Probeklausur

Technische Grundlagen der Informatik (TechGI) - Digitale Systeme

Universität:	Technische Universität Berlin
Kurs/Modul:	Technische Grundlagen der Informatik (TechGI) - Digitale Systeme
Bearbeitungszeit:	120 Minuten
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.

(a) Gegeben sei der logische Ausdruck $F(A, B, C) = AB + \overline{A}C$. Erstellen Sie die vollständige Wahrheitstafel für die Eingangsvariablen A, B, C und tragen Sie die Werte in die Spalte F ein.

A	B	C	$F(A, B, C)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(b) Vereinfachen Sie $F(A, B, C)$ mit Booleschen Regeln und geben Sie die minimale Form an. Formulieren Sie Ihre Schritte in kurzer textueller Form.

(c) Beschreiben Sie eine minimal realisierte Schaltung für F mit nur logischen Gattern (AND, OR, NOT). Geben Sie die benötigten Zwischengliederungen und die logische Struktur an.

Aufgabe 2.

(a) Beschreiben Sie die Übergangslogik eines D-Flipflops. Formulieren Sie die Abhängigkeit der nächsten Zustand Q^+ von dem aktuellen Zustand Q und dem Eingangsbit D in Textform. Begründen Sie kurz, warum ein Taktsignal erforderlich ist.

(b) Gegeben sei ein einfacher synchroner Zähler, der aus einem D-Flipflop besteht und bei jedem Taktinkrement um eins zählt. Geben Sie die Art der Zählfolge an und erläutern Sie die Rolle der D-Eingänge in dieser Schaltung.

(c) Zeigen Sie die Beziehung zwischen Taktperiode T, Setup-Zeit t_s und Hold-Zeit t_h in einem typischen Taktsystem, und geben Sie eine allgemeine Formel für die minimale Taktperiode $T_{\min}$ an. Erläutern Sie kurz, welche Parameter beeinflusst werden dürfen, um eine korrekte Funktion zu gewährleisten.

Aufgabe 3.

(a) Betrachten Sie eine zweistufige Schaltwerk-Konfiguration mit Flipflops Q_1 und Q_2 und dem folgenden Übergangsverhalten: - $Q_1^+ = D_1 = \overline{Q_1} \ \wedge \ Q_2$ - $Q_2^+ = D_2 = Q_1$

Formulieren Sie die Zustandsgleichungen der Schaltung und erklären Sie, wie sich ein Anfangszustand auf die spätere Sequenz auswirkt.

(b) Bestimmen Sie die Anzahl der Zustände der obigen Schaltwerk-Konfiguration und geben Sie eine kurze Begründung.

(c) Skizzieren Sie in Textform eine Moore- oder Mealy-Variante dieses Schaltwerks und erläutern Sie den Unterschied in der Abhängigkeit von Ausgabewerten Z zu dem internen Zustand.

Aufgabe 4.

(a) Entwerfen Sie eine 1-Bit-AlU-Sektion als Auszug einer 4-Bit-ALU, die additionieren kann. Definieren Sie die Signale: - Eingaben: A, B (je 1 Bit), $CarryIn$ - Ausgaben: $Sum, CarryOut$
Geben Sie die grundlegende Gleichung für Sum und eine Form für $CarryOut$ an.

(b) Zeigen Sie, wie eine 4-Bit-Ripple-Carry-Addition durch Verknüpfung von vier 1-Bit-ALU-Teilen realisiert wird. Skizzieren Sie die Carry-Kette in Textform und erläutern Sie die Vor- und Nachteile dieser Architektur.

(c) Geben Sie eine kurze Beschreibung, wie eine subtract-Funktion in einer ALU durch Invertieren von B und Setzen von $CarryIn = 1$ realisiert werden kann. Formulieren Sie die resultierende Gleichung für die Differenz $A - B$ in zwei Schritten.

Aufgabe 5.

(a) Beschreiben Sie kurz die Funktionsweise einer programmierbaren Logikbaustein-Struktur (PLA) mit drei Eingängen x_1, x_2, x_3 und zwei Ausgängen y_1, y_2. Geben Sie an, wie Logikfunktionen dort implementiert werden können, ohne eine konkrete Schaltungszeichnung zu liefern.

(b) Gegeben seien zwei Ziel-Funktionen $F_1(x_1, x_2, x_3)$ und $F_2(x_1, x_2, x_3)$ zu implementieren. Beschreiben Sie die generelle Vorgehensweise, wie ein PLA diese Funktionen durch entsprechende Minterm-Zuordnungen realisieren kann. Erläutern Sie, welche Informationen erforderlich sind, damit die Zuordnung gelingt.

(c) Diskutieren Sie kurz Vor- und Nachteile von PLA-basierten Implementierungen im Vergleich zu festen logischen Schaltkreisen (ASIC, FPGA-Ansätze) hinsichtlich Flexibilität, Leistung und Lieferumfang.

Lösungen

Lösung zu Aufgabe 1.

(a) Die Wahrheitstafel für $F(A,B,C) = AB + \overline{A}C$ ergibt sich direkt durch Auswerten der beiden Terme AB und $\overline{A}C$ und deren ODER-Verknüpfung. Die Werte von F in Abhängigkeit von A, B, C sind:

A	B	C	$F(A,B,C)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(b) Vereinfachung von $F(A,B,C) = AB + \overline{A}C$:

$$F(A,B,C) = AB + \overline{A}C$$

Durch Shannon-Entkopplung über A erhält man

$$F = A \cdot F|_{A=1} + \overline{A} \cdot F|_{A=0} = A\,B + \overline{A}\,C.$$

Dies ist auch die minimale Form, da sich die beiden Terme nicht durch Absorption oder Konsensus weiter reduzieren lassen (Karnaugh-Map zeigt zwei voneinander unabhängige Gruppen: $\overline{A}C$ für $A = 0$ und AB für $B = 1$ mit beliebigem C).

(c) Minimal realisierte Schaltung (nur AND, OR, NOT):

- Erzeuge $\overline{A}$ mittels NOT.
- GIBT G1 = A $\wedge B.GIBTG2 = \overline{A} \wedge C$.

- F = G1 $\vee G2$.

Damit entsteht eine zweiprozentige Summe eines Produkts, bestehend aus zwei UND-Gates (einschließlich eines NOT-Eingangs) und einem ODER-Gate.

Lösung zu Aufgabe 2.

(a) Ein D-Flipflop ist ein Taktsignalgesteuertes Speicherelement mit der Eigenschaft, dass der Eingangswert D auf die Zellent651 (Q) beim Auftreten eines aktiven Flankenübergangs des Takt-Eingangs übernommen wird. Formal gilt bei einem positiven Edge-Trigger:

$$Q^+ = D \quad \text{(bei dem passenden Flankenfenster)}$$

Zwischen zwei Taktsignalen bleibt der Ausgang Q konstant. Das Taktsignal ist erforderlich, um die Zustandsänderungen synchron und deterministisch zu timen; ohne Takt würden Geraden und Störsignale zu unvorhersagbaren Zuständen führen.

(b) Ein einfacher synchroner Zähler, der aus einem D-Flipflop besteht und bei jedem Takt inkrementiert, realisiert typischerweise eine Zählfolge $Q^+ = Q + 1 \bmod 2$. Das bedeutet für den einzigen Flipflop:

$$D = \overline{Q} \quad \Rightarrow \quad Q^+ = \overline{Q}.$$

Die Zählfolge ist damit 0, 1, 0, 1, ... (eine 1-Bit-Paginator-/Inkrementierung). Die D-Eingänge dienen hier dem Tastverhalten des Zählers: D wird aus dem aktuellen Zustand abgeleitet, sodass der Zähler bei jedem Takt den nächsten Zustand übernimmt (in diesem Fall durch Invertierung des aktuellen Zustands).

(c) Beziehung zwischen Taktperiode T, Setup-Zeit t_s und Hold-Zeit t_h in einem typischen Taktsystem:

- Timing-Bedingungen für eine sequentielle Kette (Flip-Flop -> Logik -> Flip-Flop): - Setup-Bedingung: $T \geq t_{co} + t_{pd} + t_s$, wobei t_{co} die Clock-to-Q-Verzögerung des Quell-Flipflops und t_{pd} die maximale Verzögerung der Zwischeng funktionale Logik (und interconnect) ist. - Hold-Bedingung: $t_{co(min)} \geq t_h + t_{cd(min)}$, wobei $t_{co(min)}$ die minimale Clock-to-Q-Verzögerung und $t_{cd(min)}$ die minimale Verzögerung der logischen Pfade darstellt.
- Allgemeine Minimalkriterium für das Taktsignal:

$$T_{\min} = t_{co}^{\max} + t_{pd}^{\max} + t_s.$$

Man kann also Parameter beeinflussen, um eine korrekte Funktion zu gewährleisten: - Verringerung von $t_{co}^{\max}$ (schnellere FF-Implementierung), - Verringerung von $t_{pd}^{\max}$ (optimierte Logikpfade, Treiber), - Verringerung von t_s (weniger Setup-Verzögerung am Ziel-Flipflop), - oder Einbau zusätzlicher Pipelines/Speicherelemente, um die maximale Nutzungsdauer zu verringern.

Lösung zu Aufgabe 3.

(a) Gegeben ist das zweistufige Schaltwerk mit den Übergangs- (Zustands-)Details

$$Q_1^+ = D_1 = \overline{Q_1} \ \wedge \ Q_2, \qquad Q_2^+ = D_2 = Q_1.$$

Die Zustandsgleichungen lauten damit explizit:

$$Q_1^+ = \overline{Q_1} \cdot Q_2,$$
$$Q_2^+ = Q_1.$$

Aus diesen Gleichungen ergibt sich das Folgende zum Zustandsablauf: - Für $Q_1Q_2 = 00$: $Q_1^+ = 1 \cdot 0 = 0$, $Q_2^+ = 0$ nächster Zustand 00 (Fixed Point). - Für $Q_1Q_2 = 01$: $Q_1^+ = 1 \cdot 1 = 1$, $Q_2^+ = 0$ nächster Zustand 10. - Für $Q_1Q_2 = 10$: $Q_1^+ = 0 \cdot 0 = 0$, $Q_2^+ = 1$ nächster Zustand 01. - Für $Q_1Q_2 = 11$: $Q_1^+ = 0 \cdot 1 = 0$, $Q_2^+ = 1$ nächster Zustand 01.

Damit ergibt sich eine Folge, in der 00 eine feste Ecke bildet, während 01 <-> 10 zyklisch aufeinander folgen (eine 2-Zustands-Schleife). Das Anfangs-"Zustand" beeinflusst also, ob man in die feste 00-Stellung oder in die 2-Zustands-Schleife gerät.

(b) Anzahl der Zustände: Es existieren 4 mögliche Zustände des Schaltwerks (Q1,Q2) 00, 01, 10, 11. Aufgrund der Übergangsrelationen werden alle vier Zustände rein formal als Zustände geführt, allerdings führen 01 und 10 zu einer zyklischen Folge, während 00 stabil bleibt.

(c) Moore- vs. Mealy-Variante (Textform):

- Moore-Variante (Zustandsabhängige Ausgabe): Der Ausgang Z hängt nur vom aktuellen Zustand der Flipflops ab, d. h. Z = f(Q1,Q2). Eine mögliche Zuweisung wäre z. B.

$$Z = Q_2.$$

Die Ausgabe ändert sich nur bei Taktflanken, im Zustand Q1Q2 selbst. Vorteile: einfache Analyse, stabiler Ausgangsverlauf; Nachteile: ggf. weniger flexible Reaktion auf externe Signale.

- Mealy-Variante (Ausgabe hängt zusätzlich von Eingangsgröße ab): Die Ausgabe Z kann auch vom externen Eingang X abhängen, z. B.

$$Z = f(Q_1, Q_2, X) = X \wedge Q_2,$$

oder eine andere Funktion, die den aktuellen Zustand mit einem Eingang verknüpft. Vorteil: direktere, potenziell schnellere Reaktion auf Eingaben; Nachteil: der Ausgang kann störanfälliger sein, er kann schon zwischen Taktflanken variieren, da er nicht nur vom Zustand, sondern auch von Eingängen abhängt.

Im konkreten Fall kann man eine Moore-Variante bevorzugen, wenn stabile, regelbare Ausgaben benötigt werden. Für reaktive Signalmuster, die rasch auf Eingaben reagieren sollen, bietet sich eine Mealy-Variante an.

Lösung zu Aufgabe 4.

(a) Eine 1-Bit-AlU-Sektion (Auszug einer 4-Bit-ALU) mit Addition-Funktion besitzt Eingänge A, B (je 1 Bit) sowie CarryIn, und Ausgänge Sum und CarryOut. Die grundlegende Gleichung lautet:

$$\begin{aligned} \text{Sum} &= A \oplus B \oplus \text{CarryIn}, \\ \text{CarryOut} &= (A \wedge B) \ \vee \ (\text{CarryIn} \wedge (A \oplus B)). \end{aligned}$$

(b) 4-Bit-Ripple-Carry-Addition realisiert man, indem vier 1-Bit-ALU-Segmente wie oben hintereinandergeschaltet werden. Die Carry-Ausgabe eines Stufenelements wird als CarryIn in das nächste Segment verwendet:

- C_0 (CarryIn des LSB-Segments) ist extern vorgegeben (typischerweise 0). - $C_1 = \text{CarryOut}_0$, - $C_2 = \text{CarryOut}_1$, - $C_3 = \text{CarryOut}_2$, - $C_4 = \text{CarryOut}_3$ (MSB-Ausgabe).

Vorteile: einfache Architektur, geringe Komplexität. Nachteile: langsamer bei längeren Wortbreiten aufgrund der Carry-Latenz (Carry-Propagation-Delay); Dämpfungen beim Taktfrequenzanstieg.

(c) Subtraktions-Funktion via Addierer:

$$A - B = A + (\overline{B}) + 1.$$

Zwei Schritte:

1. Invertiere B zu $B' = \overline{B}$.
2. Setze CarryIn $=\ 1$ und berechne Sum $= A \oplus B' \oplus 1$ (und CarryOut entsprechend).

Damit realisiert der 1-Bit-ALU-Slice zusätzlich die Subtraktion durch Umstellung der Eingaben; der Borrow-Out/Carry-Out verhält sich in der Subtraktionsinterpretation entsprechend (typischerweise als Borrow-Out betrachtet).

Lösung zu Aufgabe 5.

(a) Die Architektur eines PLA (Programmierbarer Logikbaustein) mit drei Eingängen x_1, x_2, x_3 und zwei Ausgängen y_1, y_2 besteht typischerweise aus zwei Ebenen: - Eine AND-Ebene (Programmable-AND-Plane), die Produktterme (Minterme) erzeugt, z. B. x_1, $\overline{x_1}x_2$, x_1x_3, etc. - Eine OR-Ebene (Programmable-OR-Plane), die eine oder mehrere dieser Produktterme zu den Ausgängen kombiniert.

Die Implementierung beliebiger Logikfunktionen $y_1 = f_1(x_1, x_2, x_3)$ und $y_2 = f_2(x_1, x_2, x_3)$ erfolgt durch Programmieren der passende Minterm-Ausdrücke in der AND-Ebene und das anschließende OR-Verknüpfen in der OR-Ebene. Die konkrete Zuordnung der Produktterme zu y_1 bzw. y_2 wird durch die Verbindungen in der PLA bestimmt (keine Schaltungszeichnung erforderlich).

(b) Gegeben seien zwei Ziel-Funktionen $F_1(x_1, x_2, x_3)$ und $F_2(x_1, x_2, x_3)$. Die generelle Vorgehensweise in einem PLA lautet:

1. Aus der Wahrheitstafel oder aus canonical forms die Mengen der Minterms bestimmen, die jeweils zu F_1 bzw. F_2 gehören (also die Indizes der Zeilen mit Output 1).
2. Diese Minterms im PLA-AND-Plane entsprechend anwenden, so dass die gewählten Minterms zu den jeweiligen Ausgängen führen.
3. Die OR-Ebene zuordnen, damit alle gewählten Minterms für jeden Ausgang zusammengefasst werden.
4. Gegebenenfalls Don't-Care-Terms nutzen, um eine noch einfachere Zuordnung zu ermöglichen.

Für eine 3-Eingang-PLA stehen maximal 8 Minterms (von 000 bis 111) zu Verfügung. Die Zuordnung gelingt, indem man festlegt, welche Minterms zu F_1 bzw. F_2 gehören; danach programmiert man die entsprechende Zuordnung in der PLA.

(c) Vor- und Nachteile von PLA-basierten Implementierungen gegenüber festen Schaltkreisen (ASIC, FPGA):

- Vorteile: - Höhere Flexibilität und einfache Anpassbarkeit, besonders in Prototypen, Tests und kleinem Produktionsmaßstab. - Kein spezieller Front-End-ASIC-Entwurf erforderlich; schnelle Implementierung von Logikänderungen.
- Nachteile: - In der Regel geringere Dichte und höhere Fläche pro Funktion im Vergleich zu optimierten ASIC/FPGA-Logikblöcken. - Geringere Taktraten und höhere Leistungsaufnahme im Vergleich zu optimierten ASIC-Implementierungen. - PLA-Architekturen bieten möglicherweise weniger Optimierungsspielraum als vollsynthetisierte Logik in modernen FPGAs/ASICs. - Lieferumfang: PLA-Lösungen können teurer in der Entwicklung sein, weniger standardisiert als fertige FPGA-Logikblöcke.