Computer Engineering – Submission due 27.10.2020

Jan Vogt, Yannik Köllmann - in gleichen Teilen

23. Oktober 2025

1 Basics of Logic Design

1.1 Boolean Algebra

Tasks

1. Prove $A + \overline{A} = 1$ using perfect induction. Apply only one axiom per step and name it.

   Beweis durch perfekte Induktion:

   Fall 1: $A = 0$

   $$\begin{aligned} A + \overline{A} &= 0 + \overline{0} \\ &= 0 + 1 \quad \text{(Axiom: NOT)} \\ &= 1 \quad \text{(Axiom: OR/AND)} \end{aligned}$$

   Fall 2: $A = 1$

   $$\begin{aligned} A + \overline{A} &= 1 + \overline{1} \\ &= 1 + 0 \quad \text{(Axiom: NOT)} \\ &= 1 \quad \text{(Axiom: OR/AND)} \end{aligned}$$

   Da für beide möglichen Werte von A das Ergebnis 1 ist, gilt $A + \overline{A} = 1$.

2. Prove $A \cdot A = A$ using perfect induction. In every step apply only a single axiom. State which axiom you are using.

   Beweis durch perfekte Induktion:

   Fall 1: $A = 0$

   $$\begin{aligned} A \cdot A &= 0 \cdot 0 \\ &= 0 \quad \text{(Axiom: OR/AND)} \end{aligned}$$

   Fall 2: $A = 1$

   $$\begin{aligned} A \cdot A &= 1 \cdot 1 \\ &= 1 \quad \text{(Axiom: OR/AND)} \end{aligned}$$

   Da für beide möglichen Werte von A gilt $A \cdot A = A$, ist die Aussage bewiesen.

3. Prove $\overline{A + B} = \overline{A} \cdot \overline{B}$ using perfect induction. In every step apply only a single axiom. State which axiom you are using.

   Beweis durch perfekte Induktion:

   Fall 1: $A = 0, B = 0$

   $$\begin{aligned} \overline{A + B} &= \overline{0 + 0} \\ &= \overline{0} \quad \text{(Axiom: OR/AND)} \\ &= 1 \quad \text{(Axiom: NOT)} \end{aligned}$$

$$\begin{aligned}\overline{A} \cdot \overline{B} &= \overline{0} \cdot \overline{0} \\ &= 1 \cdot 1 \quad \text{(Axiom: NOT)} \\ &= 1 \quad \text{(Axiom: OR/AND)}\end{aligned}$$

Fall 2: $A = 0, B = 1$

$$\begin{aligned}\overline{A + B} &= \overline{0 + 1} \\ &= \overline{1} \quad \text{(Axiom: OR/AND)} \\ &= 0 \quad \text{(Axiom: NOT)}\end{aligned}$$

$$\begin{aligned}\overline{A} \cdot \overline{B} &= \overline{0} \cdot \overline{1} \\ &= 1 \cdot 0 \quad \text{(Axiom: NOT)} \\ &= 0 \quad \text{(Axiom: OR/AND)}\end{aligned}$$

Fall 3: $A = 1, B = 0$

$$\begin{aligned}\overline{A + B} &= \overline{1 + 0} \\ &= \overline{1} \quad \text{(Axiom: OR/AND)} \\ &= 0 \quad \text{(Axiom: NOT)}\end{aligned}$$

$$\begin{aligned}\overline{A} \cdot \overline{B} &= \overline{1} \cdot \overline{0} \\ &= 0 \cdot 1 \quad \text{(Axiom: NOT)} \\ &= 0 \quad \text{(Axiom: OR/AND)}\end{aligned}$$

Fall 4: $A = 1, B = 1$

$$\begin{aligned}\overline{A + B} &= \overline{1 + 1} \\ &= \overline{1} \quad \text{(Axiom: OR/AND)} \\ &= 0 \quad \text{(Axiom: NOT)}\end{aligned}$$

$$\begin{aligned}\overline{A} \cdot \overline{B} &= \overline{1} \cdot \overline{1} \\ &= 0 \cdot 0 \quad \text{(Axiom: NOT)} \\ &= 0 \quad \text{(Axiom: OR/AND)}\end{aligned}$$

Da für alle vier möglichen Wertekombinationen von A und B gilt $\overline{A + B} = \overline{A} \cdot \overline{B}$, ist die Aussage bewiesen.

4. Simplify $\overline{A}(A + B) + (B + A)(A + \overline{B})$. State the law used in each step.

   Vereinfachung:

$$\begin{aligned}\overline{A}(A + B) + (B + A)(A + \overline{B}) &= \overline{A}(A + B) + (A + B)(A + \overline{B}) \quad \text{(Commutativity)} \\ &= \overline{A}(A + B) + [A + (B \cdot \overline{B})] \quad \text{(Distributivity)} \\ &= \overline{A}(A + B) + [A + 0] \quad \text{(Inverse)} \\ &= \overline{A}(A + B) + A \quad \text{(Identity)} \\ &= \overline{A} \cdot A + \overline{A} \cdot B + A \quad \text{(Distributivity)} \\ &= 0 + \overline{A} \cdot B + A \quad \text{(Inverse)} \\ &= \overline{A} \cdot B + A \quad \text{(Identity)} \\ &= A + \overline{A} \cdot B \quad \text{(Commutativity)} \\ &= (A + \overline{A}) \cdot (A + B) \quad \text{(Distributivity)} \\ &= 1 \cdot (A + B) \quad \text{(Inverse)} \\ &= A + B \quad \text{(Identity)}\end{aligned}$$

1.2 Wires and Gates

Tasks

1. Complete Table 2.2.1.

   Vervollständigte Wahrheitstabelle:

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	0

2. Formulate function $\mathcal{F}(A, B, C) = (D, E, F)$ through Boolean equations, i.e., find Boolean equations which encode the provided textual descriptions.

   Boolean-Gleichungen:

$$
\begin{aligned}
D &= A \cdot B \cdot C \\
E &= A + B + C \\
F &= \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C}
\end{aligned}
$$

3. Design a combinational circuit in CircuitVerse that implements the function $\mathcal{F}$. Label the inputs (A, B and C) and outputs (D, E, F).

   `(siehe ./src)`

4. If not already done in the previous task: Design a similar circuit which only uses two-input gates.

   `(siehe ./src)`

1.3 Universal Gates

Tasks

1. Prove the universality of {NOR} using Boolean equations.

   Beweis der Universalität von NOR:

   Um zu zeigen, dass NOR universal ist, müssen wir beweisen, dass alle grundlegenden logischen Operationen (NOT, AND, OR) nur mit NOR-Gattern ausgedrückt werden können.

   NOT:

$$
\overline{A + A} = \overline{A} \quad \text{(Idempotency)}
$$

   OR:

$$
\begin{aligned}
\overline{\overline{A + B}} &= \overline{\overline{A} \cdot \overline{B}} && \text{(DeMorgan)} \\
&= \overline{\overline{A}} + \overline{\overline{B}} && \text{(DeMorgan)} \\
&= A + B && \text{(Double Complement)}
\end{aligned}
$$

   AND:

$$
\begin{aligned}
\overline{\overline{A} + \overline{B}} &= \overline{\overline{A}} \cdot \overline{\overline{B}} && \text{(DeMorgan)} \\
&= A \cdot B && \text{(Double Complement)}
\end{aligned}
$$

   Da alle grundlegenden Operationen (NOT, OR, AND) nur mit NOR-Gattern ausgedrückt werden können, ist {NOR} universal.

2. In CircuitVerse, implement the logical operations AND, OR and NOT using only two-input NOR gates.

   `(siehe ./src)`

1.4 Equality Comparator

Tasks

1. Implement an equality comparator for the two 4-bit inputs $A_{[3:0]}$ and $B_{[3:0]}$ in CircuitVerse.

   `(siehe ./src)`

2. Showcase your design by running a simulation with the following inputs:

   (a) $A_{[3:0]} = 1011_2$ and $B_{[3:0]} = 1001_2$, and
   (b) $A_{[3:0]} = 1101_2$ and $B_{[3:0]} = 1101_2$.

   `(siehe ./src)`