dass das Inkrement einer n-bit Zahl $n+1$ Bits benötigt. Geben Sie schließlich die Multiplikationsmatrix analog zur Vorlesung explizit an.

(b) Wir müssen nun n Partialprodukte effizient aufsummieren. Wie gehen Sie vor?

(c) Testen Sie Ihren Multiplikationsalgorithmus mit den beiden 6-Bit Zahlen $[a] = -10$ und $[b] = 18$. Geben Sie explizit Ihre Multiplikationsmatrix an.

Aufgabe 5.4: Arithmetic Logic Unit

(a) Geben Sie eine rekursive Beschreibung einer Schaltung an, die eine n-bit Sequenz auf Gleichheit mit 0^n überprüft. Geben Sie rekursive, sowie nicht-rekursive Formeln für die Kosten und Tiefe Ihrer Schaltung an. Zeigen Sie mithilfe von Induktion, dass Ihre nicht-rekursiven Formeln korrekt sind (bzgl. der rekursiven Formeln).

(b) Erweitern Sie die ALU aus der Vorlesung (Foliensatz 8, Folie 24) um Funktionen für die Vergleichsoperationen $>, \geq, =, <, \leq, \neq$ auf Zahlen in 2-Komplementdarstellung. Die ALU soll 0^n1 liefern wenn der Vergleich wahr ist, andernfalls 0^{n+1}. Überlegen Sie sich zunächst, wie viele `select`-Bits Sie benötigen. Geben Sie eine Tabelle analog zu Folie 21 an, die die Funktionsauswahl beschreibt. Wie erweitern Sie den ALU-Schaltkreis? Verwenden Sie möglichst bereits vorhandene Elemente sowie den 0-Tester aus (a) und weitere Multiplexer.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Assignment 5

You may submit your solutions via the CMS until **10:00 a.m. on Wednesday, May 26, 2021**.
Please state on your solutions your tutorial, and the names and matriculation numbers of all team members.

Problem 5.1: Improved Full Adder

In the following, we will consider a set of gates AND, OR, XOR, NAND, NOR, NOT, as well as a 1-bit multiplexer, each with cost 1. In the lecture, we developed a full adder that has cost 5 because it uses 5 gates. Find a variant that has a cost of only 3.

Problem 5.2: Carry-Lookahead Adder

In the lecture, we discussed the carry lookahead adder. It distinguishes two types of carries: p and g. To compute these carries, we use a parallel prefix computation of the following operation

$$(g_1, p_1) \circ (g_2, p_2) = (g_1 \vee p_1 \wedge g_2, p_1 \wedge p_2)$$

Our fast implementation of the parallel prefix requires the operation to be associative. Show that the above operation is indeed associative. Specify the rule used in each step. Show *all* intermediate steps and use only one rule per step.

Problem 5.3: Multiplication of Numbers in Two's Complement Representation

In the lecture, we developed a circuit for the multiplication of unsigned binary numbers. In this assignment, we consider the multiplication of two signed binary numbers in two's complement representation. Let $[a] = [a_{n-1}a_{n-2}\dots a_0]$ and $[b] = [b_{n-1}b_{n-2}\dots b_0]$ be two numbers in two's complement representation.

First, we want to represent $[a] \cdot [b]$ as a sum of partial products:

$$[a] \cdot [b] = -2^{n-1} b_{n-1} [a] + \sum_{i=0}^{n-2} [a] b_i 2^i$$

We have a negative partial product. From the previous assignment sheet we know that $-[a] = [\overline{a}] + 1$.

$$[a] \cdot [b] = ([\overline{a}] + 1)\, b_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} [a] b_i 2^i$$

(a) We can now construct the multiplication matrix, which consists of n rows with $2n$ bits each. Multiplication by 2^i is realized by shifting by i bits and padding with 0. Some summands, e.g., $[a]b_0$, must be padded to $2n$ bits without changing the represented value. How have the rows to be padded? Recall the calculation rules of the previous assignment sheet. Also keep in mind that the increment of an n-bit number requires $n + 1$ bits. Finally, state the multiplication matrix explicitly, analogously to the lecture.

(b) We now have to sum up n partial products efficiently. How do you proceed?

(c) Test your multiplication algorithm with the two 6-bit numbers $[a] = -10$ and $[b] = 18$. State your multiplication matrix explicitly.

Problem 5.4: Arithmetic Logic Unit

(a) Develop a recursive circuit that checks an n-bit sequence for equality with 0^n. Give recursive, as well as non-recursive formulas for the cost and depth of your circuit. Show that your non-recursive formulas are correct (with respect to the recursive formulas) using induction.

(b) Extend the ALU from the lecture (slide deck 5, slide 24) to include functions for the comparison operations $>, \geq, =, <, \leq, \neq$ on numbers in two's complement representation. The ALU shall output 0^n1 if the comparison is true, and 0^{n+1} otherwise. First, think about how many `select` bits are required. Give a table analogous to the table on slide 21 that describes the functions. How can you extend the ALU circuit? If possible, use the existing elements, as well as the 0 tester from part (a) and additional multiplexers.