

Überblick

- Einleitung
 - Lit., Motivation, Geschichte, v. Neumann-Modell, VHDL
- Befehlsschnittstelle
- Mikroarchitektur
- Speicherarchitektur
- Ein-/Ausgabe
- Multiprozessorsysteme, ...

JR - RA - SS2002

Kap. 5

5/1

Kap. 5 Ein-Ausgabe

Ein-/Ausgabeeinheiten

- **Kommunikation** zwischen MP und Umwelt (auch Peripherie-Geräte):
 - Terminal (Tastatur und Bildschirm)
 - Drucker
 - externe Speichermedien (Diskette, Magnetplatte, ...)
 - Meßwerterfassungssysteme
 - ...

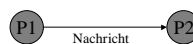
JR - RA - SS2002

Kap. 5

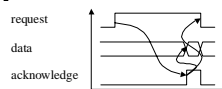
5/3

Kommunikation und Interface

- Kommunikation über Kanäle
 - Datentransfer zwischen Sender- und Empfängerprozess
 - hohe Abstraktionsebene



- Interfaces
 - Schaltungen, Kommunikationsprotokolle
 - niedrige Abstraktionsebene



JR - RA - SS2002

Kap. 5

5/4

Asynchrone/synchrone Kommunikation

- Asynchrone Kommunikation



- synchrone Kommunikation



JR - RA - SS2002

Kap. 5

5/5

Interface-Einheiten

- Peripherie-Geräte normalerweise *nicht direkt* an MP, sondern Interface-Schaltungen
 - **Datenpufferung**, wenn Datenübertragungsrate unterschiedlich
 - **Synchronisation** notwendig
 - **Datenkonvertierung** (A/D- und D/A-Wandlung oder seriell nach parallel)

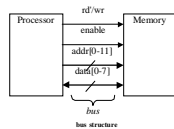
JR - RA - SS2002

Kap. 5

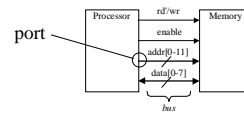
5/6

Kommunikation über Busse

- Drähte:
 - uni- oder bidirektional
 - eine Leitung kann mährerer Drähre repräsentieren
- Bus
 - ein Bündel von Drähnten mit festgelegter Funktion
 - address bus, data bus
 - festgelegtes Protokoll (Regeln für die Kommunikation)



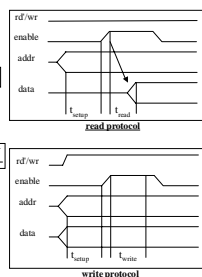
Ports



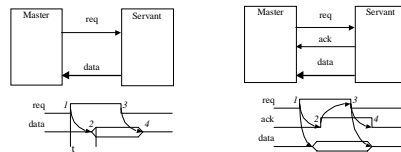
- Leitendes Medium zur Umwelt
- verbindet das System mit den Bussen
- wird oft als *pin* bezeichnet
 - Stift, der in eine Vertiefung auf der Leiterplatte versenkt wird
 - manchmal metallische Bälle anstatt der Stifte
 - bei SMD (surface mounted device) liegt eine Stift auf der Leiterplatte auf

Spezifikation mit Timing Diagrammen

- Gebräuchlichste Methode um Interface-Protokolle zu beschreiben
- Zeit vergeht in Richtung der X-Achse
- Kontrollsignale: low oder high
 - active high / active low
 - besser *assert* (aktiv) und *deassert*
- Datensignale: not valid oder valid
- ein Protokoll kann Unterprotokolle haben
 - diese werden auch als bus cycles bezeichnet, z.B., read und write
 - können sich über mehrere Taktzyklen erstrecken

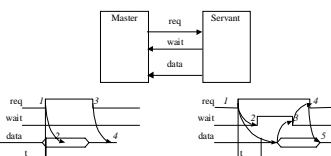


Grundlegende Protokoll Methoden



1. Master asserts *req* to receive data
 2. Servant puts data on bus **within time t_{access}**
 3. Master receives data and deasserts *req*
 4. Servant ready for next request
- Strobe protocol**
1. Master asserts *req* to receive data
 2. Servant puts data on bus **and asserts *ack***
 3. Master receives data and deasserts *req*
 4. Servant ready for next request
- Handshake protocol**

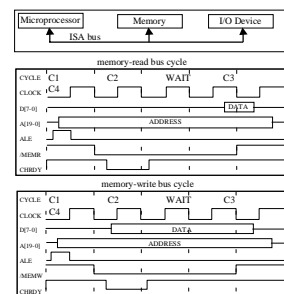
Kompromiss aus *handshake* und *strobe*



1. Master asserts *req* to receive data
 2. Servant puts data on bus **within time t_{access}**
 3. Master receives data and deasserts *req*
 4. Servant ready for next request
- Fast-response case**
1. Master asserts *req* to receive data
 2. Servant can't put data within t_{access} : **asserts *wait*** (*wait* line is unused)
 3. Servant puts data on bus and **deasserts *wait***
 4. Master receives data and deasserts *req*
 5. Servant ready for next request
- Slow-response case**

ISA bus protocol – memory access

- ISA: Industry Standard Architecture
 - benutzt in 80x86'er
- Eigenschaften
 - 20-Bit Adresse
 - Kompromiss *strobe/handshake*
 - Standard sind 4 Zyklen
 - außer wenn *CHRDY* deasserted, dann zusätzliche Zyklen (bis zu 6)



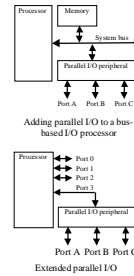
E/A-Adressierung

- Kommunikation mit anderen Komponenten über einige Pins
 - Port-based I/O (parallel I/O)
 - Prozessor hat mehrere N-Bit-Ports
 - Softwarezugriff auf Ports über spezielle Register/Speicherbereiche oder spezielle I/O-Befehle
 - Port spezifiziert direkt eine externe Komponente
 - Bus-based I/O
 - Prozessor hat Adress-, Daten- und Kontrollport, die zusammen einen Bus bilden
 - Protokoll im Prozessor integriert
 - eine einfache Operation führt das Lese-/Schreibprotokoll auf dem Bus durch
 - Angabe der Empfängeradresse

JR - RA - SS2002 Kap. 5 5/13

Kompromisse/Erweiterungen

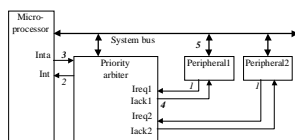
- Parallele E/A-Kommunikation
 - Prozessor unterstützt nur busbasierte Kommunikation, aber parallele Kommunikation benötigt
 - zusätzliche parallel arbeitende Peripheriegeräte am Bus
- Erweiterung der parallelen Ein-/Ausgabe
 - Prozessor unterstützt port-basierte E/A aber es werden mehr Ports benötigt
 - zusätzliche parallel arbeitende Peripheriegeräte an einem oder mehreren Ports



JR - RA - SS2002 Kap. 5 5/14

Arbitrierung mit Prioritäten

- Situation: Mehrere externe Komponenten wollen gleichzeitig den Bus benutzen. Wer darf zuerst arbeiten?
- Arbitrierung mit Prioritäten
 - externe Komponente fragt beim Arbitrer an, Arbitrer fragt beim Prozessor an, Arbitrer erteilt exklusiven Zugriff
 - Arbitrer ist nur für konfigurationszwecke am Systembus angeschlossen



JR - RA - SS2002 Kap. 5 5/15

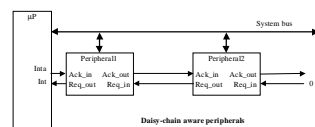
Arbitrierung mit Prioritäten

- Feste Priorität
 - jede Komponente hat einen eindeutigen "Rang"
 - bei gleichzeitigen Anfragen gewinnt die Komponente mit höchstem Rang
 - wird eingesetzt wenn sich diese Rangordnung statisch definieren läßt
- Rotierende Priorität (round-robin)
 - die höchste Priorität (token) wird der Reihe nach vergeben
 - bessere und gerechtere Verteilung wenn mehrere Komponenten gleiche Priorität haben

JR - RA - SS2002 Kap. 5 5/16

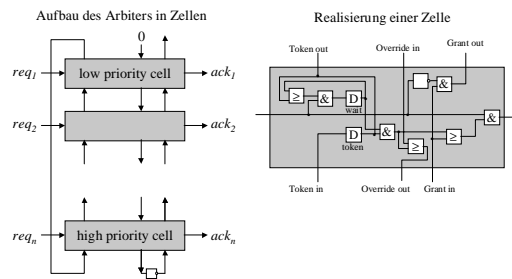
Daisy-chain Arbitrierung

- Arbitrierung in den externen Komponenten
 - In Komponente eingebaut oder durch extra Logik
 - jede Komponente hat req Eingang und ack Ausgang
- Komponenten werden in einer Kette verknüpft
 - eine Komponente ist mit der Resource verbunden
 - req „fließt“ zur Resource: "downstream"
 - ack „fließt“ zu den externen Komponenten: "upstream"
 - nächste Komponente hat höchste Priorität



JR - RA - SS2002 Kap. 5 5/17

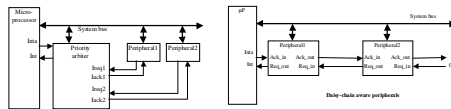
Daisy-chain Arbitrer mit priority + round-robin



JR - RA - SS2002 Kap. 5 5/18

Daisy-chain Arbitrierung: Pros/Cons

- Neue Komponenten ohne Redesign integrierbar
- defekte Komponente legt alle tieferen Komponenten lahm



JR - RA - SS2002

Kap. 5

5/19

Datenverkehr (1)

- Zwischen MP und Peripherie
- Drei Vorgehensweisen:
 - **programmierte Ein-/Ausgabe**
 - ┆ Programm steuert die Datenübertragung
 - **Interrupt**
 - ┆ externes Signal erzwingt Unterbrechung des laufenden Programms
 - ┆ Datenübertragung wird durch spezielle Routine ausgeführt
 - **DMA (=direct memory access)**
 - ┆ durch speziellen Prozessor wird separater Datenweg zwischen Speicher und Peripherie geschaffen
 - ┆ Zentraleinheit wird entlastet

JR - RA - SS2002

Kap. 5

5/20

Datenverkehr (2)

- Bei programmierter Ein-/Ausgabe und Interrupt:
 - ┆ besondere *E-/A-Tore (I/O-ports)*
 - ┆ enthalten Register zur Pufferung von Datenwörtern

JR - RA - SS2002

Kap. 5

5/21

Datenverkehr (3)

- Ansprechen durch MP:
 - ┆ **'memory mapped I/O'**-Methode
 - ┆ für I/O-port *ausgezeichnete, eindeutige Adresse* (nicht in RAM oder ROM)
 - ┆ Datentransfer vergleichbar mit Zugriff auf Speicherzelle
 - ┆ immer anwendbar, wenn MP *keine speziellen Befehle* für I/O hat

JR - RA - SS2002

Kap. 5

5/22

Datenverkehr (4)

- ┆ **'I/O mapped'**-Methode
 - ┆ Voraussetzung ist, daß MP über spezielle I/O-Befehle verfügt
 - z.B. IN <addr>, OUT <addr>
 - ┆ eigenen Adreßbereich für I/O-Geräte
 - ┆ eigene Steuersignale zur Unterscheidung
 - ┆ I/O-port mit Steuerbus verbunden
 - ┆ neben eigentlichen Daten transferieren von Synchronisationssignalen

JR - RA - SS2002

Kap. 5

5/23

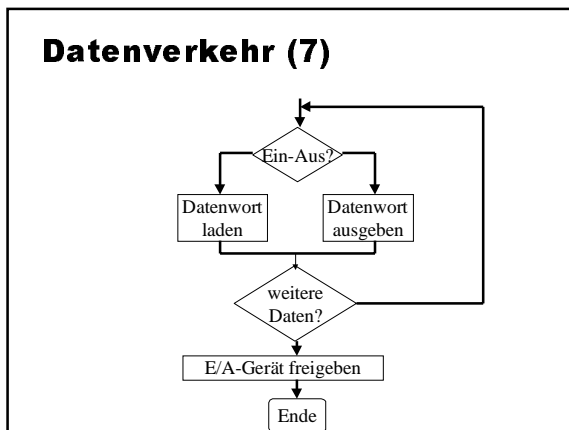
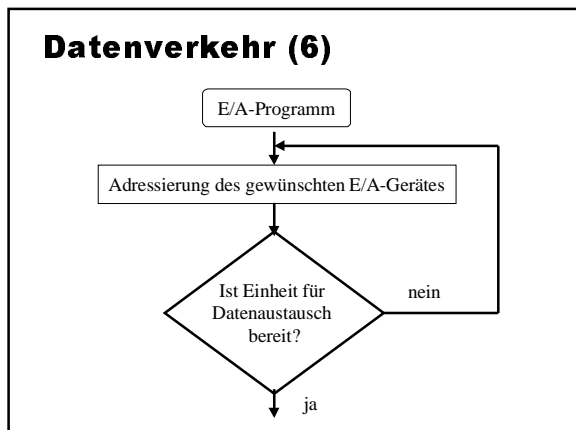
Datenverkehr (5)

- Spezielle E/A-Programme für programmierten Datenaustausch
- Programmgesteuerte Synchronisation
 - ┆ Anstoßverfahren (**strobing**)
 - ┆ Statusabstimmung (**polling**)
 - ┆ gegenseitige Abstimmung (**handshaking**)

JR - RA - SS2002

Kap. 5

5/24



Datenverkehr (8)

I Anstoßverfahren

- I bei *unidirektionaler* Übertragung
- I Übertragungsgeschwindigkeit des Empfängers **größer** als die des Senders
- I Sender gibt Datenwort aus
 - I 'strobe signal' an Empfänger
- I Sender verläßt sich darauf, daß Empfänger *immer* empfangsbereit

JR - RA - SS2002 Kap. 5 5/27

Datenverkehr (9)

I Statusabstimmung

- I Übertragungsgeschwindigkeit des Senders *größer* als die des Empfängers
- I Überprüfung des **polling signals** (signalisiert Bereitschaft des Empfängers)
- I falls empfangsbereit, Datenwort ausgeben (sonst warten)
- I muß permanent **polling signal** abfragen (**busy waiting**)

JR - RA - SS2002 Kap. 5 5/28

Datenverkehr (10)

I Gegenseitige Abstimmung

- I hohe Übertragungssicherheit
- I *Synchronisation* von Sender und Empfänger (polling **und** strobing)
- I Sender wartet bis Empfänger bereit
- I dann wartet Empfänger auf Sender
- I anschließend Datenaustausch

JR - RA - SS2002 Kap. 5 5/29

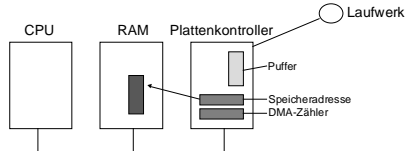
Datenverkehr (11)

I Interrupt

- I Bisher: Initiative geht von MP aus
- I Nachteil:
 - I busy waiting
- I MP reagiert auf Anfrage
- I MP unterbricht aktuelles Programm
- I Wichtig bei Interrupts:
 - I **Prioritäten** (Maskierung von Interrupts)
 - I **interrupt enable**
 - I **interrupt acknowledge**

JR - RA - SS2002 Kap. 5 5/30

Datenverkehr-DMA



■ DMA-Verfahren

■ 'ideale DMA'

■ Zweitortspeicher (dual ported RAM)

- sehr teuer
- Inkonsistenzen bei gleichzeitigen Schreibzugriffen

■ 'cycle stealing'-Verfahren

- DMA-Einrichtung hält MP an
- oftmals über Interrupt

JR - RA - SS2002

Kap. 5

5/31

Datenverkehr-DMA

■ 'memory idle'-Verfahren

■ Ablauf einer Befehlsverarbeitung

- Befehl holen
- Befehl ausführen

■ in 2. Phase greift MP **nicht** auf Speicher zu

■ nutze 2. Phase für DMA

■ Vorteil: keine Unterbrechung der CPU

■ zusätzliche Hardware:

- 'Speicher ungenutzt'-Signal von MP an DMA-Einheit

JR - RA - SS2002

Kap. 5

5/32

Datenverkehr

■ Datenübertragungsgeschwindigkeit

■ Einheit: Bit pro Sekunde (Bez.: **Baud**)

■ abhängig von Busbreite

- parallele Leitungen

■ 'beliebige' Erweiterung nur bedingt möglich

- z.B. 'pin limitation problem'

■ es gibt MPs bei denen Breite von Daten und Adressen *größer* als die des Datenbusses (z.B. 68000 oder 80386sx)

JR - RA - SS2002

Kap. 5

5/33

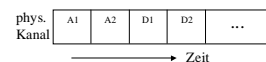
Datenverkehr-Zeitmultiplex

■ Beispiel

- Datenbusbreite n
- Datenwortbreite $n \cdot m$
- unterteile Datum in m Gruppen zu n Bits
- übertrage *nacheinander*
- einzelnen Gruppen müssen *eindeutig identifizierbar* sein

■ **zeitmultiplexe Übertragung**

- schmale Busse
- längere Übertragung



JR - RA - SS2002

Kap. 5

5/34