DS 17 – Addition im Binärsystem und Halbaddierer

Informatik 10 – MINT-Zweig | Informatiksysteme

Lernziel

Ich kann die Addition zweier natürlicher Zahlen im Binärsystem durchführen und den Aufbau und die Funktionsweise eines Halbaddierers erläutern.

🕘 Stundenverlaufsplan (Doppelstunde, 90 Minuten)

Orientierungsrahmen für 11 SuS: Die ersten 80 Minuten bilden den Kern. Die letzten 10 Minuten sind bewusst flexibel und werden nicht mit neuem Pflichtstoff belegt.

Phase	Zeit	Inhalt/Aktion	Sozialform
Einstieg und Aktivierung	10'	Eine bekannte Dezimaladdition mit Übertrag auf Binärzahlen übertragen.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Die vier Fälle der Addition zweier Bits gemeinsam entwickeln.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	Aus Summenbit und Übertrag die Belegungstabelle und den Halbaddierer herleiten.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Halbaddierer in CircuitVerse bauen und alle vier Belegungen testen.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Funktion von XOR und AND im Halbaddierer erläutern lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	Weitere Handrechnungen oder optionale Python-Kontrolle.	Flexibel; bei Bedarf für Fragen, Fehlersuche oder Abschluss nutzen

🗣️ Sprachhilfe

- „Für die Eingangswerte … liefert die Schaltung den Ausgangswert …“
- „Der Term/die Tabelle repräsentiert die Schaltung, weil …“
- „Die Rückkopplung bzw. Gruppierung bewirkt …“

📌 Merksätze

- Die **binäre Addition** funktioniert wie die schriftliche Addition im Dezimalsystem, nur mit den Ziffern `0` und `1`: `0+0=0`, `0+1=1`, `1+0=1`, `1+1=10` (also `0` schreiben, `1` **Übertrag**)
- Ein **Halbaddierer** addiert **zwei einzelne Bits** (`A`, `B`) und liefert zwei Ausgänge:
  - **Summe (S):** `A XOR B`
  - **Übertrag (C, Carry):** `A AND B`
- "Halb"-Addierer deshalb, weil er **keinen eingehenden Übertrag** aus einer vorherigen Stelle berücksichtigen kann – das kann erst der Volladdierer (DS 18).

💡 Beispiel: Binäre Addition von Hand

```
  0101   (= 5)
+ 0011   (= 3)
-------
  1000   (= 8)
```

Rechts beginnend: 1+1=10 (0, Übertrag 1) → 0+1+1(Übertrag)=10 (0, Übertrag 1) → 1+0+1(Übertrag)=10 (0, Übertrag 1) → 0+0+1(Übertrag)=1

```python
# Kontrolle mit Python (nur zur Überprüfung, nicht als Ersatz fürs Kopfrechnen!)
a = int("0101", 2)
b = int("0011", 2)
summe = a + b
print(bin(summe))
```

```
0b1000
```

📝 Übung 1 – Von Hand rechnen

Berechnet **von Hand** (auf Papier, Schritt für Schritt mit Übertrag) folgende binäre Additionen und kontrolliert anschließend mit Python:

1. `0110 + 0101`
2. `1011 + 0111`
3. `1111 + 0001`

```python
# TODO: Übung 1 - Kontrolle eurer Handrechnungen
aufgaben = [("0110", "0101"), ("1011", "0111"), ("1111", "0001")]
for a_str, b_str in aufgaben:
    a = int(a_str, 2)
    b = int(b_str, 2)
    print(a_str, "+", b_str, "=", bin(a + b))
```

```
0110 + 0101 = 0b1011
1011 + 0111 = 0b10010
1111 + 0001 = 0b10000
```

🖥️ Praxisteil: Halbaddierer in CircuitVerse bauen

1. Legt zwei Input-Schalter (`A`, `B`) an.
2. Fügt ein **XOR-Gatter** hinzu (Eingänge: `A`, `B`) → Ausgang ist die **Summe S**.
3. Fügt ein **AND-Gatter** hinzu (Eingänge: `A`, `B`) → Ausgang ist der **Übertrag C**.
4. Verbindet beide Gatterausgänge mit je einer Output-LED.
5. Testet alle vier Belegungen und vergleicht mit der Merksatz-Formel oben.

📝 Übung 2 – Halbaddierer in Python nachbilden

Schreibt eine Funktion `halbaddierer(a, b)`, die ein Tupel `(summe, uebertrag)` zurückgibt, und testet sie für alle vier Belegungen.

Lösung

✅ Lösung Übung 2

Lösung

```python
def halbaddierer(a, b):
    summe = a != b        # entspricht XOR
    uebertrag = a and b   # entspricht AND
    return int(summe), int(uebertrag)

for a in [0, 1]:
    for b in [0, 1]:
        print(a, b, "->", halbaddierer(a, b))
```

```
0 0 -> (0, 0)
0 1 -> (1, 0)
1 0 -> (1, 0)
1 1 -> (0, 1)
```