Eksamen 2012  
3. semester  
I3MPS - Microprocessor systemer  
Kasper Nissen

**Spørgsmål 3**

**Busser**

**Spørgsmål**

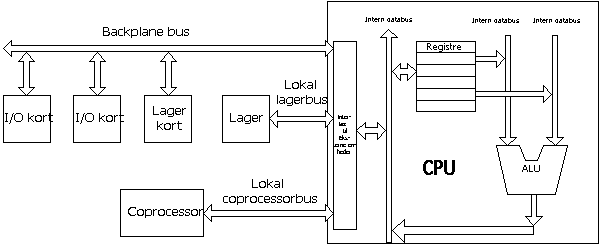
* Hvad er en bus?
* Hvilke signaler benytter man typisk på en parallel bus?
* Hvordan adresserer man på en bus?
* Hvilke ting skal konfigures på en host for at kunne tale med et devise på en bus?
* Beskriv et læse/skrive scenarie og de timing parametre som indgår.

**Pensum:**

* Structured Computer Organization, Tanenbaum: 174-189
* Simon 45-57

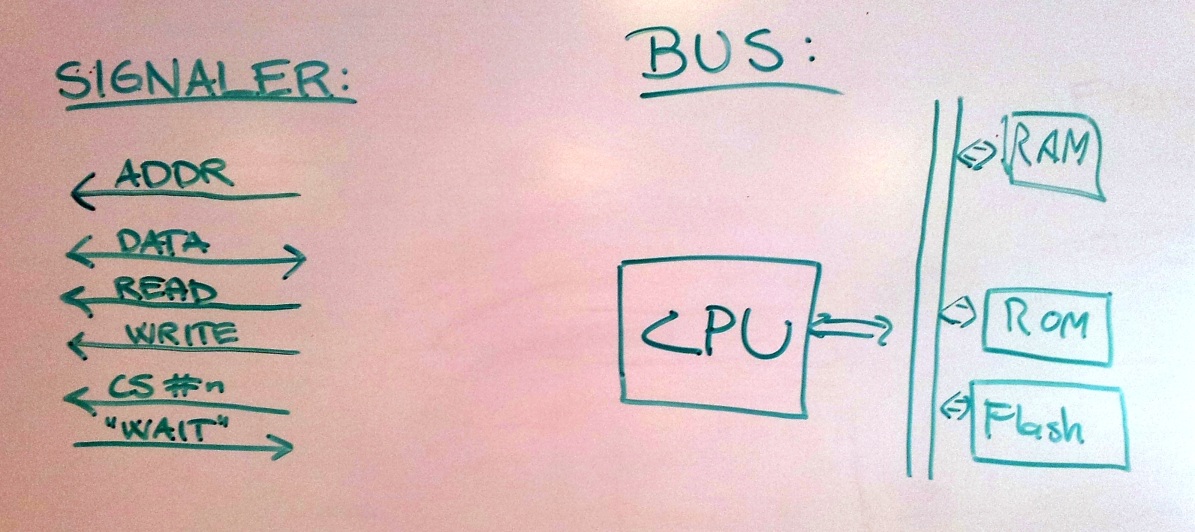
**Exercise 2**

**Hvad er en bus?**



Man kan sige en bus er navnet for den vej data bevæger sig mellem nogle enheder. Det er altså noget er forbinder f.eks. CPU’en med en lagerenhed. En bus kan også være et kabel som er forbundet mellem nogle enheder, hvor der i hver enden er en eller anden form for interface.

**Bus**

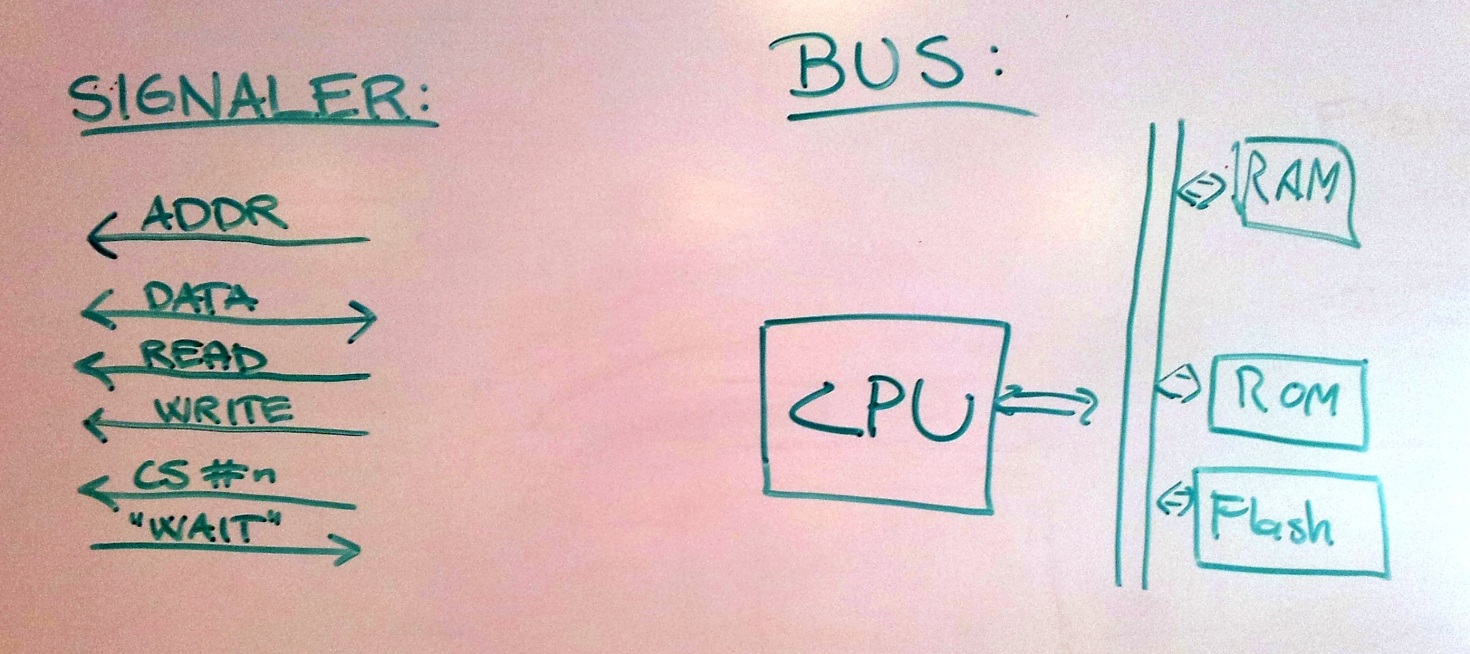
* En forbindelse mellem komponenter
* Både intern og ekstern
  + Internt
    - SPI, I2Cl
  + Ekstern
    - eSATA
    - USB(Universal Serial Bus)
* Internt i en chip
  + Mange parallelle forbindelser kan indgå
  + Høj båndbredde
  + F.eks. internt i CPU
* Busser bliver ofte delt mellem flere enheder.

Hvis vi kigger på en CPU, så kan dennes pins opdeles i 3 hovedgrupper:

* Adresse pins
* Data pins
* Control pins

Disse pins er forbundet til andre enheder der har lignende pins, det kunne være memory eller I/O enheder.Forbindelsen mellem disse enheder er en række af parallelle ledninger, som til sammen kaldes for en **parallelbus.**

Som det ses på figuren ovenfor er der både busser internt og eksternt.

**En Bus’ kontrol signaler**

Følgende signaler bruges til at kontroller bussen:

* RD (OE) / WR (WE)
  + Instructs device that this is a read / write access
* ALE / CLE
  + Address- / Command Latch Enable. For multiplexed busses
* BE0 / BE1
  + Byte Enable Low/High. Allows byte access on 16-bit interface
* WAIT / BUSY
  + Signal from device, requesting for time to prepare data
* CLK
  + Clock output for bus subsystem

**Et eksempel på hvordan CPU’en snakker med memory:**

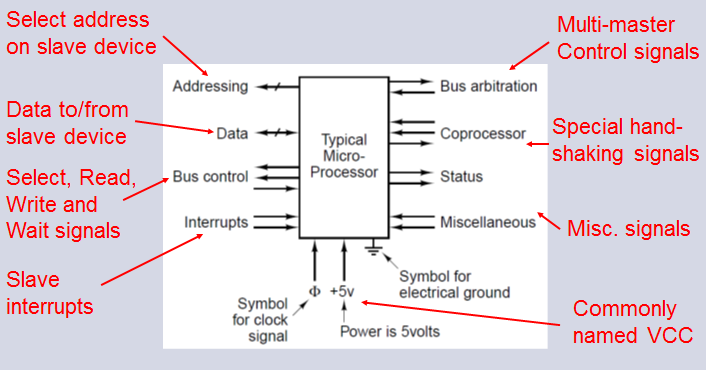
Hvis vi tager udgangspunkt i at vi har en CPU der skal hente en instruktion fra en plads i hukommelsen.

Først ligger CPU’en adressen på instruktionen ud på dens adresse pins, dernæst bruger den control pins til at signalere til memory at den gerne vil læse f.eks. et ord. Memory svarer ved at sende det forespurgte ord tilbage på CPU’ens datapins og signalere at nu er den færdig. Når CPU’en ser dette signal, acceptere den ordet på databusen og udfører instruktionen.

2 parametre som er vigtige i forhold til at vurdere en CPU, er antallet adresse og datapins. En chip med m adresse pins kan tilgå 2m hukommelses lokationer. Mest brugte værdier af m er 16,20,32 eller 64. Det samme ses med antallet af data pins, hvor en chip med n data pins kan skrive eller læse et n-bit ord i EN operation. Mest brugte værdier af n er 8,16,32, 36 og 64. Dette betyder altså at hvis vi har en CPU med 8 data pins, så vil det tage 4 operationer at læse et 32-bit ord, hvorimod havde den haft 32 datapins, var det kun nødvendigt med en operation. Denne chip vil selvfølgelig være meget hurtigere, dog vil den også være en del dyrer.

Kontrol pins bruges til at styre flow og timing af data til og fra CPU’en. Kontrol pins har dog også andre opgaver som:

* Interrupts
* Bus arbitration
* Coprocessor signaling
* Status
* Mm.



**Master / Slave**

Aktive enheder tilsluttet en bus kaldes for master enheder, mens passive enheder kaldes for slave enheder. Hvis f.eks. en CPU beordre en disk til at skrive eller læse en blok, arbejder CPU’en om master enhed, mens disken er slave enhed.Men denne disk kan senere arbejde som master, hvis den f.eks. skal tilgå noget hukommelse.

**Bustyper**

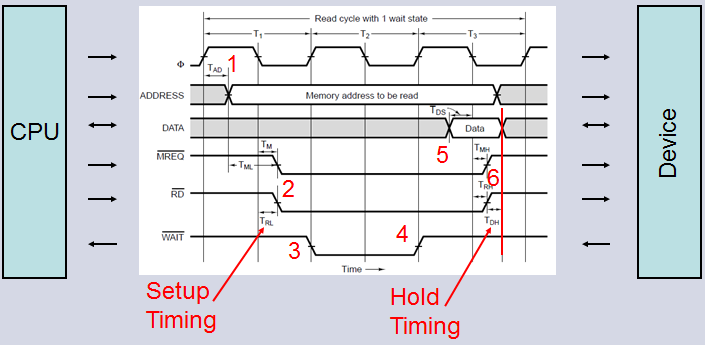
Der findes 2 hovedgrupper som man kan inddeler busser i efter måden de er clocket på, nemlig:

* Synkrone busser
* Asynkrone busser

En synkron bus har en clock drevet af en crystal oscillator som typisk består af et firkant signal med en frekvens på mellem 5 Mhz og 100 Mhz.Alle aktiviteter på den bus tager et integrale af tiden, denne tid kaldes for bus cycles.

En asynkron bus derimod har ikke master clock. Bus cycles kan derfor variere og kan derfor være lige så lange som nødvendigt.

Eksempel på synkron timing:



Eksempel på asynkron timing:



Bruges som regel sammen med langsomme enheder som flash-memory eller andre langsomme enheder.

Timingen konfigures af CPUen.

**Hvordan adressere man en på bus?**

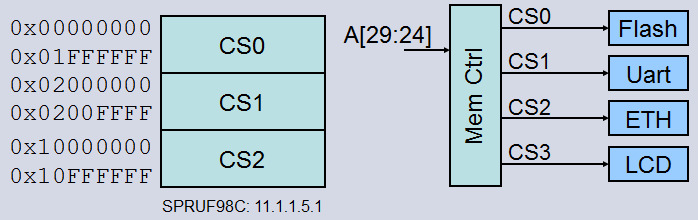
* Der benyttes en adresse bus
* For at kunne adressere f.eks. 64K memory skal der benyttes 16 adresselinier(216)
* READ fra host er forbundet til OE(output enable) på modtageren
* WRITE til WE(Write Enable)
* Host ved reelt ikke hvad der sidder på hvilke addresser

**Hvilke ting skal konfigures på en host for at kunne tale med et devise på en bus?**

* Adressen på bussen
* Chipselect til den specifikke enhed
* Read/write – rettigheder.
* Timing. Clock hastigheden skal være den samme på master og slave.

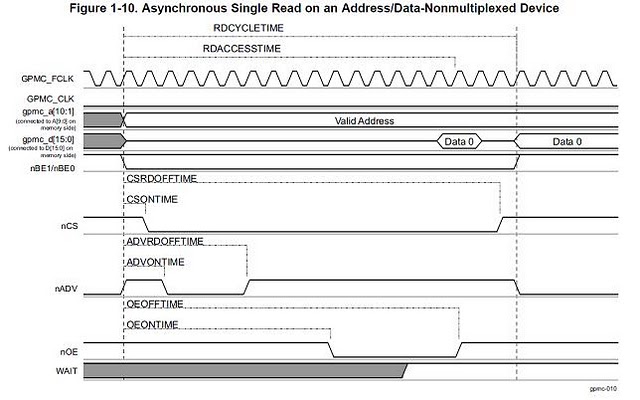
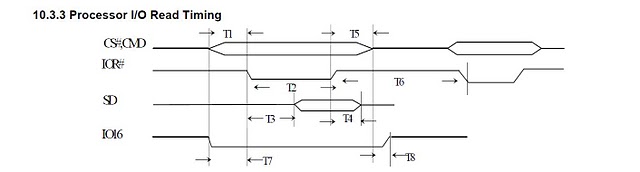
**Chip select**

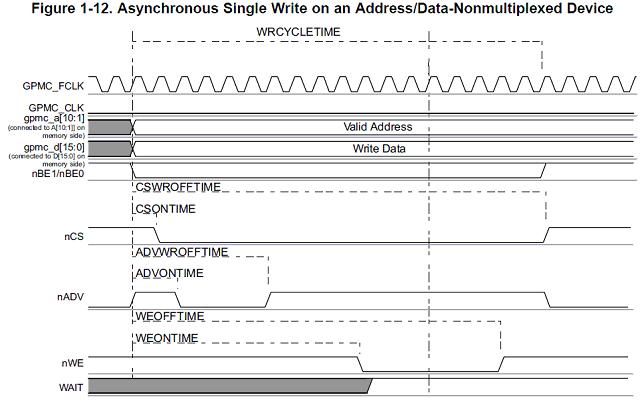
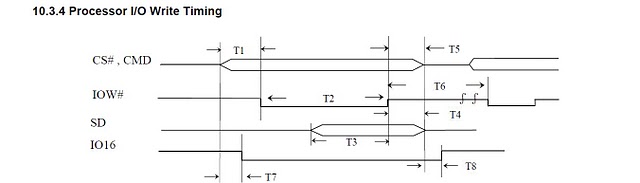
Chip select er måden hvorpå man vælger en specifik endhed på en bus.Hver enhed har en et antal adresse bits, som bruges til at vælge netop den enhed.



**Et eksempel på et læse/skrive scenarie og de timing parametre der indgår.**

**Kort beskrivelse af hvordan at OMAP og slave enheden skal forbindes. (Hvilke bus signaler? hvor mange data bits?)**  
  
  
OMAP og slaveenheden Ethernet controller(DM9000)  
Slaveenheden skal forbindes til OMAP vha. bussignalerne: D[0:15] som går til benene BD[0:15]  
Antal databits = 16  
  
Ved at kigge i databladet for DM9000 ses det at CS(Chip Select) skal være default aktivt lav for at aktivere enheden. Polariteten kan modificeres af EEPROM indstillingerne.

**Relevante timing diagrammer**  
  
  
*OMAP read timing*  
  
*DM9000 read timing*  


OMAP write timing  
  
  
*DM9000 write timing*  
  
  
  
  
**Beregninger af konstanter**  
  
Read-Cycle:

* CSONTIME
  + CSONTIME = 0, da CS#CMD ligger oven i hinanden jf. databladet for DM9000
* OEONTIME
  + OEONTIME > T1 > 0 ns => tOEONTIME = 10 ns => OEONTIME = 1 clk.  
    T1 skal ifølge datablad være min 0, og vælges derfor til 10 ns = 1 clk for en sikkerhedsskyld
* OEOFFTIME
  + OEOFFTIME > T1 + T2  
    OEOFFTIME > 10 ns + 10 ns > 20 ns eller 1 clk + 1clk > 2 clk derfor får vi  
    OEOFFTIME = 20 ns + 10 ns = 30 ns eller 2 clk + 1 clk = 3 clk.  
    T2 er opgivet i databladet til min. 10 ns.
* CSRDOFFTIME
  + CSRDOFFTIME > T1 + T2 + T5  
    CSRDOFFTIME > 30 ns + 10 ns > 40 ns eller 3 clk + 1 clk > 4 clk  
    CSRDOFFTIME = 40 ns + 10 ns = 50 ns eller 4 clk + 1 clk = 5 clk.  
    T5 skal ifølge datablad være min 0, og vælges derfor til 10 ns = 1 clk for en sikkerhedsskyld.
* RDACCESSTIME
  + RDACCESTIME = OEOFFTIME = 3 clk eller 30 ns.
* RDCYCLETIME
  + RDCYCLETIME > CSRDOFFTIME
  + RDCYCLETIME = 50 clk + 1 clk = 60 clk
* CYCLE2CYCLEDELAY
  + CYCLE2CYCLEDELAY = T6 - T5 - T1  
    CYCLE2CYCLEDELAY =   
    4 clk - 1 clk - 1 clk = 2 clk

Write-Cycle:

* WEONTIME
  + WEONTIME > T1  
    WEONTIME > 0 => 10 ns eller 1 clk
* WEOFFTIME
  + WEOFFTIME > T1 + T2
  + WEOFFTIME > 10 ns + 10 ns > 20 ns
  + WEOFFTIME = 30 ns
* CSWROFFTIME
  + CSWROFFTIME > T1 + T2 + T5
  + CSWROFFTIME > 30 + 10 ns > 40 ns
  + CSWROFFTIME = 40 ns + 10 ns = 50 ns
* WRCYCLETIME
  + WRCYCLETIME > CSWROFFTIME  
    WRCYCLETIME = 50 ns + 10 ns = 60 ns
* CYCLE2CYCLEDELAY
  + CYCLE2CYCLEDELAY > T6 - T5 - T1  
    CYCLE2CYCLEDELAY > 20 ns - 10 ns - 10 ns > 0 ns   
    Da CYCLE2CYCLEDELAY giver 0, kan den næste cycle ikke startes med det samme, hvorfor derfor være større end 0.   
    CYCLE2CYCLEDELAY = 10 ns