Lernzettel

Logische Schaltungen und Hardwarebausteine: Addierer, Speicherzellen, Flip-Flops

Universität: Technische Universität Berlin
Kurs/Modul: Einführung in die Informatik
Erstellungsdatum: September 19, 2025

Lernzettel: Logische Schaltungen und Hardwarebausteine

(1) Grundlagen der digitalen Logik.
Digitale Schaltungen arbeiten mit zwei Zuständen: 0 und 1. Sei $A, B \in \{0, 1\}$. Die Hauptgatter sind:

$$\text{AND} : A \cdot B, \quad \text{OR} : A + B, \quad \text{NOT} : \neg A.$$

Nützliche Operatoren:

$$\text{NAND} = \neg(A \cdot B), \quad \text{NOR} = \neg(A + B).$$

Wichtige Gesetze (Boolesche Algebra) helfen beim Vereinfachen von Schaltungen, z. B.:

$$A \cdot 1 = A, \quad A + 0 = A, \quad \neg(\neg A) = A, \quad A \cdot (A + B) = A.$$

Gatter-Symbole und einfache Schaltpläne helfen beim Entwurf.

(2) Addierer.
Addierer verarbeiten Binärzahlen bitweise und erzeugen eine Summe sowie einen Übertrag.

Half-Adder (HA): zwei Eingänge A, B.

$$S = A \oplus B, \quad C = A \cdot B.$$

Full-Adder (FA): drei Eingänge A, B, Cin.

$$S = A \oplus B \oplus \text{Cin}, \qquad C_{\text{out}} = (A \cdot B) + (A \cdot \text{Cin}) + (B \cdot \text{Cin}).$$

Ripple-Carry-Addierer: Verkette n FA-Blöcke. - Der Übertrag aus Stufe k wird zu Cin der Stufe k+1. - Gesamtergebnis besteht aus $\{S_0, \ldots, S_{n-1}\}$ und dem letzten Cout. Formeln je Stufe:

$$S_k = A_k \oplus B_k \oplus C_k, \quad C_{k+1} = (A_k \cdot B_k) + (A_k \cdot C_k) + (B_k \cdot C_k).$$

(3) Speicherzellen.
Zustände werden in bistabilen Bausteinen gehalten.

SR-Latch (NOR-basiert): zwei Eingänge S (Set) und R (Reset), zwei Ausgänge Q und Q'. Typische Funktionsweise (NOR-basiert):

$$Q = \neg(S \vee Q'), \quad Q' = \neg(R \vee Q).$$

Ungültig ist der Zustand S=1, R=1.

D-Latch (datenabhängige Speicherung): Gating durch Clock (CLK). Die Eingangszustände werden nur beim hohen CLK gespeichert.

$$Q_{t+1} = \text{CLK} \cdot D + \overline{\text{CLK}} \cdot Q_t.$$

Register aus Reihe von Latches/Flip-Flops: Mehrere Bits speichern, z. B. ein 8-Bit-Register.

(4) Flip-Flops.
Flip-Flops speichern zuverlässig bei Clock-Events.

D-Flip-Flop: - Das Ergebnis Q folgt dem Eingang D zum Rand des Clock-Signals.

$$Q^+ = D \quad \text{bei einer Taktflanke (edge).}$$

Typisch: synthetisch als synchroner Speicher.

JK-Flip-Flop: - J und K steuern das Verhalten; bei J=K=1 toggelt Q.

$$Q^+ = J\overline{Q} + \overline{K}Q.$$

T-Flip-Flop (Toggle): - Mit T=1 wird Q bei jedem Takt-Zyklus invertiert.

$$Q^+ = T \oplus Q.$$

(5) Praktische Beispiele.
4-Bit Ripple-Carry-Addierer: vier Full-Adder hintereinandergeschaltet. - Bitsignale: (A_0, A_1, A_2, A_3), (B_0, B_1, B_2, B_3). - Start-Übertrag Cin $= 0$. - Summenbits S_0, S_1, S_2, S_3 ergeben die Summe $A+B$. - Letzter Übertrag C_4 ist der Zusatzbit (Carry-out).

Formell pro Stufe k:

$$S_k = A_k \oplus B_k \oplus C_k, \qquad C_{k+1} = (A_k \cdot B_k) + (A_k \cdot C_k) + (B_k \cdot C_k).$$

(6) Typische Anwendungen und Design-Hinweise.
- Addierer bilden zentrale Bausteine in Rechenwerken, arithmetischer Logik und Adressgeneratoren.
- Speicherzellen ermöglichen Register, L1-/L2-Caches, Zwischenspeicher für Befehle und Operanden.
- Flip-Flops sind die Bausteine für sequentielle Logik, Zustandsautomaten und Steuerwerke.

(7) Kurzüberblick: Boolesche Gegenüberstellungen vs. Hardware.
- Boolesche Algebra dient der Reduktion und Vereinfachung.
- Gatterbild ist die direkte Umsetzung in Hardware.
- Von einer logischen Gleichung zu einem physischen Schaltkreis braucht es oft mehrere Stufen (Optimierung, Timing, Hazards).

Zusammenfassung in Kompaktdarstellung.
- Addierer arbeiten bitweise; Full-Adder addiert A, B und Cin.
- Speicherzellen speichern Bits in bistabilen Elementen.
- Flip-Flops liefern synchronen bzw. asynchronen Speicher, adressierbar über Clock.