

TGI, Technische Grundlagen der Informatik

- Moore'sches Gesetz:
- Anzahl Transistoren pro Chip $2 \times / 2 \text{ J}$
 - Verarbeitungsleistung Prozessoren $2 \times / 18 \text{ Mo}$
 - Für gleichen Preis, $2 \times$ Leistung in $< 2 \text{ Jah}$

Komponenten + Struktur = gewünschtes Verhalten
 \Rightarrow "Teile und herrsche!"

Entwurf + Analyse, Abstraktion

Top-Down, Bottom-Up

Rechnerarithmetik

Dezimal, Dual, Oktal, Hexadezimal
 $B = 10, 2, 8, 16$

Euklidische Algorithmus

sukzessiv, start mit höchster Ziffer, bei Komma einfach weiter 126

① finde höchste Potenz, die noch enthalten ist 2

② $y_i = z_i : b^i, R_i = z_i \bmod(b^i)$ (der Rest)

③ wiederhole bis fertig

$$\begin{aligned} 126 &: 2^6 = 1 \\ 62 &: 2^5 = 1 \\ 30 &: 2^4 = 1 \\ 14 &: 2^3 = 1 \\ 6 &: 2^2 = 1 \\ 2 &: 2^1 = 1 \\ 0 &: 2^0 = 0 \end{aligned}$$

Horner Schema

ganzzahliger/gebrochener Teil separat betrachten

① durch Basis b dividieren, der Rest ist die letzte Zahl

② weiter bis fertig, mit vorherigem Resultat

$$\begin{aligned} 99_{10} \\ 99 : 16 &= 6 \\ 6 : 16 &= 0 \\ 63_{16} \end{aligned}$$

Basis $b \rightarrow$ Dezimal

einfach $b^i * z$ aufsummieren $1001_2 \Rightarrow 2^0 \cdot 1 + 2^1 \cdot 0 + 2^2 \cdot 0 + 2^3 \cdot 1 = 9$

⚠ Spezialfall: Basis ist Potenz anderer Basis, einfach mehrere Stellen zusammennehmen oder umgekehrt

Zahlendarstellung

- Betrag + Vorzeichen (MSB für Vorzeichen (1=-))
* Sonderbetrachtung Vorzeichen-Bit + Zwei 0 (+0, -0)
- Einerkomplement (Stellenkomplement) ($z = 0100_2$, $-z = \sim(z) = 1011$)
* immernoch zwei 0 (+0, -0)
- Zweierkomplement (Einerkomplement, dann 1 addieren)
* unsymmetrischer Zahlenbereich $[-2^{n-1}, +2^{n-1}-1]$
- Offset-Dual-Darstellung (Exzess-Darstellung)
⇒ kleinste negative Zahl als 0...0, durch Addition von Offset

Festkomma

- fester Platz zugeordnet
- nicht mehr HW verwendet

VS

Gleitkomma

$$X = \pm \text{Mantisse} \cdot b^{\text{Exponent}}$$

↳ Normalisiert (erste 1 implizit)

▲ dichte auf dem Zahlenstrahl!

maxreal, minreal, smallreal

Schaltnetze

- rein kombinatorische logische Schaltungen
- kein Speicherverhalten

\wedge UND, \vee ODER, \neg NOT ⇒ vollständiges Operatorensystem

Booleschen Ausdruck, Interpretation, Aussage

Tautologie, Kontradiktion

⇒ Funktionstabelle, symbolische Form, Graph (Shannon-Baum)

KNF Konjunktive

DNF Disjunktive

Normal Form

MinTerm: \wedge Verknüpfung aller Variablen ($a=1, \bar{a}=0$)

MaxTerm: \vee Verknüpfung aller Variablen: negiert ($\bar{a}=1, a=0$)

KUF: 1 Verknüpfung der MaxTerme (Ergebnis 0)

DUF: \vee Verknüpfung der MinTerme (Ergebnis 1)

n-Variablen $\Rightarrow 2^n$ Min/Max-Terme

Shannonsche-Entwicklungssatz \Rightarrow siehe Beispiel

Minimalformen \rightarrow so kurze Ausdrücke wie möglich
 \rightarrow weniger Kosten für Realisierung Schaltung


DMF

KMF

NAAND/NOR-Konversion \rightarrow vollständige Operatorensysteme
 \rightarrow einfachere Realisierung Schaltung

Konversion durch Negation (doppelt) + De Morgan

Gatter: \Rightarrow AND \Rightarrow OR \Rightarrow NOT \Rightarrow XOR usw. 0 für Negation

 Entwurf: - Leistungsaufnahme - Platzbedarf
- Schaltzeit/verzögerung - Material
- Lebensdauer - Wärmeentwicklung
- Hasards/Wettläufe - begrenzte I/O
- Entwurfsaufwand / Kosten minimieren
- Realisierungskosten minimieren, wenig Gatter/Fläche
- Betriebskosten usw. minimieren

Technisch Kriterien
 \Downarrow KOMPR
 \Uparrow Ökon. Kriterien

KV-Diagramm \Rightarrow minimierung!

\Rightarrow Primimplikanten finden (grösstmögliche Blöcke 2^k)
 \Rightarrow so wenig wie möglich, grösste nehmen!

Multiplexer ≥ 2 Inputs, 1 Output, Steuerleitung, $2^n:1$

$\Rightarrow 2^n:1$ Multiplexer kann auch $n+1$ Variablen Funktionen implementieren (Wahl von einigen Vars als Steuersignale)

Demultiplexer 1 Input, ≥ 2 Outputs, $1:2^n$ (auch Dekoder)

festverdrahtete Logik VS mikroprogrammierte Logik
(Personalisierbar/Programmierbar)

Speicherbausteine

\rightarrow Adressierbar

ROM / PROM / EPROM / EEPROM
 Programmable Erasable Electrically

RAM \Rightarrow r/w, muss durch Spannung versorgt sein

DRAM / SRAM
dynamic static

- mem refresh
- Kondensator
- 1 Transistor
- no refresh
- Flip-flop
- 6-8 Transistoren

PLA (Programmable Logic Array)

FPLA (Field PLA), durch Benutzer programmierbar

PAL (Programmable Array Logic)

Laufzeiteffekte \Rightarrow zeitlicher Signalverlauf

- Totzeitmodell (Totzeitbaustein/glied hat Verzögerung)

Hasards \Rightarrow mehrfache Änderung Signal bis Stabilisierung

Eingabewechsel \Rightarrow Übergang \Rightarrow Ruhezustand

Hasardfehler \Rightarrow Hasard ABER Hasard \wedge Verzögerung \Rightarrow Hasardfehler

Funktionshasard / Strukturhasard / statisch/dynamisch

Schaltwerke (sequentielle Schaltungen)

Abhängig von vergangenen Werten \Rightarrow Zustand

\Rightarrow Automaten: Mealy, Moore

Moore (Ausgabewert hängt von Zustand ab)

Mealy (auch Eingabebelegung wird berücksichtigt)

\Rightarrow formale Beschreibung

- Zeitdiagramm

- Ablauftabelle

- Automatentabelle

- Automatengraph

Zustandsspeicher \Rightarrow Rückkopplung

Synchrone Schaltwerke (Takt) (Flanken/pegel-gesteuert)

Asynchrone Schaltwerke

(\downarrow wichtig wegen Taktverteilung)

(aber sehr störempfindlich) \rightarrow Normal Fundamental Mode

\downarrow
Latches

- asynchrones RS-FlipFlop
- pegelgesteuertes RS-Latch
- D-FlipFlop (D-Latch)
- MS-FlipFlop (Master-Slave, $2 \times$ D-Latch)
- JK-FlipFlop
- T-FlipFlop

Zustandskodierung

Synchrone Schaltwerke \rightarrow Serienaddieren

RT-Ebene \rightarrow Register-Transfer-Ebene

- Register zur Speicherung von Daten

- Funktionale Einheiten (Zähler, Addierer, ...)

- Datenwort
(Bitvektor)

Datenfluss, Blockdiagramm, VHDL

Register lineare Anordnung von Flipflops zur Speicherung mehrerer Bits (Bitvektor), gem. Takt, gem. Steuersig.

Schieberegister Kette von in Reihe geschalteten Register
shift (Div./Mult. durch/mit 2)
Umlaufspeicher / Ringzähler (I mit O verbunden)

Zähler Impulse abzählen, Adressieren aufeinanderfolgend, Frequenzteiler
Asynchrone Zähler (ripple counter), langsamer, simpler

Rechnerarithmetik: Addition und Subtraktion

Addition Grundlage, da Subtraktion \Rightarrow Addition negativer Zahl

Halbaddierer

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$s = (\bar{a} \wedge b) \vee (a \wedge \bar{b})$
 $c = a \wedge b$

Volladdierer zusätzlicher Eingang für Übertrag (Carry) der vorherigen Zahlenstelle

Carry-lookahead-Addierer Überträge direkt aus Inputs bestimmen, ^{VORTEIL} kleinere Verzögerung

Überlauferkennung: wenn $\text{CarryIn} \neq \text{CarryOut}$

Multiplikation

AddSub im Zweierkomplement praktisch, Vorzeichen muss nicht speziell beachtet werden

⇒ bei Multiplikation aber schon, kein Vorteil!

- Zahlen umwandeln, Vorzeichen/Betrag separat
- spezielle Algorithmen, z.B. Booth

Division analog Multiplikation

Teildivision: $0 \rightarrow \text{Divisor} > \text{aktueller Dividend}$

$1 \rightarrow \text{Divisor} \leq \text{aktueller Dividend}$

- Δ Division durch 0 Ausnahmezustand

- Multiplikations-Schaltung auch für Division brauchbar (teilweise)

ALU (Arithmetische-Logische Einheit)

In: Daten, Steuersignale Out: Resultat, Statussignale

Bitscheiben-ALU (für 4/8 bits), dann verkettet für größere Zahlen

von Neumann Architektur

- Rechner (CPU)
• Control Unit
• ALU
• Register

- Speicher (Memory)

- Ein/Ausgabe Komponente (Input/Output)

EVA-Prinzip Eingabe, Verarbeitung, Ausgabe

CPU, RAM, ROM, Busse/Schnittstellen, Chipsatz
↳ Datenleitungen

Register Daten, Adressen, IP=Instruction pointer, Stack, Befehle

⇒ Stack-Speicher (LIFO)

- Status/zähler
- parameter
- kurzzeitige Daten

Stack-Register

PUSH
POP

Maschinenbefehle ⇒ Assembler (symbolisch)

(Befehlsätze) ⇒ binär (OPCodes, ...)

Speicher-Adressierung Register, Ein-, Zweistufig

immediate, absolute, relative ← memory
register
program count

⇒ nicht-linear, Sprünge, Aufrufe, Threads, Interrupts

HW ↙ SW ↘

⇒ Exceptions (Fehler, Ausnahmesituationen)

↳ Interrupt System, Interrupt Service Routine, Interrupt Table

Speicher ⇒ RAM (random access!)

Zugriffszeit (access)
IW Addr → Data OUT

Zykluszeit (cycle)
IW Addr → Addr IW

Big endian (Network) VS

⇒ "natural" order

Little endian

⇒ inverted order

Massenspeicher ⇒ Magnetisch, Optisch, SSD

HD

CD

⇒ Speicherhierarchie Klein, schnell → gross, langsam

Cache, Virtueller Speicher

BIOS Firmware auf IBM-PCs: POST, HW Kommunikation, MBR geladen

BUS transport von Daten, mehrere Bits parallel

Datenbus, Adressbus, Steuerbus

⇒ Polling: Prozessor fragt regelmässig ab

⇒ IRQs (Interrupt Requests): Gerät sendet Prozessor Signale

⇒ BUS-Arbitern - zentrale Vergabe, fester Algorithmus

- Daisy-Chain, Kette, der Erste gewinnt

Übertragung Daten

- I/O Port bzw. Basisadressen (Programmed I/O)
 - vereinbarter Adressblock für Austausch Daten
- Memory Mapped I/O
 - Adressblock liegt im Arbeitsspeicher
- DMA Channels (Direct Memory Access), Bus-Mastering
 - Direkter Austausch Gerät ↔ Arbeitsspeicher, ohne durch die CPU zu gehen (viel schneller)

Erweiterungskarten

PCI, AGP, PCMCIA, PCI-X, PCI-Express, ISA

Laufwerkanschlüsse

EIDE, SCSI, Wide-SCSI, S-ATA, SAS

Geräteanschlüsse (Hot-Plugging)

USB (1.0, 1.1, 2.0, 3.0), IEEE-1394 (FireWire), PS/2

⇒ serial VS parallel Bus
↳ weniger Ströme, größere Distanzen

HD - Platten - Lese/Schreibköpfe - Spindel-Motoren

CD - Laser - Pits/Land - ~700M

DVD - 4.7, 8.5, 9.4, 17 GB - mehrere Standards

Technologieentwicklung

- Parallel arbeiten
- Pipelining, arbeit "verweben"
 - SISD (Single Instr. Single Data), senell
 - ⇒ SIMD (Single Instr. Multiple Data), parallel auf mehreren Daten
 - ⇒ MIMD (Multiple Instr. Multiple Data), mehrere Befehle parallel
- Mehrprozessorsysteme
- Feldrechner
- Funktionsspezialisierte Prozessoren (z.B. Crypto)
- Pipeline-Struktur
 - ⇒ Befehlspipeline
- Alternative Rechnerarchitekturen (z.B. Neuronale-Netz)