Übungsaufgabe

ALU-Grundlagen: Funktionscodes, Operationen, Carry-Propagation und Statusbits

Universität:	Technische Universität Berlin
Kurs/Modul:	Technische Grundlagen der Informatik (TechGI) - Digitale Systeme
Erstellungsdatum:	September 6, 2025

Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

https://study.AllWeCanLearn.com

Technische Grundlagen der Informatik (TechGI) - Digitale Systeme

Aufgabe 1: ALU-Grundlagen: Funktionscodes, Operationen, Carry-Propagation und Statusbits

In dieser Aufgabe betrachten Sie eine 4-Bit-ALU mit Eingängen A, B, C_{in} und einem 2-Bit-Funktionscode $f \in \{00, 01, 10, 11\}$. Die vier Operationen entsprechen ADD, SUB, AND, OR. Die Carry-Propagation erfolgt bei der ADD-/SUB-Logik, und die Statusbits Z (Zero), N (Negative), C (Carry) sowie V (Overflow) werden gemäß gängiger Konvention gesetzt.

a) Funktionscodes und Operationen

Gegeben: $A = 1011_2$, $B = 0101_2$, $C_{\text{in}} = 0$. Bestimme für jeden Funktionscode $f \in \{00, 01, 10, 11\}$ $die4-Bit-SummeS$ bzw. das 4-Bit-Summe-Ergebnis und den Carry-out C_{out} der ALU.

b) Carry-Propagation

Betrachte eine Ripple-Carry-Verknüpfung aus vier 1-Bit-Addierern. Gegeben: $A = 1100_2$, $B = 0011_2$, $C_{\text{in}} = 0$. Bestimme die Zwischen-Carries c_1, c_2, c_3 und den End-Cout C_{out} für die Operation ADD ($f = 00$).

c) Statusbits

Berechne zu den jeweiligen Ergebnissen aus Unteraufgabe a) bzw. b) die Statusbits Z (Zero), N (Negative), C (Carry) und V (Overflow) der ALU. Gib jede Kombination als eigene Zeile an.

Aufgabe 2: Anwendungsszenarien und Interpretation der Statusbits

Ziel ist es, zu verstehen, wie die ALU in einem einfachen Rechenpfad arbeitet und wie Statusbits genutzt werden.

a) Testfall ADD vs. OR

Gegeben: $A = 1010_2$, $B = 0101_2$, $C_{\text{in}} = 0$. Bestimme für die Operationen ADD ($f = 00$) und OR ($f = 11$) jeweils das Ergebnis S und die Statusbits Z, N, C, V.

b) SUB-Operation mit Vorzeichen

Gegeben: $A = 0110_2$, $B = 1001_2$, $C_{\text{in}} = 0$. Bestimme das Ergebnis S und die Statusbits Z, N, C, V bei der Subtraktion ($f = 01$).

c) Interpretation der Flags in einer Bedingungenlogik

Erkläre kurz, wie die Flags Z, N, C und V genutzt werden könnten, um bedingte Sprünge oder Verzweigungen in einer einfachen CPU zu steuern, wenn eine ALU mit den vier Operationen verwendet wird. Diskutiere dabei insbesondere, wann Overflow (V) bei Vorzeichenoperationen relevant ist und wie das Carry-Bit (C) sich bei Subtraktionen verhalten kann.

Lösungen

Aufgabe 1: ALU-Grundlagen: Funktionscodes, Operationen, Carry-Propagation und Statusbits

Lösung:

a) Funktionscodes und Operationen

- f = 00 (ADD): $A + B + C_{\text{in}}$

  $$A = 1011_2, \quad B = 0101_2, \quad C_{\text{in}} = 0$$

  Berechnung: $1011_2 + 0101_2 + 0 = 0000_2$ mit Carry-out $C_{\text{out}} = 1$. Die einzelnen Carries im Ripple-Cary-Verlauf (LSB zuerst) ergeben: $c_1 = 1,\ c_2 = 1,\ c_3 = 1,\ C_{\text{out}} = 1$. Ergebnis: $S = 0000_2$, $C_{\text{out}} = 1$.

- f = 01 (SUB): $A - B$ implementiert über $A + (\neg B) + C_{\text{in}}$ (mit $C_{\text{in}} = 0$ wie vorgegeben)

  $$\neg B = 0101_2 \to 1010_2, \quad A + (\neg B) + 0 = 1011_2 + 1010_2 + 0$$

  Ergebnis: $S = 0101_2$ und $C_{\text{out}} = 1$. Carries: $c_1 = 0,\ c_2 = 1,\ c_3 = 0,\ C_{\text{out}} = 1$.

- f = 10 (AND): Bitweise-Verknüpfung

  $$A \wedge B = 1011_2 \wedge 0101_2 = 0001_2$$

  Carry-Output hier nicht relevant (typischerweise $C_{\text{out}} = 0$ für logische Operationen).

- f = 11 (OR): Bitweise-Oder

  $$A \vee B = 1011_2 \vee 0101_2 = 1111_2$$

  Carry-Output hier ebenfalls $C_{\text{out}} = 0$.

b) Carry-Propagation

Gegeben: $A = 1100_2$, $B = 0011_2$, $C_{\text{in}} = 0$. Operation: ADD ($f = 00$).

Rechnerisch ergibt sich:

$$S = 1100_2 + 0011_2 + 0 = 1111_2, \quad C_{\text{out}} = 0.$$

Die Carry-Ketten (4 Bit) ergeben:

$$c_1 = 0, \quad c_2 = 0, \quad c_3 = 0, \quad C_{\text{out}} = 0.$$

c) Statusbits

Berechne zu den jeweiligen Ergebnissen aus Unteraufgabe a) bzw. b) die Statusbits Z (Zero), N (Negative), C (Carry) und V (Overflow). Gib jede Kombination als eigene Zeile an.

- $f = 00$: $S = 0000_2$, $C_{\text{out}} = 1$

  $$Z = 1, \quad N = 0, \quad C = 1, \quad V = c_3 \oplus C_{\text{out}} = 1 \oplus 1 = 0.$$

- $f = 01$: $S = 0101_2$, $C_{\text{out}} = 1$

  $$Z = 0, \quad N = 0, \quad C = 1, \quad V = c_3 \oplus C_{\text{out}} = 0 \oplus 1 = 1.$$

- $f = 10$: $S = 0001_2$, $C_{\text{out}} = 0$

  $$Z = 0, \quad N = 0, \quad C = 0, \quad V = c_3 \oplus C_{\text{out}} = 0 \oplus 0 = 0.$$

- $f = 11$: $S = 1111_2$, $C_{\text{out}} = 0$

  $$Z = 0, \quad N = 1, \quad C = 0, \quad V = c_3 \oplus C_{\text{out}} = 0 \oplus 0 = 0.$$

Aufgabe 2: Anwendungsszenarien und Interpretation der Statusbits

Lösung:

a) Testfall ADD vs. OR

Gegeben: $A = 1010_2$, $B = 0101_2$, $C_{\text{in}} = 0$.

- ADD ($f = 00$):

$$S = A + B + C_{\text{in}} = 1010_2 + 0101_2 + 0 = 1111_2, \quad C_{\text{out}} = 0.$$

  Statusbits: $Z = 0,\ N = 1,\ C = 0,\ V = 0$.

- OR ($f = 11$):

$$S = A \vee B = 1010_2 \vee 0101_2 = 1111_2, \quad C_{\text{out}} = 0.$$

  Statusbits: $Z = 0,\ N = 1,\ C = 0,\ V = 0$.

b) SUB-Operation mit Vorzeichen

Gegeben: $A = 0110_2$, $B = 1001_2$, $C_{\text{in}} = 0$. Operation: SUB ($f = 01$).

$$S = A + (\neg B) + C_{\text{in}} = 0110_2 + 0110_2 + 0 = 1100_2, \quad C_{\text{out}} = 0.$$

Statusbits: $Z = 0,\ N = 1,\ C = 0,\ V = 1$ (Overflow tritt auf, da Vorzeichen der Operanden und Vorzeichen des Ergebnisses widersprüchlich sind).

c) Interpretation der Flags in einer Bedingungenlogik

- Z (Zero): Ist der Wert des Results Null, kann Z genutzt werden, um z. B. gleichsprungige Verzweigungen zu ermöglichen. Eine Bedingung wie "wenn Z gesetzt" entspricht gleichwertigem Vergleich “R == 0”.

- N (Negative): Zeigt das Vorzeichen des Ergebnisses (MSB) an. In Vorzeichenarithmetik steuert N bedingte Sprünge basierend auf dem Vorzeichen des Resultats.

- C (Carry): Wichtig für unaries/unsigned Vergleiche. Bei ADD entspricht C der Carry-out und signalisiert unsigned Overflow des Summenwerts. Bei SUB (implementiert als A + (B) + Cin) kann C als Indikator dienen, ob ein borrow aufgetreten ist (in vielen Architekturen gilt: C=1 bedeutet kein Borrow, C=0 Borrow).

- V (Overflow): Relevanter Indikator für signed Arithmetic. Overflow tritt auf, wenn das Vorzeichen des Ergebnisses inkorrekt ist bezogen auf die Vorzeichen der Operanden. Für Addition gilt V = Carry into MSB XOR Carry out of MSB; analog gilt diese Beziehung auch in der SUB-Transformation, wenn man die Carry-Ketten betrachtet. In obigem Beispiel (SUB mit A=$0110_2, B = 1001_2$)$istV = 1, dadassignierteErgebnisauerhalbdesdarstellbarenBereichesliegt.$