(version 13-10-11)

Utdrag ur:

Grundläggande digital- och datorteknik

Del 2 – Datorteknik

Kapitel 13:

- Anslutning av minnes- och I/O-moduler till CPU12-buss

13 Anslutning av minnes- och I/O-moduler till CPU12-buss

Varje processor har minst ett ***adressrum***. Detta definieras som den mängd möjliga lagringsställen som kan adresseras via adressbussen och det kallas i fortsättningen för adressrum M (minne, memory). I kapitel 11 sågs att en stor del av maskininstruktionerna arbetar mot detta adressrum, som huvudsakligen rymmer primärminnet. Via programräknaren eller instruktioner kan processorn hämta eller placera binära ord på någon av adressrummets adresser under förutsättning att det finns ett läs- eller skrivbart register på adressen. Att hämta eller placera ett ord på en adress är liktydigt med att ***läsa*** (eng. ***read***) respektive ***skriva*** (eng. ***write***) på adressen.

Hur adressrummet används (är fyllt) beror mycket på den tillämpning i vilken processorn skall användas. Den största delen används för primärminnet. Vad som menas med primärminne beskrevs i avsnitt 9.5. Man kan se primärminnet som en stor mängd numrerade register.

Två typer av minnesmoduler kan användas. I den ena är registren både läs- och skrivbara. Sådana minnen kallas för ***läs/skrivminnen***, **RWM** (eng. ***read/write memories***I den andra är registren enbart läsbara och de kallas för ***läsminnen***, **ROM** (eng. ***read only memories***).

En mindre del av adressrummet brukar användas för ***separata register***, som inte tillhör primärminnet utan används för andra ändamål. Några av dessa är register som processorn använder för att kommunicera med omvärlden. Denna kommunikation kallas för ***in- och utmatning av data*** (eng. ***input and output*** eller **I/O**). Register som används för in- och utmatning av data kallas ofta ***I/O-portar***. För processorn är en I/O-port enbart läsbar (en inport) eller enbart skrivbar (en utport) i normalfallet.

På varje adress i adressrummet kan det således finnas antingen ett register som är både läs- och skrivbart (som i RWM) eller två register av vilka det ena enbart är läsbart från processorn (en inport) och det andra enbart är skrivbart från processorn (en utport).

Detta kapitel behandlar hur adressrummet kan utnyttjas dels för primärminne, dels för in- och utmatning av data. För att göra detta behöver man god kännedom om hur processorn arbetar mot sitt adressrum M. Kapitlet inleds med en beskrivning av dels begreppen "random access" och "random access"-minnen, dels hur CPU12 arbetar mot sitt adressrum. Detta för att CPU12 används flitigt i kursen för exemplifiering.

Mål: Efter att ha studerat kapitlet skall man kunna:

- konstruera ett CPU12 baserat system med önskvärd kombination av RWM- och ROM-moduler, samt I/O-portar
- konstruera adressavkodare för ett givet innehåll i adressrummet M
- konstruera system med endera fullständig eller ofullständig adressavkodning

13.1 Något om "random access"-minnen

Det skall vara lika lätt att komma åt innehållet på en adress i adressrummet oavsett var adressen är. Med detta menas att åtkomsttiden (= accesstiden) skall vara densamma för alla adresser. Principen kallas för "random access". Minnen som svarar upp mot detta kallas för ***"random access"-minnen,*** **RAM** (eng ***random access memories***) (access = åtkomst).

I dagens datorer utgöres minnesmodulerna av ***halvledarminnen*** (eng. ***semiconductor memories***). Detta är integrerade kretsar som innehåller en stor mängd minnesregister av något slag. Som tidigare sagts skiljer man på läs/skrivminnen, RWM och läsminnen, ROM. Båda är "random access"-minnen.

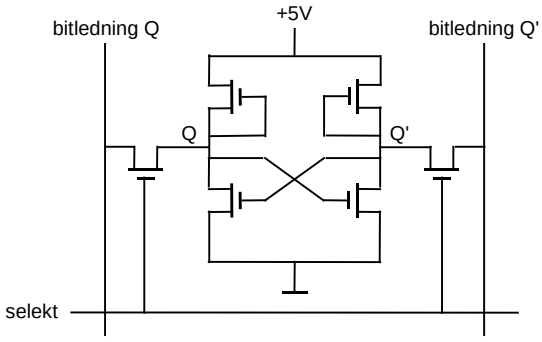
RWM ger användaren möjlighet att byta ut program och data eller att göra ändringar i dataareor. Halvledar-RWM förlorar sitt informationsinnehåll när matningsspänningen slås av och sägs därför vara ***flyktigt*** (eng. ***volatile***).

Exempel 13.1

Här exemplifieras två typiska bitceller för läs/skrivminnen, RWM. De är realiserade i en halvledarteknologi som kallas nMOS (n-channel Metal-Oxide-Semiconductor).

6-transistorcell

Kopplingen i Figur 13.1 innehåller sex nMOS-transistorer. Den mittre delen är en modern variant av den triggerkoppling (latch) som Eccles och Jordan presenterade 1919 (jfr figur 5.9 i kapitel 5. Det är en latch som både går att ettställa och nollställa. De två omgivande transistorerna är switchar som möjliggör att man kan komma åt latchen.



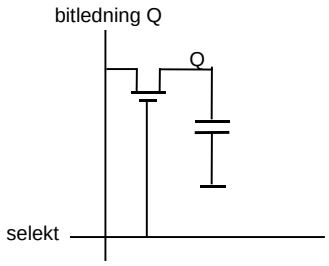
Figur 13.1

Kopplingen används som bitcell i statiska läs/skrivminnen (de kallas på engelska static RAMs).

Cellen är ***statisk*** för att den behåller sitt värde så länge den har matningsspänning (+5V). Innehållet i cellen kan läsas eller ändras genom att aktivera selektledningen och därefter på lämpligt sätt använda bitledningarna.

1-transistorcell

Kopplingen i Figur 13.2 har inga likheter med latchen. Istället lagras bitvärdet som laddning i en kondensator. Laddning i kondensatorn svarar mot "1" och ingen laddning svarar mot "0".



Figur 13.2

nMOS-transistorn fungerar som switch och möjliggör att man kan komma åt kondensatorn. Via selekt kan då kondensatorn kopplas till eller från bitledningen. När kondensatorn är bortkopplad från bitledningen så "läcker" den långsamt, dvs laddningen försvinner långsamt. Man säger att en kondensator är en ***dynamisk*** bitcell just för att den läcker. Funktionen som bitcell bygger därför på att kondensatorn används ofta, dvs laddas upp (med "1") eller ur (med "0") ofta. Innehållet i cellen kan läsas eller ändras genom att aktivera selektledningen och bitledningen.

Vid läsning laddas kondensatorn ur. Efter läsningen har den således ingen laddning. Det lästa värdet måste således återskrivas i cellen, dvs kondensatorn måste återladdas, annars har ju cellen "tappat minnet".

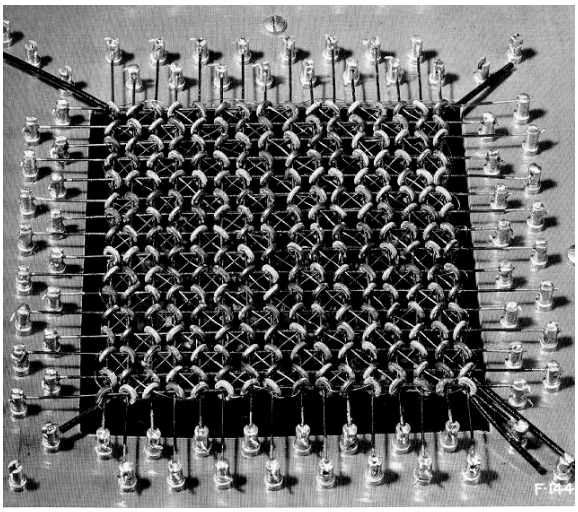
Kopplingen används som bitcell i dynamiska läs/skrivminnen (dynamic RAMs). Den är trots sina egenheter mycket attraktiv för att den är enkel och tar liten plats på ett chip. Den tar ju bara upp en sjättedel så stor yta som den statiska bitcellen.

Kärnminnet - gårdagens RWM

I kapitel 9 gavs en del viktiga milstolpar i datorernas utveckling. En av dem löd:

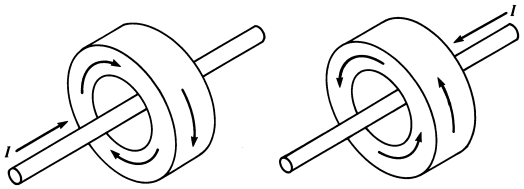
1953 installeras ringformade magnetiska kärnor att användas för lagring av bitvärden, s k kärnminnen, för första gången i en dator

I mer än 20 år fram till mitten av 1970-talet när halvledarminnen fick sitt stora genombrott utgjordes primärminnet av ***kärnminnesmatriser***. En sådan visas i Figur 13.3. I dessa utgjordes bitcellen av en liten ringformad kärna av magnetiserbart material (ferrit). Kärnans diameter var typiskt 0,5 mm.



Figur 13.3

Kärnan kan magnetiseras endera medsols eller motsols med hjälp av ström I, såsom visas i Figur 13.4.



ettställning nollställning

Figur 13.4

Magnetiseringen blir kvar när strömmen slås av och kärnan minns därför vilken riktning strömmen hade senast. Man kan således med strömriktningen lagra värdet "0" eller "1" i kärnan.

För att läsa vad som finns i kärnan måste magnetiseringsriktningen detekteras och detta gjorde man genom att slå på strömmen I i en av riktningarna, säg nollriktningen, och mäta om magnetiseringen ändrades eller inte. Ändrades den inte så innehåller cellen "0". Ändrades magnetiseringen så innehöll cellen "1", men läsningen har nu ändrat innehållet till "0". När cellen innehöll "1" så måste värdet "1" återskrivas efter läsningen. Vid läsning har således kärnan samma egenskaper som kondensatorn i den dynamiska cellen.

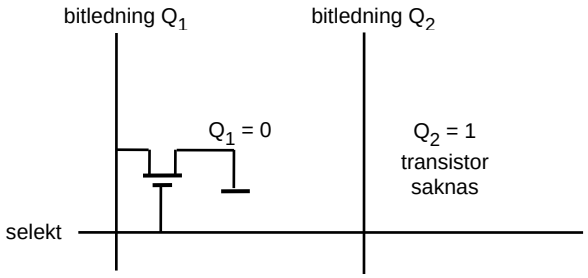
ROM är förprogrammerade och används för program och/eller data, som alltid skall vara tillgängliga i systemet, och för vilka inget behov av utbytbarhet föreligger. Läsminnen behåller sitt informationsinnehåll även när matningsspänningen är frånslagen. Man säger att ett läsminne är ***icke-flyktigt*** (eng. ***non volatile***).

Exempel 13.2

Läsminnen (ROM) kännetecknas av att de behåller sitt innehåll även om matningsspänningen slås av. De finns olika varianter och de klassificeras efter det sätt som minnesmodulen ges sitt innehåll. Följande varianter är vanliga.

Mask-programmerade ROM får sitt innehåll när modulen tillverkas. Innehållet kan inte ändras. Det är dyrt att skapa den mask som definierar innehållet, så denna typ av minne lämpar sig bäst vid tillverkning av stora serier med ett och samma innehåll.

I Figur 13.5 visas de två typiska fallen på en bitcells utseende. Den är som synes en 1-transistorcell och tar därför upp liten yta på ett chips. Detta kännetecknar även bitcellerna för varianterna som följer.



Figur 13.5

Fält-programmerbara ROM ges sitt innehåll ute på "fältet" (där de skall användas) innan de används. När de väl fått ett innehåll så går det inte att ändra. Modulerna kallas PROM (programmable ROM). Programmeringen sker i en speciell utrustning kallad PROM-brännare.

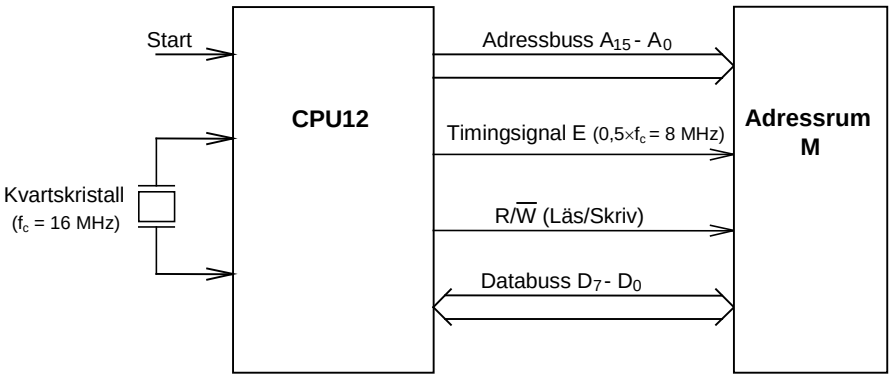
Raderbara ROM ges sitt innehåll ute på fältet innan de används. De kan raderas med ultraviolet ljus och omprogrammeras på elektrisk väg. För att göra detta måste modulen plockas ut ur det system i vilket det används. Radering och omprogrammering kan ske många gånger. Dessa moduler kallas EPROM (Erasable Programmable ROM).

Elektriskt raderbara ROM ges sitt innehåll ute på fältet innan de används. De kan raderas och omprogrammeras på elektrisk väg utan att ta ut modulen ur systemet i vilket de används. Dock behövs ett elektriskt specialarrangemang för att göra ändringarna. Ändringarna görs ord för ord, ett i sänder. Dessa moduler kallas också EEPROM (Electrically Erasable PROM)

"Flash"-minnen är en form av elektriskt raderbara ROM. I dessa raderas block av ord istället för ett ord isänder.

13.2 Processorn CPU12

I Figur 13.6 visas de signaler CPU12 har för arbete mot adressrum M.

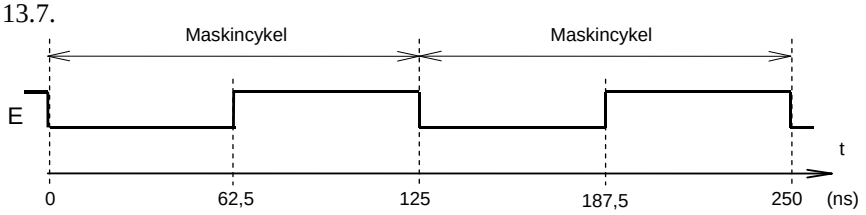


Figur 13.6 CPU12 kopplad till adressrum M.

Läsning respektive skrivning kallas för minnesreferenser och sker vanligen under en klockcykel. Namnet minnesreferens kommer av att adressrum M används för såväl primärminne som andra minnesregister.

CPU12-klockning (timing)

CPU12 genererar "timing"-signalen E under hela tiden den arbetar. Den bildas ur signalen från en kristalloscillator med frekvensen 16 MHz i normalfallet. Frekvensen hos E är halva kristallfrekvensen, dvs 8 MHz i normalfallet, se Figur 13.7.



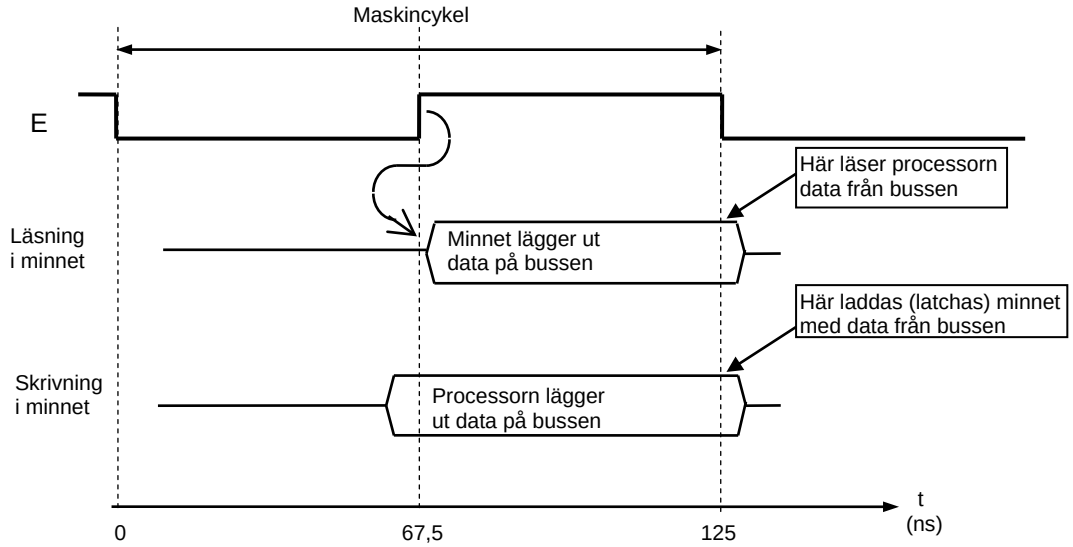
Figur 13.7 Timingsignalen E hos CPU12.

När CPU12 läser eller skriver i minnet utför den en ***maskincykel***. Den inleds när ***signalen* E** går från hög till låg nivå (negativ flank) och avslutas med nästa negativa flank på E-signalen, såsom visas i Figur 13.7.

En läsning i minnet (en läscykel) inleds med att processorn lägger ut adressen, där läsningen skall ske, på adressbussen samtidigt som signalen ***Läs***/$\overline{\textbf{\textit{Skriv}}}$ (***R***/$\overline{\textbf{\textit{W}}}$) ges värdet "1". Det sker vid negativ flank på E-signalen.

På grund av att det finns interna fördröjningar i processorn så kommer varken adressbitar eller R/$\overline{W}$-signalen att få rätt värde förrän ***efter*** E-signalens negativa flank. Dessutom kommer de inte att få rätt värde samtidigt på grund av att fördröjningarna är olika för de olika signalerna. Vid tidpunkten för E-signalens positiva flank är dock adressen och R/$\overline{W}$-signalen garanterat korrekta (stabila) och först då bör minnet börja "plocka" fram data från den aktuella adressen och placera det på databussen, vilket visas i Figur 13.8. Processorn läser sedan av data från databussen vid slutet av maskincykeln dvs vid E-signalens nästa negativa flank.

Ett liknande resonemang kan föras för adresser och R/$\overline{W}$-signalen vid skrivning. Skrivningen inleds med att processorn lägger ut adressen, där skrivningen skall ske, på adressbussen samtidigt som signalen ***Läs***/$\overline{\textbf{\textit{Skriv}}}$ (***R***/$\overline{\textbf{\textit{W}}}$) ges värdet "0". Därefter lägger processorn ut data på databussen på det sätt som visas i Figur 13.8. Minnet laddas sedan med data från databussen vid slutet av maskincykeln dvs vid E-signalens nästa negativa flank.



Figur 13.8 Läsning och skrivning med CPU12-processorn

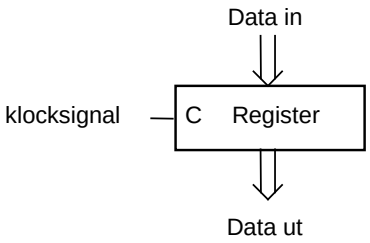
Vid läsning och skrivning är således adressen och R/$\overline{W}$-signalen, som läggs ut från processorn vid E-klockans negativa flank, inte garanterat korrekta (stabila) förrän vid E-klockans positiva flank. Under resterande halvan av maskincykeln är dock adressen stabil. Adressignalerna och R/$\overline{W}$ signalen har således stabila värden under tiden som E är "1". E- signalen är därför en **VMA-signal** (***Valid Memory Address***). För CPU12 används signalen VMA = E ofta som grindsignal vid arbete mot adressrum M.

När minnet är anslutet till processorns bussar enligt principen i figurerna 13.6 och 13.8 är det processorn som bestämmer timing vid läsning och skrivning i minnet. Det förutsätts då att minnet hinner med att hantera data vid

läsning och skrivning. Denna princip kallas **synkron bussanslutning**. Vid **asynkron bussanslutning** måste processorn vänta på klarsignal från minnet vid läsning och skrivning.

13.3 Minnesmoduler

Primärminnet byggs upp av moduler. I detta avsnitt beskrivs den principiella uppbyggnaden av en minnesmodul som är både läsbar och skrivbar, dvs en RWM-modul.

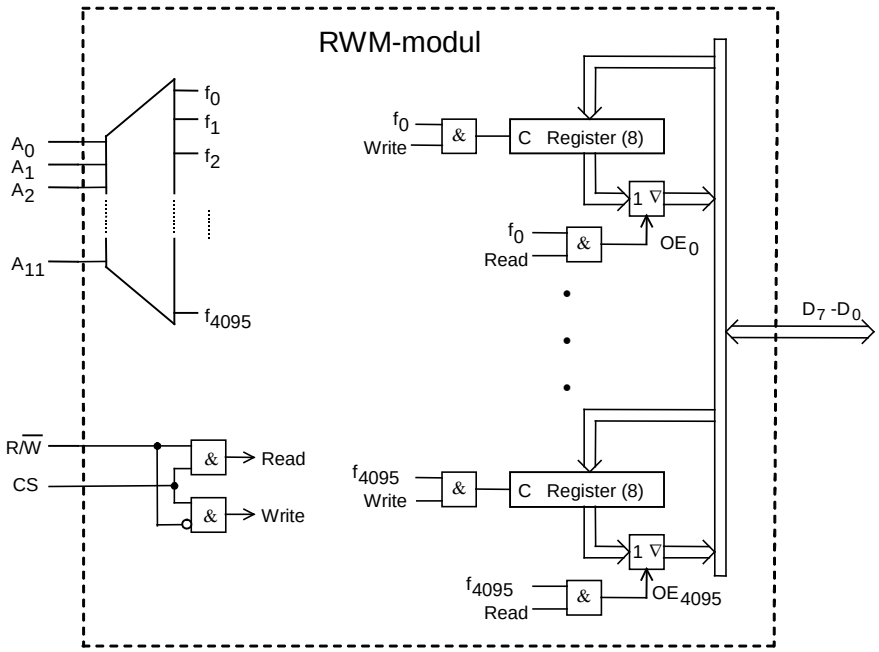


Figur 13.9 Blocksymbol för register uppbyggt med klockade D-latchar.

RWM-moduler innehåller register av latchtyp. Registren är därför uppbyggda med enkla klockade D-latchar, sådana som tidigare visats i figur 5.13. För ett sådant register används här blocksymbolen i Figur 13.9. Kom ihåg att ett sådant register inte är flanktriggat utan att innehållet kan ändras via "Data in"-ingångarna under hela den tid som klocksignalen är "1".

I Figur 13.10 visas schematiskt hur man kan tänka sig att en minnesmodul är uppbyggd med denna registertyp.

Ett register i modulen adresseras via adressledningarna. Eftersom var och en av dem kan anta värdena "0" och "1", så är antalet ord i modulen en heltalspotens av 2. Modulen i figuren har 12 st adressledningar och innehåller således 2^{12} = 4096 st ord.



Figur 13.10 Principiellt utseende av en 4k RWM-modul.

Orden har vanligen åtta bitar, dvs ordlängden är en byte. Vanliga modulstorlekar ligger i området från 1kbyte till 256 kbyte (1kbyte = 1024 bytes = 2^{10} bytes). Modulen i figur har således 4 kbytes (= 4·1024 bytes).

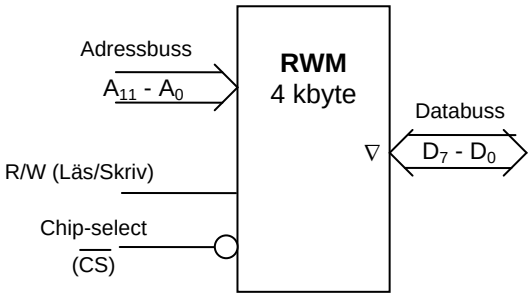
Det är nödvändigt att minnesmoduler har "three-state"-buffertar på registerutgångarna, som i Figur 13.10, ty varje modul skall kunna kopplas till en gemensam databuss för att lägga ut data på den. Detta sker internt via respektive registers OE-signal.

I en modul som också skall ta emot data från databussen måste registren kunna laddas via en signal. I Figur 13.10 sker detta internt via en signal till respektive registers grindingång C.

För CPU12 kan dessa interna signaler (till C och till "three-state" buffertar) skapas m h a signalen $R/\overline{W}$, som är "1" vid läsning och "0" vid skrivning.

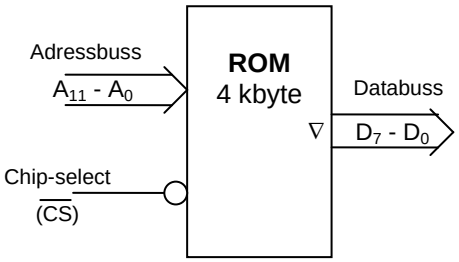
En modul som skall aktiveras (för läsning eller skrivning) skall kunna pekas ut (väljas) och det görs på en ingång som normalt kallas för ***"Chip Select"*** **(CS)** eller "Chip Enable" (CE) och den är oftast aktiv låg ($\overline{CS}$ eller $\overline{CE}$).

Minnesmodulens adressledningar (12 st), A_{11}-A_0, är inuti modulen kopplade till en stor avkodare (12/4096), såsom visas i Figur 13.10, för att välja ut rätt minnesregister för läsning eller skrivning. I fortsättningen kommer blocksymbolen i Figur 13.11 att användas för en 4 kbyte RWM-modul.



Figur 13.11 Blocksymbol för läs-/skrivminnesmodul (RWM) om 4 kbyte.

Man kan på ett liknande sätt beskriva den principiella uppbyggnaden av en ROM-modul, dvs en modul som enbart är läsbar. Det görs dock inte här. Dock ges blocksymbolen för en 4 kbyte ROM-modul i Figur 13.12. Den har som synes ingen insignal för att avgöra om läsning eller skrivning skall ske. Detta för att den enbart är läsbar.



Figur 13.12 Blocksymbol för läsminnesmodul (ROM) om 4 kbyte.

13.4 Anslutning av minnesmoduler till CPU12

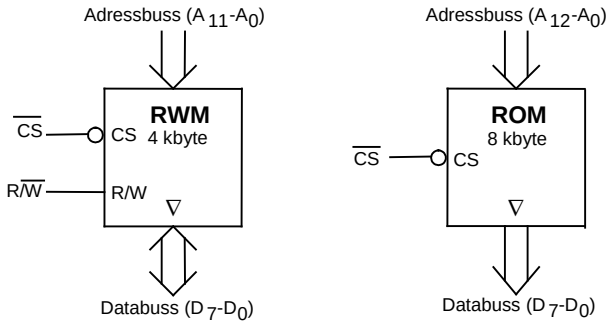
När minnesmoduler, lika de i Figur 13.11 och Figur 13.12, placeras i processorns adressrum M ansluts adressingångarna till motsvarande ledningar i adressbussen. De övriga adressledningarna i bussen, A_{15} -A_{12}, används för att välja rätt minnesmodul genom att man låter "Chip Select"-signalen bildas med hjälp av dem. Därmed aktiverar "Chip Select"-signalen minnesmodulen i det 4k-ords adressintervall, inom vilket processorn skall adressera minnesmodulen.

Eftersom flera moduler är kopplade till samma databuss måste man se till att inte två eller fler moduler lägger ut data på bussen samtidigt. Det är sedan tidigare känt att adressvärdet som processorn släpper ut inte är korrekt (stabilt) omedelbart efter E-signalens negativa flank. För att undvika "busskollisioner" på grund av "falska" adressvärden i början av maskincyklerna måste man förutom de högsta adressbitarna även använda VMA = E som grindsignal till det logiknät som bildar "Chip Select"-signalerna.

En konstruktör, som skall göra adressavkodningen för en dator, utgår ifrån hur stort minne och vilken typ av minne som behövs. Därefter avgörs hur stora minnesmoduler som skall användas. Valet bestäms ofta av sådana faktorer som snabbhet, pris, leveranstid m m. Nästa steg är att placera in de olika modulerna i processorns adressrum. Hur det görs beskrivs i ett exempel.

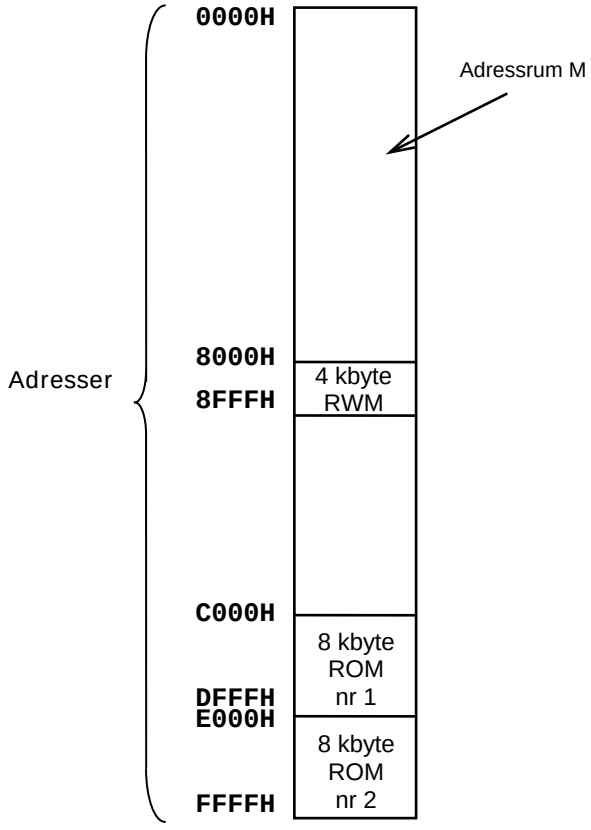
Exempel 13.3

För en viss CPU12-tillämpning behövs ett primärminne om 4 kbyte RWM och 16 kbyte ROM. Man har bara tillgång till RWM- och ROM-moduler av den typ som visas i Figur 13.13.



Figur 13.13 Tillgängliga RWM- och ROM-moduler.

Inplaceringen i adressrummet skall vara enligt Figur 13.14. Lägg märke till att modulerna placerats in på ett systematiskt sätt så att deras adresser börjar på "bra" adresser ex vis 8000H, C000H, E000H etc.



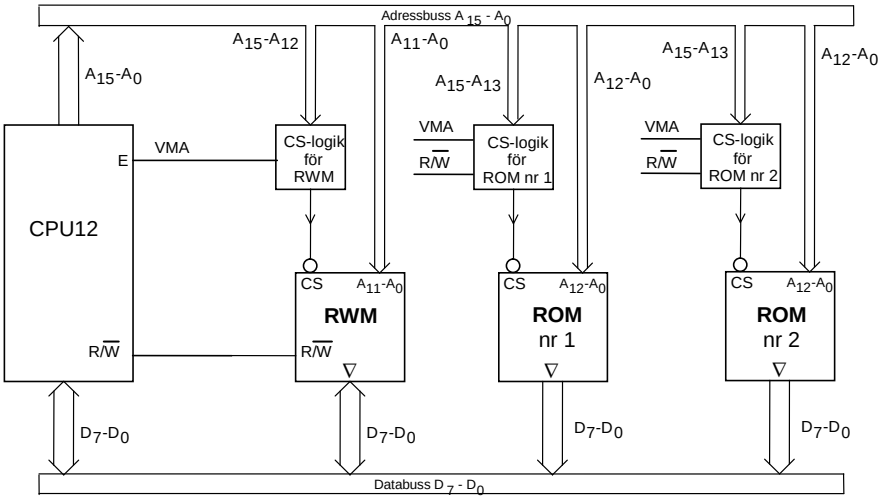
Figur 13.14

Inplaceringen kan också sammanfattas i tabellform:

Modul	*Adress*	*Adressignal nr*															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWM	8000H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	8FFFH	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 1	C000H	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	DFFFH	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 2	E000H	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	FFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

De olika modulerna skall adresseras inom unika adressintervall (områden). Inom respektive minnesmoduls adressområde är ett litet antal av de mest signifikanta adressbitarna konstanta. De är markerade med en ram. De har samma (unika) värden för alla adresser inom området och kan sägas bilda en ID-kod för respektive område. ID-koden kan därför användas för att "peka ut" respektive minnesmodul, dvs för att bilda CS-signalen, som är aktiv när processorn skall läsa eller skriva i just denna modul.

Modulerna ansluts till adress- och databussen enligt Figur 13.15.

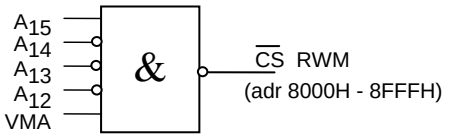


Figur 13.15 Anslutning av minnesmoduler till processorn CPU12.

Adressavkodningen (CS-logiken) sker i logikblock med de mest signifikanta adressbitarna och VMA, samt när så behövs signalen R / $\overline{W}$, som insignaler. Utsignalerna består av "chip select"-signaler till de olika modulerna.

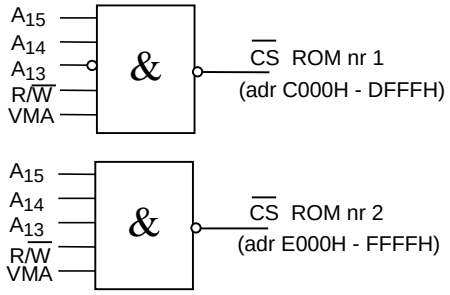
Börja med att betrakta adressavkodning för RWM-modulen. När modulstorleken är 4k = 2^{12} används 12 st adressbitar för intern adressavkodning inom modulen. Detta framgår av Figur 13.15. De övriga fyra adressbitarna, A_{15} - A_{12}, används för att bestämma var någonstans i processorns adressrum modulen placeras.

Enligt disponeringen av adressrummet i Figur 13.14 skall RWM-modulen placeras i adressintervallet 8000H - 8FFFH. I detta har adressbitarna A_{15} - A_{12} bitvärdena 1000 (se tabellen). Modulen kan då pekas ut med CS-signalen, som bildas enligt Figur 13.16. 1000 kan således ses som ID-koden för modulen



Figur 13.16 CS-logik för 4kbyte RWM-modul.

De två ROM-modulerna om vardera 8 kbyte skall ligga i adressområdena C000H-DFFFH (ROM nr 1) respektive E000H-FFFFH (ROM nr 2). I Figur 13.17 visas hur CS-signalerna, för att peka ut respektive ROM, bildas.



Figur 13.17 CS-logik för två 8 kbyte ROM.

Här används även R / $\overline{W}$ -signalen vid bildandet av CS eftersom man då kan undvika problem vid ett eventuellt försök till skrivning (av processorn) på en ROM-adress. Om R / $\overline{W}$ -signalen inte används på detta sätt så skulle en eventuell skrivning i en ROM-modul innebära att både processorn och ROM-modulen släpper ut data på databussen och då fås en "busskollision". Försök till skrivning i ROM kan ex vis inträffa på grund av ett programmeringsfel.

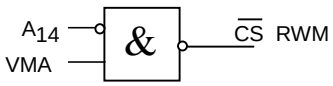
I Exempel 13.3 används adressbussens alla bitar i minnesavkodningen. Detta kallas för ***fullständig adressavkodning***. För att förenkla adressavkodningsblocken kan man ibland låta bli att använda alla adressbitarna vid avkodningen. Detta får till följd att adressavkodningen inte blir entydig dvs samma CS-signal kan bli aktiv i fler än ett adressintervall. Man kallar detta förfarande för ***ofullständig adressavkodning***

Har man valt att göra adressavkodningen ofullständig så blir det också svårt att vid en senare tidpunkt bygga ut minneskapaciteten i datorn. En annan nackdel med ofullständig adressavkodning är att samma modul kan adresseras på olika ställen i adressrummet. Detta kan ställa till problem om fel gjorts vid programskrivningen.

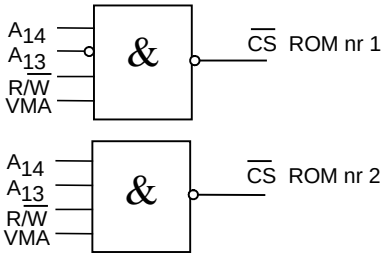
Exempel 13.4

Om avkodningen i Exempel 13.3 istället gjorts med CS-avkodare enligt Figur 13.18 a och b så är den ofullständig.

a) CS-logik för 4kbyte RWM-modulen



b) CS-logik för de två 8 kbyte ROM-modulerna



Figur 13.18

Enligt Figur 13.18 så finns det RWM överallt där adressbiten A_{14} är 0 och ROM överallt där A_{14} är 1. Detta har markerats i följande tabell, som avbildar processorns hela adressrum M.

Modul	*Adress*	*Adressignal nr*															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWM	0000H	**0**	0	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	0FFFH	**0**	0	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	1000H	**0**	0	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	1FFFH	**0**	0	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	2000H	**0**	0	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	2FFFH	**0**	0	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	3000H	**0**	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	3FFFH	**0**	0	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 1	4000H	**0**	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	5FFFH	**0**	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 2	6000H	**0**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	7FFFH	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RWM	8000H	**1**	0	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	8FFFH	**1**	0	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	9000H	**1**	0	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	9FFFH	**1**	0	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	A000H	**1**	0	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	AFFFH	**1**	0	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
RWM	B000H	**1**	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	BFFFH	**1**	0	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 1	C000H	**1**	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	DFFFH	**1**	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM nr 2	E000H	**1**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	:									:							
	FFFFH	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Det ofullständiga framgår av att modulerna, 4kbyte och 2×8 kbyte, nu "tar upp" (blockerar) mycket mer av utrymmet i adressrummet M, närmare bestämt hela utrymmet M, vilket ses i tabellen. Utöver de områden som är önskvärda så tar modulerna vid denna ofullständiga adressavkodning även upp de områden som gråmarkerats i tabellen.

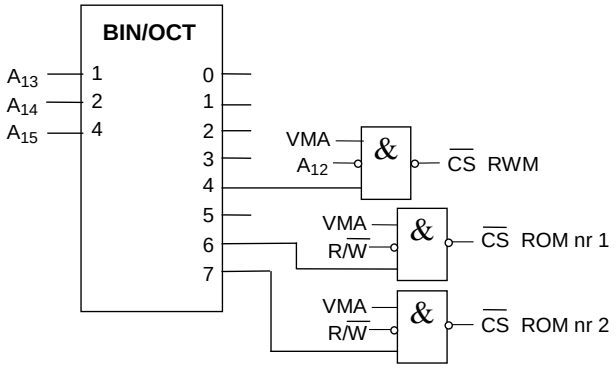
Anledningen till detta är att de adressbitar som markerats med fetstil i tabellen inte ingår i adressavkodningen. De skall därmed betraktas som "don't care"-värden.

I stället för att bygga upp adressavkodningen med diskreta grindar som i de visade exemplen använder man i praktiken ofta färdiga avkodare, fördelare, snabba ROM, PLA- eller PAL-kretsar för att bilda CS-signalerna. PLA- och PAL-kretsar är exempel på kretsar med ***programmerbar logik*, PLD (*Programmable Logic Device*).**

Exempel 13.5

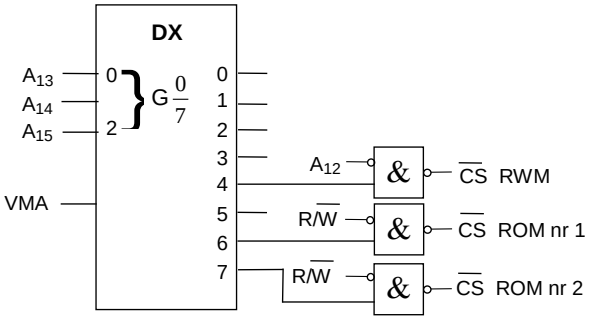
Avkodningen för de tre modulerna i Exempel 13.3 kan anordnas med en färdig funktionsmodul och kompletterande grindelement.

I Figur 13.19 är den ordnad m h a "NBC till en av åtta" avkodare (3/8 avkodare)



Figur 13.19

och i Figur 13.20 m h a en fördelare. Lägg märke till hur VMA-signalen ansluts.



Figur 13.20

I båda fallen ser man att såväl avkodaren som fördelaren också har outnyttjade utgångar som kan användas om systemet behöver utvidgas med ytterligare moduler.

Detta avsnitt skall nu avslutas med att betrakta ett ibland förekommande avkodningsproblem. Det uppstår när en modul behöver överlappa en annan i adressrummet M. Därvid täcks en del av adressrummet av två moduler. I avkodningen måste då en av modulerna prioriteras i det gemensamma adressområdet. CS-logiken för den prioriterade modulen skall då dölja (skymma) det gemensamma området i den oprioriterade modulen. Tekniken för detta kan man kalla för ***prioriterad avkodning***. Tillvägagångssättet studeras via ett exempel.

Exempel 13.6

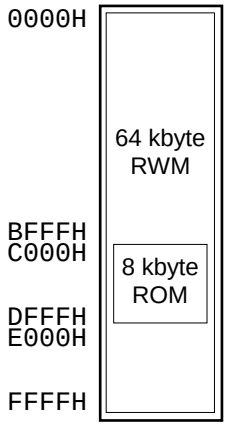
Exemplet är ett fall där ett CPU12 system innehåller dels en 64kbyte RWM-modul, dels en 8kbyte ROM-modul.

Avsikten är att adressrummet skall innehålla 8kbyte ROM och 56kbyte RWM (64k – 8k) med inplacering enligt Figur 13.21.

Anledningen till att man valt en 64kbyte RWM-modul kan vara att det är billigare än att välja flera mindre moduler.

När det gäller CS-bildningen för modulerna ser man av figuren att CS-logiken för ROM-modulen måste dölja motsvarande adressdel av RWM-modulen.

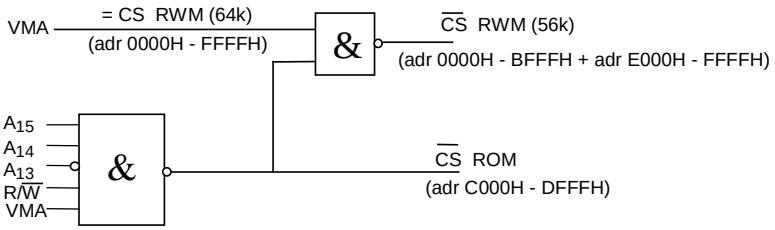
I Figur 13.22 visas hur detta görs. Avkodningen för ROM-modulen blir densamma som tidigare visats för ROM nr 1 i Figur 13.17.



Figur 13.21

Avkodningen för 64kbyte RWM (hela modulen) blir enkel, CS = VMA ty modulen tar upp hela adressområdet.

Tekniken för att dölja adressområdet C000H – DFFFH i RWM är enkel. Eftersom CS-signalen för ROM är 0 endast när detta område adresseras och 1 annars, så används den som grindsignal för att förhindra att CS-signalen för RWM samtidigt är 0. På så sätt används CS-signalen för ROM för att dölja RWM i det gemensamma området.



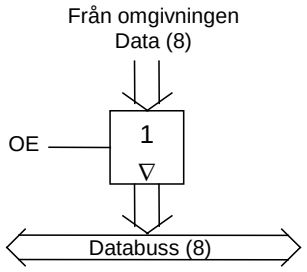
Figur 13.22

13.5 In- och utmatning av data

En dators arbete är utan värde om datorn inte kan ta emot indata från och släppa ut utdata till omgivningen. Varje dator behöver därför anordningar för att sköta kontakten med omvärlden.

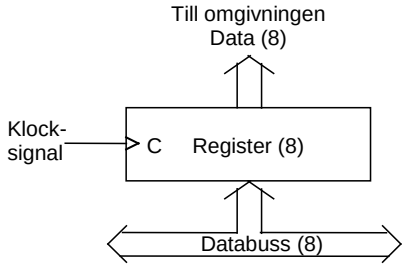
Principiellt påminner in- och utmatning av data mycket om läsning och skrivning i ett primärminne med skillnaden att minnets register bytts ut mot separata register. Man kan helt enkelt bestämma att vissa adresser används för in- och utportar istället för minnesregister.

En ***inport*** behöver dock inte vara ett register utan består ibland bara av en "three-state"-buffert som är kopplad till databussen, såsom visas i Figur 13.23. När processorn läser på "inportadressen" aktiveras buffertens OE-ingång så att data från omgivningen läggs in på databussen och läses av processorn. "Three-state"-bufferten beskrevs i kapitel 7.



Figur 13.23 Principen för en inport.

En ***utport*** är ett register vars D-ingångar är kopplade till databussen och vars vippor laddas parallellt vid den aktiva klockflanken (beskrevs i kapitel 5), se Figur 13.24. Klocksignalen skall aktiveras när processorn skriver på "utportadressen". Databussens innehåll kommer då att placeras i utportregistret. Innehållet i registret förblir sedan oförändrat tills nästa skrivning görs på samma adress.



Figur 13.24 Principen för en utport.

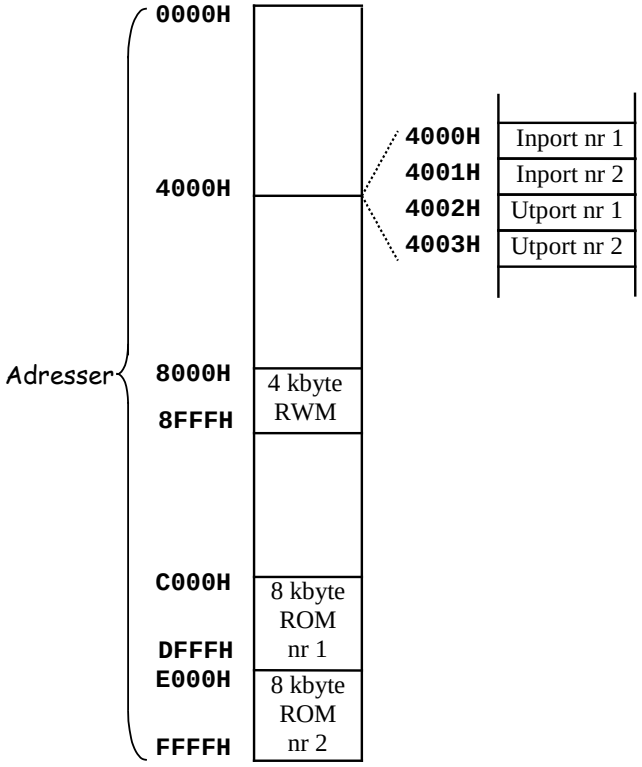
När en processor i likhet med CPU12 behandlar in- och utmatning på samma sätt som läsning och skrivning i minnet kallas detta ***minnesadresserad in- och utmatning*** (eng. ***memory mapped I/O***). Vissa andra processorer har ***speciella I/O-instruktioner*** och därmed också speciella styrsignaler för in- och utmatning. Detta gäller t ex mikroprocessorerna 80X86, som i många år använts i IBM-kompatibla persondatorer. Principen kallas ***separatadresserad in- och utmatning*** (eng. ***isolated I/O***). Fördelen med separatadresserad in- och utmatning är att man inte behöver inkräkta på minnets adressutrymme samt att adressavkodningen för in- och utmatning blir enklare. I sådana fall har man alltså två adressrum.

13.6 Anslutning av in- och utmoduler till CPU12

Beskrivningen görs här genom en utvidgning av systemet i Exempel 13.3 med in- och utmoduler.

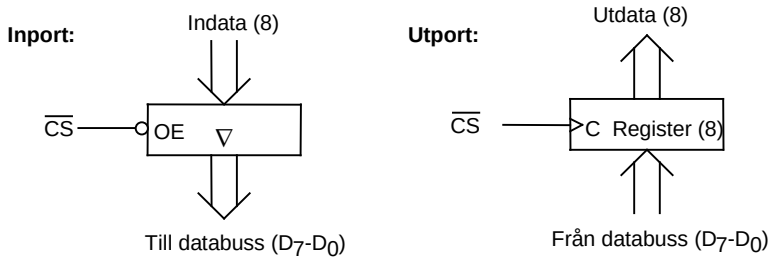
Exempel 13.7

Systemet i Exempel 13.3 skall nu utökas med två in- och två utportar, som skall placeras i adressrummet på adresserna 4000H - 4003H enligt Figur 13.25.



Figur 13.25

Portarna skall vara av den typ som ges i Figur 13.26.

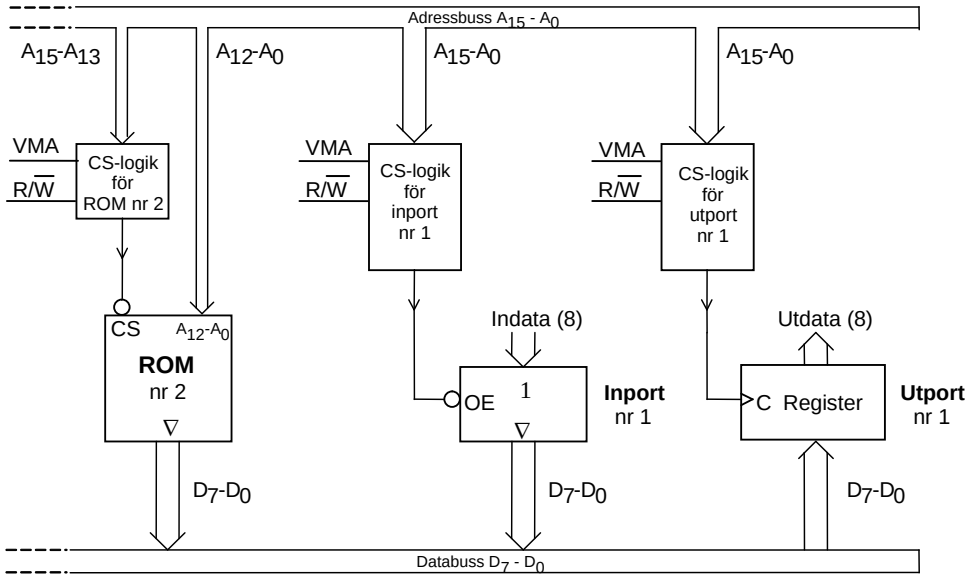


Figur 13.26 Blocksymboler för inports- och utportsmoduler.

Inplaceringen kan sammanfattas i tabellform:

Modul	***Adress***	***Adressignal nr***															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Inport 1	4000H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Inport 2	4001H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Utport 1	4002H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Utport 2	4003H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

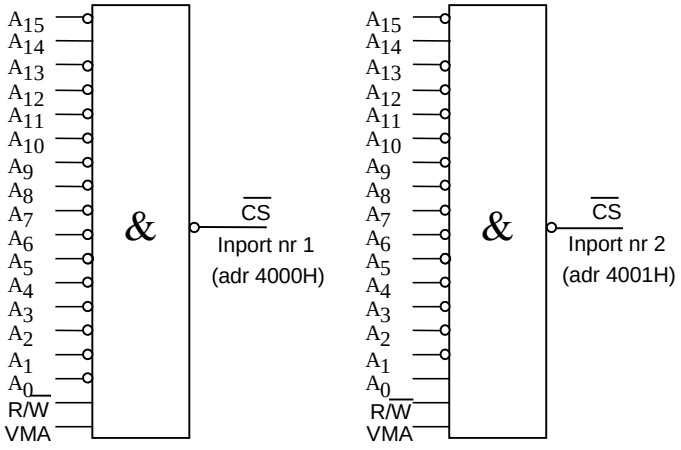
Av Figur 13.26 och tabellen framgår att varje port har en egen (unik) adress. För in- och utportarna krävs alla 16 bitarna i adressen för att "peka ut" rätt enhet, vilket visas i Figur 13.27.



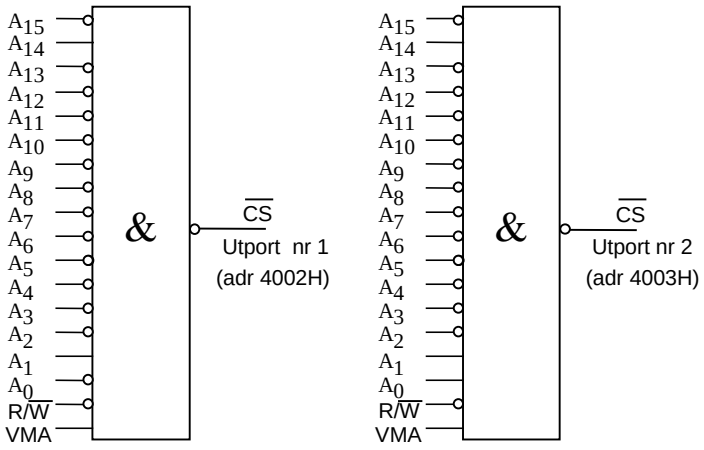
Figur 13.27 Anslutning av minnes- och in-/utmoduler till CPU12.

Som syns i Figur 13.27 sker adressavkodningen (i CS-logiken) för in- och utportarna i logikblock med signalerna A_{15}-A_0, R / $\overline{W}$ och VMA som insignaler. Utsignalerna är "chip-select"-signaler till de olika modulerna.

Figur 13.28 och Figur 13.29 visar schematiskt logiken för adressavkodningen för in- och utportarna enligt den användning av adressrummet som ges i Figur 13.25.



Figur 13.28 "Chip-select"-bildning för inportarna.



Figur 13.29 "Chip-select"-bildning för utportarna.

En jämförelse mellan de fyra "chip-select"-blocken i Figur 13.28 och Figur 13.29 ger att de är mycket lika.

Skillnaden mellan blocken i Figur 13.28 respektive Figur 13.29 är att adressbit 0 är inverterad i det vänstra blocket.

Den principiella skillnaden mellan Figur 13.28 (inportar) och Figur 13.29 (utportar) är att R / $\overline{W}$ -signalen är inverterad i Figur 13.29. Detta medför att blocken i Figur 13.28 är avsedda för läsning i adressrummet (inportar: R / $\overline{W}$ = 1) medan blocken i Figur 13.29 är avsedda för skrivning (utportar: R / $\overline{W}$ = 0).

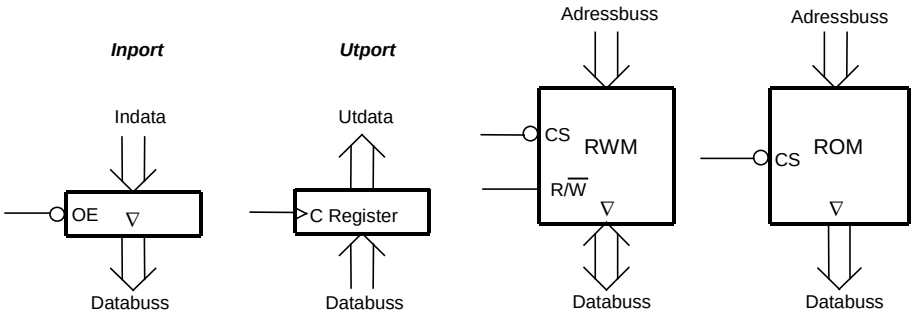
13.7 Resumé

Kapitlet behandlar hur minnes- och I/O-moduler kan placeras i ett adressrum M som definieras av processor-egenskaper som läs/skrivprincip och adressbussens utseende. Hur adressrummet används (är fyllt) beror mycket på den tillämpning i vilken processorn skall användas. Läsaren har därmed fått en inblick i hur en processors adressrum M kan användas dels för primärminne, dels för I/O-portar.

Speciellt har ***läs/skrivminnesmoduler***, **RWM** och ***läsminnesmoduler***, **ROM** beskrivits och använts i exempel på hur modulerna avkodas för att kunna anpassas till processorns egenskaper. Detsamma gäller enkla I/O-portar.

Övningsuppgifter

Om inget annat anges så skall moduler av denna typ användas i uppgifterna.



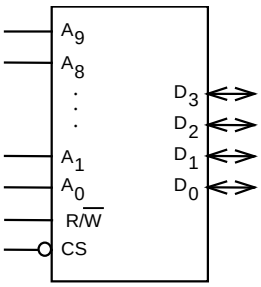
13.1 Figuren visar in- och utgångarna på en typisk minneskapsel.

a) Hur många minnesord finns det i kapseln? Hur breda är minnesorden?

b) Till vilka bussar är de visade in- och utgångarna vanligen anslutna när de används i ett mikrodatorsystem med mikroprocessorn CPU12?

c) Hur många kapslar av denna typ krävs för att bilda fullständiga dataord?

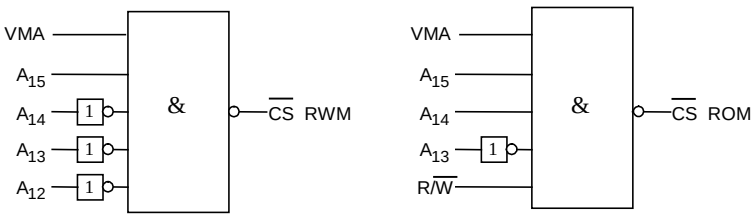
d) Beskriv kortfattat hur CS-ingången används.



13.2 Här ges logiknäten för adressavkodningen för två fullständigt avkodade minneskapslar.

a) Ange i vilket adressområde respektive kapsel är placerad. De två minneskapslarna aktiveras när CS-ingångarna har låg logiknivå (0).

b) Ange minneskapaciteten (antal ord) för respektive minneskapsel.



13.3 En ROM-kapsel om 16 kbyte och en RWM-kapsel om 8kbyte skall anslutas till CPU12.

a) Konstruera CS-logiken för ROM-kapseln. Den skall ha begynnelseadressen C000H.
Visa hur samtliga signaler skall kopplas mellan processorn och minneskapseln.

b) Konstruera CS-logiken för RWM-kapseln. Den skall ha begynnelseadressen 8000H.
Visa hur samtliga signaler skall kopplas mellan processorn och minneskapseln.

Inverterare och NAND-grindar med valfritt antal ingångar får användas. Fullständig adressavkodning skall användas.

13.4 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 32 kbyte ROM-kapsel, 1 st 2 kbyte ROM-kapsel, 1 st 4 kbyte RWM-kapsel, 1 st 8-bitars "three-state"-buffert som inport och ett 8-bitars register som utport. Modulerna har valts för att passa CPU12's timing.

32 kbyte ROM-kapseln skall placeras med början på adress 0000H.
4 kbyte RWM-kapseln skall placeras med början på adress 8000H.
2 kbyte ROM-kapseln skall placeras så att den slutar på adress FFFFH.

In- och utporten skall båda placeras på adressen C000H.

Uppgift: Konstruera den logik som krävs för "chip-select"-avkodningen. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Adressintervallet för varje modul skall anges i hexadecimal form. Använd ofullständig adressavkodning.

13.5 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 8 kbyte ROM-kapsel, 2 st 4 kbyte RWM-kapslar, 1 st 8-bitars "three-state"-buffert som inport och ett 8-bitars register som utport.

Placera ROM-kapseln så att sista adressen hamnar på adress FFFFH. Placera själv övriga komponenter i adressrummet på lämpligt sätt.

Rita adressavkodningslogik för samtliga komponenter. Det valda adressintervallet för varje komponent skall anges i hexadecimal form.

13.6 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 32 kbyte ROM-kapsel med slutadressen FFFFH, 1 st 16 kbyte RWM-kapsel med placering direkt före ROM-kapseln, 1 st 8-bitars "three-state"-buffert på adressen 0000H som inport och 1 st 8-bitars register på adressen 0001H som utport.

Modulerna har valts för att passa CPU12's timing.

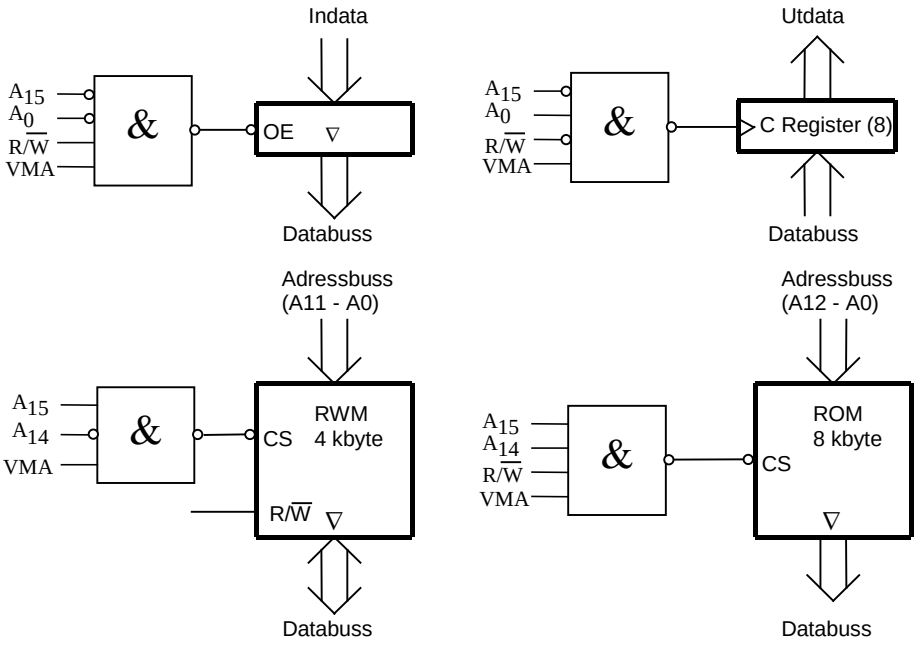
Uppgift: Konstruera den logik som krävs för "chip-select"-avkodningen. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Adressintervallet för varje modul skall anges i hexadecimal form.

13.7 Antag att ett mikrodatorsystem med CPU12 innehåller en ROM-kapsel och en RWM-kapsel för att lagra program och variabler, samt en 8-bitars "three-state"-buffert som inport och ett 8-bitars register som utport.

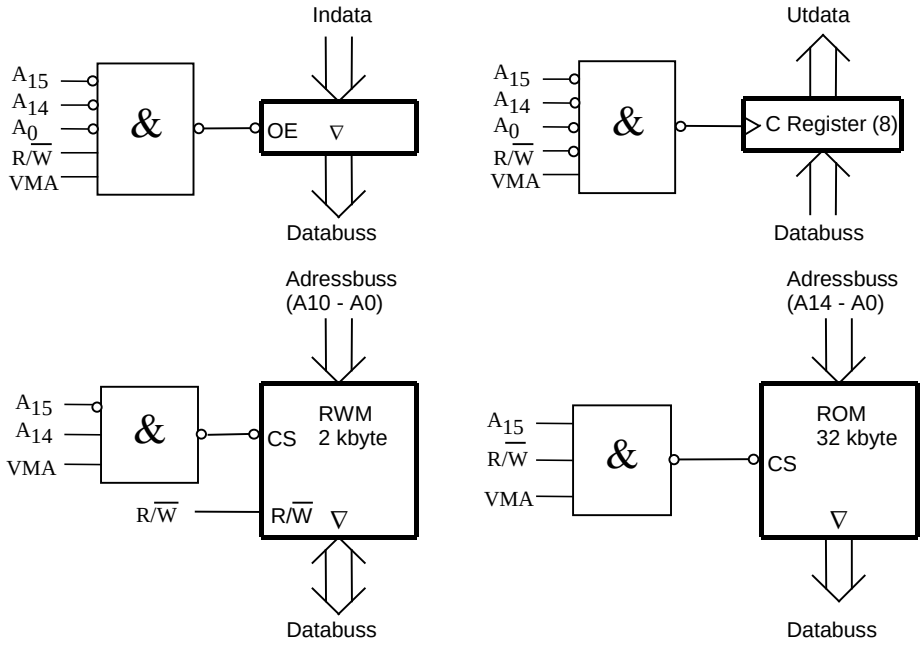
I figurerna a), b) och c) visas för tre fall hur dessa moduler är anslutna till processorbussen, samt hur “chip-select”-avkodningen utförs.

Redogör för var i adressrummet processorn kan läsa eller skriva i de olika enheterna. Alla addresser, där en enhet kan nås, skall redovisas.

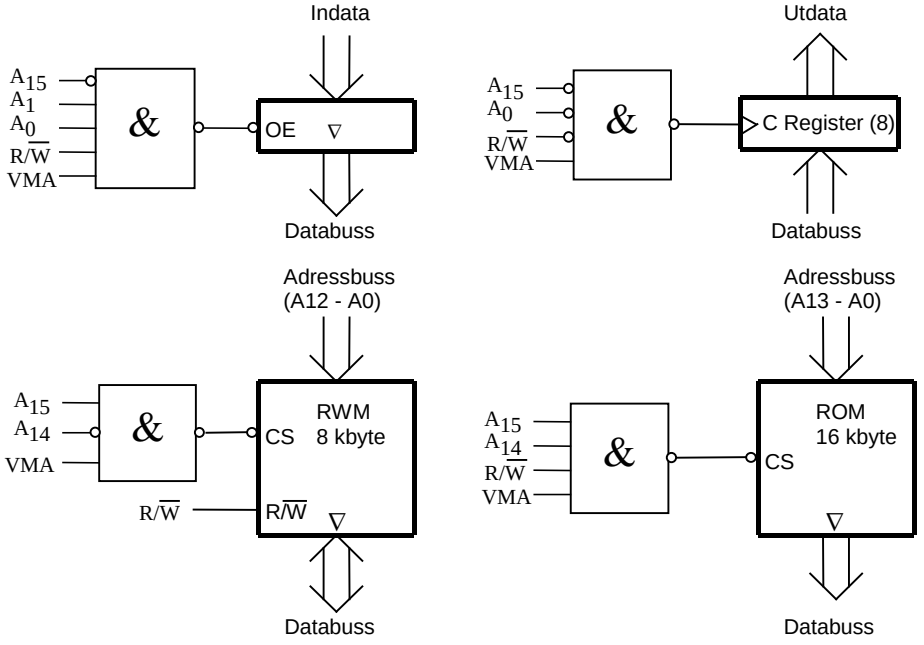
a)



b)


Indata
A15
A14
A0
R/W
VMA
&
OE
Databuss
Utdata
C Register (8)
Adressbuss
(A10 - A0)
RWM
2 kbyte
CS
Adressbuss
(A14 - A0)
ROM
32 kbyte

c)


Indata
A15
A1
A0
R/W
VMA
&
OE
Databuss
Utdata
C Register (8)
Adressbuss
(A12 - A0)
RWM
8 kbyte
CS
Adressbuss
(A13 - A0)
ROM
16 kbyte

13.8 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 16 kbyte ROM-kapsel, 1 st 16 kbyte RWM-kapsel, 1 st 8-bitars "three-state"-buffert som inport och ett 8-bitars register som utport.

ROM-kapseln skall placeras på de högsta adresserna i adressrummet.

Mikrodatorsystemet skall tillverkas i två olika versioner, en dyrare utbyggbar och en billigare som inte skall byggas ut. I den utbyggbara versionen kan i extremfallet samtliga adresser i adressrummet fyllas med minnesmoduler eller moduler för in- och utmatning.

Konstruera minimala logiknät för CS'-signalerna för:

a) Den utbyggbara, dyrare versionen.

b) Den billigare versionen, som inte skall byggas ut.

NAND-grindar med valfritt antal ingångar samt INVERTERARE får användas. Adressintervallet för varje modul skall anges i hexadecimal form. Modulerna har valts för att passa CPU12's timing.

Modulerna, förutom ROM-modulen, får placeras på lämpligt sätt i adressrummet. Alla program som är körbara i den dyrare versionen skall också kunna köras i den billigare.

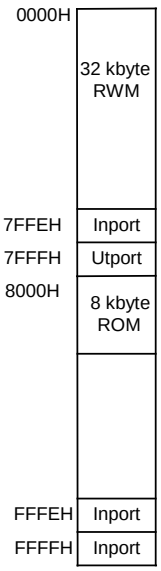
13.9 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 8 kbyte ROM-kapsel, 1 st 32 kbyte RWM-kapsel, 3 st 8-bitars "three-state"-buffertar som inportar och 1 st 8-bitars register som utport.

I figuren visas var i adressrummet modulerna skall placeras.
Modulerna har valts för att passa CPU12's timing.

Av figuren framgår det att inporten med adressen 7FFEH och utporten med adressen 7FFFH överlappar de två sista adresserna i RWM-modulen. För att avkodningen skall fungera måste därför inporten prioriteras så att en läsning på adressen 7FFEH inte aktiverar RWM-modulen.

Uppgift: Konstruera den logik som krävs för "chip-select"-avkodningen. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Adressintervallet för varje modul skall anges i hexadecimal form.

Inportarna på adresserna FFFEH och FFFFH skall användas på ett speciellt sätt.
Förklara hur du tror att de skall användas!

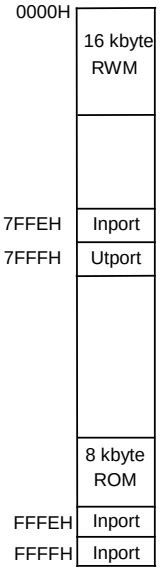


13.10 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 8 kbyte ROM-kapsel, 1 st 16 kbyte RWM-kapsel, 3 st 8-bitars "three-state"-buffertar som inport och 1 st 8-bitars register som utport.

I figuren visas var i adressrummet modulerna skall placeras. ROM-modulen skall placeras så att dess sista adress blir FFFFH. Modulerna har valts för att passa CPU12's timing.

Av texten och figuren framgår att inportarna på adresserna FFFEH och FFFFH överlappar de två sista adresserna i ROM-modulen. För att avkodningen skall fungera måste därför inportarna prioriteras s läsningar på adresserna FFFEH och FFFFH inte aktiverar ROM-modulen.

Uppgift: Konstruera den logik som krävs för "chip-select"-avkodningen. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Adressintervallet för varje modul skall anges i hexadecimal form.

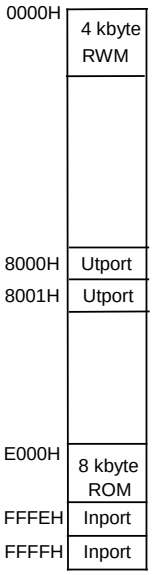


13.11 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 8 kbyte ROM-kapsel, 1 st 4 kbyte RWM-kapsel, 2 st 8-bitars "three-state"-buffertar som inportar och 2 st 8-bitars register som utportar.

I figuren visas var i adressrummet modulerna skall placeras.
Modulerna har valts för att passa CPU12's timing.

Av figuren framgår att inportarna på adresserna FFFEH och FFFFH överlappar de två sista adresserna i ROM-modulen. För att avkodningen skall fungera måste därför inportarna prioriteras så att läsningar på adresserna FFFEH och FFFFH inte aktiverar ROM-modulen.

Uppgift: Konstruera den logik som krävs för "chip-select"-avkodningen. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Adressintervallet för varje modul skall anges i hexadecimal form.



13.12 Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 32 kbyte ROM-kapsel och 1 st 32 kbyte RWM-kapsel!

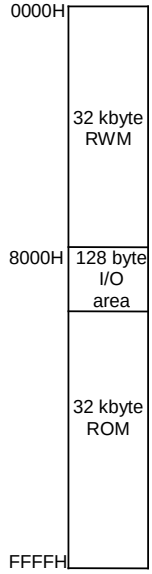
Inom ROM-kapselns adressområde skall dessutom i början finnas ett adressintervall för in- och utmatn

I figuren visas var i adressrummet modulerna skall placeras. Vissa moduler har valts för att passa CPU1 timing. Utportarna har negativ flanktriggning.

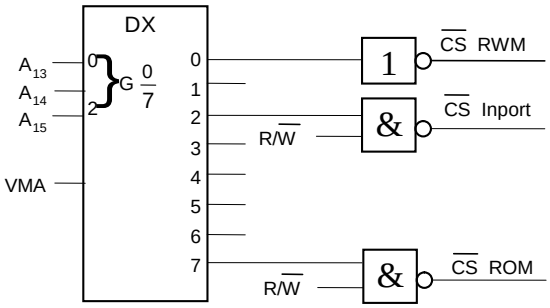
För att undvika busskollisioner måste inportarna prioriteras så att läsningar i de adressintervall, som innehåller eller kan innehålla inportar, inte aktiverar ROM-modulen.

Uppgift: Konstruera CS-logiken för RWM- och ROM-modulen. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Det användbara adressint vallen för de två minnesmodulerna skall anges i hexadecimal form.

CS-signaler för inportar och utportar behöver inte konstrueras!



13.13 För ett CPU12-system med 1 st 16 kbyte RWM och 1 st 4 kbyte ROM har CS-logiken i figuren konstruerats. Analysera den!



a) Ange hur adressrummet utnyttjas.

b) Förklara vad det är för fördel med att använda en fördelare (demultiplexer) för adressavkodning.

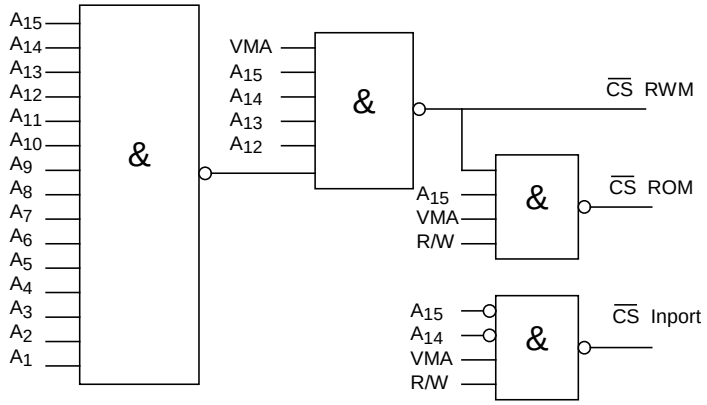
c) Vad är mindre lyckat med kopplingen i figuren?

d) Visa hur man kan åtgärda "felet" i **c)**.

e) Anslut en utport till systemet. Använd ofullständig adressavkodning så att porten ser ut att finnas på var och en av adresserna 4000H - 5FFFH.

f) Anslut ytterligare port, en inport, till adress 60F0H. För den skall adressavkodningen vara fullständig och realiseras den m h a fördelaren.

13.14 I samband med ett CPU12-system har man funnit CS-logiken i figuren. I systemet finns 1 st 2 kbyte RWM och 1 st 4 kbyte ROM. Analysera CS-logiken!



a) Visa hur adressrummet utnyttjas.

b) Vad finns det för skäl till denna disposition av adressrummet?

c) Visa hur man kan koppla in en utport till detta system. Välj lämplig(a) adress(er).

d) Vilka värden får N, Z, V och C- flaggan i CC-registret efter följande sekvens av instruktioner?

```
LDAA   $8000      (Fungerar på samma sätt som i FLISP, med undantaget att adresserna har 16 bitar.)
CMPA   $C000
```

Gör lämpliga antaganden, som stämmer överens med datorsystemet ovan.