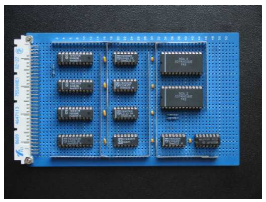
Digital- och datorteknik



Föreläsning #22

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

Bussar i datorsystem

Översikt

Datorsystemet är uppbyggd av enheter som utbyter information med varandra via elektriska ledare. En grupp av sådana ledare kallas buss.

- Enheterna utgörs av centralenheten (CPU), minnesenheter och periferikretsar (gränssnitt mot yttre enheter).
- Informationsutbyte görs normalt via tre bussar: adressbuss, databuss och styrbuss.
- I normalfallet generar endast centralenheten adresser på adressbussen. Däremot kan centralenhet, minnesenheter och periferikretsar placera data på databussen samt genera signaler på styrbussen.

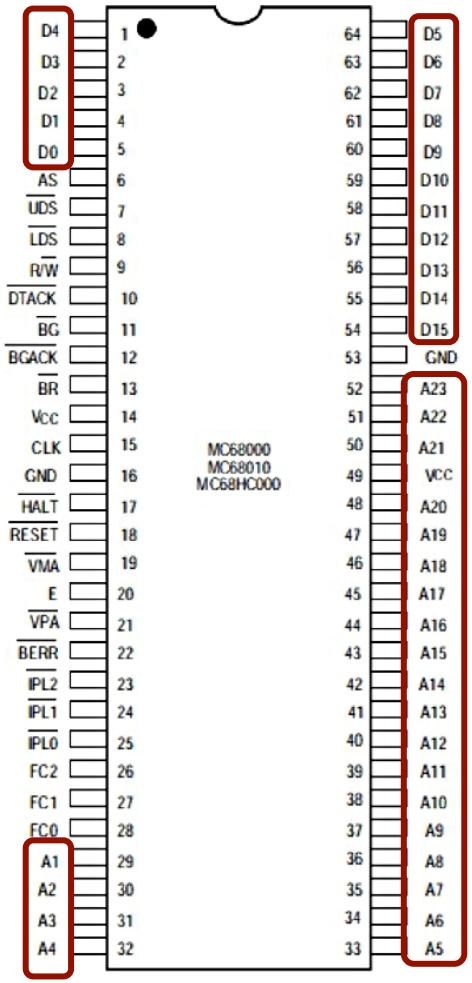
Bussar i datorsystem

Traditionellt buss-system

I traditionella system brukar samtliga bitar på data- och adressbuss vara tillgängliga på centralenhetens kapselpinnar.

Då kan data- och adressbuss anslutas direkt till minnen och periferikretsar.

Exempel: Motorola 68000 med 24 bitars adressbuss och 16 bitars databuss.



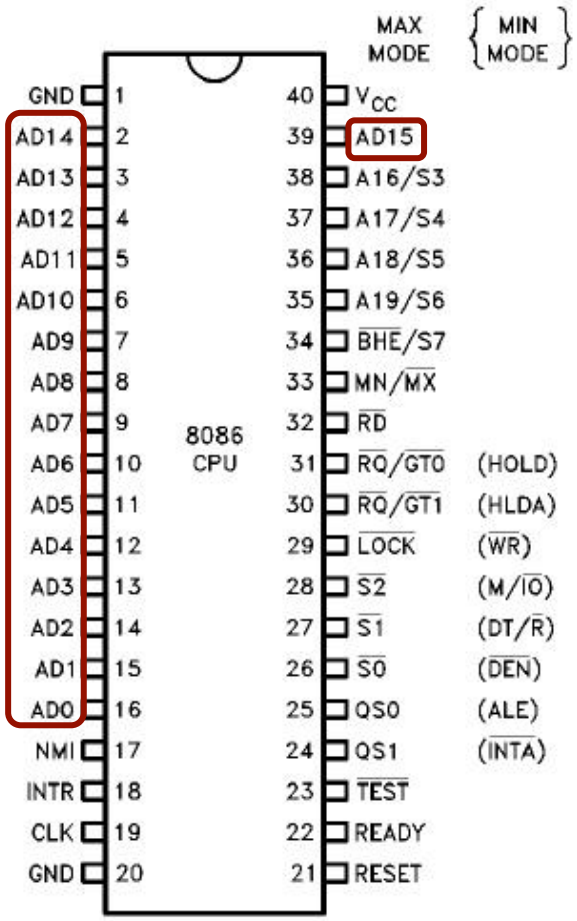
Bussar i datorsystem

System med multiplexade bussar

I andra system kan antalet pinnar på kapseln vara begränsat, vilket gör att det är nödvändigt att tidsdela data- och adressbuss på gemensamma pinnar.

I dessa fall kommer styrbussen att innehålla speciella signaler som måste användas för att låsa data- respektive adressbitar i separata externa register.

Exempel: Intel 8086 med 16 bitars adressbuss och 16 bitars databuss.

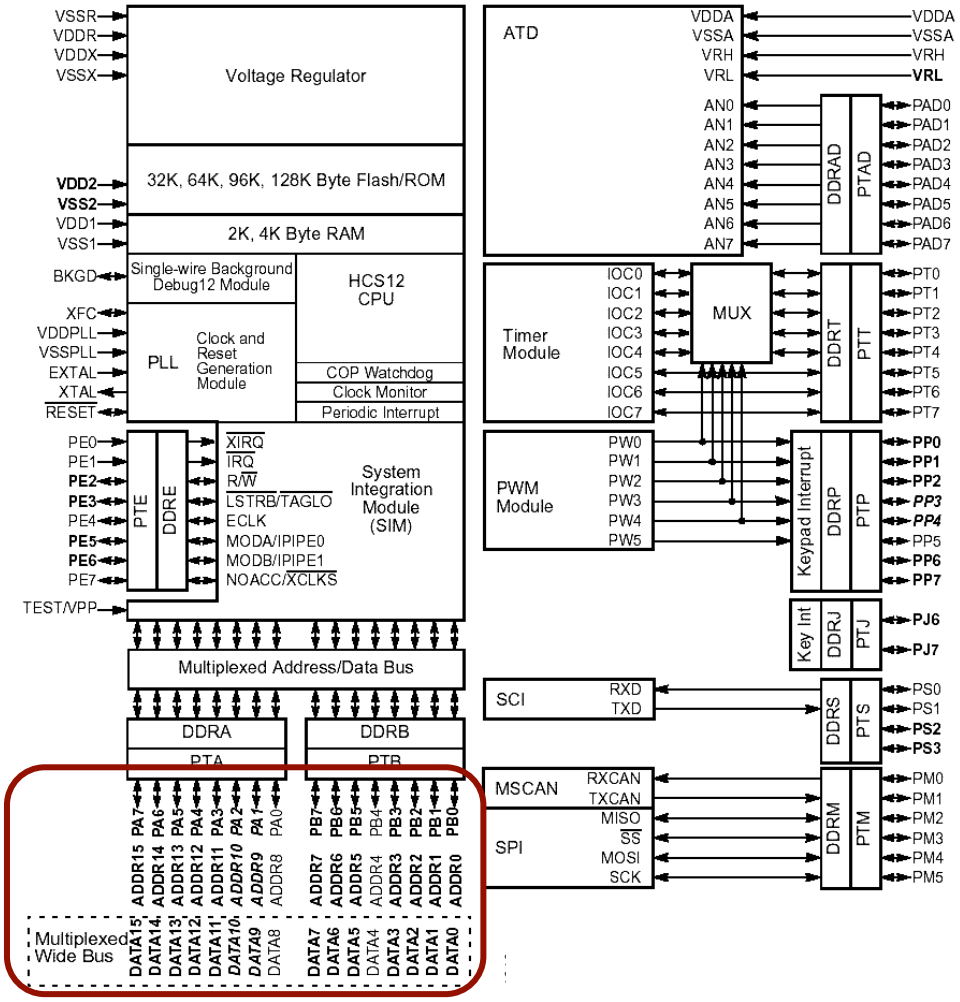


Bussar i datorsystem

Microcontrollers

I vissa fall integreras en centralenhet med såväl minnen som periferikretsar i samma kapsel. Då brukar data- och adressbuss samt vissa periferikretsar dela på några av kapselns pinnar.

Exempel: Motorola 9S12 microcontroller, med HCS12 CPU.



Bussar i datorsystem

Seriebussar

Av utrymmesskäl kan det vara nödvändigt att begränsa antalet pinnar på en minneskapsel till ett minimum. Då måste data- och adressbitar tidsdelas på en seriebuss, d v s överföras bit för bit på en enstaka ledning med hjälp av en klocksignal.

Exempel: 512 Kbit SPI Bus Serial EEPROM.

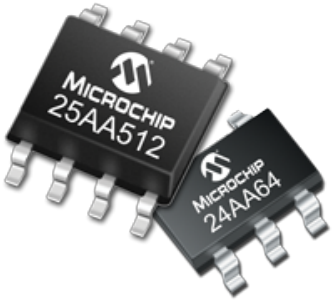
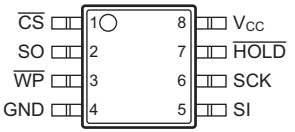


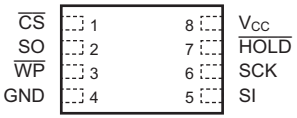
Table 1. Pin Configurations

Pin Name	Function
$\overline{CS}$	Chip Select
GND	Ground
$\overline{HOLD}$	Suspends Serial Input
SCK	Serial Data Clock
SO	Serial Data Output
SI	Serial Data Input
$\overline{WP}$	Write Protect
V_{CC}	Power Supply

8-lead SOIC
(Top View)



8-pad UDFN/XDFN
(Top View)



Bussar i datorsystem

Adressbussen

Antalet bitar på adressbussen bestämmer systemets adressrum.

- Med 16 bitar omfattar adressrummet 2^{16} = 64 kbyte
- Med 32 bitar omfattar adressrummet 2^{32} = 4 Gbyte
- Med 64 bitar omfattar adressrummet 2^{64} = 18 exabyte

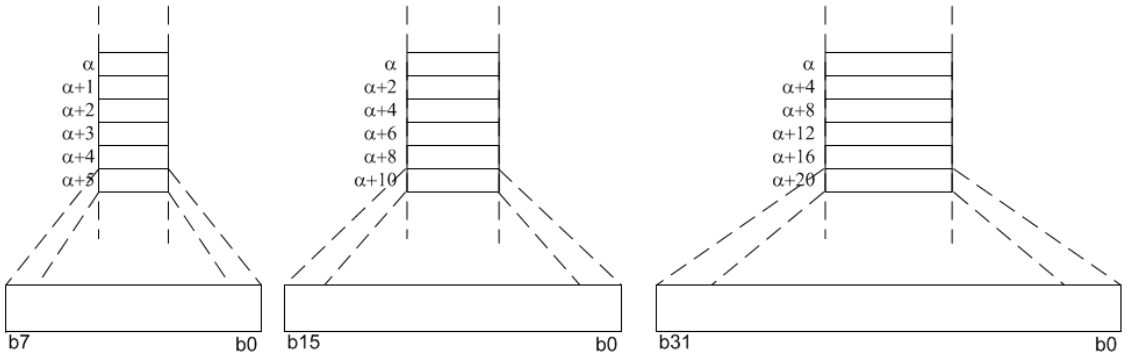
Med större adressrum blir det mer sannolikt att inte alla adresser är anslutna till en minnesenhet. Om processorn gör ett försök att hämta en instruktion eller data från en adress där varken minne eller periferikrets finns ansluten kan detta detekteras via signaler på styrbussen. I sådana fall kan ett undantag (”address error”) genereras hos processorn.

Bussar i datorsystem

Databussen

Antalet bitar på databussen följer oftast adressbussens bredd. Med 16 eller fler bitar på databussen kan man därför arbeta med flera bytes parallellt och därmed öka systemets prestanda.

Att arbeta med flera bytes parallellt brukar dock kräva vissa restriktioner. T ex krävs att läsning och skrivning av 32-bitars dataord bara sker på adresser som slutar med en multipel av 4. I annat fall kan ett undantag (”bus error”) genereras.



Bussar i datorsystem

Styrbussen

Exempel på kontrollsignaler som genereras på styrbussen:

- Skriv- och lässignaler (MR, MW).
- Signal som indikerar att informationen på data- och adressbuss är giltiga (VA, ”valid address”).
- Handskakningssignal vid asynkron busskommunikation, som indikerar om adressen på adressbussen är ansluten till en minnesenhet (ACK).
- Signal för hantering av multiplexad data- och adressbuss.
- Selektorsignaler (”chip select”) för minnes- och periferikretsar, som genereras från grindnätet för adressavkodning.

Bussar i datorsystem

Bussprotokoll

Ett protokoll är en överenskommelse (konvention) mellan olika parter i en kommunikation. För bussar i ett datorsystem kan ett sådant protokoll omfatta:

- Vilka logiknivåer som gäller på bussarna

  T ex Aktiv hög = logisk ”1” betyder ”sann”
  Aktiv låg = logisk ”0” betyder ”sann”

- Hur bitar organiseras i en byte (eller i ett ord)
- Hur bytes organiseras i ett ord

  ”Little-endian byte order” respektive ”big-endian byte order”

- Tidshändelser

  Asynkron respektive synkron busskommunikation.

Bussprotokoll

Organisation av bitar i en byte

Fakta:

- Mest signifikanta bit: bitpositionen i ett binärtal som är mest värd vid binär evaluering av talet

Några viktiga frågor:

- I vilken ordning skriver vi (på papper eller i ett dokument) de bitar som ingår i ett binärtal?
- I vilken ordning skickar vi de bitar som ingår i ett binärtal vid seriell överföring (t ex till ett EEPROM eller i ett nätverk)?

Bussprotokoll

Organisation av bitar i en byte

Fakta:

- Mest signifikanta bit: bitpositionen i ett binärtal som är mest värd vid binär evaluering av talet

Vanligaste konvention:

- Mest signifikanta bit skrivs längst till vänster
- Mest signifikanta bit skickas först vid seriell överföring
- Minst signifikanta bit i ett binärtal benämns *bit 0*.
- Mest signifikanta bit i ett n-bitars tal benämns *bit n-1*

Bussprotokoll

Organisation av bytes i ett ord

Fakta:

- Mest signifikanta byte: de 8 bitar i ett 16-, 32- eller 64-bitars binärtal som är mest värda vid binär evaluering av talet

Några viktiga frågor:

- I vilken ordning skriver vi (på papper eller i ett dokument) de bytes som ligger lagrade i ett minne?
- I vilken ordning skriver vi (på papper eller i ett dokument) de bytes som ingår i ett 16-/32-/64-bitars binärt tal?
- I vilken ordning skickar vi de bytes som ingår i ett 16-/32-/64-bitars binärt tal vid seriell överföring (t ex i ett nätverk)?

Bussprotokoll

Organisation av bytes i ett ord

Fakta:

- Mest signifikanta byte: de 8 bitar i ett 16-, 32- eller 64-bitars binärtal som är mest värda vid binär evaluering av talet

Vanligaste konvention:

- Bytes i ett minne skrivs ut från vänster till höger med ökande minnesadresser
- Mest signifikanta byte skrivs längst till vänster
- Mest signifikanta byte skickas först vid seriell överföring (konventionen kallas ”network byte order”)

Bussprotokoll

Organisation av bytes i ett ord

Givet att moderna processorer arbetar med ordlängder större än 8 bitar måste vi nu ställa oss följande fråga:

- Vilket är det lämpligaste sättet att lagra ett 16-/32-/64-bitars binärt tal i en minnesenhet?

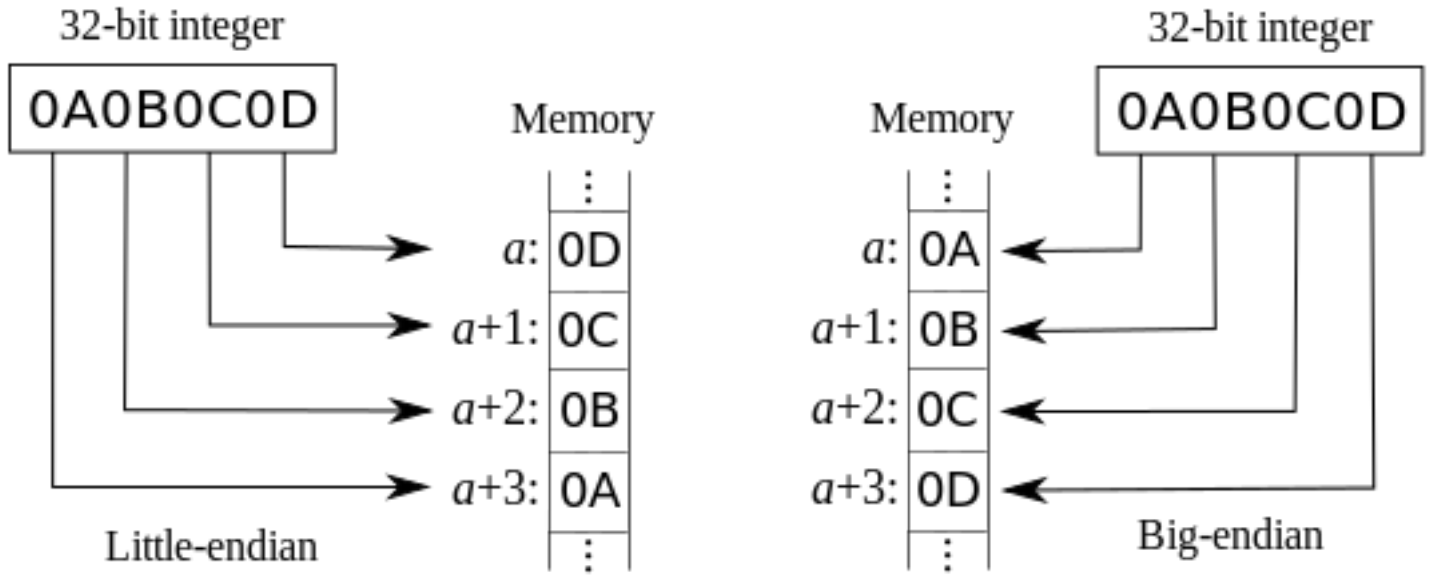
Vanligaste konventioner:

- Minst signifikanta byte lagras på lägre minnesadress
  - Detta kallas för ”little-endian byte order”
  - Har använts i processorer från t ex Intel och Digital
- Mest signifikanta byte lagras på lägre minnesadress
  - Detta kallas för ”big-endian byte order”
  - Har använts i processorer från t ex Motorola och Sparc

Bussprotokoll

Organisation av bytes i ett ord

Exempel: Lagring av talet $0A0B0C0D_{16}$ enligt konventionerna för little-endian respektive big-endian byte order:



Bussprotokoll

Organisation av bytes i ett ord

Kommentarer:

- Utskrift av minnesinnehåll matchar *big-endian*-konventionen
- Överföring av 16-/32-/64-bitars binärtal i nätverk (”network byte order”) matchar *big-endian*-konventionen

  Vid kommunikation mellan datorer som tillämpar olika konventioner måste alltså ordningen på de bytes som ingår i binärtalen kastas om vid sändning och mottagning i den dator som inte arbetar enligt *big-endian*-konventionen.
- Det finns processorer som kan konfigureras till att arbeta som antingen *big-endian* eller *little-endian*

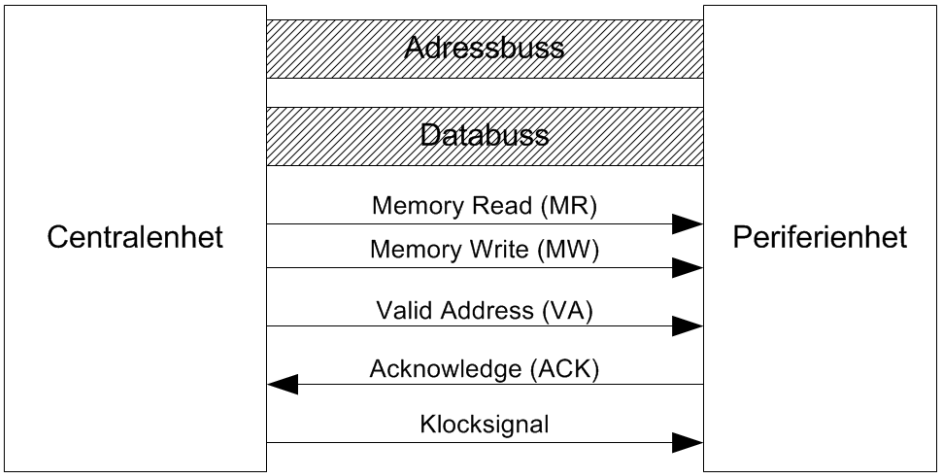
  Exempel: processorer från ARM- och PowerPC-familjerna

Bussprotokoll

Asynkron buss

Asynkron busskommunikation är den mest flexibla varianten av bussprotokoll, i den meningen att man kan ansluta ett stort utbud av minnesenheter och periferikretsar till centralenheten.

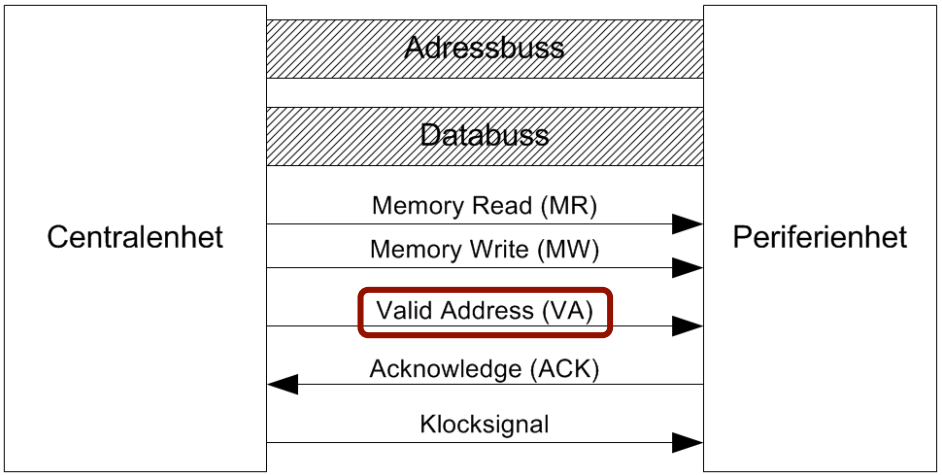
Speciellt användbar är asynkrona bussar om man har olika typer av minnen med stor skillnad i åtkomsttid.



Bussprotokoll

Asynkron buss

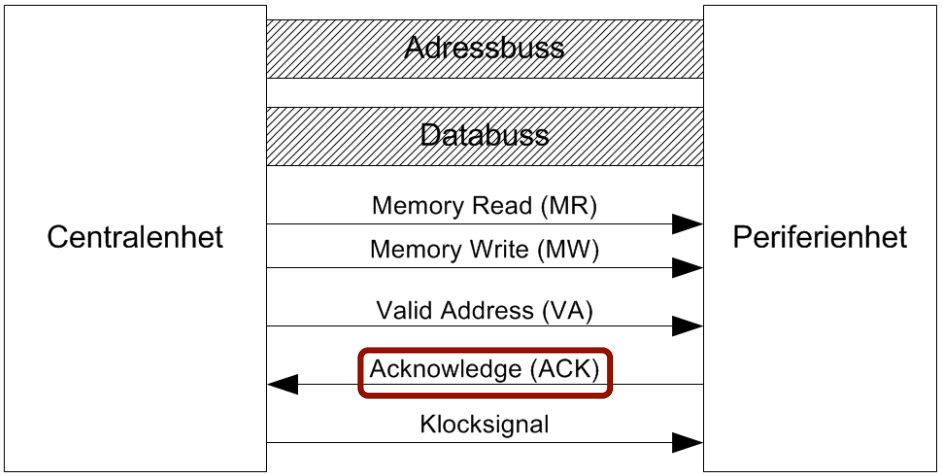
På en asynkron buss används så kallade handskakningssignaler för kommunikation mellan enheterna. Centralenheten signalerar till minnes- och periferienheterna att giltig information finns på adressbuss (och vid skrivning, databuss) genom att aktivera en signal, ”Valid Address” (VA).



Bussprotokoll

Asynkron buss

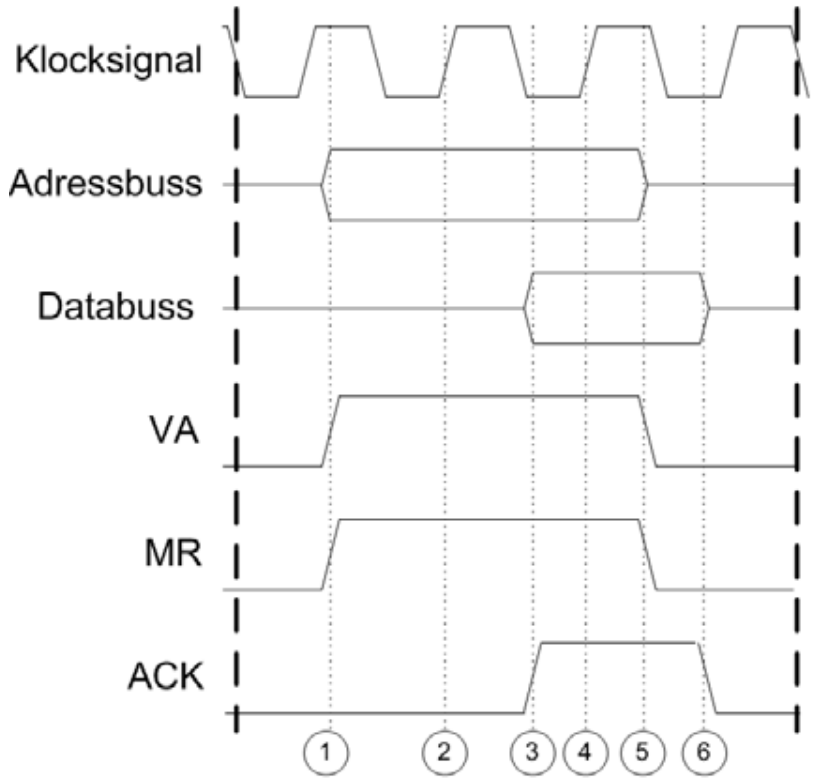
Minnes- och perferienheter läser av bussarna, och därefter måste den enhet som adresseras generera en signal, Acknowledge (ACK), tillbaks till centralenheten, för att meddela centralenheten att adressen är igenkänd och att data kan utväxlas.



Bussprotokoll

Asynkron buss – läsning

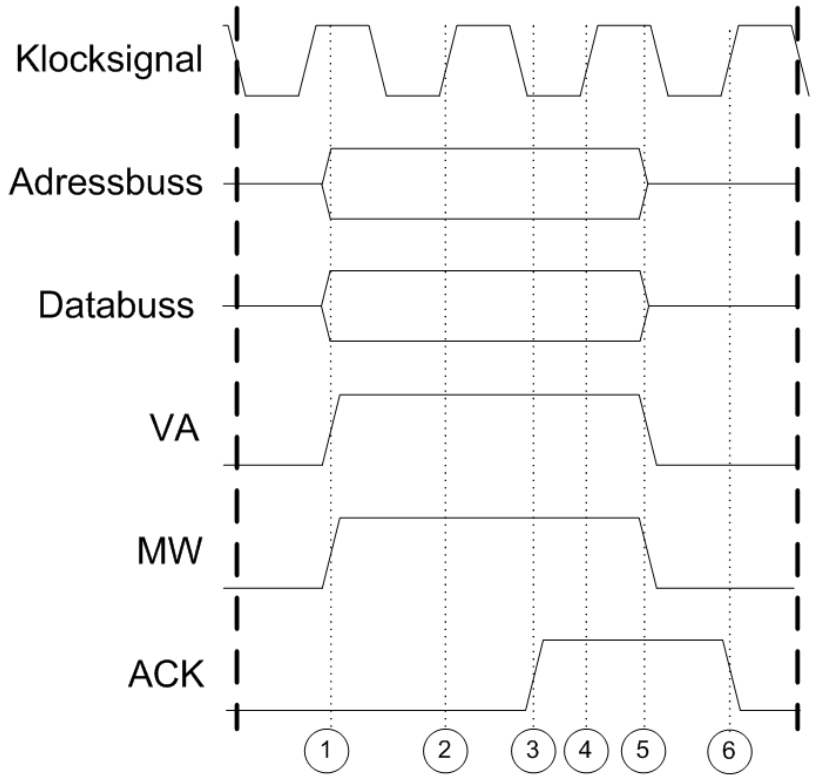
1. Centralenhet lägger ut adress och styrsignaler.
2. Signal VA detekteras och adressbuss avkodas.
3. Periferienhet visar att den har adresseras och lägger ut data samt aktiverar signal ACK.
4. Centralenhet detekterar ACK och klockar in data.
5. Centralenhet deaktiverar adress och styrsignaler.
6. Periferienhet deaktiverar ACK.



Bussprotokoll

Asynkron buss – skrivning

1. Centralenhet lägger ut adress, data och styrsignaler.
2. Signal VA detekteras, adress avkodas och data klockas in i periferienhet.
3. Periferienhet visar att den har adresserats genom att aktivera signal ACK.
4. Centralenhet detekterar ACK.
5. Centralenhet deaktiverar adress och styrsignaler.
6. Periferienhet deaktiverar ACK.

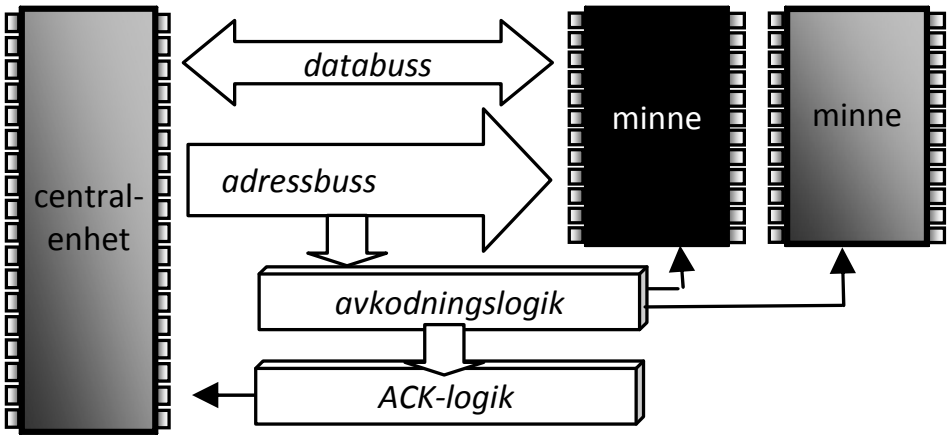


Bussprotokoll

Asynkron buss – väntecykler

Det är inte alla typer av minnes- eller periferienheter som själva kan generera ACK-signal. Då måste ett speciellt logiknät för detta syfte konstrueras i samverkan med avkodningslogiken. Logiken för ACK-generering kan då anpassas till åtkomsttiden för varje ansluten enhet genom lägga in ett lämpligt antal väntecykler innan ACK aktiveras (tidpunkt 3).

Centralenheter med asynkron buss kan (via time-out-logik) detektera en utebliven ACK, vilket då ger upphov till ett ”address error”-undantag.

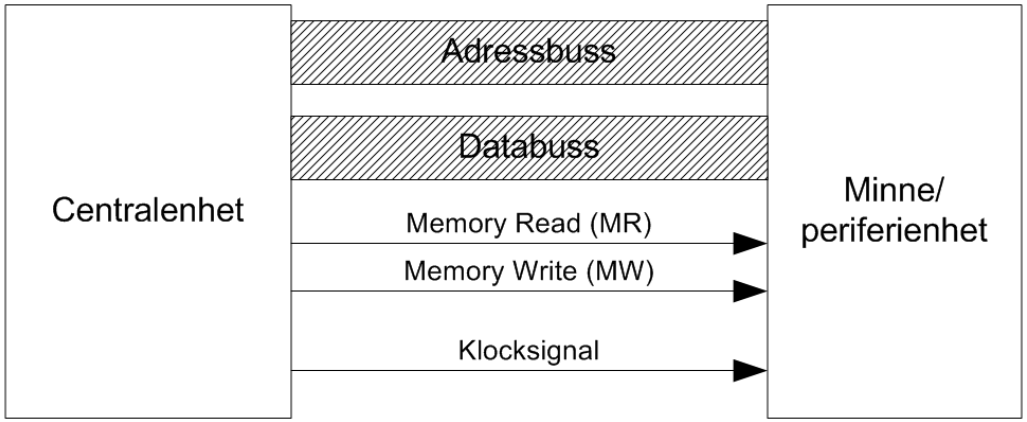


Bussprotokoll

Synkron buss

Synkron busskommunikation är den enklaste varianten av bussprotokoll, i den meningen att man inte behöver använda handskakningssignaler. Dataöverföringen förutsätter alltså att de andra enheterna hänger med i centralenhetens arbetstakt.

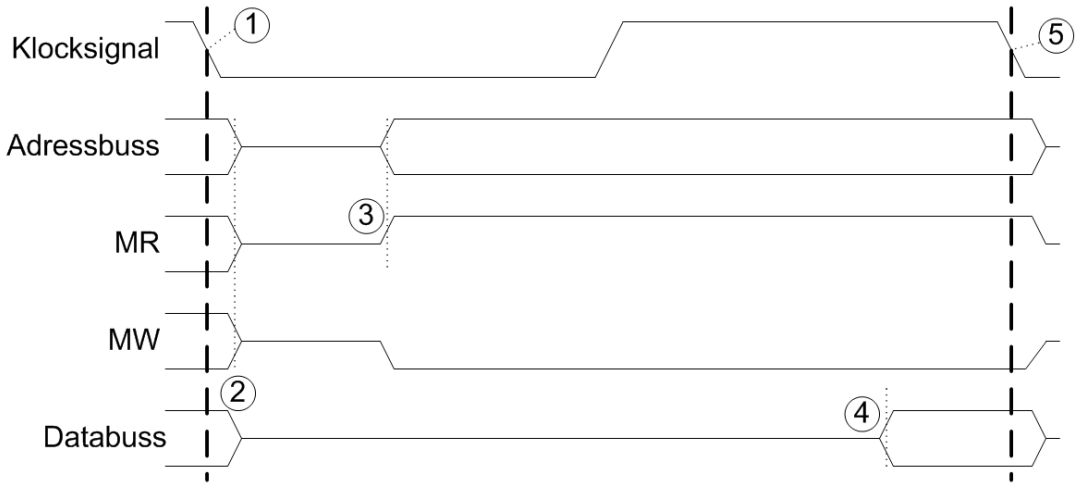
En nackdel med detta är att det kan vara svårt att blanda olika typer av minnen med stor skillnad i åtkomsttid.



Bussprotokoll

Synkron buss – läsning

1. Läscykel inleds med negativ klockflank.
2. Bussarna övergår till odefinierat tillstånd.
3. Centralenhet lägger ut adress och styrsignaler.
4. Periferienhet har adresserats och lägger ut data.
5. Läscykel avslutas med negativ klockflank, och centralenheten klockar in data.



Bussprotokoll

Synkron buss – skrivning

1. Skrivcykel inleds med negativ klockflank.
2. Bussarna övergår till odefinierat tillstånd.
3. Centralenhet lägger ut adress och styrsignaler.
4. Centralenhet lägger ut data och periferienheten adresseras.
5. Skrivcykel avslutas med negativ klockflank, och periferienhet klockar in data.