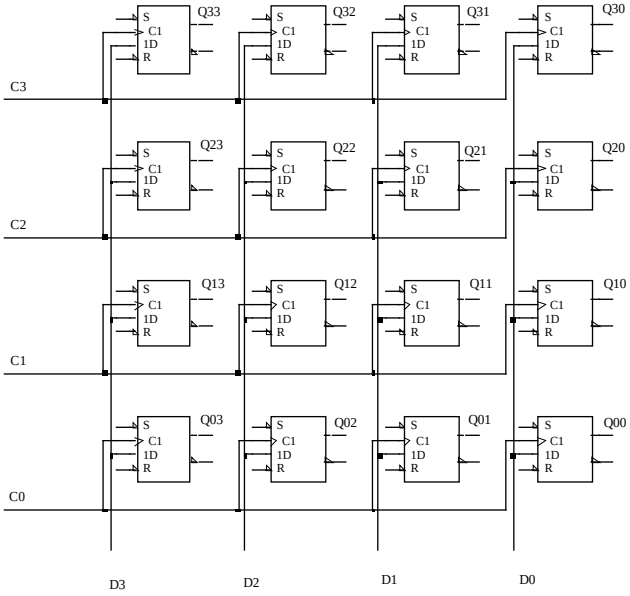
Opbouw van een halfgeleidergeheugen.

Een halfgeleidergeheugen bestaat uit een aantal D-FF's.. Veronderstel dat we 16 FF's. willen stoppen in één IC., dan heeft de geïntegreerde schakeling 50 pinnen nodig.
16 ingangen
16 uitgangen
16 comandoingangen
2 voedingsspanningen
totaal 50 pinnen

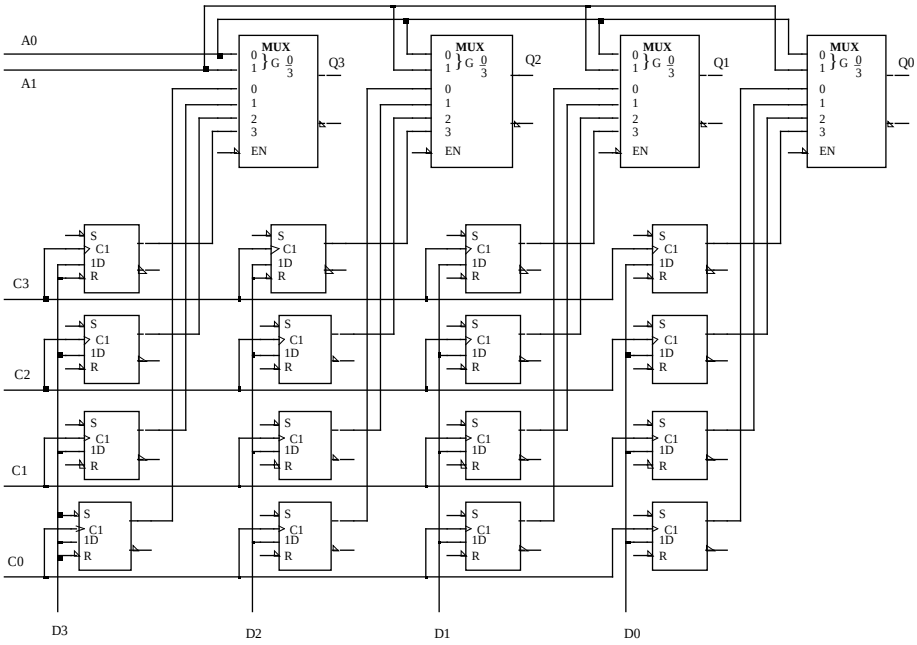


Figuur 1.1

Dit IC heeft te veel pinnen en zou te duur zijn. Vandaar tracht men het aantal pinnen te reduceren door de FF. te groeperen zoals in figuur 1.1.

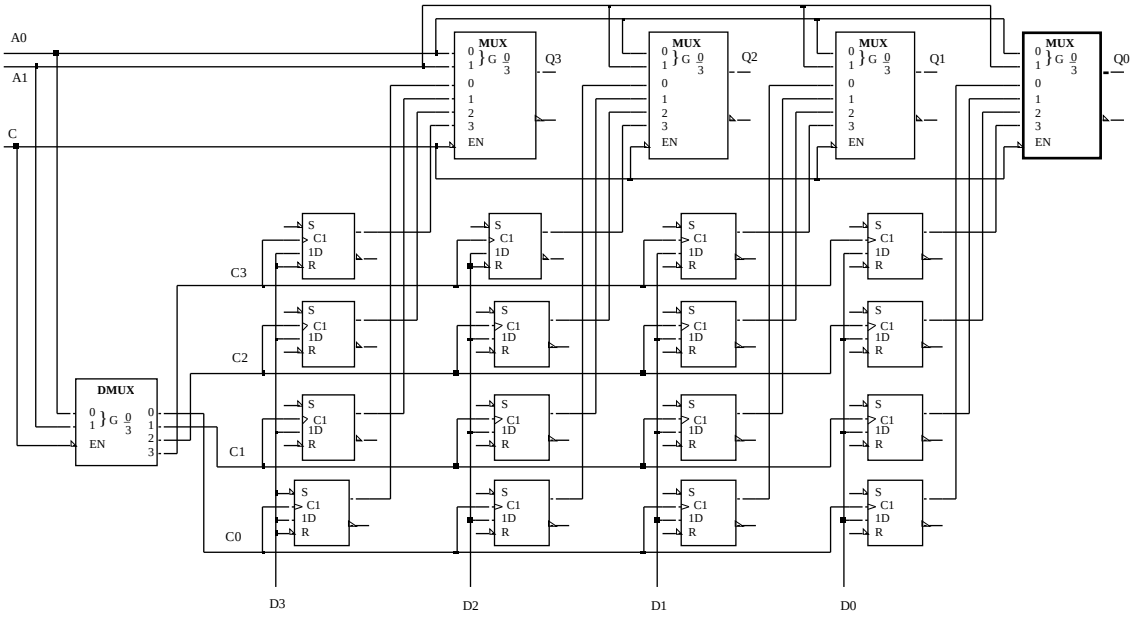
Hierdoor wordt het aantal klok en data ingangen gereduceerd tot acht, maar het aantal uitgangen blijft onveranderd.
Eigelijk is het de bedoeling om vier woorden van vier bits te kunnen adresseren.
Het aantal uitgangen kunnen we beperken door de uitgangen naar een multiplexer te sturen.

In de schakeling van figuur 1.2 is het onmogelijk om alle uitgangen tegelijkertijd naar buiten te brengen. Men kan maar 4 uitgangen tegelijkertijd uitlezen. Elke rij van 4 uitgangen heeft nu een adres, gevormd door A0 en A1, gekregen. A0 En A1 vormen een binair getal en dit getal geeft de rij aan welke we uitlezen. Wil men bv. de nulde rij uitlezen dan maakt met A1=0 en A0=0 en zal de multiplexer er voor zorgen dat de uitgangen van de nulde rij FF. naar buiten gebracht worden.



Figuur 1.2

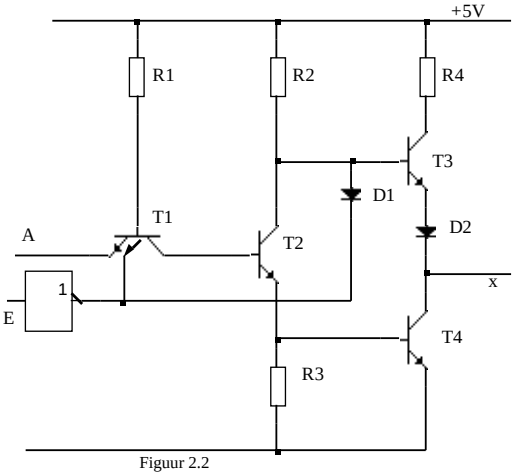
Door nu tevens de klok door een demultiplexer te sturen kan men het aantal klokingangen eveneens reduceren.



Figuur 2.1

Om data in de FF. van een bepaalde rij in te latchen moet men nu reeds het adres van de rij aanleggen en dan een klokimpuls geven. Het IC. bevat nu nog.
4 D-ingangen
4 Q-uitgangen
1 klokingang
2 adresingangen
2 voedingsspanningen
totaal 13 pinnen.

Dit aantal pinnen gaan we nog verminderen. Een geheugenelement dient om informatie te bewaren, dus is het niet nodig dat men in een geheugen tegelijkertijd kan lezen (Q-uitgangen) en schrijven (D-ingangen). Aangezien men maar één van beide bewerkingen tegelijkertijd moet doen, kan men de D-ingangen en de Q-uitgangen samen nemen. Om dit samen nemen te begrijpen bestuderen we eerst een **tri-state buffer.**

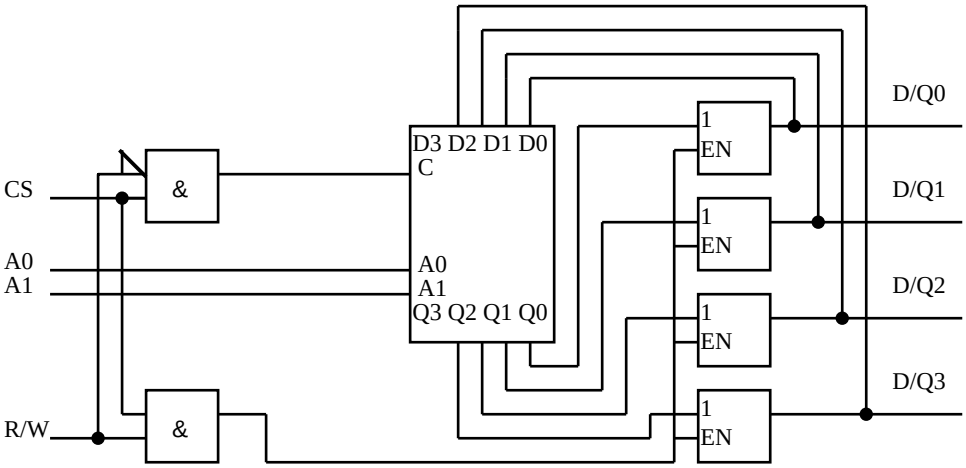


Figuur 2.2

Voor E = 0 ontstaat een 1 op de tweede emittor van T1. De 1 laat toe dat de transistor T1 gestuurd wordt door ingang A, m.a.w. de schakeling werkt als invertor.

Voor E = 1 wordt de tweede emittor van T1 op een laag niveau aangesloten. Door de werking van de schakeling spert T4. De sturing van transistor T3 wordt echter via D1 afgeleid zodat T3 eveneens zal sperren. We zien dus dat beide transistoren sperren, zodat de uitgang x zweeft, of in HI.

Deze tri-state buffer plaats men achter de Q-uitgangen van de FF's. en de tri-state uitgangen koppelt men aan de D-ingangen van de FF's. Tevens voegt men twee en poorten toe.

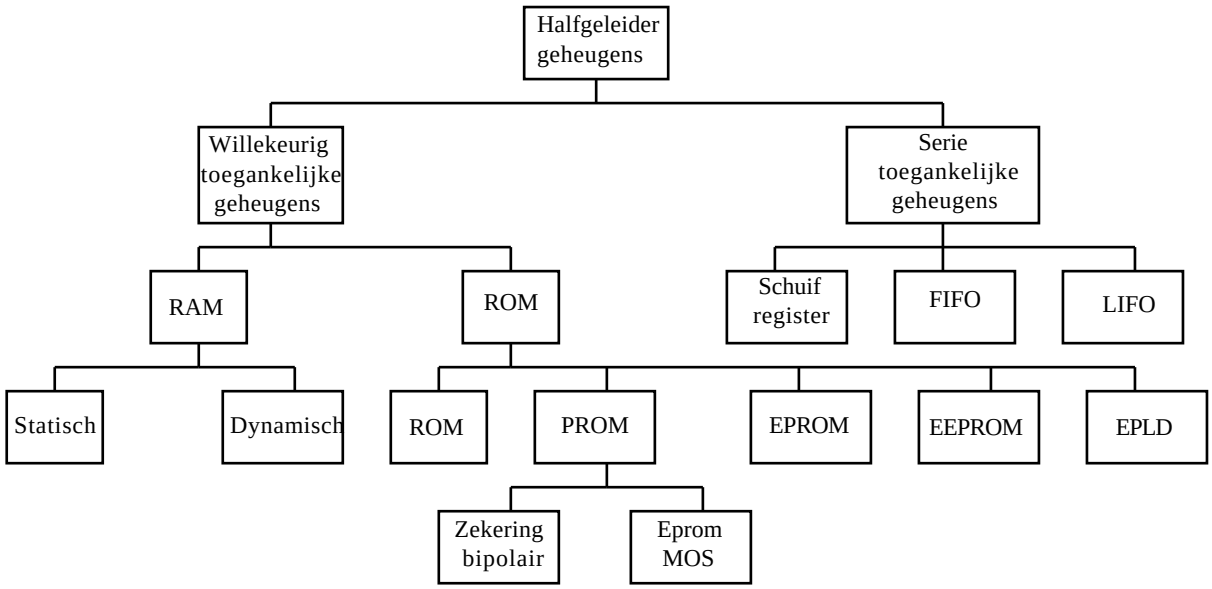


Figuur 3.1

De R/W lijn moet men 0 maken om in het geheugen te schrijven, en 1 maken om te lezen, terwijl men de chip-select (CS) 1 moet maken om te kunnen lezen of schrijven in dit IC. Om in dit geheugenelement te schrijven moet men nu de R/W-lijn laag maken, het adres van de rij aanleggen, de in te schrijven data D0 tot D3 aanleggen aan de D/Q pinnen, en dan de CS lijn even hoog maken.

Om een geheugen element uit te lezen maakt men de R/W lijn 1, legt het adres van de rij aan, en maakt de CS lijn hoog, dan kan men op de D/Q uitgangen de data uit de FF. lezen.

Indeling van de halfgeleidergeheugens



Figuur 3.2

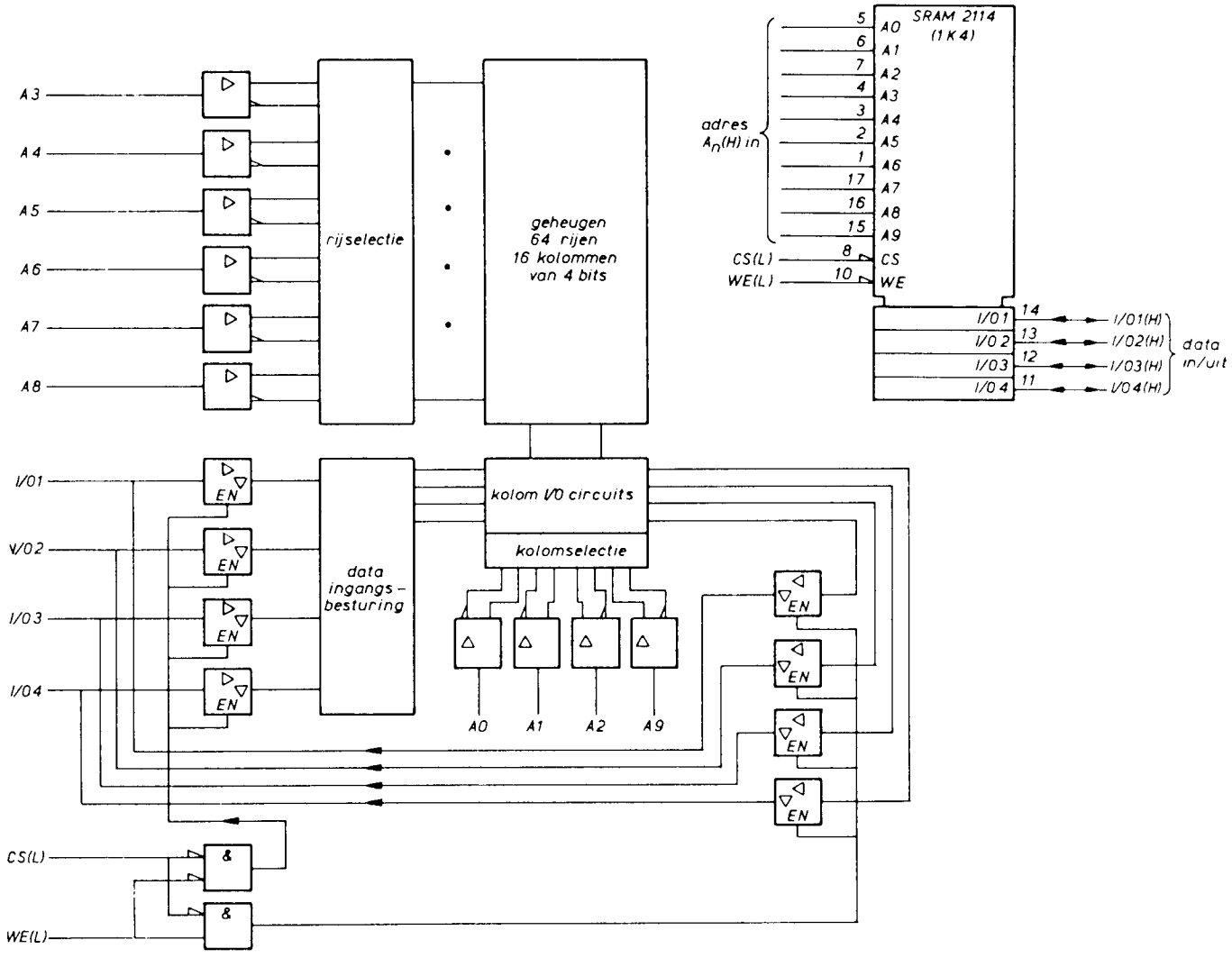
RAM (Random Acces read/write Memory)

Vrij uitleesbare/inschrijfbare geheugens. In dit geheugen waarin men zowel snel kan lezen als schrijven is elke geheugenplaats willekeurig toegankelijk, dit in tegenstelling tot de serie toegankelijke geheugens zoals bv. een digitale cassetterecoreder of een schuifregister.

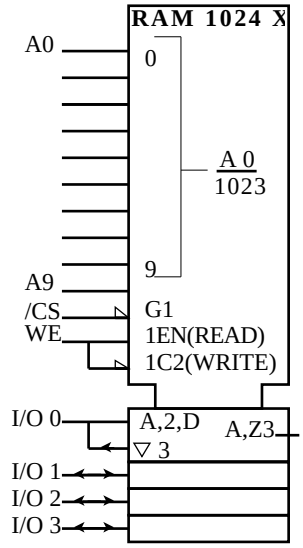
Statische RAM

Deze IC's bestaan uit FF's. opgebouwd uit vier transistoren. De geheugeninhoud blijft bewaard zolang de voedingsspanning behouden blijft. Dit nadeel kan men opvangen door het geheugen van een batterijvoeding te voorzien. Daar NMOS IC's te veel stroom vergen zal men steeds meer en meer CMOS geheugens gebruiken. Dit omdat CMOS geheugens hun informatie kunnen bewaren bij verlaagde spanning en de benodigde stroom bestaat uit een kleine lekstroom. CMOS IC's zijn heden even snel als NMOS geheugens vandaar dat deze laatste van de markt verdwijnen. Dit maakt dat de moderne komputers nog enkel CMOS statische RAM's gebruiken.

Praktisch voorbeeld: Blokschema van het 1K4-RAM type 2114 met het IEC symbool.



De **S.RAM 2114** bevat:



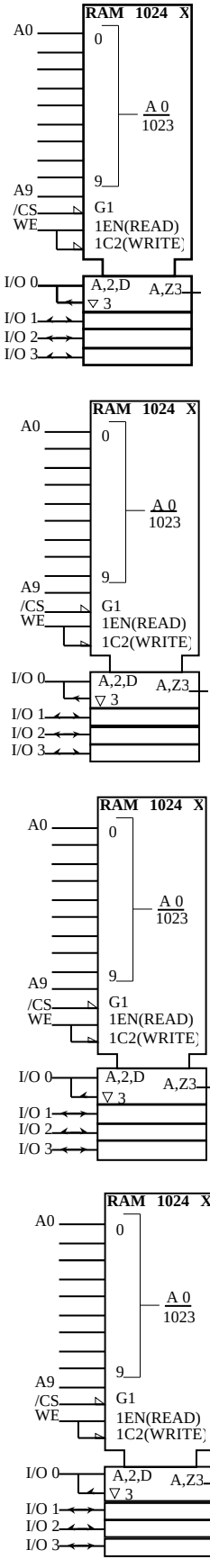
10 Adresingangen (2^{10} = 1024), genummerd van A0 tot en met A9 voor de adressen 0 tot en met $3FF.
Een /CS-ingang of de chip-select ingang. Deze ingang wordt gebruikt om de chip te selecteren.
Een WE-ingang of de lees/schrijfingang.
Vier I/O-lijnen. Die aanduiding leert ons dat de data in- en uitgelezen wordt via dezelfde lijnen. Deze lijnen zijn dus bidirectioneel. Dat de uitgangen "3-status uitgangen" zijn, zien we aan de omgekeerde driehoek en het cijfer 3.
De data wordt in de RAM ingelezen als /CS = 0 en WE = 0 zijn. Dit zien we in het symbool door de polariteitsindicatoren en de aanduiding "write".
Het uitlezen van de data gebeurt door de chip te activeren /CS = 0 en de lees/schrijfingang hoog te maken (WE = 1).
Om de uitgangen in HI. toestand te brengen volstaat het /CS hoog te maken.
Dergelijk geheugen kan gebruikt worden als data-geheugen van een microprocessor-syteem.

Onderstaande tabel geeft het verband weer tussen de geheugencapaciteit, het aantal bits, het aantal adreslijnen, en het eindadres in hex.-decimale notatie.

geheugen in M bits	geheugen in K bits	aantal bits	A0 tot An	eindadres in HEX.
	1	1.024	10	$3FF
	2	2.048	11	$7FF
	4	4.096	12	$FFF
	8	8.192	13	$1FFF
	16	16.384	14	$3FFF
	32	32.768	15	$7FFF
	64	65.536	16	$FFFF
	128	131.072	17	$1 FFFF
	256	262.144	18	$3 FFFF
	512	524.288	19	$7 FFFF
1M	1.024	1.048.576	20	$F FFFF
2M	2.048	2.097.152	21	$1F FFFF
4M	4.096	4.194.304	22	$3F FFFF
8M	8.192	8.388.608	23	$7F FFFF
16M	16.384	16.777.216	24	$FF FFFF
32M	32.768	33.554.432	25	$1FF FFFF
64M	65.536	67.108.864	26	$3FF FFFF
128M	131.072	134.217.728	27	$7FF FFFF
256M	262.144	268.435.456	28	$FFF FFFF
512M	524.288	536.870.912	29	$1FFF FFFF
1G	1.048.576	1.073.741.824	30	$3FFF FFFF

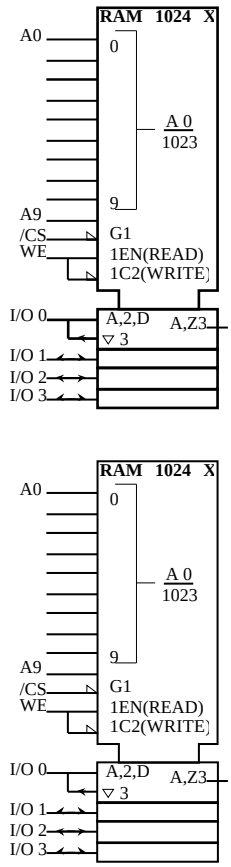
Oefening 6.1

Bouw met 4 maal de 2114 een 4K4 geheugen.



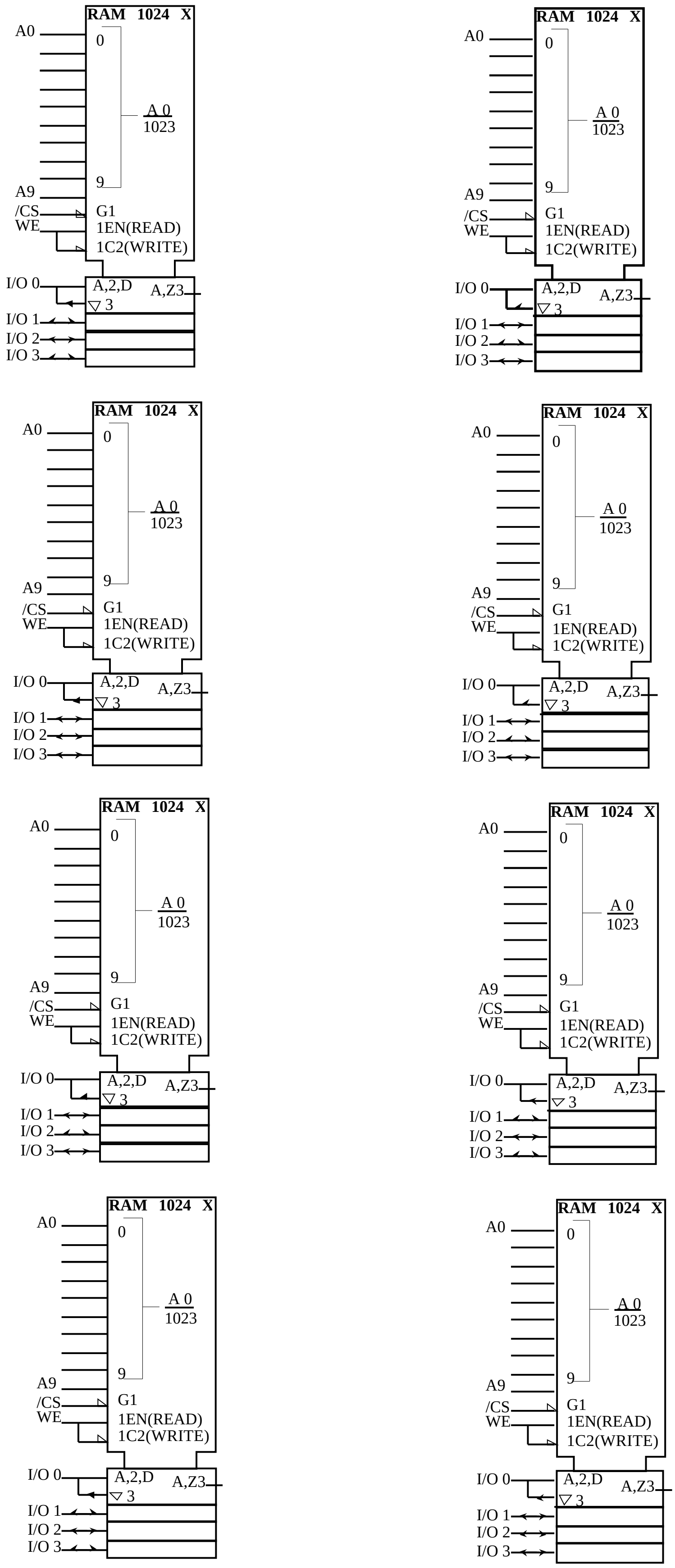
Oefening 7.1

Bouw met 2 maal de 2114 een 1K8 geheugen.



Oefening 8.1

Bouw met 8 maal de 2114 een 4K8 geheugen.

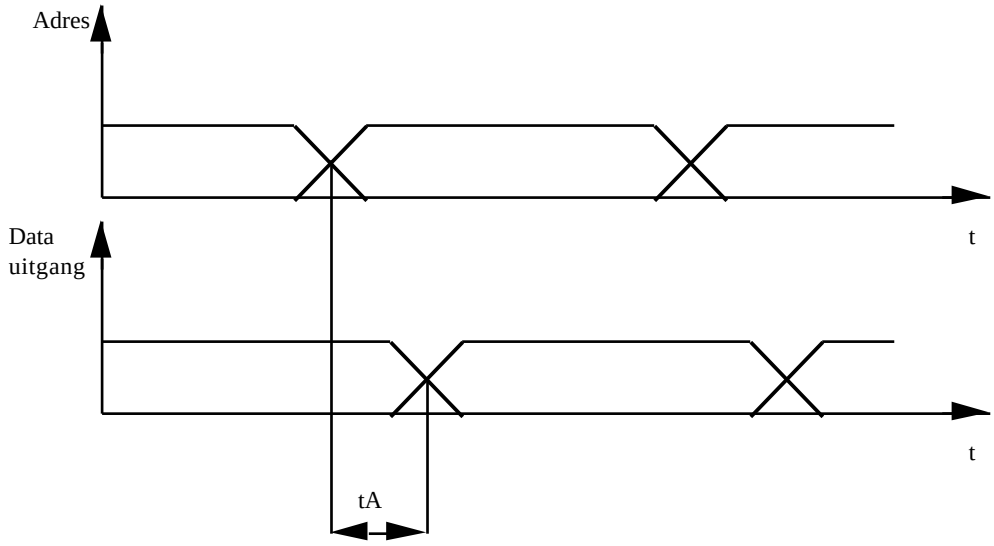


Tijdvolgorde diagrammen

Het selecteren van geheugenbits in een RAM dient strikt te gebeuren volgens de specificaties van de fabrikant. Deze informatie wordt aan de verbruiker medegedeeld o.a. onder de vorm van tijdvolgorde diagrammen. Wij bestuderen de belangrijkste tijden.

De toegangstijd (accestime tA)

De toegangstijd is de tijd die verloopt tussen het aanleggen van een adres (bij een reeds geselecteerde RAM) en het ogenblik waarop de informatie stabiel aanwezig is in iedere geheugencel (bij de inleesoperatie).



figuur 9.1

Bipolaire geheugens bezitten een typische toegangstijd van 40 ns; MOS geheugens zijn veel trager en bezitten toegangstijden rond 300 ns.

De cyclustijd (cycle-time)

De cyclustijd is de tijd dat een adres stabiel aanwezig moet zijn opdat een lees- of schrijfoperatie kan uitgevoerd worden.

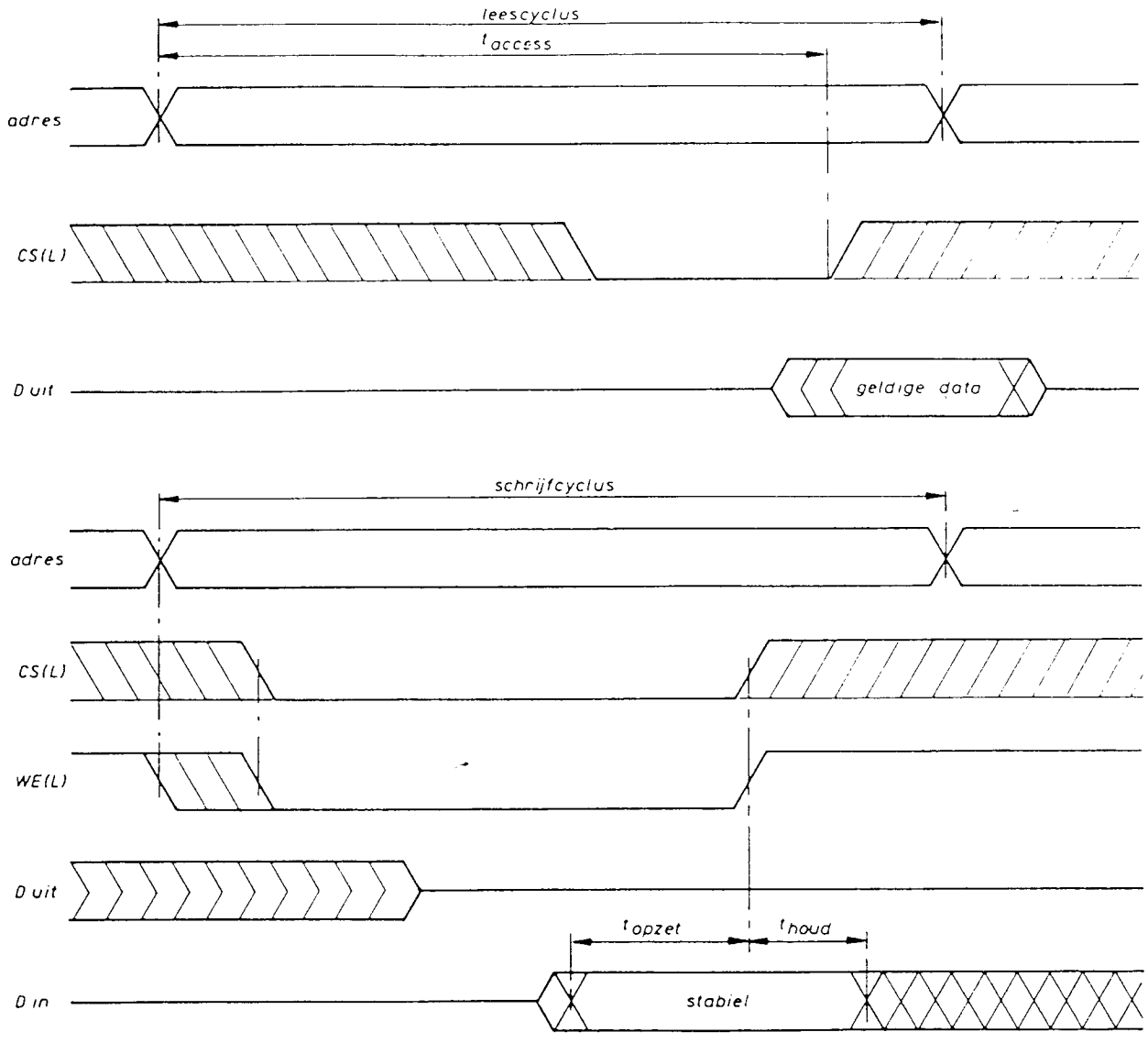
De leescyclus (read-cycle-time, t_{RC})

Dit is de tijd gemeten vanaf het moment dat een chipselectie plaatsvindt tot het moment dat een te lezen data stabiel aan de uitgangen beschikbaar is.

De schrijfcyclus (write-cycle-time, t_{WC})

De tijd gemeten vanaf het moment dat een chip-selectie plaatsvindt tot het moment dat de data in de cellen stabiel aanwezig is.

Deze tijden worden weergegeven op de volgende bladzijde met de gebruikelijke symboliek voor tijdsdiagrammen. Deze tijdsdiagrammen vinden we terug in de meeste databoeken.



tijdsdiagram symbool	consequentie voor de ingang	consequentie voor de uitgang
	moet stabiel zijn, hoog of laag	stabiel, hoog of laag
	veranderingen van hoog naar laag toegestaan	zal veranderen van hoog naar laag gedurende aangegeven interval
	veranderingen van laag naar hoog toegestaan	zal veranderen van laag naar hoog gedurende aangegeven interval
	niet relevant	status onbekend of veranderend
	(niet van toepassing)	middenlijn betekent uitgang zwevend

betekenis

Dynamische RAM

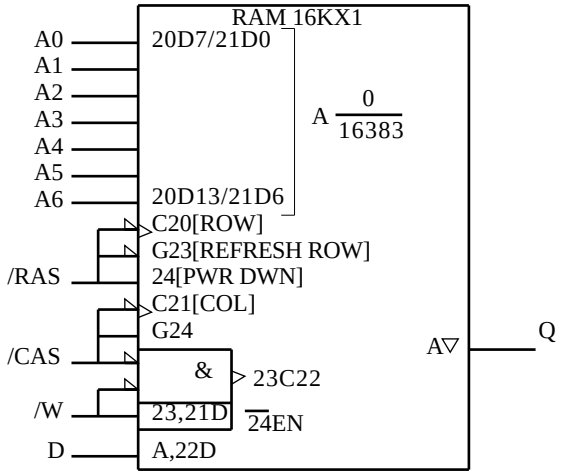
In dynamische RAM's heeft men het aantal transistoren per bit kunnen beperken tot 1. Vandaar dat er in een Dynamische RAM chip 4 X meer bits zitten dan in statische RAM.
Een geheugenbit in dynamische RAM bestaat uit een kondensator.
Een opgeladen kondensator komt overeen met een "1".
Een ontladen kondensator komt overeen met een "0".
Deze kondensatoren lekken echter leeg zodat men om de 2 ms de informatie terug moet schrijven in de dynamische RAM. Op de chip is er echter extra hardware voorzien om dit herinschrijven te vergemakkelijken. Zo volstaat het bij de meeste dynamische RAM's om 128 opéénvolgende adressen aan te bieden om alle bits uitgelezen en heringeschreven te hebben. Dit herinschrijven noemt men de refresh. De refresh kan op verschillende manieren gebeuren:
- Ofwel zorgt de microprocessorhardware ervoor.
- Ofwel zorgt een CRT controller ervoor.
- Ofwel zal men er een refreshcontroller bijbouwen welke de microprocessor om de 2 ms even laten stoppen en de 128 opéénvolgende adressen aanbiedt aan de geheugens, en waarna de microprocessor verder kan werken.

Dynamische RAM gebruikt men voor grotere geheugens. Omdat grote geheugens nogal veel adreslijnen hebben zal men het adres in twee delen aanbieden. Hierdoor reduceert men het aantal pinnen tot 16 voor een 256 K X 1 bit geheugen.

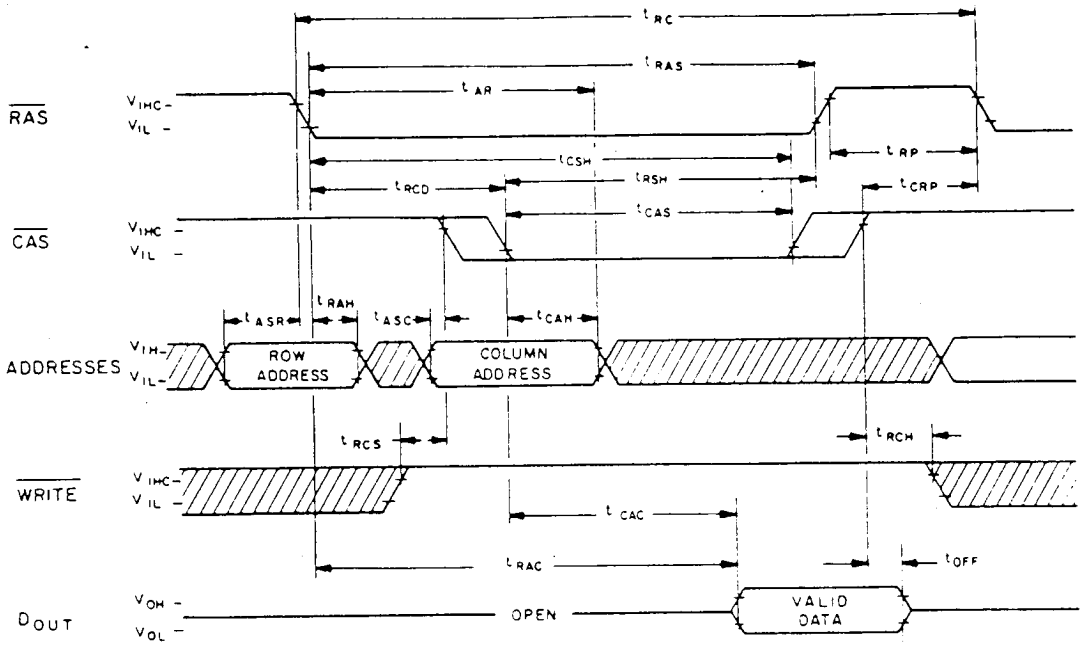
De dynamische RAM 4116

De 4116 is een dynamische RAM die 16384 woorden bevat van 1 bit.
In het symbolenschema zien we dat er 7 adresingangen beschikbaar zijn. Nochtans verwachten wij hier 14 ingangen gezien 2^{14} = 16384.
In de 4116 wordt hiervoor als volgt tewerk gegaan:
Eerst moeten de rij-adressen aangesloten worden aan de ingangen; door het /RAS-signaal (Row Address Strobe) 0 te maken wordt de rij-adresinformatie in de chip opgenomen en bewaard in een tussengeheugen.
Daarna is een gelijkaardige bewerking nodig voor de kolom-adresinformatie met de /CAS-ingang (Colom Adress Strobe).
De adresinformatie moet echter stabiel aanwezig zijn vóór de 1-0 overgang van het /RAS of /CAS-signaal.
In het IC zijn alle in-en uitgangen TTL compatible. De uitgang kan in een HI toestand geschakeld worden teneinde uitbreiding mogelijk te maken.

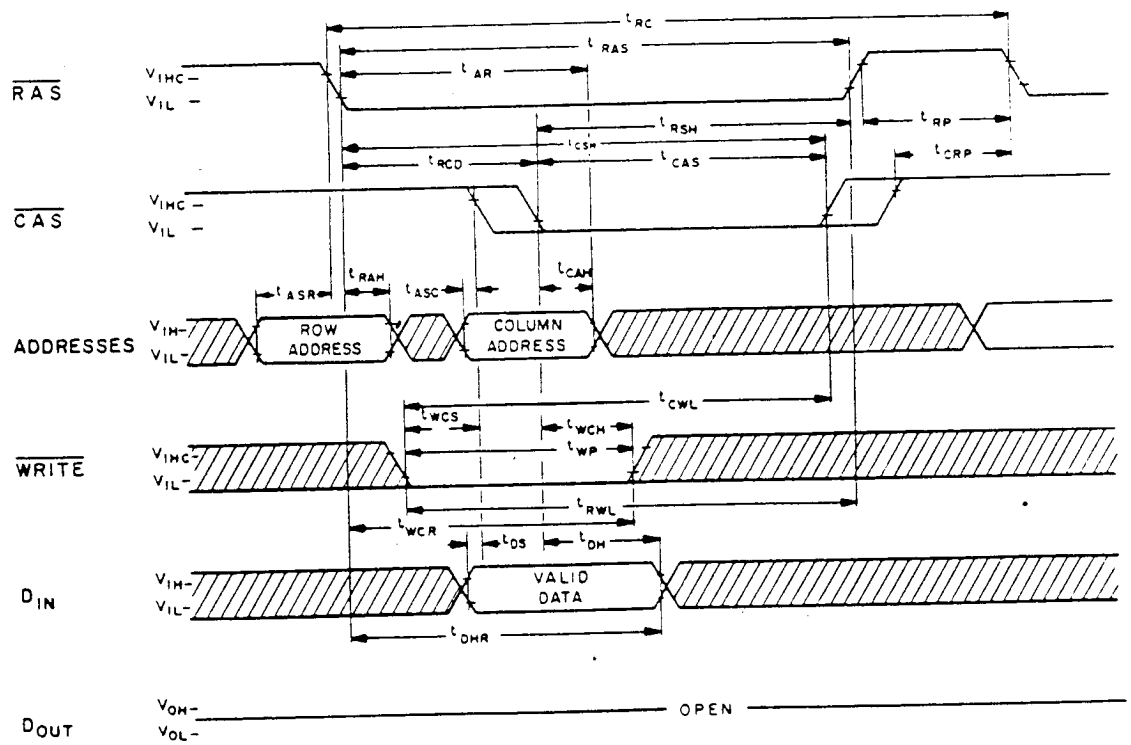
Via de tijdsdiagrammen van volgende bladzijde kunnen we de lees- en schrijf cyclus volgen.



- Read cycle



- WRITE-cyclus



ROM (Random Acces Read Only Memory)

Vrij uitleesbare geheugens. In deze geheugens kan men enkel lezen. Het programma is er door de fabrikant eens en voor altijd ingestopt bij de fabrikatie ervan, dit betekend dat het programma hierin niet meer gewijzigd kan worden. Ook zal een fabrikant deze pas maken wanneer men van deze chips er duizenden nodig heeft.

PROM (Programmable Random Acces Read Only Memory)

Programmeerbare vrij uitleesbare geheugens. Deze geheugens zijn door de gebruiker zelf te programmeren, vandaar dat men deze ook voor enkele stuks bruikbaar zijn. Het programma wordt hierin geschreven door het opblazen van zekeringen. Het is dus onmogelijk om het programma te veranderen eens het er in geschreven is.

EPROM (U.V. Erasable Programmable Random Acces Read Only Memory)

Ultra violet uitwisbare programmeerbare vrij uitleesbare geheugens. Men kan deze PROM's zelf programmeren.
Het programmeren gebeurt door middel van spanningspieken, waarbij elektronen gevangen worden op een geïsoleerde gate.
Het uitwissen van deze PROM's gebeurt door het belichten met U.V. - licht van het IC gedurende een twintigtal minuten. Hierdoor verdwijnt de lading op de geïsoleerde gate.
Praktische uitvoeringen van UV-PROM's zijn o.a.:
- De 2708: 1024 woorden van 8 bits
- De 2716: 2048 woorden van 8 bits
- De 2732: 4096 woorden van 8 bits
- De 2764: 8192 woorden van 8 bits

EEPROM (Electrically Erasable Programmable Random Acces Read Only Memory)

Elektrisch uitwisbare programmeerbare vrij uitleesbare geheugens. Het programmeren en het uitwissen gebeurt met elektrische impulsen die toegepast worden op de uitwendig bereikbare gate.

EAPROM (Electrically Alternable Programmable Random Acces Read Only Memory)

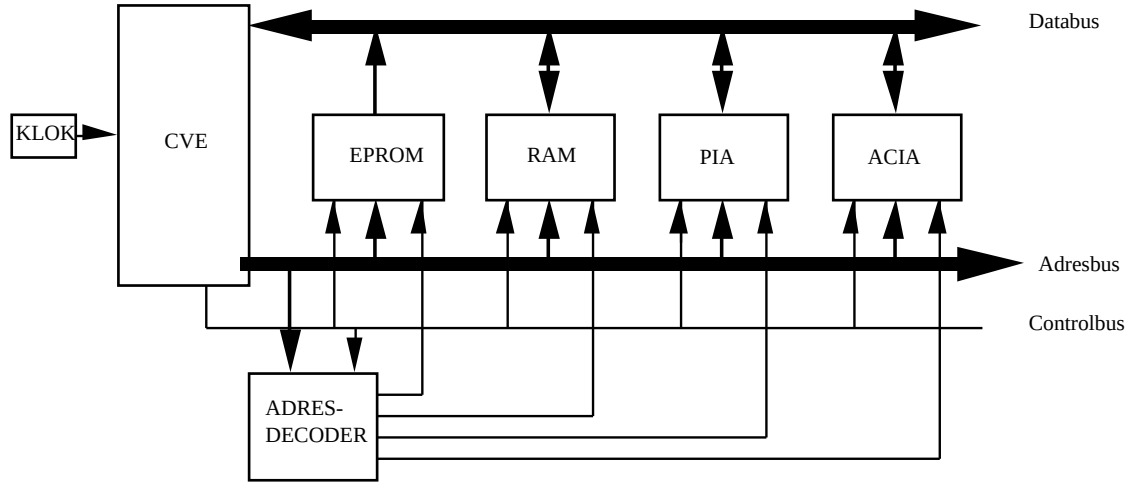
De EAPROM kan uitgewist en geprogrammeerd worden terwijl hij in het systeem aanwezig blijft.
Het uitwissen van de data gebeurt in nagenoeg 10 ms.
Het programmeren gebeurt op een ritme van 1 byte per ms.
In deze IC' moet men bij een programmafout niet de ganse IC wissen, men kan ook één enkele byte veranderen.

PLA (Programmable Logic Arrays)

Om logische vergelijkingen praktisch uit te voeren hebben we eerst de klassieke combinatorische logica gebruikt. Verder is het mogelijk deze logische vergelijking uit te voeren met multiplexers. Combinatorische schakelingen kunnen ook uitgevoerd worden in een PLA; dit zijn programmeerbare componenten om complexe schakelingen er in onder te brengen.
De PLD's (Programmable Logic Device) is de verzamelnaam voor de verschillende programmeerbare componenten. Deze zullen bestudeerd worden is een afzonderlijk hoofdstuk.

Adresdecodering

Het globaal blokschema van een klassiek microcomputersysteem kunnen we voorstellen zoals in figuur 14.1.



Figuur 14.1

Een microcomputersysteem bestaat uit minimaal een microprocessor, RAM ROM en I/O.

De Parallel-Interface-Adapter (PIA) is een programmeerbare I.C. die ons de mogelijkheid biedt tweemaal 8 bits brede (parallel) invoer en uitvoer van data te bedrijven. Daarbij zijn per poort twee handshake-lijnen en twee onderbrekingslijnen (interrupt-request-lijnen) voorzien.

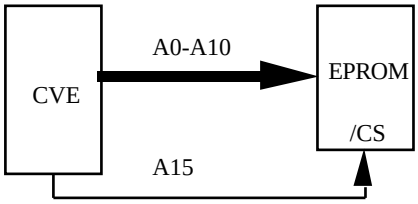
De Asynchrone-Communication-Interface-Adapter (ACIA) is een programmeerbare I.C. die ons de mogelijkheid biedt om seriële input en output te bedrijven. Hij kan daarbij aangesproken worden als een gewone geheugenplaats.

Om de communicatie tussen de verschillende blokken mogelijk te maken moet de CVE (Centrale Verwerkings Eenheid) of µP. een bepaalde geheugenplaats kunnen selecteren in één van de aangesproken blokken. Dit selecteren gebeurt via de adresbus. Het is logisch dat er nooit twee (of meer) I.C.'s op hetzelfde moment mogen aangesproken worden.
Gezien al de gebruikte I.C.'s op één en dezelfde adresbus aangesloten worden moet er in de hardware een schakeling voorzien zijn die er voor zorgt dat er altijd slechts één I.C. op het juiste moment actief is. Die schakeling noemen we de **"ADRESDECODERING".**

De verzameling van adreslijnen noemt men de **adresbus**. De adresbus in **unidirektioneel** daar het adres enkel naar het geheugenelement toe kan stromen.
De verzameling van datalijnen noemt men de **databus**. De databus is **bidirektioneel** daar de data zowel naar het geheugen toe als er van weg kan stromen.
De binaire code van de adresbus wordt in de adresdecoder omgevormd naar een signaal dat een bepaalde I.C activeert. Dit signaal noemen we het **"chip-select"** of **"chip-enable"** signaal.
De controlesignalen zorgen ervoor dat een bepaalde I.C. op het juiste moment aangesproken wordt.
De adresdecodering mag immers maar werken als de aangesloten adressen stabiel aanwezig zijn. Bij sommige microprocessoren is een controlelijn voorzien die het geldig zijn van de adressen signaleert, de "VMA"-lijn (Valid-Memory-Adress).

Rechtstreekse adresdecodering

Veronderstel dat we een 2K-EPROM gebruiken in een bepaald systeem. Van de 16 adreslijnen van de CVE hebben we dan 11 lijnen nodig (A0 t.e.m. A10) om de geheugenplaatsen van de EPROM aan te duiden. Daarbij is een adreslijn nodig (bijv. A15) om de EPROM, via de /CS-lijn, te selecteren. Zie figuur 15.1.



figuur 15.1

Vermits de EPROM actief is als /CS = "0" is zal het beginadres zich bevinden op adres $\%0000\ 0000\ 0000\ 0000_{\mathbf{2}}$ en het eindadres op $\%0111\ 1111\ 1111\ 1111_{\mathbf{2}}$
of van \$0000 tot \$7FFF.
De adreslijn A15 moet dus laag zijn terwijl er voor al de overige combinaties een bepaalde geheugenplaats aangesproken wordt. Dit wordt als volgt voorgesteld:

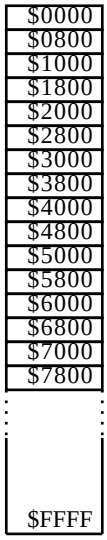
$(0XXX\ XXXX\ XXXX\ XXXX)_{\mathbf{2}}$ (Met X = Don't care)

Gezien de EPROM maar 11 adresingangen heeft is een duidelijker voorstelling van het adrespatroon als volgt:

$(0XXX\ X***\ ****\ ****)_{\mathbf{2}}$ (Met $*$ = Inwendige adressen)

De EPROM is actief vanaf \$0000 tot \$7FFF. Dit is een bereik van 32 K en de I.C. bevat slechts 2K. Voor de 16 combinaties van de adreslijnen A11 tot A14 zal de EPROM aangesproken worden. Dit betekent dus dat we in feite op 16 verschillende adressen dezelfde data vinden. De geheugenmap ziet er dan uit zoals in figuur 15.2.

Door een invertor te schakelen in de aansluiting naar de /CS-ingang wordt de EPROM actief op 16 verschillende plaatsen die gelegen zijn vanaf \$8000 tot \$FFFF dit wil zeggen de tweede helft van het totale geheugenbereik van 64K. Die methode van werken is aan te raden, omdat de resetvectoren geprogrammeerd worden op de gebruikelijke plaatsen \$FFFE en \$FFFF.



Figuur 15.2

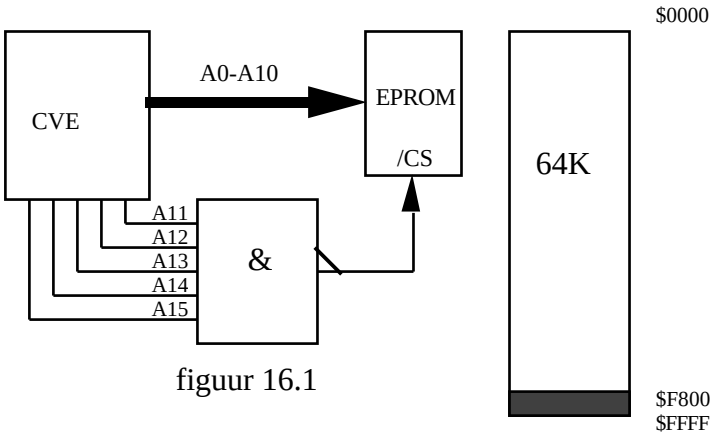
Oefening 15.1

Bepaal het adressenpatroon en teken de geheugenmap voor de opstelling van figuur 15.1 als de /CS-lijn verbonden is met de adreslijn A14.

Volledige adresdecodering.

Omdat we dezelfde inhoud van een bepaald geheugenadres op 16 verschillende plaatsen terug vinden, zoals in figuur 15.1, noemen we deze adresdecodering een **onvolledige adresdecodering.**

Willen we een **volledige adresdecodering**, dan moeten de niet inwendige adreslijnen van het geheugen aangesloten zitten op de adresdecodering. Zie figuur 16.1.



figuur 16.1

Oefening 16.1.

Zoek het schema op als we een 4K-EPROM willen selecteren vanaf adres $0000.

Adresdecodering met decoder I.C.'s

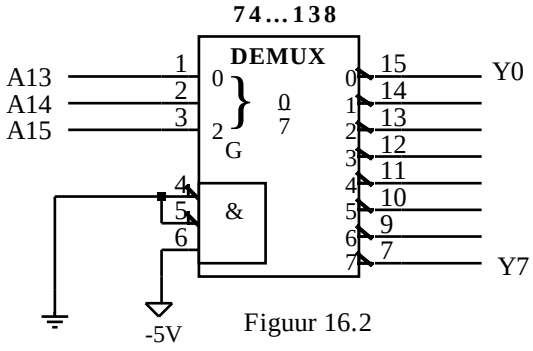
Het gebruik van decoders laat ons toe op een soepele manier het geheugen in te delen.
Met een 2 naar 4 decoder bijv. kunnen we met behulp van 2 selectieïngangen 4 plaatsen kiezen.
Met een 3 naar 8 decoder bijv. kunnen we met behulp van 3 selectieïngangen 8 plaatsen kiezen. (bv. 74...138)
In de eenvoudigste vorm kunnen we m.b.v. deze decoder een adresdecodering maken zoals in figuur 16.2. is gegeven. Hierdoor ontstaan 8 uitgangssignalen waarvan slechts één laag actief is afhankelijk van A13, A14 en A15. Op die manier verdelen we het 64K geheugen in 8 blokken van elk 8K.

Oefening 16.2

Geef de geheugenmap en het adrespatroon van de uitgangen Y0 tot en met Y7.

Oefening 16.3

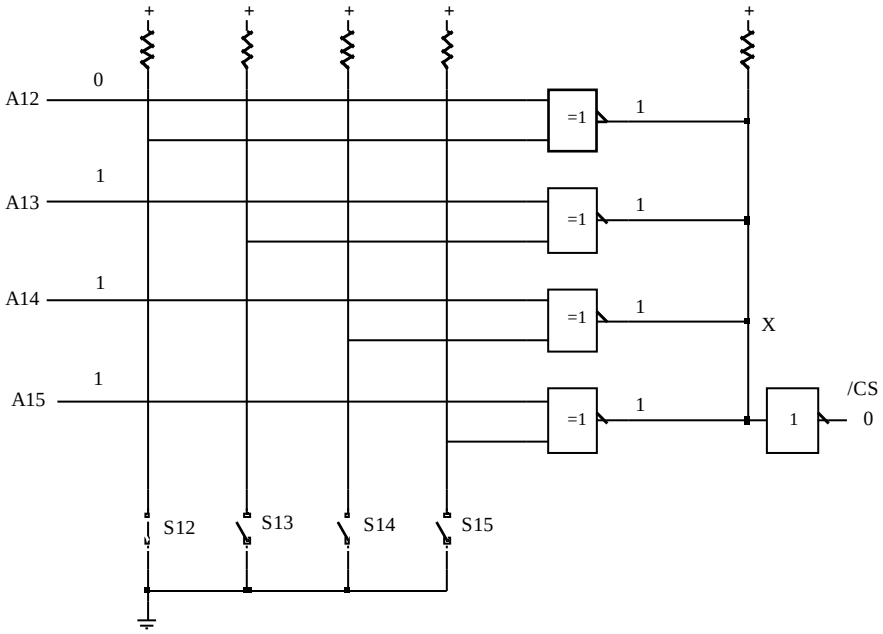
Veronderstel dat we een 64K geheugen willen indelen in 8 blokken van 1K gebruik makend van de 74...138. De laatste geheugenplaats moet gelegen zijn op adres $FFFF



Figuur 16.2

Adresdecodering met XNOF-poorten.

In sommige gevallen zoals bij experimentele opstellingen is het gewenst dat het adresbereik op een flexibele manier instelbaar is bijv. m.b.v. een aantal schakelaartjes. De tot hiertoe besproken systemen hebben dat niet.
Een mogelijke eenvoudige oplossing hiervoor is een adresdecoder met XNOF-poorten. Veronderstel de schakeling van figuur 17.1



figuur 17.1

In de schakeling kan door het /CS-signaal plaats voorzien worden voor 2^4 = 16 adresplaatsen die elk 64/16 = 4K groot zijn.
Een andere instelling wordt bekomen door meer of mindere adreslijnen te schakelen naar een even-groot aantal XNOF-poorten.
De XNOF-poorten vergelijken het ingesteld adres met het werkelijk adres van de adresbus. Als iedere adreslijn afzonderlijk gelijk is aan de logische toestand van de ingestelde schakelaars komen de uitgangen van al de poorten logisch "1". Dan, en dan alleen, is X = "1"; hierdoor komt het /CS-signaal logisch "0" en wordt de I.C. geactiveerd. In het gegeven voorbeeld is voor A12, A13, A14, A15 = 0111.

Oefening 18.1

Teken een blokschema van een microcomputersysteem en benoem de fundamentele delen.
Integreer in het blokschema tevens een adresdecoderingssysteem dat de 64K memorymap opdeelt in vier stukken van 16K.
Om te beginnen teken eerst de geheugenmap van 64K.

Oefening 18.2

Ontwerp een adresdecoderingssysteem dat de 64K memorymap opdeelt in 8 gelijke memoryblokken van 8K.
- Het RAM-gedeelte zit vanaf adres $0000.
- De PIA zit vanaf f adres $A000.
- De ACIA zit vanaf adres $B000.
- Viermaal 4K EPROM zit vanaf adres $C000.

Om te beginnen teken eerst de geheugenmap van 64K.

Oefening 18.3

Ontwerp een adresdecoderingssysteem voor een 16K memoryblok, bestaande uit 8 memory's van 2K.
De adresdecodering moet ervoor zorgen dat de memoryblok van 16K begint vanaf adres $8000.
Het I/O-gedeelte volgt direct achter het 16K memoryblok, en er moet 8K EPROM voorzien zijn vanaf adres $E000.
Integreer dit adresdecoderingssysteem in een blokschema van een microcomputersysteem.
Om te beginnen teken eerst de geheugenmap van 64K.

Oefening 18.4

Ontwerp een adresdecoderingssysteem dat een blok van 20K bestaande uit 5 memory's van 4K begint vanaf adres $7000.
Integreer dit adresdecoderingssysteem in een blokschema van een microcomputersysteem.
Om te beginnen teken eerst de geheugenmap van 64K.

Oefening 18.5

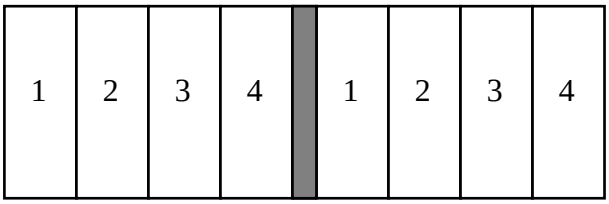
Ontwerp een adresdecoderingssysteem met de 74...138 voor 8 memory's van 4K.
Het geheugenblok van 32K moet beginnen op adres $8000.
Om te beginnen teken eerst de geheugenmap van 64K.

SIMM (Single In line Memory Module)

In de moderne PC's worden als geheugens dynamische RAM's ingezet (behalve voor het cache-geheugen). Deze RAM's zitten op een soort insteekmodules (SIMM genoemd), kleine printjes met 30 of 72 kontakten. De intussen wat oudere 30-pens SIMM's zijn altijd 8 bits breed (9 bits met pariteitsbit), terwijl de nieuwere 72- pens PS/2-SIMM's een breedte van 32 bits bezitten (36 bits met pariteitsbit). Afhankelijk van het type hoofdprint zitten er één of meerdere SIMM's in speciale voetjes op de print.

Bij processoren met een busbreedte van 32 bits (80386, 80486) zijn er altijd en veelvoud van vier SIMM's in de voetjes. (zie figuur 19.1) Tabel 19.1 geeft de mogelijke kombinaties van SIMM's die kunnen gebruikt worden.

32-BIT µP 30 pens SIMM's



figuur 19.1

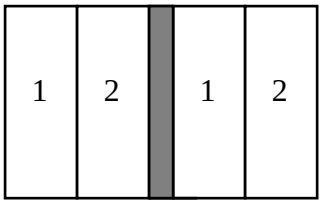
B0	B1	Tot. Memory
4x1Mx9	x	4MB
x	4x1Mx9	4MB
4x1Mx9	4x1Mx9	8MB
4x4Mx9	x	16MB
x	4x4Mx9	16MB
4x1Mx9	4x4Mx9	20MB
4x4Mx9	4x1Mx9	20MB
4x4Mx9	4x4Mx9	32MB
4x16Mx9	x	64MB
x	4x16Mx9	64MB
4x16Mx9	4x1Mx9	68MB
4x1Mx9	4x16Mx9	68MB
4x16Mx9	4x4Mx9	80MB
4x4Mx9	4x16Mx9	80MB
4x16Mx9	4x16Mx9	128MB

tabel 19.1

Bij de Pentium met zijn busbreedte van 64 bits zijn er minimaal 2 PS/2-SIMM's nodig.

Tabel 19.2 geeft de mogelijke kombinaties van 72-pens SIMM's die kunnen gebruikt worden.

PENTIUM 64-BIT µP PS/2-SIMM's 72 pens

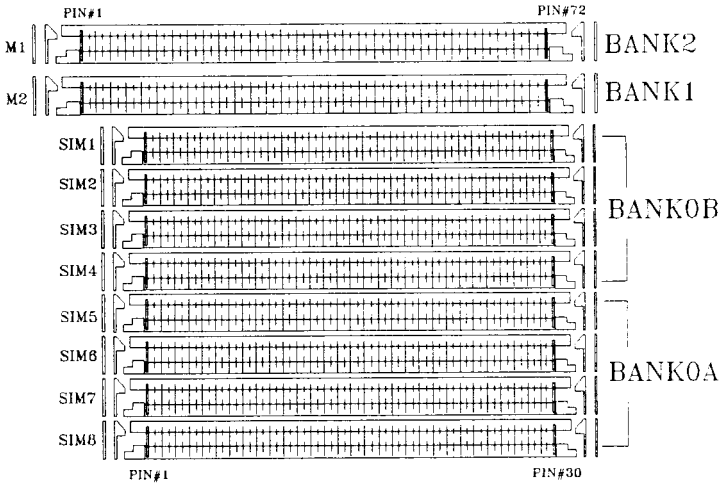


figuur 19.2

B0	B1	Tot. Memory
x	2x4MB	8MB
2x4MB	x	8MB
x	2x8MB	16MB
2x4MB	2x4MB	16MB
2x8MB	x	16MB
2x4MB	2x8MB	24MB
2x8MB	2x4MB	24MB
x	2x16MB	32MB
2x16MB	x	32MB
2x16MB	2x16MB	32MB
2x4MB	2x16MB	40MB
2x16MB	2x4MB	40MB
2x8MB	2x16MB	48MB
2x16MB	2x8MB	48MB
x	2x32MB	64MB
2x16MB	2x16MB	64MB
2x32MB	x	64MB
2x4MB	2x32MB	72MB
2x32MB	2x4MB	72MB
2x8MB	2x32MB	80MB
2x32MB	2x8MB	80MB
2x16MB	2x32MB	96MB
2x32MB	2x16MB	96MB
2x32MB	2x32MB	128MB

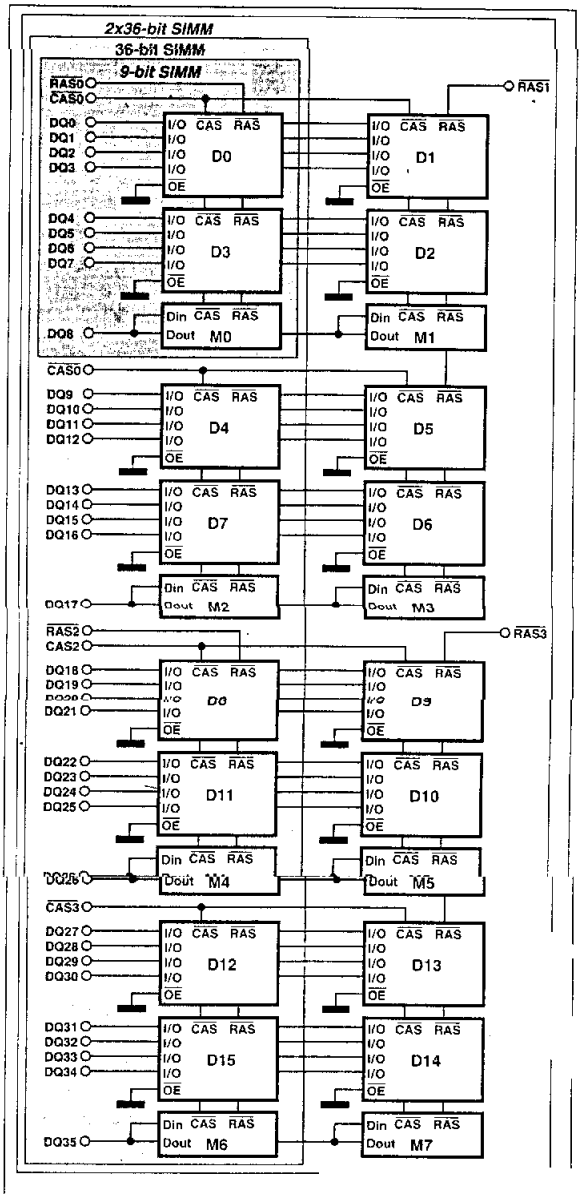
tabel 19.2

Nevenstaande figuur geeft een voorbeeld van een geheugensysteem waarbij zowel de 30-pens als de 72-pens SIMM's kunnen gebruikt worden.
Het zijn vier geheugenbanken:
Twee banken van acht 30 pens SIMM's van 1Mx9, 4Mx9, of 16Mx9.
Twee banken van twee 72 pens SIMM's van 1Mx36, 2Mx36, 4Mx36 of 8Mx36.



Een SIMM is een printje waarop diverse dynamische RAM's zijn samengevoegd, zodat uiteindelijk een 8 of een 32 bit brede databus gerealiseerd kan worden. Wordt de pariteitsbit gebruikt, dan is de breedte respectievelijk 9 of 36 bits. De dynamische RAM's zelf kunnen typen zijn met een breedte van 1 of 4 bits. Een SIMM van het type 1Mx9 kan daarom zowel in een uitvoering met 9 als 3 chips geleverd worden. Bij de 8(9) bits brede SIMM's zijn de RAS- en CAS-lijnen van de aanwezige IC's parallel geschakeld en worden als twee aansluitingen naar buiten gevoerd.

Bij de PS/2-SIMM's (SIMM's met 72 aansluitpennen) gebeurt dit per byte, eventueel aangevuld met een paritietsbit. Ieder byte heeft daarbij een aparte CAS-lijn (CAS0...CAS3), terwijl RAS per twee bytes (in de volgorde RAS0, RAS1, RAS2, en RAS3) aangeboden wordt. Indien RAS1 en RAS3 gebruikt worden, zijn twee RAM-IC's parallel geschakeld. Op de print van de SIMM is dat meestal te zien doordat beide zijden voorzien zijn van IC's. In nevenstaande figuur is schematisch de opbouw van zo'n print weergegeven, voor de duidelijkheid zijn de adreslijnen weggelaten. De pariteitsbit heeft tegenwoordig dankzij de hoge betrouwbaarheid van de geheugenbouwstenen bijna geen zin meer.



Inhoud

Opbouw van halfgeleider geheugens
Indeling van de halfgeleidergeheugens
RAM
Statische RAM
Uitbreiding van geheugens
Tijdvolgorde diagrammen
Dynamische geheugens
ROM, PROM, EPROM, EEPROM EAPROM, PLA
Adresdecodering
Rechtstreekse adresdecodering
Volledige adresdecodering
Adresdecodering met decoder I.C.'s
Adresdecodering met XNOF-poorten.
Oefeningen op adresdecodingssystemen.
SIMM (Single In line Memory Modul)