FH Studiengang SE **WS2025/26**
EIR1

Übung 6 Arithmetic-Logic-Unit (ALU)

Unter einer ALU versteht man jenen Teil eines Prozessors (CPU-Central Processing Unit), der die eigentlichen arithmetischen Operationen (Addition, Subtraktion, ...) und logischen Operationen (UND, ODER, ...) ausführt. Bild 1 zeigt eine ALU, die aus zwei Operanden *A* und *B* ein Ergebnis *F* berechnet (im Folgenden werden mit a_i, b_i und f_i einzelne Stellen von *A*, *B* und *F* bezeichnet). Die Auswahl der tatsächlich gewünschten Berechnung wird über die Steuerleitungen *S* bewerkstelligt.

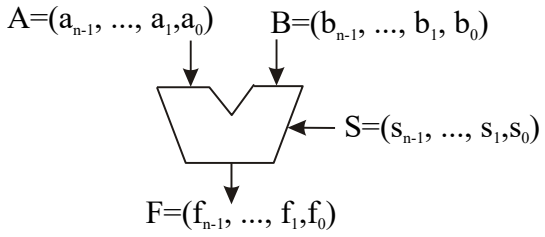

Bild 1 ALU Schaltsymbol.

Der Funktionsumfang einer minimalen ALU wird durch folgende Tabelle beschrieben:

Operation	**Beschreibung**	s_2	s_1	s_0
Addition	$F = A + B$	0	0	0
Subtraktion	$F = A - B$	0	0	1
Inkrement	$F = A + 1$	0	1	0
Dekrement	$F = A - 1$	0	1	1
UND	$f_i = a_i$ AND b_i	1	0	0
ODER	$f_i = a_i$ OR b_i	1	0	1
NOT	$f_i = a_i$‘	1	1	0
XOR	$f_i = a_i$ XOR b_i	1	1	1

Tabelle 1 ALU Funktionstabelle.

Die ALU soll leicht für unterschiedliche Operandenlängen (unterschiedliche *n*) realisiert werden können. Dies erreicht man, indem die Operationen nach Tabelle 1 für eine einzige Stelle in einer Komponente ALU1 realisiert werden. Durch mehrfache Verwendung dieser ALU1 lässt sich jede beliebige Operandenlänge realisieren. Man bezeichnet ALU1 auch als 1-Bit-Slice. Bild 2 zeigt (a) die Komponente ALU1 und (b) deren Verwendung zur Realisierung einer ALU mit Operandenlänge *n*.

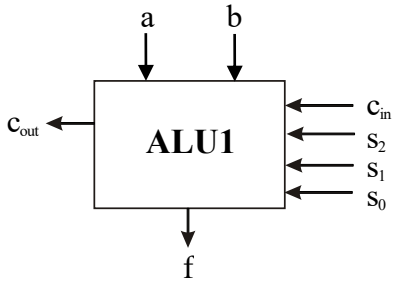

Bild 2a ALU1

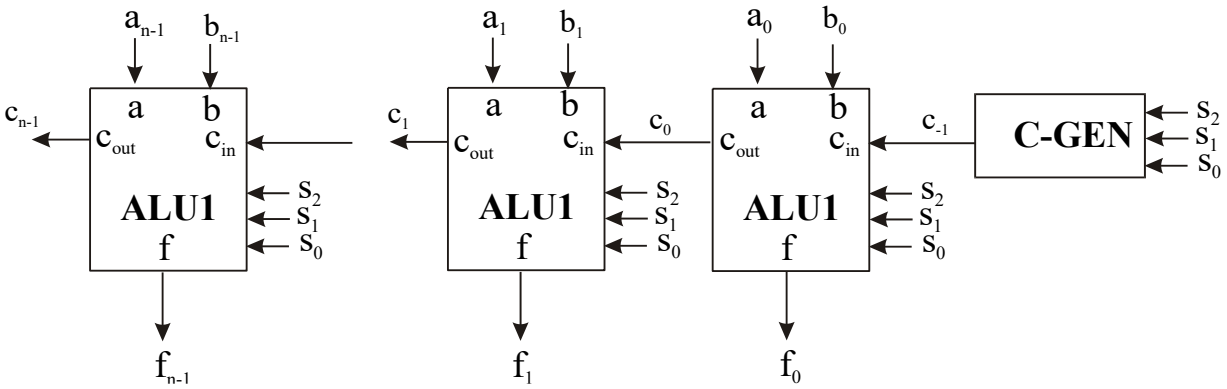

Bild 2b n-Bit ALU bestehend aus n-facher Verwendung von ALU1

Logic Unit (LU)

ALU1 lässt sich nach Bild 3 weiter zerlegen in eine Logic Unit (LU) und eine Arithmetic Unit (AU). Durch einen 2-zu-1-Multiplexer MUX2 wird über die Steuerleitung s_2 das logische oder arithmetische Ergebnis an den Ausgang f von ALU1 durchgeschaltet.

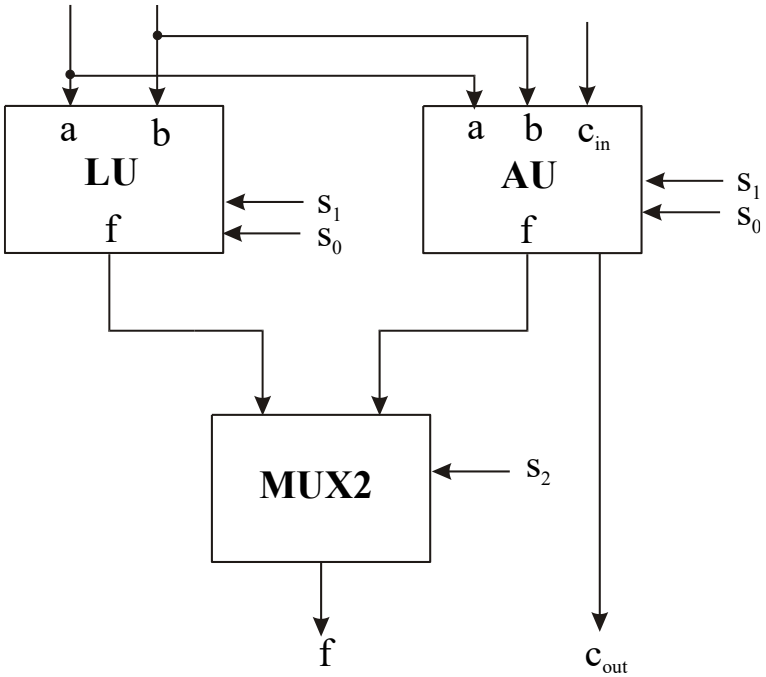

Bild 3 Zerlegung von ALU1 in eine Logik-Unit (LU) und eine Arithmetic-Unit (AU).

Mit Hilfe von s_0 und s_1 wird eine logische Verknüpfung ausgewählt (siehe Tabelle 2), die auf die Eingänge *a* und *b* angewendet und das Ergebnis am Ausgang *f* ausgegeben wird.

Name	s_1	s_0	f
UND	0	0	$a\ b$
ODER	0	1	$a + b$
NOT	1	0	a‘
XOR	1	1	$a \oplus b$

Tabelle 2 Beschreibung der LU.

Arithmetic Unit (AU)

Die logischen Operationen werden in den LUs bitweise d.h. getrennt ausgeführt. Deshalb arbeiten die LUs einer ALU *unabhängig* voneinander. Die AUs einer ALU müssen hingegen Überträge weitergeben bzw. Überträge aus niedrigeren Stellen berücksichtigen. Wie der n-Bit-Addierer sind die AUs deshalb durch Carry-Leitungen miteinander verbunden.

Zur Realisierung der Addition $A+B$ enthält jede AU einen 1-Bit-Volladdierer FA. Eine ALU (und damit jede AU) muss jedoch auch Subtraktion, Inkrement und Dekrement beherrschen. Die Subtraktion wird auf die Addition des Zweier-Komplements zurückgeführt.

> **Wiederholung**: Das Zweierkomplement einer Zahl B lässt sich durch Komplementierung der einzelnen Stellen b_i und anschließender Addition von 1 bilden.

Bild 4 zeigt, dass jede AU aus einem 1-Bit-Volladdierer FA und einer Zusatz-Logik Y-GEN besteht.

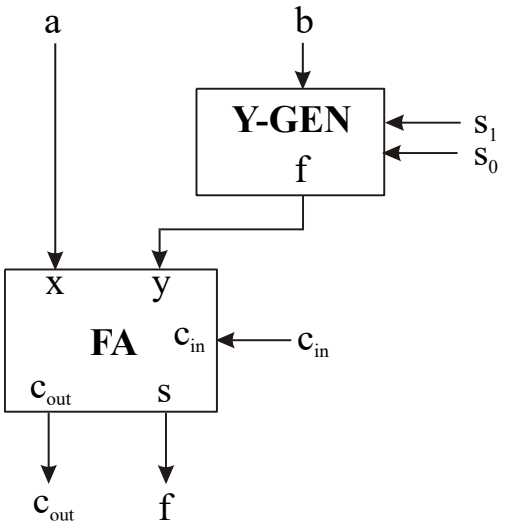

Bild 4 Struktur einer Arithmetic-Unit AU.

Die arithmetischen Operationen einer ALU werden wie folgt durch AUs implementiert: Die AUs bilden hintereinandergeschaltete 1-Bit-Volladierer mit zusätzlicher Funktionalität in den Komponenten **Y-GEN** (Bild 4) und einer zentralen Komponente **C-GEN** (Bild 2, weiter oben):

- **Addition *A+B***: keine zusätzliche Funktionalität durch die Y-GENs und den C-GEN nötig.
- **Subtraktion *A-B***: Jede AU an der Stelle i bildet in ihrem Y-GEN das Komplement von b_i, die Addition von 1 zur Bildung des Zweierkomplements von B wird von der zentralen Stelle C-GEN durchgeführt. Dazu erzeugt C-GEN an der niedrigstwertigen Stelle für c_{-1} den Wert 1.
- **Inkrement *A*+1**: Jede AU ignoriert den Wert am Eingang b. In allen Y-GENs wird damit b_i durch die Konstante 0 ersetzt. Die zentrale Komponente C-GEN legt den Wert 1 an c_{-1}. Insgesamt entsteht für B der Wert 1.
- **Dekrement *A*-1**: Jede AU ignoriert den Wert am Eingang b. In allen Y-GENs wird b_i durch die Konstante 1 ersetzt und im zentralen C-GEN der Wert 0 für c_{-1} erzeugt. Insgesamt liegt für B der Wert –1 an.

Abhängig von der auszuführenden Operation muss also jede AU den Eingangswert y für den Volladdierer FA abhängig von der selektierten arithmetischen Operation erzeugen. Gleichzeitig erzeugt die zentrale Komponente C-GEN den Übertrag c_{-1}. Tabelle 3 beschreibt die Ausgangswerte von Y-GEN und C-GEN in Abhängigkeit von s_1 und s_0:

Operation	s_1	s_0	y	c_{-1}
Addition	0	0	b	0
Subtraktion	0	1	b‘	1
Inkrement	1	0	0	1
Dekrement	1	1	1	0

Tabelle 3 Verhalten der Komponenten Y-GEN und der zentralen Komponente C-GEN.

Aufgabe 6.1

a) Entwickeln und simulieren Sie eine erste Version der Logic Unit, eine Komponte LU1 aus einem UND-, einem ODER-, einem XOR-Gatter, einem Inverter und dem Makro MUX4. Wie viele Gatter sind für die Realisierung von LU1 insgesamt notwendig?

b) Entwickeln und simulieren Sie eine verbesserte Version der Logic Unit: Erstellen Sie die Wertetabelle der Logic Unit (4 Eingänge, 1 Ausgang) und vereinfache die Funktion. Entwickeln und simulieren Sie die LU nach der vereinfachten Form. Wie viele Gatter konnten eingespart werden?

Aufgabe 6.2

Entwickeln Sie eine minimierte Gatterschaltung der Komponente C-GEN und Y-GEN entsprechend Tabelle 3. Entwickeln und simulieren Sie die komplette Gatterschaltung der Komponente ALU1. Wie viele Gatter sind für ALU1 und wie viele Gatter für eine 4-Bit-ALU insgesamt notwendig?

Aufgabe 6.3

Entwickeln und simulieren Sie eine minimierte Gatterschaltung einer gesamten 4-Bit-ALU. Überprüfen Sie die Funktionalität aller 8 Operationen (testen Sie dabei besonders „Sonderfälle“).