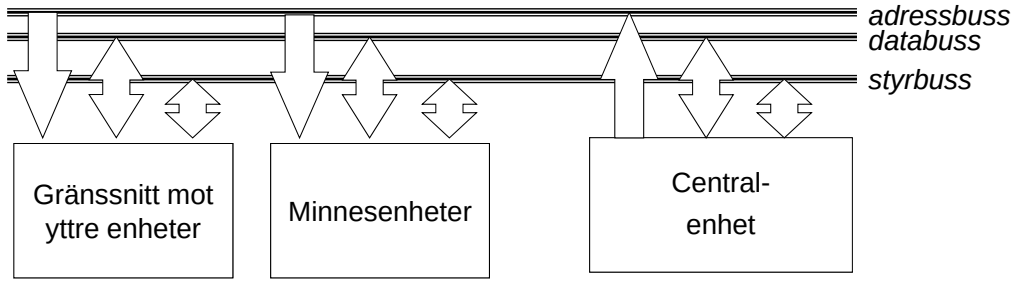
Adressavkodning - busskommunikation

Kompendie kapitel 10

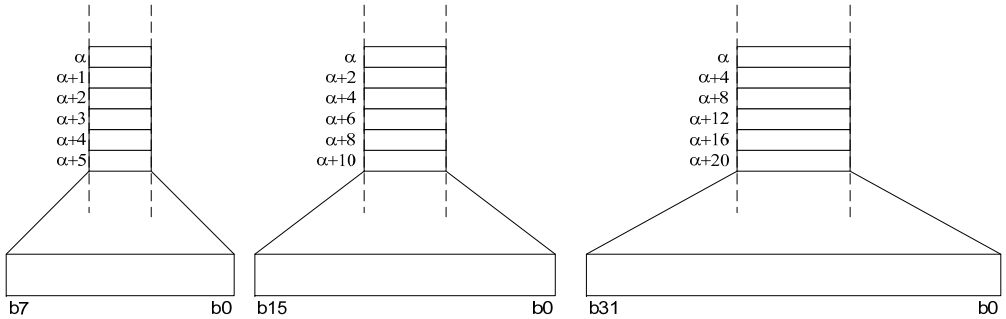
Ur innehållet:

- Bussystem, intern kommunikation i datorsystemet
- Adressavkodning, hur primärminne och I/O-enheter kan anslutas
- Olika principer för intern kommunikation (busskommunikation)

Bussystem



Primärminnets organisation

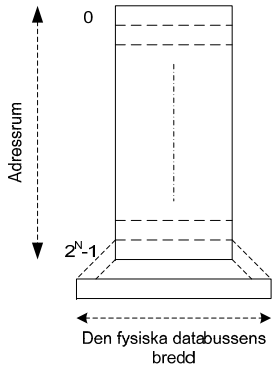


BYTE-adresserat

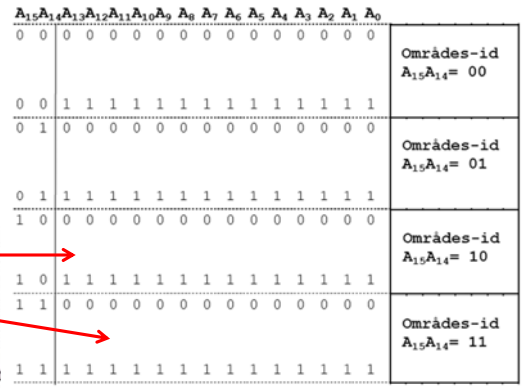
WORD-adresserat

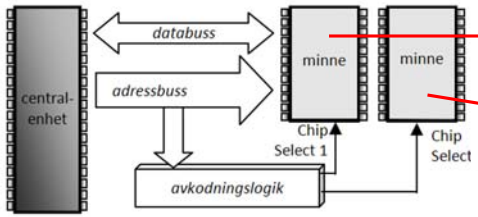
LONGWORD -adresserat

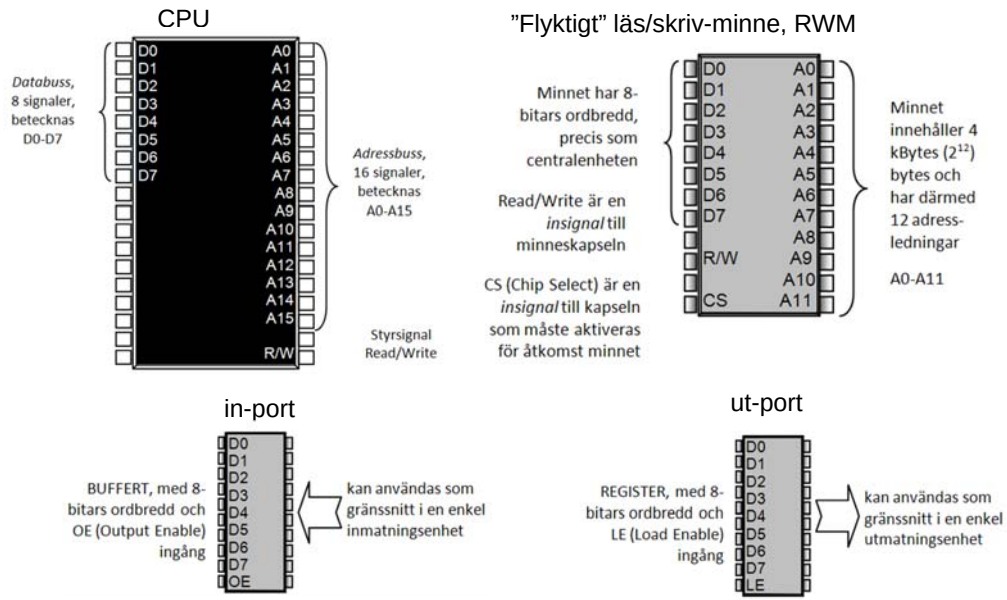
EXEMPEL: Adressavkodning



Varje minnesblock (kapsel) tilldelas sin unika del av adressrummet







Specifikation

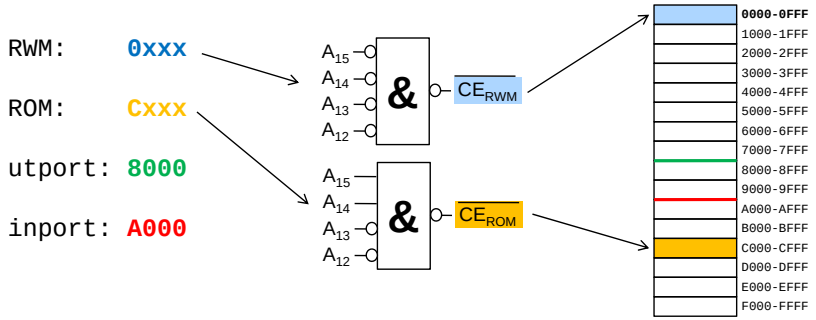
- RWM-minnet ska aktiveras om centralenheten genererar någon av adresserna 0-0FFF.
- ROM-minnet ska aktiveras om centralenheten genererar någon av adresserna C000-CFFF.
- Inporten ska aktiveras om centralenheten genererar adress A000.
- Utporten ska aktiveras om centralenheten genererar adress 8000.

Kapsel	Startadress	Slutadress
Minneskapsel RWM 4 kbyte	0000	0FFF
Minneskapsel ROM 4 kbyte	C000	CFFF
Buffert, 8 bitar som inport	A000	A000
Register, 8 bitar som utport	8000	8000

Realisering med NAND-logik

Fullständig avkodning

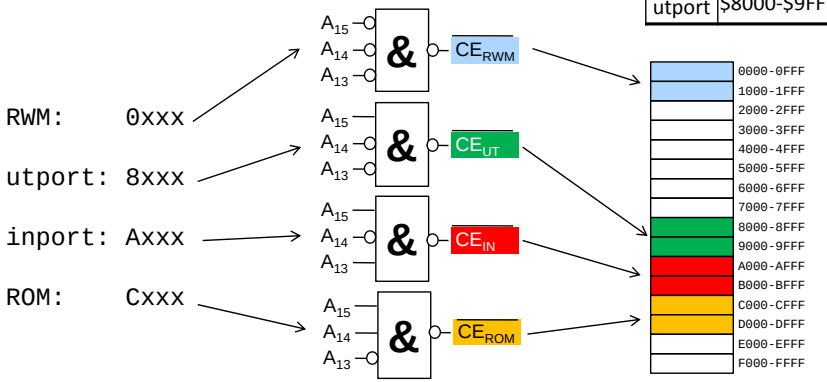
Kapsel		Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RWM	$0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROM	$CFFF	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	$C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
inport	$A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	$A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
utport	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

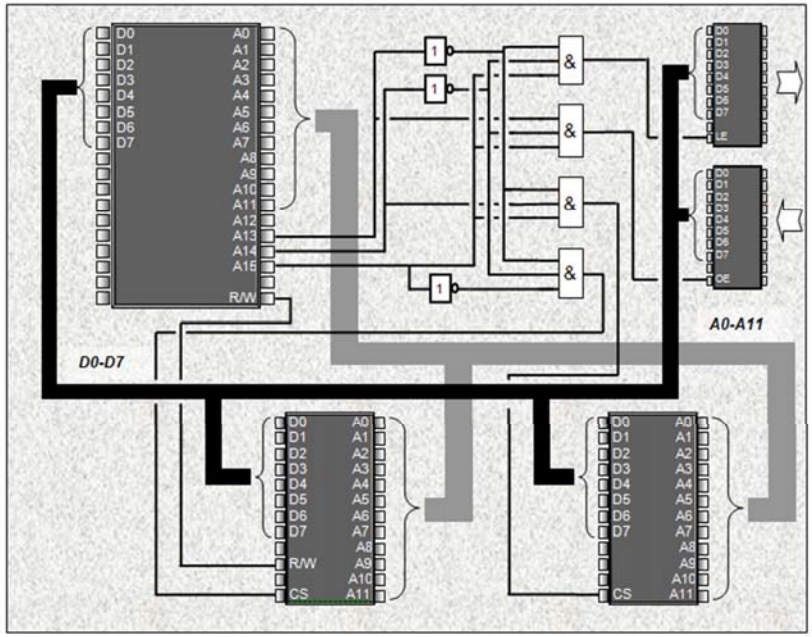


Kapsel		Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RWM	$0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROM	$CFFF	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	$C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
inport	$A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	$A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
utport	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

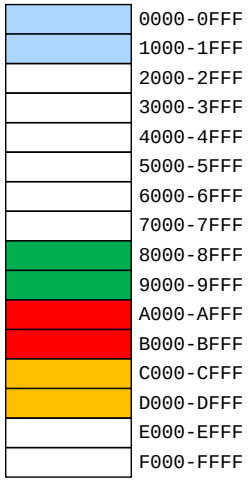
Ofullständig avkodning

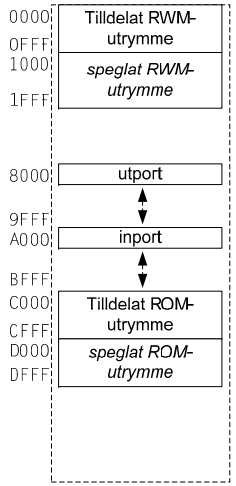
Kapsel		Adressbuss		
		A15	A14	A13
RWM	$0000-$1FFF	0	0	0
ROM	$C000-$DFFF	1	1	0
inport	$A000-$BFFF	1	0	1
utport	$8000-$9FFF	1	0	0





Ofullständig adressavkodning





Exempel:

Vi har ett system med 16 bitars adressbuss och 8 bitars databuss. Till centralenheten vill vi ansluta en ROM-modul (32 kbyte), en RWM-modul (16 kbyte) och en I/O-port (en inport och en utport). I/O-porten skall placeras på adress $4000, ROM-modulen placeras med start på adress $8000, RWM-modulen placeras med start på adress 0000 i adressrummet.

Konstruera adressavkodningslogiken, dvs. ange booleska uttryck för ”chip select” (CS)- signalerna. Använd **ofullständig adressavkodning**. Alla ”chip select” signaler (CS_{ROM}, CS_{RWM}, CS_{IN} och CS_{UT}) är aktiva låga.

Vi löser detta på tavlan…

Vi sammanfattar resultaten:

Modul		Adressbuss		
		A15	A14	A13....
ROM	$FFFF	1	1	1
	$8000	1	0	0
RWM	$3FFF	0	0	1
	0	0	0	0
IO-port	$4000	0	1	0
	$4000	0	1	0

$$\overline{CS_{ROM}} = \overline{A15}$$

$$\overline{CS_{RWM}} = \overline{\overline{A15} \cdot \overline{A14}}$$

$$\overline{CS_{IN}} = \overline{\overline{A15} \cdot A14 \cdot R/\overline{W}}$$

$$\overline{CS_{UT}} = \overline{\overline{A15} \cdot A14 \cdot \overline{R/\overline{W}}}$$

Fullständig adressavkodning

Modul		Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RWM	$3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IO-port	$4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Adressbitar som EJ skiljer sig åt i intervallet används i avkodningslogiken.

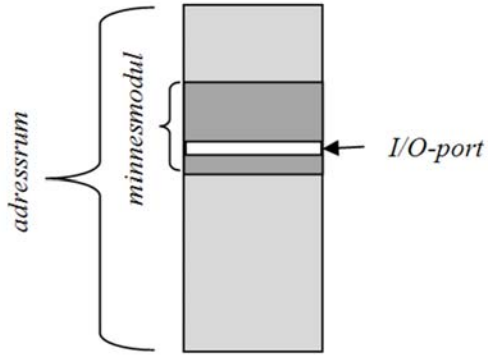
$$\overline{CS_{ROM}} = \overline{A15}$$

$$\overline{CS_{RWM}} = \overline{\overline{A15} \cdot \overline{A14}}$$

$$\overline{CS_{IN}} = \overline{\overline{A15} \cdot A14 \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5} \cdot \overline{A4} \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \cdot R/\overline{W}}$$

$$\overline{CS_{UT}} = \overline{\overline{A15} \cdot A14 \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5} \cdot \overline{A4} \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \cdot \overline{R/\overline{W}}}$$

Överlagrad minnesavbildning



Metod:
Bilda fullständig avkodningssignal för IO-porten.

Använd inversen av IO-portens CS-signal i avkodningslogiken för minnesmodulen.

Exempel:

Vi har ett system med 20 bitars adressbuss och 8 bitars databuss. Till centralenheten finns ansluten en 128 kbytes ROM-modul med start på adress $20000. En I/O-port, som upptar 32 bytes, ska överlagras på adress $22000.

Konstruera adressavkodningslogiken, dvs. ange booleska uttryck för ”chip select” (CS)- signalerna. Båda ”chip select” signalerna (CS_{ROM} och CS_{IO}) är aktiva låga.

Lösningsmetod

Bestäm start och slutadresser och ställ upp en tabell:

Modul		Adressbuss																			
		A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM	$3FFFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$20000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IO-port	$2201F	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1
	$22000	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Ur tabellen får vi direkt den fullständiga avkodningslogiken för I/O-porten:

$$\overline{CS_{IO}} = \overline{A19 \cdot \overline{A18} \cdot A17 \cdot \overline{A16} \cdot \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5}}$$

Avkodningslogiken för ROM-kapseln fås på samma sätt men vi grindar dessutom in inversen av $\overline{CS_{IO}}$.

$$\overline{CS_{ROM}} = \overline{\overline{A19} \cdot \overline{A18} \cdot A17 \cdot \overline{CS_{IO}}}$$

Busskommunikation

Datorns buss-kommunikation måste vara fastlagd i ett *protokoll*. Med protokoll menas helt enkelt regler för hur kommunikationen sker, dvs. hur information, *data*, utbytes mellan olika enheter i systemet.
Regelverket inbegriper alltså överenskommelser om:

- ”Språket”, betydelsen av hög respektive låg logiknivå
- Organisation av bitarna i ett ord, ”endian bit order”
- Organisation av bytes i ett ord ”endian byte order”
- Tidsegenskaper, händelser korrekt ordnade i tid

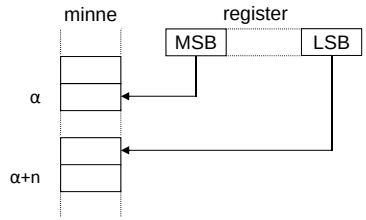
Bit-ordning

big-endian (”LSB 0”) | b_7 | b_6 | b_5 | b_4 | b_3 | b_2 | b_1 | b_0 |

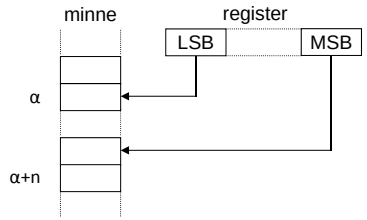
little-endian (”MSB 0”) | b_0 | b_1 | b_2 | b_3 | b_4 | b_5 | b_6 | b_7 |

8-bitars ord där b_7 är den MEST signifikanta biten och b_0 den MINST signifikanta biten

Byte-ordning

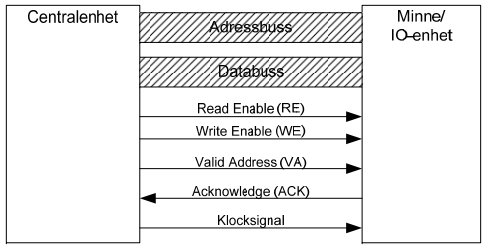


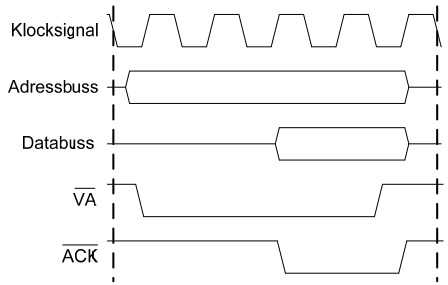
big-endian byte ordning



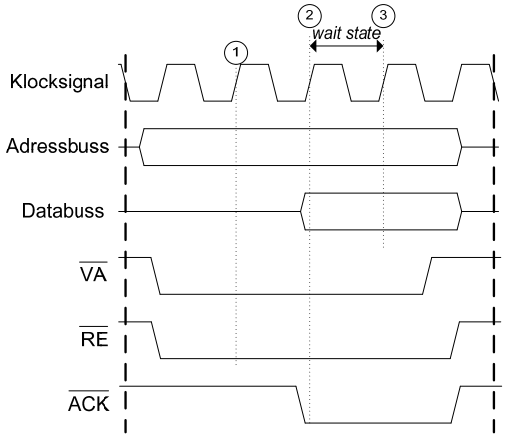
little-endian byte ordning

Asynkron buss





Läscykel, asynkron buss

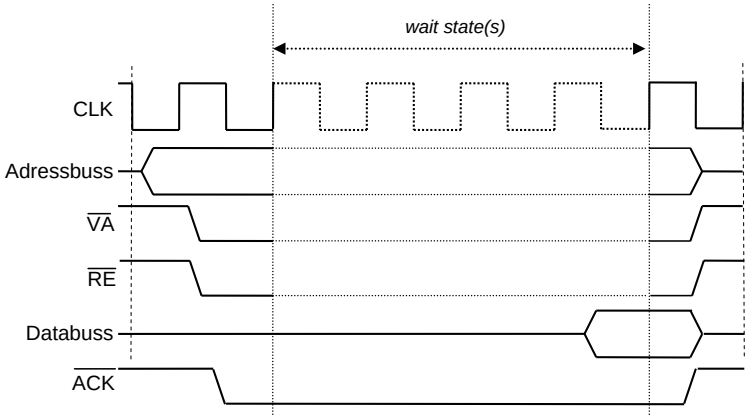


Vid ”1” börjar periferienheten avkoda adressen från adressbussen,.

ACK-signalen upptäcks av centralenheten vid ”2” och databussen läses av vid nästa positiva klockflank ”3”.

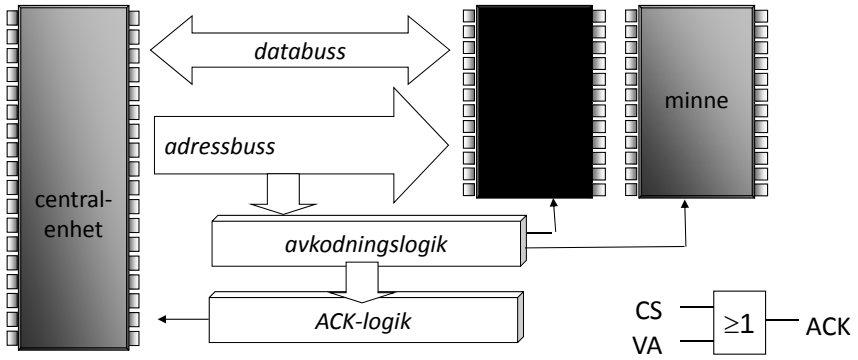
Antalet klockcykler från ACK, tills data klockas in från databussen kallas *väntecykler* (”*wait states*”).

Väntetillstånd



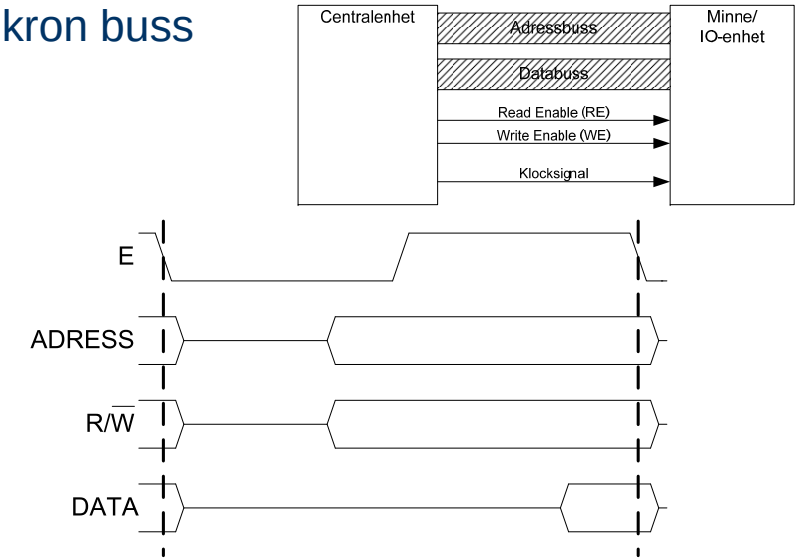
Generering av ACK-signal

Minneskretsar genererar som regel inte denna signal…

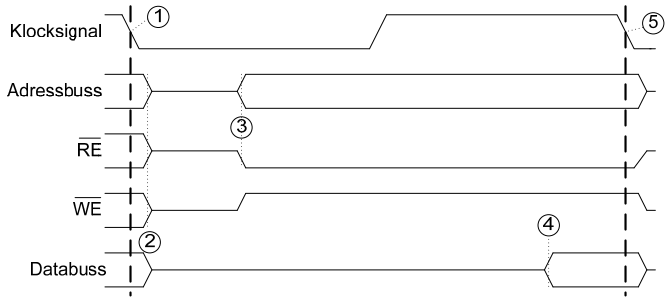


Utebliven ACK-signal indikerar ”Bussfel”…

Synkron buss

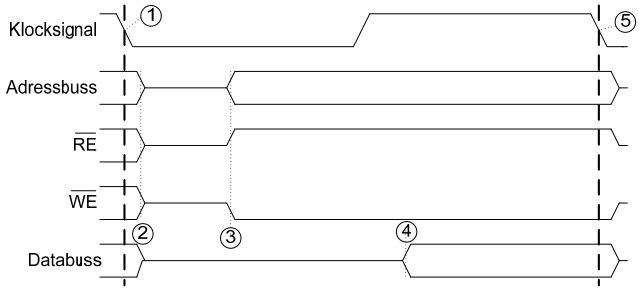


Läscykel, synkron buss



1. Läscykeln inleds med en negativ flank hos klocksignalen.
2. Signaler som var giltiga under föregående cykel tas bort och bussarna är nu i odefinierade tillstånd.
3. Centralenheten sätter upp adress och styrsignaler för minnet.
4. Minnet har avkodat adressbussen och lägger ut data på databussen.
5. Läscykeln avslutas med en negativ flank hos klocksignalen. Data klockas in till centralenheten från databussen.

Skrivcykel, synkron buss



1. Skrivcykeln inleds med en negativ flank hos klocksignalen.
2. Signaler som var giltiga under föregående cykel tas bort och bussarna är nu i odefinierade tillstånd.
3. Centralenheten sätter upp adress och styrsignaler för minnet.
4. Centralenheten lägger ut data på databussen, adressbussen avkodas och aktiverar minneskapsel.
5. Skrivcykeln avslutas med en negativ flank hos klocksignalen. Data klockas in till minnet från databussen.

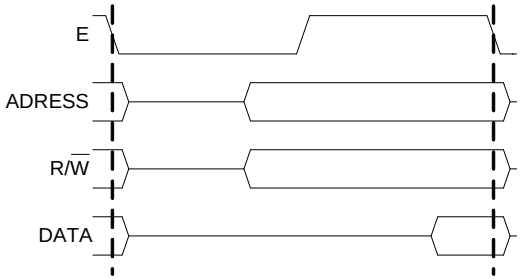
Exempel

Vi har ett synkront system med 16 bitars adressbuss och 8 bitars databuss. Data klockas i systemet vid negativ flank hos signalen E.

Till centralenheten ska anslutas:

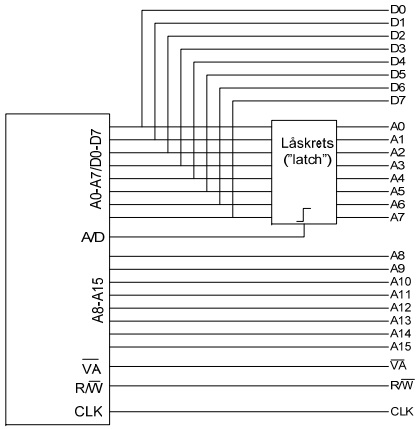
- 8 kbyte ROM med start på adress $E000
- 16 kbyte RWM med start på adress $8000
- 512 byte I/O, med start på adress $E000

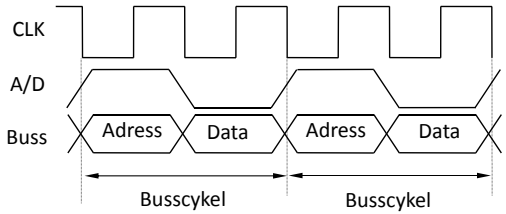
Konstruera fullständig adressavkodningslogik, dvs. ange booleska uttryck för ”chip select” (CS)-signalerna. Alla ”chip select” signalerna (CS_{ROM}, CS_{RWM} och CS_{IO}) är aktiva låga.

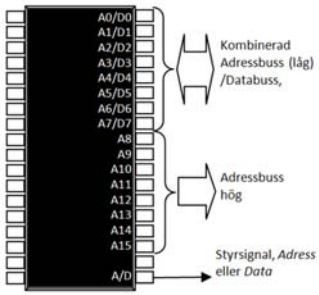


Vi löser detta på tavlan...

Multiplexad buss







Modul		Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0