Einführung in die Technische Informatik 2. Semester 1.Test

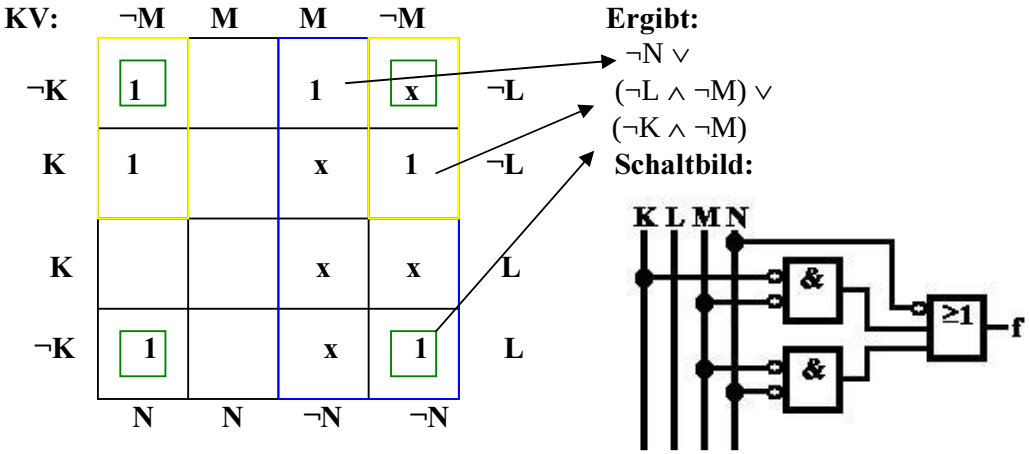
1.Punkt: optimale Schaltung
Die Angabe besteht aus einer Funktion die für 0 bis 15 jeweils logisch 0, logisch 1 oder don't care sein kann.

Funktion f: soll (mit Leitungen KLMN mit K lsb und N msb)
log 1 sein bei: 1,2,4,8,9,10
und log 0 bei: 11,12,13,14,15 sonst don't care

Vorgehensweise:
1.Schritt: Tabelle mit den 4 Variablen aufstellen sowie als 5.Zeile die Funktion mit logisch 1, logisch 0 und don't care
2.Schritt: KV Diagramm zeichnen und alle log. 1 und don't care von f eintragen
3.Schritt: Don't cares sinnvoll interpretieren um möglichst 8er ,4er,2er Blöcke aus kleineren logisch-1er-Blöcken zu bekommen.
4.Schritt: KV Diagramm lösen und das Ergebnis als Schaltbild aufzeichnen wobei hier die 4 Leitungen über & sowie ≥1 Schaltelementen und Negationskreisen bei deren Eingängen (falls nötig) verknüpft zu f laufen.
Damit ist Punkt 1 gelöst!

Beispiel mit Angaben von oben:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K:	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
L:	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
M:	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
N:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
f:	x	1	1	x	1	x	x	x	1	1	1	0	0	0	0	0



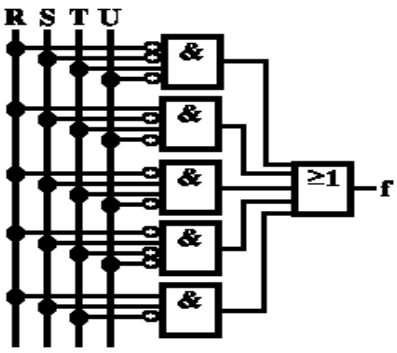
Für die Dont' Cares wurde hier überall ein 1er angenommen da sich hierdurch ein 8er Block bilden lässt.

Zusätzlich zum Schaltbild könnte die Darstellung der Lösung als PLA (Programmable Logic Array) Schaltung verlangt sein. Hierbei einfach eine Matrix aufzeichnen und die jeweilig korrekten Verknüpfungen durch Knoten anzeichnen (Seite 43 im Skriptum).

Hinweis: Wer Probleme mit KV Diagrammen hat sollte sich noch mal das entsprechende Kapitel aus Gründzüge der Informatik anschaun.

2.Punkt: Schaltung optimieren

Die Angabe besteht aus einer logischen Schaltung welche optimiert werden soll



Vorgehensweise:
1.Schritt: KV-Diagramm zeichnen
2.Schritt: Jede UND-Verknüpfung im KV-Diagramm eintragen, falls dabei einzelne Verknüpfungen weniger Eingangsvariablen als die Anderen verwenden, so jeden möglichen Zustand mit dieser Kombination als 1er annehmen. *
3.Schritt: KV-Diagramm lösen und Terme als Schaltung anschreiben.
Damit ist Punkt 2 gelöst!

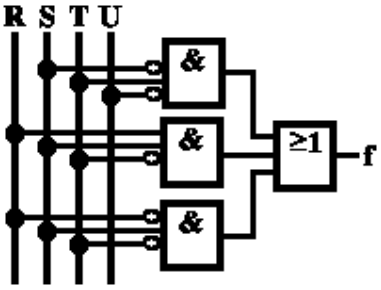
Beispiel mit Angaben von oben:

KV:

	¬U	**U**	**U**	**¬U**	
¬R	1				**¬S**
R	1				**¬S**
R			1*	1*	**S**
¬R	1			1	**S**
	T	**T**	**¬T**	**¬T**	

Ergibt:
(¬S ∧ T ∧¬U)∨
(R ∧ S ∧ ¬T) ∨
(¬R ∧ S ∧ ¬U)

Schaltbild:



3.Punkt: Übergangsdiagramme

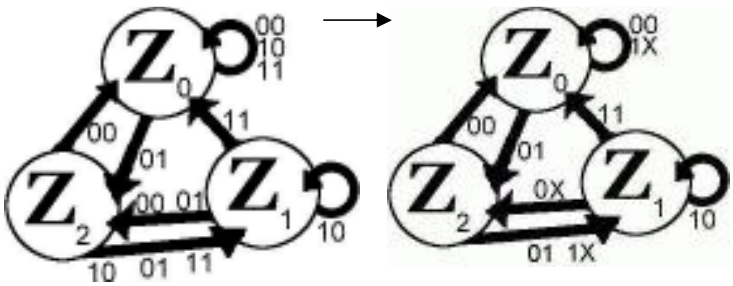
Die Angabe besteht aus einer Übergangstabelle welche in ein Zustandsdiagramm umgesetzt werden soll

A:	0 1 0 1
B:	0 0 1 1
Z_0:	Z_0 Z_0 Z_2 Z_0
Z_1:	Z_2 Z_1 Z_2 Z_0
Z_2:	Z_0 Z_1 Z_1 Z_1

Vorgehensweise:
1.Schritt: Alle Zustande getrennt als Knoten aufschreiben
2.Schritt: Von den Zuständen alle Übergänge einzeichnen und auf die Kanten die Eingangswerte schreiben
3.Schritt: Gegebenenfalls doppelte Kanten mit don't cares Vereinfachen.
Damit ist Punkt 3 gelöst!

Beispiel mit Angaben von oben:

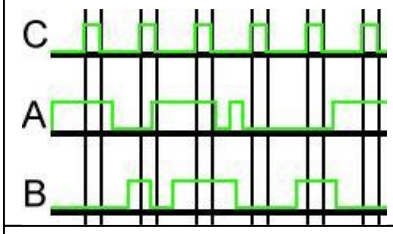
Da z.B. bei Z_0 10 und 11 wieder auf Z_0 führen kann hier ein don't care 1X gesetzt werden, denn sobald A eins ist Z_0 -> Z_0 gilt.



4.Punkt: Zeitdiagramme

Die Angabe besteht aus einer Schaltung oder einem Zustandsdiagramm und einem Zeitverlauf aus dem zeitabhängige Ausgangswerte bestimmt werden sollen.

Interpretieren sie das Zustandsdiagramm aus Bsp 3 als Schaltung mit positiver Flankensteuerung, den Eingangswerten A,B sowie Ausgangswerten X und Y. A,B haben folgenden Verlauf:



Kodierung:

	X	Y
Z_0	0	0
Z_1	0	1
Z_2	1	0

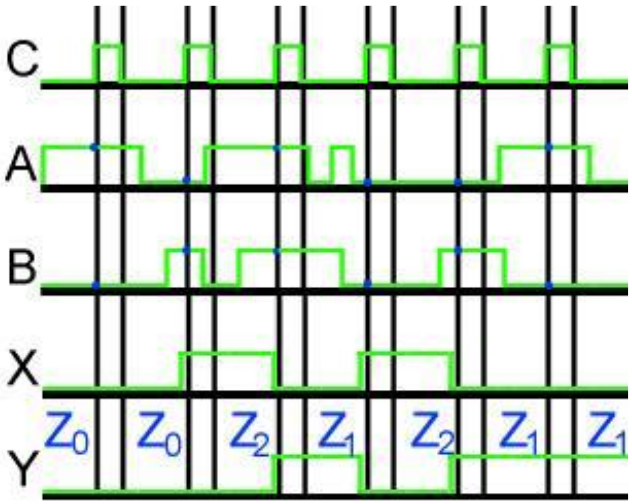
Anfangszustand: Z_0

Vorgehensweise:
1.Schritt: Zeitdiagramm anlegen
2.Schritt: Bei jedem Anstieg von C in den darunter liegenden Zeilen die Eingangswerte zu Ausgangswerten verknüpfen und eine Gerade bis zum nächsten Anstieg von C zeichnen
Damit ist Punkt 4 gelöst!

Beispiel mit Angaben von oben:

Die Übergänge an den positiven Flanken (= wo C nach oben geht) entscheiden über die Zustände:

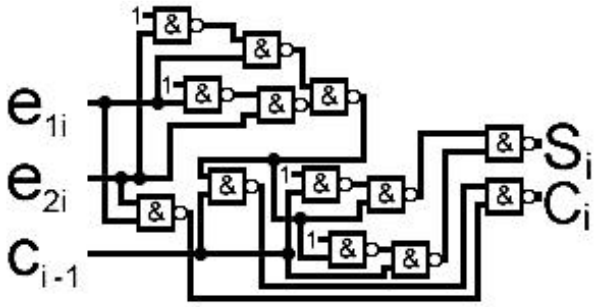
vorher	X	Y	AB	nachher
Z_0	0	0	10	Z_0
Z_0	0	0	01	Z_2
Z_2	1	0	11	Z_1
Z_1	0	1	00	Z_2
Z_2	1	0	01	Z_1
Z_1	0	1	10	Z_1
Z_1	0	1		



Der 45 minütige Test beinhalten zu den Praxisfragen auch Theoriefragen über den Stoff der Vorlesung bis einschließlich Kapitel 4.1 des Skriptums. Einige Fragen zum üben:

1. Was unterscheidet positive Logik von negativer Logik?
2. Was ist der Fan Out und der Fan In Wert?
3. Welche Schaltkreisfamilien gibt es und wodurch unterscheiden sie sich?
4. Wie sieht das Schaltbild eines Volladdierers mit ausschließlich NAND Gattern aus?
5. Was ist ein prioritätsgesteuerter Codierer?
6. Wie viele Steuereingänge braucht ein 1 zu 16 Demultiplexer und was ist seine Aufgabe?
7. Wozu benötigt man einen Taktgeber in einer Schaltung?
8. Wie funktioniert ein RS Latch und wie unterscheidet es sich von einem D Latch?
9. Was bedeutet *parallel load* bei Registern?
10. Wie ist eine Speicherzelle aufgebaut?
11. Wozu benutzt man *Tristate-Outputs* ?
12. Welche Arten von Speicherbausteinen gibt es und wodurch unterscheiden sie sich?
13. Wodurch unterscheiden sich PLA, LCA und PLD-Schaltungen?
14. Was bedeutet VHDL und wie ist es entstanden?
15. Welche Achsen besitzt das Y-Diagramm und welche Aussage steckt dahinter?
16. Was bedeutet *entity, architecture* und *configuration* bei VHDL?
17. Was versteht man unter einem endlichen Automat bzw. deterministischen Automat?

Antworten:

2. Bei positiver Logik entspricht logisch 1 einer hohen Spannung(Vcc), bei der negativen Logik einer niedrigen Spannung (GND).
3. Der Fan Out Wert gibt an wie viele Standarteingänge an den Ausgang des Schaltelements angeschlossen werden dürfen ohne durch eine zu große Lastkapazität Signalverfälschungen zu erhalten. Der Fan In Wert gibt an wie vielen Standarteingänge der Eingang des Schaltelements entspricht.
4. Es gibt: TTL *Transitor-Transitor Logic* (großer Fan out aber langer Propagation delay)
   ECL *Emiter Coupled Logic* (geringste Prop. Delay und Störsicher aber teuer)
   MOS *Metal-Oxide Semiconductor* (für hoch integrierte Schaltungen)
   CMOS *Complementary MOS* (niedrige Leistungsaufnahme, hohe Spannung)
5. 

6. Solche Codiererschaltungen verwerten bei mehrdeutigen Ergebnissen den größten Index.
7. Ein Demultiplexer schaltet einen Eingang auf einen von mehreren Ausgängen je nach den angelegten Steuerbits. Bei 1 zu 16 benötigt man 4 Eingänge (2^4 =16).
8. Der Taktgeber liefert ein diskretes alternierendes Signal. So können komplexe Vorgänge aufeinander abgestimmt werden (synchrone Schaltungen).
9. Ein RS-Latch kann mit 2 NOR-Gattern, bei denen die Ausgänge auf die Eingänge des anderen geschaltet sind, realisiert werden. Ein D Latch verhindert den Zustand R=S=1 mit einem Kontrolleingang C und einem Informationseingang D.
10. Bei *parallel load* werden alle Informationen/Bits gleichzeitig ins Register geladen.
11. Eine Speicherzelle (*Binary Cell*) besitzt einen select Eingang, einen read/write Eingang sowie den input Eingang und output Ausgang. Aufgebaut wird dies z.B. mit einem RS-Latch mit & Schaltelementen.
12. *Tristate-Outputs* benutzt man wenn mehrere Ausgänge zusammengeschaltet sind um Überlagerung von Informationen und Kurzschlüssen vorzubeugen denn es wird hier möglich Ausgänge abzuschalten (3.Zustand zu log. 1 und log. 0 -> tristate).
13. Es gibt:
    Statisches RAM: Read Only Memory aus Speicherzellen bestehend
    Dynamisches RAM: speichert mit Kondensatoren
    ROM: Read Only Memory Inhalt des Speichers kann nicht geändert werden
    PROM:Programmable ROM Speichers kann nur einmal „beschrieben“ werden
    EPROM: Erasable PROM Durch Bestrahlung kann wieder beschrieben werden
    EEPROM: Electrically EPROM Speicher kann elektrisch wieder beschrieben werden
    Flash-EPROM: ähnlich EEPROM, hier ist nur das löschen des ganzen Speichers möglich

Bei RAMs bleibt die Information nach Stromunterbrechung nicht erhalten, bei ROMs schon, jedoch sind die ROMs nur sehr begrenzt wieder beschreibbar und relativ langsam.

14. LCAs (*Logic Cell Arrays*) sind neuer als PLAs (*Programmable Logic Arrays*) und können adaptiert werden. PLDs (*Programmable Logic Devices*) sind ähnlich wie PLAs besitzen aber zusätzlich noch Latches.
15. VHDL bedeutet Very (High Speed Integrated Circuit) Hardware Description Language. Diese Entwurfssprache wurde in den 80ern vom Amerikanischen Department of Defense entwickelt um die Dokumentation und Kompatibilität von verschiedensten elektronischen Systemen zu verbessern.
16. Die Achsen des Diagramms sind Verhalten, Struktur und Geometrie welches einige der Hauptgesichtspunkte, die man bei der Schaltungsentwicklung berücksichtigt, ausdrückt.
17. Unter *entity* wird die Schnittstelle beschrieben, unter *architecture* ist eine Verhaltensbeschreibung und unter *configuration* steht die Angabe von Parametern.
18. Ein endlicher Automat hat eine begrenzte Anzahl an Zuständen, ein deterministischer Automat besitzt bei jedem Zustand für jede Eingabe nur genau eine relevante Kante.