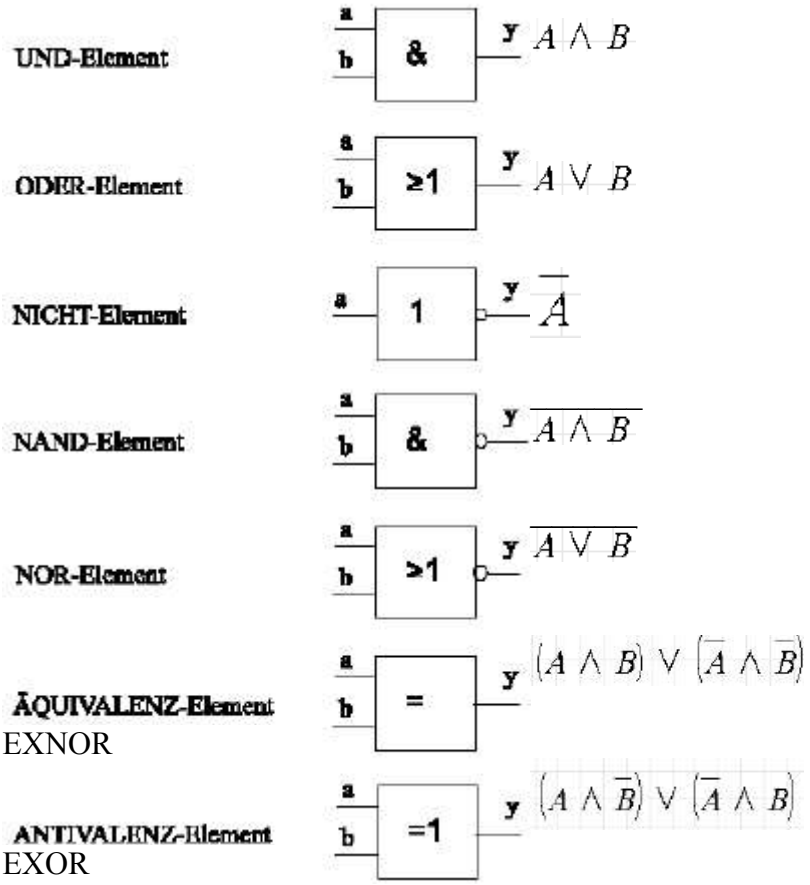
1.Schaltungstechnische Grundlagen

Definition Schaltnetz: Ein Schaltwerk (Funktionseinheit zum Verarbeiten von Schaltvariablen) , dessen Wert am Ausgang zu irgend einem Zeitpunkt nur vom Wert am Eingang zu diesem Zeitpunkt abhängt.
Bei der obigen Definition werden Laufzeiteffekte in Schaltnetzen (HAZARDS) nicht berücksichtigt.

1.1 Grundfunktionen und –elemente & Schaltalgebra



Rechenregeln:

$0 \wedge a = 0$ $\quad$ $a \vee a = a$
$1 \wedge a = a$ $\quad$ $\overline{a} \vee a = 1$
$a \wedge a = a$ $\quad$ $\overline{\overline{0}} = 0$
$\overline{a} \wedge a = 0$ $\quad$ $\overline{\overline{1}} = 1$
$0 \vee a = a$ $\quad$ $\overline{\overline{a}} = a$
$1 \vee a = 1$

$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

$\overline{a \vee b \vee c} = \overline{a} \wedge \overline{b} \wedge \overline{c}$

1.2 Vereinfachung von Schaltfunktionen

Definitionen: - Eine **Schaltfunktion** ist eine eindeutige Zuordnungsvorschrift, die jeder Wertekombination von Variablen einen Wert zuordnet.
- **Normalformen** beschreiben eine Schaltfunktion ausgehend von einer Wertetabelle in Gleichungsform
- **Minterme** (1) sind UND-Verknüpfungen, die alle Schaltvariablen einmal enthalten, wobei diese negiert oder nicht negiert vorkommen können.
- **Maxterme** (0) sind ODER-Verknüpfungen, die alle Schaltvariablen einmal enthalten, wobei diese negiert oder nicht negiert vorkommen können.
- **kanonisch disjunktive Normalform (KDNF)**
  alle Minterme UND-(disjunkt) verknüpft.
- **kanonisch konjunktive Normalform (KKNF)**
  alle Maxterme ODER- (konjunktiv) verknüpft.

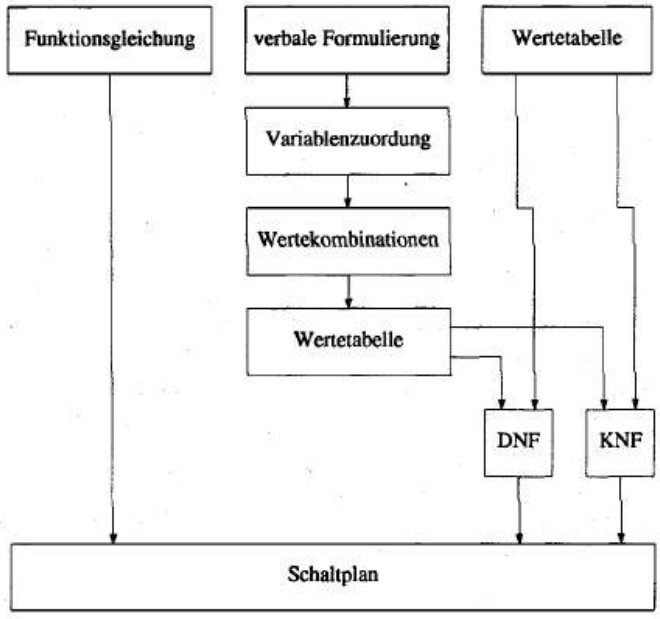
Es gibt 3 Verfahren, mit denen man Schaltfunktionen vereinfachen kann.
- die Gesetze der Schaltalgebra: sehr schnell bei wenig Variablen
- KV-Diagramme: schnelle, übersichtliche Methode bei bis zu 6 Variablen, bei mehr wird's unübersichtlich
- Quine/McCluskey Methode: langsamste Methode, man kann aber viele Variablen haben.
positive Logik: L=0, H=1; H dominiert
negative Logik: H=0, L=1; L dominiert
Wired Technik: Logikverknüpfungen ohne gatter
Arbeitstabelle: beinhaltet H,L und ist somit unabhängig von neg. bzw. pos. Logik
logische Aussage: KV, Wertetabelle, Schaltfunktion, Primimplikanten, Schaltnetz
Mit NAND und NOR lassen sich alle logischen Aussagen formulieren

Eingangslastfaktor **(Fan-in):**
FI = 1; d.h. eine "normale" Eingangsbelastung wird verursacht (von Schaltkreisfamilie abhängig)
Ausgangslastfaktor **(Fan-out):**
FQ gibt an, wie viele "normale" Eingänge maximal an einen Ausgang angeschlossen werden dürfen, damit keine undefinierten Pegel entstehen
(bei Standardbauelementen ist FQ üblicherweise 10, bei Leistungsgliedern 30)

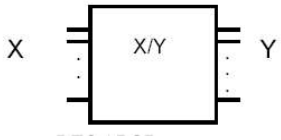
2.Schaltnetze (kombinatorische Logik)

Definitionen:
Ein **Schaltnetz** ist die techn. Realisierung einer Schaltfunktion der Schaltalgebra.
Eine **Funktionsgleichung** ist eine Gleichung der Schaltalgebra, die die Abhängigkeit einer Ausgangsvariablen von Eingangsvariablen beschreibt.
Eine **Wertetabelle** enthält eine Zusammenstellung aller möglichen Wertekombinationen der Eingangsvariablen und Ausgangsvariablen von Funktionsgleichungen.
Das **KV-Diagramm** ist eine zweidimensionale matrixförmige Anordnung der Wertetabelle.

In der **Synthese** wird aus einer gegebenen Aufgabenstellung der Schaltplan für ein Schaltnetz entworfen. 3 Typen von Aufgabenstellung: **verbale Formulierung, Funktionsgleichung, Wertetabelle.**
Ablauf einer Synthese:



Code-Umsetzer:



Ein **Codierer** ist eine Binärschaltung, die eine Menge von Eingangswerten in eine Menge von Ausgangswerten übersetzt.
Ein **Decodierer** ist ein Code-Umsetzer mit mehreren Eingängen und Ausgängen, bei denen für jede Kombination von Eingangssignalen immer nur je ein Ausgang ein Signal abgibt.

Gray-Code: Man schreibt die Zahl um eine Stelle nach rechts versetzt noch mal hin und macht eine EXOR Verbindung

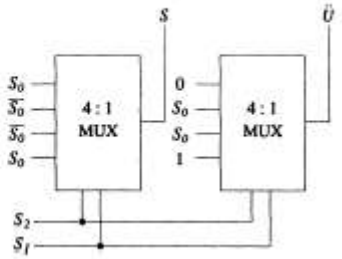
Paralleladdierer: # Der **Normalformparalleladdierer** ist ein Addiernetz mit minimaler Addierzeit aber maximalem Hardwareaufwand
Der **Ripple-Carry-Adder** ist ein Addiernetz, bei dem Addierzeit und Hardwareaufwand linear zur Stellenzahl n wachsen. (= **Serienaddierer**)
Der **Carry-Look-Ahead-Adder** ist Ein Kompromiss zwischen Normalformparalleladdierer und Ripple-Carry-Adder. Der Normalformparalleladdierer berechnet die Summen. Die Überträge werden im Carry-Look-Ahead-Generator bearbeitet

Multiplexer, Demultiplexer, Komparator

Volladdierer mit zwei 4-zu-1 Multiplexern

Multiplexer-Belegung für die Dateneingänge

Adresse S_2 S_1	MUX 1 (S)	MUX 2 ($\ddot{U}$)
0 0	S_0	0
0 1	$\overline{S_0}$	S_0
1 0	$\overline{S_0}$	S_0
1 1	S_0	1



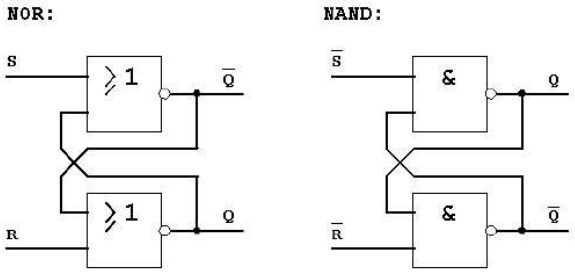
Realisierungsformen: # **ROM**: Durch Produktion festgelegte Funktion
PROM: Wenn eine Verbindung nicht erwünscht wird, dann eine Überspannung auf die Wort und Datenleitung. Für immer so
EPROM: Das gleiche wie PROM blos andere Bauart, kann also wieder glöscht werden.

	UND-Matrix	ODER-Matrix
ROM	fest	fest
PROM	fest	programmierbar
PAL	programmierbar	fest
PLA	programmierbar	fest

dynamischer HAZARD: Ist die Entstehung unerwünschter Signalflanken durch Laufzeiteffekte
statischer HAZARD: statischer hat immer eine Schwankung, dynamische mehrere. Ein dynamischer Hazard kann nur auftreten wenn vorher ein statischer da war.

3. Kippschaltungen als Speicherglieder

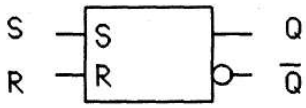
Basis-Flipflops:



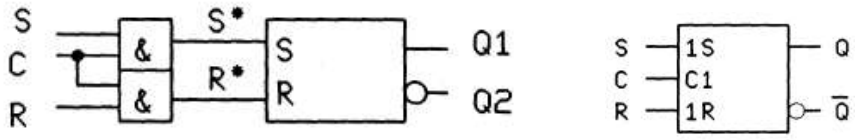
NOR-FF				NAND-FF		
S	R	Q^+	Funktion	$\overline{S}$	$\overline{R}$	Q^+
1	0	1	Setzen	0	1	1
0	1	0	Rücksetzen	1	0	0
0	0	Q	Speichern	1	1	Q
1	1	-	(verboten)	0	0	-

RS-FLIPFLOP

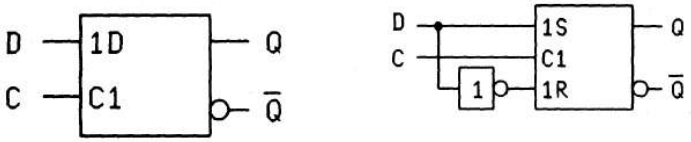
Schaltsymbol RS-FF



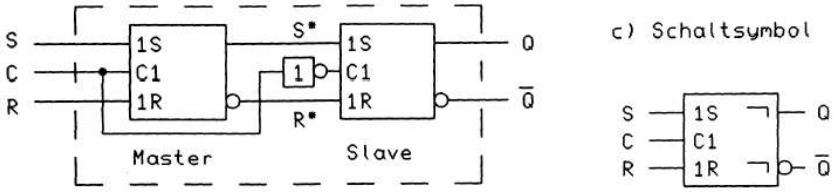
RS-Flipflop mit Zustandssteuerung (Taktsteuerung):



D-Flipflop (Das D-Flipflop mit Zustandsteuerung funktioniert nur dann richtig, wenn während der Taktimpulsdauer C=1 die Eingangsvariable D konstant bleibt)

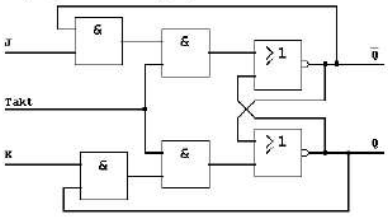


RS-kippglied mit Zwei-Zustandssteuerung (Die Ausgangsvariable nimmt den Wert 1 erst dann an, wenn die Eingangsvariable (Taktsignal C) wieder vom Wert 1 auf den Wert 0 zurückkehrt.)



JK-Flipflop

Zustandsgesteuertes RS-FF mit Rückkopplung der Ausgänge auf die Eingänge



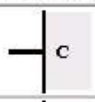
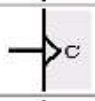
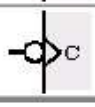
J	K	Q^+	Funktion
0	0	Q	Speichern
0	1	0	Rücksetzen
1	0	1	Setzen
1	1	$\overline{Q}$	Toggle

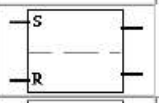
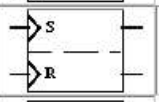
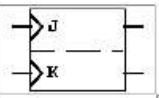
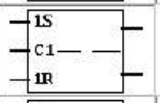
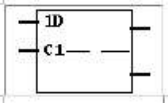
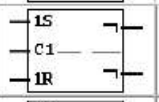
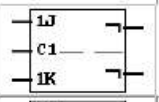
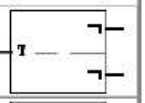
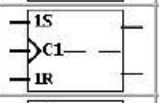
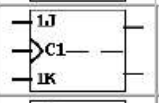
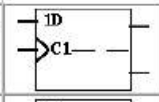
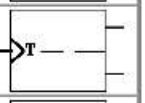
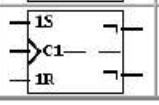
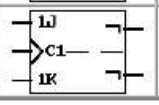
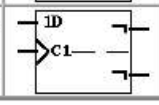
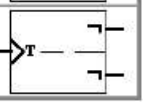
Vorteil: Dadurch, dass Q und Nicht Q immer komplementär sind, wird vermieden, dass der unzulässige Zustand R=S=1 eintreten kann.

T-Flipflop

Sieht ähnlich aus wie JK-Flipflop. Bloß dass der Eingang J = nicht Q ist, und K=Q. Es hat also keine Eingänge, ausser dem Takt. Es wechselt also bei jedem Takt den Ausgangszustand.

Taktflankensteuerung: Die Tfs (realisiert durch Verknüpfungsschaltungen) besteht darin, dass der Zwischenspeicher sofort nach der Signalübernahme über eine Rückkopplung seine Eingänge sperrt

Takt-Eingang	Symbol
Takt-Eingang mit Zustandssteuerung. Die Variablen an den Eingängen, die von C abhängen, werden bei C = 1 wirksam	
Takt-Eingang mit Flankensteuerung. Die Variablen an den Eingängen, die von C abhängen, werden beim 0-1-Übergang wirksam (positive Flanke).	
Takt-Eingang mit Flankensteuerung. Die Variablen an den Eingängen, die von C abhängen, werden beim 1-0-Übergang wirksam (negative Flanke).	

		Flipflop-Art			
Takt	Steuerung	RS	JK	D	T (datenlos)
ungetaktet	Zustands-Steuerung				
ungetaktet	Flanken-Steuerung				
getaktet	Einzustands-Steuerung				
getaktet	Zweizustands-Steuerung				
getaktet	Einflanken-Steuerung				
getaktet	Zweiflanken-Steuerung				

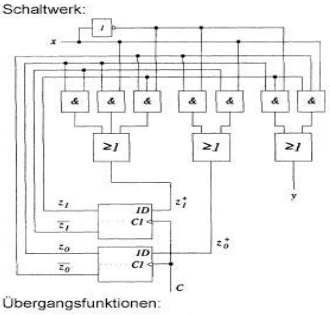
4. Schaltwerke

Schaltwerk: Eine Funktionseinheit zum Verarbeiten von Schaltvariablen, wobei der Wert am Ausgang zu einem bestimmten Zeitpunkt abhängt von den Werten am Eingang zu diesem und endlich vielen vorangegangenen Zeitpunkten.
Mealy-Automat: Y wird im Schaltnetz durch die Fkt: f(X, Z(tn)) gebildet.
Moore-Automat: Y wird nur von der Zustandsfkt. F(Z(tn)) gebildet.
Beschreibungsmöglichkeiten für Schaltwerke: Zustandsfolgetabellen; KV-Diagramme; Schalt- oder Vektorfunktionen (enhalten die Aus- und Übergangsfunktionen); Zustandsgraphen

Bsp:

.1 Analyse eines Schaltwerkes mit D-Flipflops

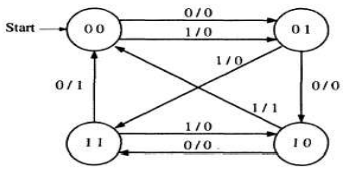
Schaltwerk:



Zustandsfolgetabelle:

z_1	z_0	x	z_1^+	z_0^+	y
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	0	0	1
1	1	1	1	0	0

Zustandsgraph:



Übergangsfunktionen:

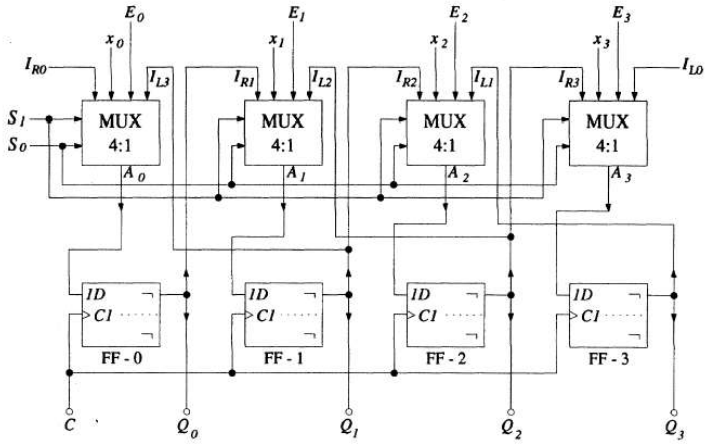
$$z_0^+ = (\overline{z}_0 \wedge \overline{x}) \vee (\overline{z}_1 \wedge x)$$
$$z_1^+ = (z_0 \wedge \overline{z}_1) \vee (z_0 \wedge x) \vee (\overline{z}_0 \wedge z_1 \wedge \overline{x})$$

Ausgabefunktion:

$$y = (z_0 \wedge z_1 \wedge \overline{x}) \vee (\overline{z}_0 \wedge z_1 \wedge x)$$

Synthese und Realisierung von Schaltwerken:
•= Festlegen der Zustandsmenge
•= Festlegen des Anfangszustandes
•= Definition der Eingangs- und Ausgabevariablen
•= Darstellung der zeitlichen Zustandsfolge in Form eines Zustandsgraphen
•= Aufstellen der Zustandsfolgetabelle
•= Erstellen der Übergangs- und Ausgabefunktion aus der Zustandsfolgetabelle
•= Darstellung des Schaltwerkes in einem Schaltwerk
•= Realisierung des Schaltwerkes

4 Bit Schieberegister mit Multiplexern realisiert:



S1=0; S0=1: x_0 bis x_3 dienen zum Löschen des Registers.
S1=0; S0=0: es wird nach rechts verschoben, d.h. I_{R0} bis I_{R3} werden auf ihre Ausgänge durchgeschaltet.
S1=1; S0=1: I_{L0} bis I_{L3} werden durchgeschaltet.→ Rechtsverschiebung
S1=1; S0=0: E_0 bis E_3 werden durchgeschaltet.

5. Rechnerarchitektur

Struktur:
Globale Systemebene
Anzahl der Prozessoren; Kopplung der Prozessoren; Realisierung des Hauptspeichers; Existenz von externen Cache;.....
Maschinenbefehlsatzebene
Anzahl und Größe der Operandenregister; Auswertbare Bedingungsregister;...
Mikroarchitekturebene
Prozessorinterner Cache; Memory Management Unit; Pipelining;.....

Organisation:
Globale Systemebene
Symmetrisches Multiprozessorsystem; hierarchische Master-Slave-Organisation;...
Maschinenbefehlsatzebene
Befehlsformat; Verzicht auf Befehle, die hohe Hardwarekomplexität erfordern;..
Mikroarchitekturebene
Aufteilung des Rechenwerkes in mehrere Funktionseinheiten; Branch-Target-Cache; Translation-Look-Aside-Buffer zur Beschleunigung der Adressübersetzung

Implementierungstechnik:
Globale Systemebene
Schaltkreis-, Verpackungs- und Kühltechnologie; Ein-/Ausgabepeipherie; Taktfrequenz; ...
Maschinenbefehlsatzebene
Delayed Branches; überlappende Registerfenster;..
Mikroarchitekturebene
Grad der Assozativität von Daten- bzw. Befehlscaches; Register zur Pufferung von Zwischenergebnissen bei Pipelining

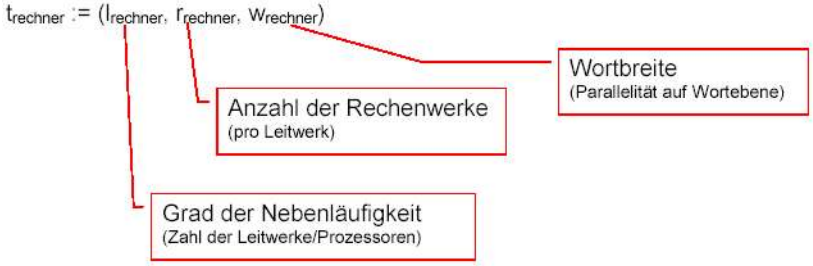
nü-Programmierung: Hardwarebaustein der Unterprogramme enthält.(statisch oder dynamisch)
Entwurfsziele von Befehlssätzen:
- •= Effizienz
  (Nutzunghäufigkeit, Verbesserung, Kosten)
- •= Orthogonalität
  (wenig Überschneidung, kombinierbar)
- •= Regularität
  (gleiche Adressierung, Symmetrie)
- •= Länge des Operationscodes
  (variabel, fest)
- •= Befehlsarten
  (Datenübertragung; Datenmanipulation, Verzweigungen; Maschinensteuerung)
- •= Adressierungsarten
  (implizit, unmittelbar, direkt indirekt, relativ)

typischer Prozessor:
IAR — Interrupt Address Register
MAR — Memory Address Register
MDR — Memory Data Register
IR — Instruction Register
PC — Program Counter

Ausführung der Befehle: Fünf Schritte (Holen, Dekodieren, Ausführen, auf Speicher zugreifen, Resultat schreiben)

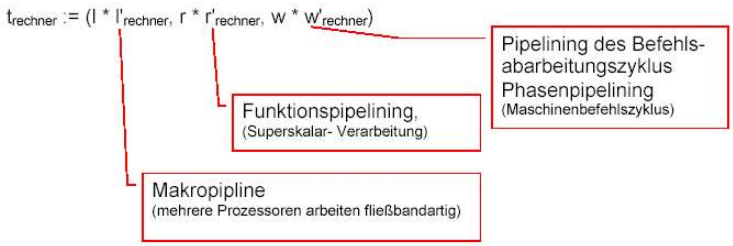
Rechnerklassifikation nach Flynn
SISD: Single Instruction Stream, Single Data Stream (1 Leitwerk, 1 Rechenwerk) (i486, M68040)
SIMD: Single I S, Multiple D S (1 Leitw., mehrere R.w.) (instruction broadcasting)
MISD: Multipl. I S, Single D S (praktisch keine Systeme)
MIMD: Multipl I S, Multipl. D S (mehrere Leitw., mehrere R.w.) (Multiprozessoren)

ESC-Klassifikation (Erlangen Classification System)
Zu beschreiben sind quantitative Eigenschaften (Nebenläufigkeit; Pipelining)
Zu betrachten sind 3 Ebenen:
- •= Leitwerksebene (Prozessor-, Programmebene)
- •= Rechenwerksebene (ALU-, Befehlsausführungsebene)
- •= Wortebene (Elemente der elementaren Verarbeitungsschritte)



t := (4, 1, 32)
4 Leitwerke mit je einem Rechenwerk und 32-Bit-Wortbreite
(entspricht einem Vier-32-Bit-Prozessorsystem)

noch zu berücksichtigen:
Pipelining, dh. Aufgabe einer Betrachtungsebene wird in mehrere Teilschritte unterteilt (nacheinander ausgeführt) und die Teilwerke arbeiten gleichzeitig.
(Im Idealfall ist bei n Teilwerken eine n-fache Beschleunigung der jeweiligen Aufgabe möglich)



z.B.:
superskalare Befehlsausführung: r' > 1, d.h. das Rechenwerk verfügt über mehrere Funktionseinheiten bzw. mehrere Befehle können gleichzeitig bearbeitet werden
(s. U- und V-Pipe von Pentium)

t_{i486} = (1, 1, 32 * 5) oder (1, 1, 32 * 5) + (0, 1, 80)

Grenzen: Speicherorganisation; Kommunikation; Softwarebezogene Aspekte

Complex Instruction Set Computer (CISC) (Prozessortyp)

Assemblerprogrammierung
mächtige Maschinenbefehle z.B. Gleitkommabefehle, komfortable Adressierungen
Kurze Maschinenprogramme
aufgrund umfangreicher Befehle und "optimaler" Befehlscodierung
Rechnerfamilien
einheitlicher Befehlssatz auf unterschiedlichen Plattformen mit unterschiedlichen Leistungsklassen
Parallelität
viele Operationen werden gleichzeitig prozessor-intern durchgeführt _ Entlastung des Systembusses (DMA, I/O)
Fehlererkennung
Fehlererkennung zur Laufzeit z.B. Unterscheidung zwischen Befehl und Daten; Überprüfung von Adress-Feld-Grenzen
Unterstützung von Betriebssystemen
z.B. Funktionen zur Speicherverwaltung als Maschinenbefehl
Compiler
effiziente Übersetzung von HLL aufgrund mächtiger Maschinenbefehle;
allerdings zeigt die Praxis: 60-80% der generierten Maschinenprogramme nutzen nur 5-10% der vorhandenen Maschinenbefehle
Mikroprogrammspeicher
komplexe Befehlssätze benötigen große Mikroprogrammspeicher auf Kosten von ALU, FPU, Cache,...
Hohe Flexibilität durch Schreib-/Lesespeicher – dynamisches Laden der µProgramme
Geschwindigkeit des Hauptspeichers
Ferritkernspeicher waren sehr langsam – mikroprogrammierte Algorithmen waren erheblich schneller; heute aufgrund von schnellen Cache-Speichern nicht mehr gültig
Zahl der Mikroschritte
komplexe Befehle erfordern eine große Anzahl von Mikroschritten _ Problem bei Interrupts

Reduced Instruction Set Computer (RISC) (Prozessortyp)

RISC Architekturen sind darauf aus, die Cycles Per Instruction zu minimieren. Das heißt, es wird versucht alle Befehle mit so wenig wie möglich Takten auszuführen.

Eine RISC-Architektur muss mindestens fünf der folgenden acht Kriterien erfüllen:
1. weniger als 50 Maschinenbefehle
2. weniger als 4 Adressierungsarten
3. weniger als 4 Befehlsformate
4. Speicherzugriffe nur über LOAD/STORE-Befehle
5. mehr als 32 Prozessorregister
6. festverdrahtete Maschinenbefehle (keine µProgrammierung)
7. Ausführung der "meisten" Befehle in einem Taktzyklus
8. höhere Programmiersprachen werden durch optimierende Compiler unterstützt

Hauptziel: 1 (oder mehr) Ergebnis pro Taktzyklus
→Befehls- bzw. Phasenpipelining
→Dreiadress-Maschine

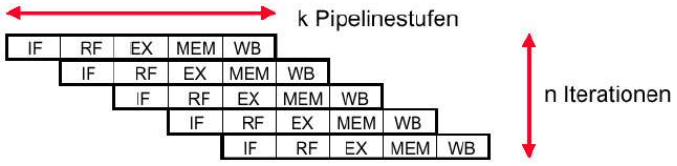
Gegenüberstellung:

Eigenschaften	CISC	RISC
Register	wenige (ca 20)	viele (bis zu 200)
Befehlssatz	ca. 300 Befehle und mehr als 50 Befehlstypen	nur rund 100 meist register-orientierte Befehle (außer LOAD/STORE)#
Adressierungsarten	ca 12 verschiedene	nur 3 bis 5 Arten; nur LOAD/STORE zum Speicher
Caches	gemeinsame Caches später auch getrennte	getrennte Daten und Befehls-Caches nach Harvard
CPI	1 bis 20 Schnitt = 4	1 bei Basisoperationen, Schnitt = 1,5
Befehlssteuerung	Mikrocode im Speicher	hartverdrahtete Mikroprogramme
Beipielprozessoren	Intel x86, AMD, Cyrix	Sun UltraSparc, PowerPC

Pipelining

$$Speedup = \frac{n \cdot k}{k+n-1}$$

$$Effizienz = \frac{Speedup}{k} = \frac{n}{k+n-1} \xrightarrow{n\to\infty} 1$$



Hindernisse: HAZARDS

1. Struktur Hazards: resultieren aus Recourcenkonflikten (z.B es kann nicht gleichzeitig ein Wort in den Speicher geschrieben werden und ein Befehl geholt werden. Abhilfe: NOP(Wartezyklus) oder Havard-Architektur.))
2. Daten Hazards: Befehl ist vom Ergebnis eines vorherigen Befehls abhängig.
3. Control Hazards: Bei Sprungbefehlen liegt Ziel noch nicht vor (Abhilfe: Branch Prediction)

Branch Prediction: Branch-Prediction-Buffer, Branch-Target-Buffer, Branch-Target-Cache
Delayed Branch: zur Behebung des Steuerfluß-Konflikts

Eigenschaften von Superskalar-Prozessoren:
→Mehrere parallel arbeitende Funktionseinheiten
→Register Renaming (logische Adresse _ physikalische Adresse)
→Warteschlangen für Funktionseinheiten z.B.: Integer, FP, Load/Store
→Registerblöcke
→Out-of-Order Abarbeitung Befehle _ µOPs;
→Spekulative Ausführung von Instruktionen mit Sprungvorhersagen

Benchmarks:
liefert Aussage über die Leistungsfähigkeit eines Systems
→ System-Benchmarks (mit E/A und Betriebssystemaufrufen)
→ CPU-Benchmarks (Leistungsmessung des Prozessor-Speicher-Systems)

zeitliche Lokalität: Auf ein gerade zugegriffenes Datum wird sicher bald wieder zugegriffen.
räumliche Lokalität: Auf Daten deren Adressen benachbart sind, wird mit hoher Wahrscheinlichkeit auch zugegriffen.
Cache-Arten:
1. Vollassoziativer Cache: aufwendige Hardware, weil bei Zugriff müssen alle Tags mit der anliegenden Adresse verglichen werden. Extrem schnell, aber teuer.
2. Direct-Mapped Cache: einfache, kostengünstige Integration, hohe Geschwindigkeit. Leider viele Konflikte, da mehrere Adressen auf die gleiche Cache-Zeile verweisen.
3. n-Wege-Satz Cache: mehrere parallel verknüpfte Direct-Mapped Caches. Vergleicht alle nTags gleichzeitig mit dem anliegenden Index. Reduziert Fehleranfälligkeit, braucht aber mehr Chipfläche.

Physischer Cache: liegt vor MMU und speichert nur physikalische Adressen
Logischer Cache: liegt zwischen CPU und MMU und speichert logische Adressen. Bei Hit enfällt die Adressumrechnung
Warum Zugriffszeit auf SRAM kleiner als bei DRAM? Keine Vorladezeiten, Auswertungszeiten der Messverstärker fallen weg.

Möglichkeiten Zugriffe zu beschleunigen:
Interleaving: Der Speicher wird in mehrere Bänke aufgeteilt und so angeschlossen, dass logisch benachbarte Speicherplätze physikalisch auf verschiedenen Speicherchips sitzen. Dadurch sich bei sequentiellem Zugriff ein Chip erholen, während auf den anderen zugegriffen wird. Mögl. 2-Weg, 4-Weg usw.
Page Mode: Sequentieller Zugriff auf die Bits einer Zeile, die Zeilenadresse bleibt stehen, nur die Spaltenadresse wechselt. Vorteil: keine Vorladezeiten zwischen den Zugriffen. Es sind bis zu 500 Zugriffe möglich, die Zykluszeit geht auf 30% zurück.
Fast Page Mode (FPM) : Der Inhalt der Schreib-/Leseverstärker wird erst bei Zeilenwechsel in die Zellen zurückgeschrieben
Enhanced Data Out (EDO): Das Ende des Lesevorgangs hier durch OE angezeigt. Dadurch wird CAS frei und kann während des Lesevorgangs schon eine neue Spaltenadresse übergeben werden: Zeitgewinn durch Überlapp.
Synchrone Data RAM, (SDRAM) Alle Signale hängen an einem gemeinsamen Takt, dadurch bessere Einbindung in das Gesamtsystem und keine Wartezeiten bei der Datenübergabe. SDRAMs arbeiten üblicherweise ebenfalls mit Burst-Reads. 8 ns-Zykluszeit bei Folgezugriffen im 125 MHz-Bustakt sind möglich. SDRAMs nutzen intern oft Interleaving.
Neuere Entwicklungen RDRAM, Rambus DRAM (intel) und SLDRAM Sync-Link DRAM, (Siemens) versprechen noch höhere Datenraten als SDRAM