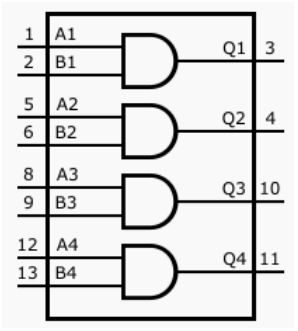
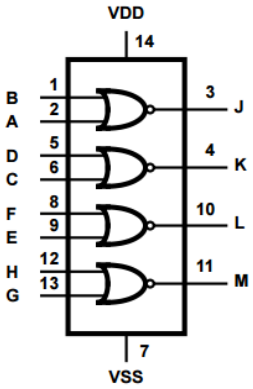
Programmering af GAL-kredse.

Vi kender allerede IC-kredse, der indeholder enten AND-gates, OR-gates osv. De er færdig-konstruerede fra fabrikken og man skal bruge dem som de er. De indeholder typisk enten et antal AND-gates, et antal NAND-gates, - osv.

De kan ikke ændres, de er ikke særlig fleksible. Skal man bruge en AND - gate og fx en NOR-gate i et krredsløb kan man blive nødt til at bruge 2 IC'er.





Det er her GAL-kredse kommer ind i billedet.

GAL-kredse er en type IC-kredse, der indeholder et stort antal GATEs. AND, NAND, OR – osv. Og endog Flip-Flops.

Men de enkelte gates i kredsen er ikke forbundne med hinanden.

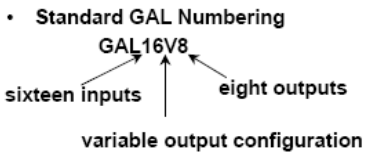
Forbindelserne mellem de forskellige gates opstår først i kredsen efter et design-forløb og en programmering.

IC'en har et antal pins der kan fungere som indgange, - og et antal pins, der både kan fungere som udgange og indgange.

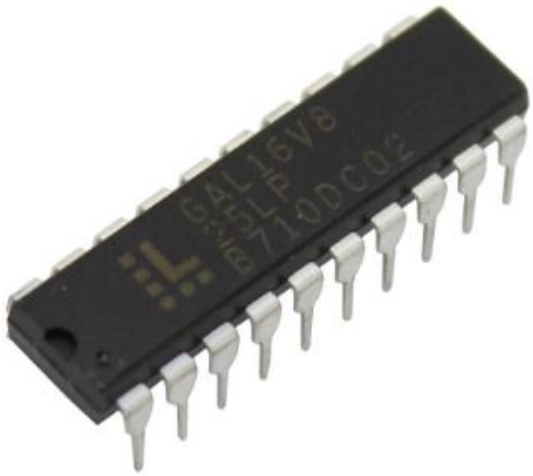
I en tekst-editor beskriver man hvordan kredsen skal udformes. Hver udgangs ønskede funktion beskrives med boolske ligninger som funktion af indgangene.

Teksten oversættes og der skabes en fil, der skal brændes ind i en GAL-kreds.

GAL-kredse har fået deres navn fra forbogstaverne for **G**eneric **A**rray **L**ogic devices. (Vist nok noget i retningen af "Generel" Array Logic ??)



Her er vist hvordan forskellige GAL-kredse-størrelser nummereres.



Bemærk: Kredsene kan kun klare 5 Volt !!!

Også på indgangene !!

Derfor skal de kredse, der styrer GAL-kredsen også arbejde på 5 Volt !!!

GAL-kredse er faktisk ret gamle, - helt tilbage fra 1970'erne ?? Men de fås stadig i nogle varianter.

De er i dag videreudviklet til FPGA, som er store komplekse kredse, der endog kan indeholde processorer !!

Hvis en GAL-kreds sammenlignes med en cykel, vil en FPGA skulle sammenlignes med en Ferrary.

Vi bruger et gammelt program, **WinCUPL** til at beskrive de boolske ligninger, og oversætte dem til en brændbar fil.

Vha. programmet skal man først navngive indgange og udgange, og dernæst angive de boolske ligninger for udgangene.

Når de boolske ligninger er defineret, kan WinCUPL oversætte kildeteksten, og generere en fil, der derefter skal flyttes over i – eller brændes - ind i GAL-kredsen.

Filen kaldes også en " Fuse-map-fil ". Filen beskriver hvor der skal laves forbindelser mellem gatene i kredsen, så ligningerne opfyldes. Filen overholder en standard, defineret af JEDEC-organisationen. (Se fodnote [1])
Filen får efternavnet ***.JED.

--

WinCUPL er et ret gammelt program, - gad vide, hvor længe det bliver ved med at køre på de nye Windows-versioner ??

WinCUPL-programmet kan hentes på min hjemmeside.

[1] *JEDEC Solid State Technology Association, tidligere kendt som Joint Electron Devices Engineering Council (JEDEC), er en uafhængig halvleder ingeniør handelsorganisation og standardiseringsorgan.*

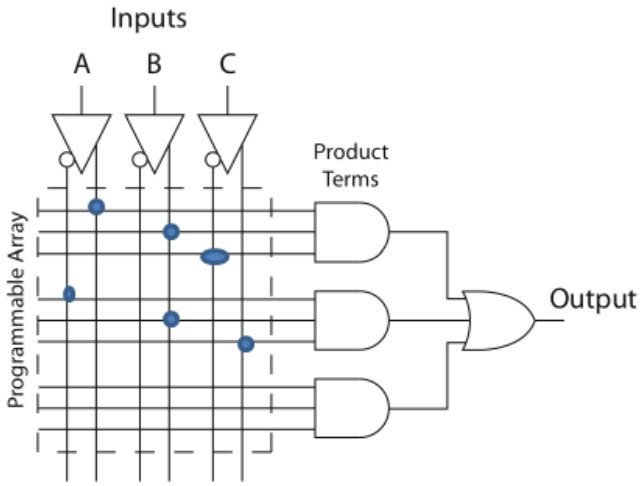
/Elektronik / GAL. http://vthoroe.dk/elektronik.htm#Gal

Eller fra Atmels hjemmeside: (der skal angives en valid emailadresse)
http://www.atmel.com/tools/wincupl.aspx

Under installeringen spørges efter et serienummer. Her kan bruges 60008009, eller 66999998.

Forbindelserne i kredsens indre:

Programmerbare kredse som fx GAL har ikke fra starten nogle indbyggede funktioner – som fx. NAND-gates, OR-gates osv. Men de har et hav af indbyggede muligheder og forbindelser, der bare endnu ikke er forbundne. Det er disse forbindelser, der skabes ved programmering.



Via programmering kan man definere output til fx at være et boolsk udtryk med forskellige indgange som variable.

Et eksempel er vis her til højre!

I viste eksempel vil output følge ligningen:

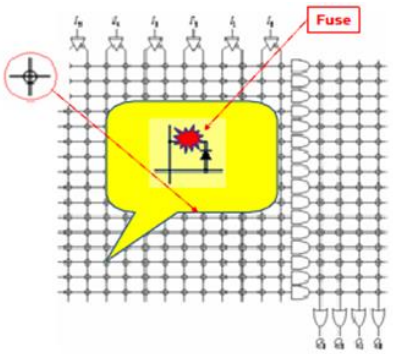
Bestem ligningen for output !!

Her er vist en skitse af, hvordan de forskellige forbindelser i kredsen i princippet laves.

I tidligere generationer af programmerbare kredse var alle forbindelser mellem gatene i kredsen intakte ved produktionen, men ved programmering ødelagde man de forbindelser, der ikke skulle bruges.

Det skete ved at tilføre en ”stor” strøm, så forbindelserne smeltede, ligesom en ”sikring smeltes”, ved for stor strøm !!

Det betød at kredsene var ”one time programable”, OTP, og følgelige kunne kasseres, hvis de blev fejlprogrammeret.



En ”Fuse” smeltes !!

Men udviklingen har betydet, at det i dag reelt set er en form for switch, der sørger for forbindelserne, hvor det ønskes.

Derfor kan GAL-kredse i dag både slettes og re-programmeres mange gange.

Programmerede kredse kan fungere mere end 20 år.

Selve Programmeringen af kredsene sker med et specielt brænderudstyr. Brænderen skal bruge en "fuse" informationsfil, en såkaldt Fusemap, eller en JEDEC-fil med extension ".JED"

--

GAL-kredse er meget fleksible kredse. De kan programmeres til på samme chip at udgøre både kombinatorisk logik, fx AND, OR, NAND osv., og logik-funktioner med flip-flops, som bruges i tællere, skifteregistere, osv.

Det vil sige, der kan laves enten simple boolske udtryk, eller der kan designes tællere vha. de indbyggede D-FF.

GAL-kredse er hurtige, dvs. med propagation delays ned til så kort som 7 ns. De kan måle sig med standard 7400 og 74LS serierne. De kan om-programmeres op til 100 gange. Sletning og programmering tager kun få sekunder.

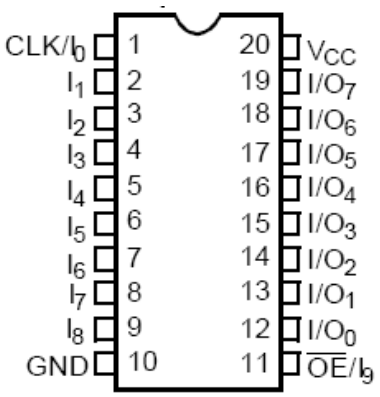
Der findes flere typer af GAL-kredse. Denne tekst dækker typen "GAL16V8"

GAL20V10 har lidt flere muligheder, men er ikke beskrevet her !!

GAL16V8 Pin-outline:

Her er vist hvordan pins er placeret i GAL16V8.

20V10 har et lignende design, har blot lidt flere I/O's

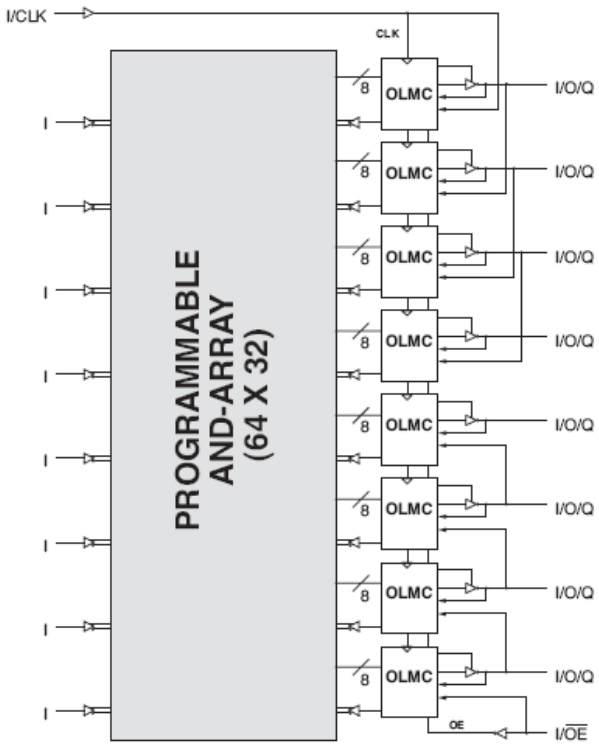


GAL16V8 skal have stel på pin 10, og +5 volt (VCC) på pin 20.

Pin 2 til 9 er altid "general purpose" input pins.

Pin 12 til 19 kan være Output eller inputs.

GAL16V8 blokdiagram.

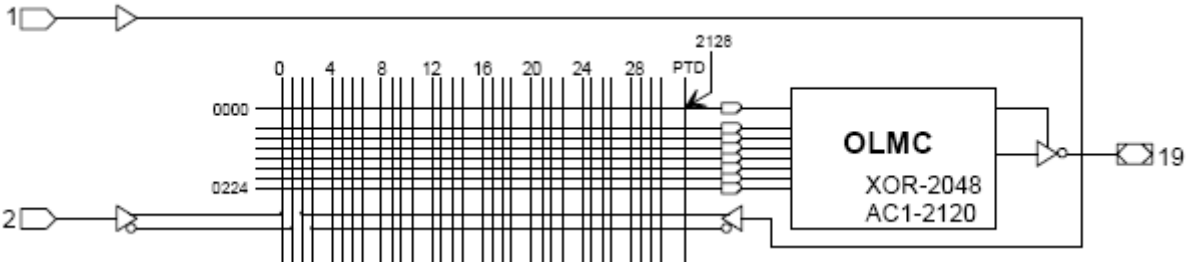


Her er vist et blokdiagram over kredsen, taget fra databladet for GAL16V8D.

I venstre side ses det programmerbare AND-array. I højre side ses at der til hver output er en kasse, kaldet en **OLMC**. Det står for "OUTPUT LOGIC MACRO CELLE".

Herunder er der "zoomet lidt mere ind"

AND-arrayet ses til venstre for én af OLMC-cellerne.



Ved hjælp af OLMC'en kan man bestemme om output-pins skal bruges som input, kombinatorisk output, register-outputs (D-FF) eller input/output pins.

Hvis en af OLMC' cellerne er konfigureret som register-output, er pin 1 Clock-input og pin 11 er Output Enable for register-outputs.

Hvis ingen af OLMC'erne bruges, kan pin 1 og 11 bruges som general purpose inputs.

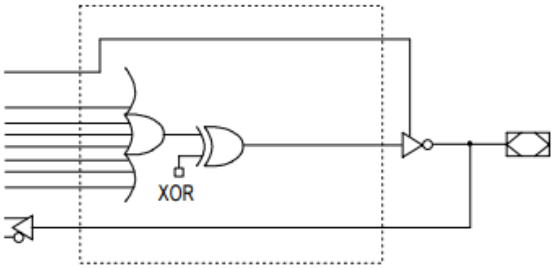
Bliver en udgang "blot" beskrevet vha. en boolsk ligning, dvs. der ikke skal bruges FF's i udgangen, bliver OLMC'erne ikke brugt !! dvs. at signalet fra gate-arrayet føres udenom direkte til udgangen !!

Det sker automatisk ved hjælp af compileren, der oversætter de boolske ligninger.

Konfigurationsmuligheder af OLMC'erne

Macrocellerne kan ved programmering konfigureres til forskellige funktioner.

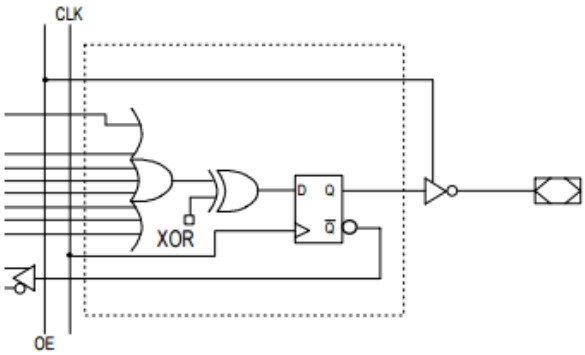
Her ses et blokdiagram for en
Output Logic Macro Cell for GAL16V8



Enten kan man føre en boolsk ligning direkte ud til et output.

Eller man kan føre det til en D-indgang i en D-FF der sidder før udgangen.

Skal D-FF'en bruges, skal man i stedet for at skrive en ligning til udgangen skrive den til "Output_name.d"



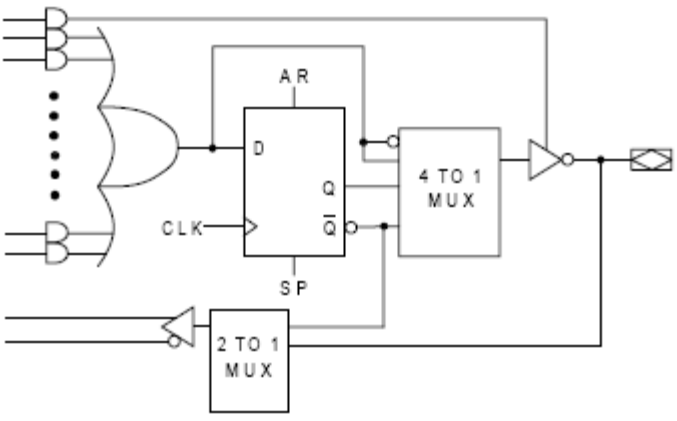
Output Enable er altid pin 11, og er aktiv lav.

Og Clk er altid pin 1.

GAL22V10

De lidt større kredse, fx GAL22V10 har tillige et Asyncron Reset og Syncron Preset.

I midten ses en D-FlipFlop. Og til højre for den en Multiplexer. Heraf indses, at outputtet kan vælges direkte fra And / OR-arrayet, evt. inverteret, eller fra Q eller inverteret Q fra D-FFén.



Brugbare ekstensions for g22v10. Bemærk: kun ".d" kan bruges i g16V8.

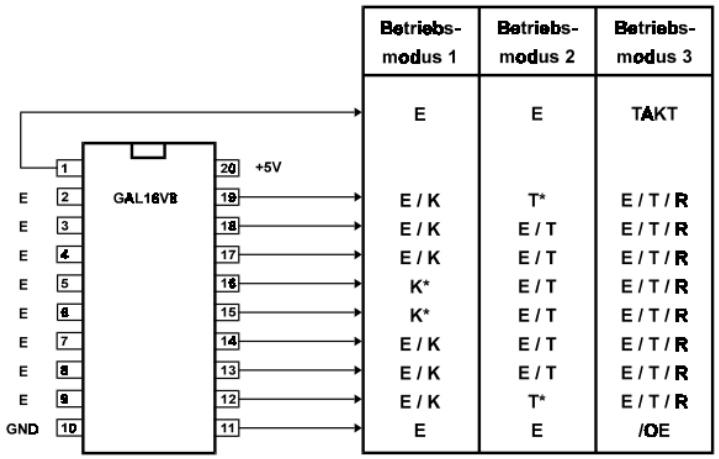
```
.AR     /* Asynkron Reset af flip-flop */
.SP     /* Synkron Preset af flip-flop */
.OE     /* Output Enable */
.D      /* D input for D-type flip-flop, kun for  */
```

Se evt: http://emp.byui.edu/FisherR/Downloads/CUPL%20Tutorial.htm
http://public.rz.fh-wolfenbuettel.de/~krausea/vl/Labor/DT_I/DISK4/PLD_Vers.pdf

I nogle tilfælde er det vist sådan, at man ikke kan få output-signal retur til AND-Arrayet.

Jeg har fx haft problemer med at kompilere et kredsløb, med D-FF koblet til udgang pin 19, og tilbageføring af signalet til Input-matrixen.

Pin 12 og 19 kan vist ikke bruges som input i registered mode.



Modus 1 er kun med kombinatorisk algebra. Der er ingen tilbagekobling fra Pin 15 og 16.

Modus 2 hvis der benyttes Output Enable Der er ingen tilbagekobling fra Pin 12 og 19

Modus 3 er med register-udgang og OE. !!

Abkürzung	Erläuterung
E	Eingang
K	Kombinatorischer Ausgang mit Rückkopplung
/OE	/ (OUTPUT ENABLE) für Registerausgänge
TAKT	Taktsignal für Registerausgänge
T	Tristate-Ausgang mit Rückkopplung
*	Ausgang ohne Rückkopplung !
R	Registerausgang

Fra: http://public.rz.fh-wolfenbuettel.de/~krausea/vl/Labor/DT_I/DISK4/PLD_Vers.pdf

Her er vist en oversigt over forskellige typer programmerbare kredse.

• ASIC	Application Specified Integrated Circuit	Anwendungsspezifisches IC
• CPLD	Complex PLD	Komplexer PLD
• EEPLD	Electrically Erasable Programmable Logic Device	Elektrisch löschbarer programmierbarer Logikbaustein
• EPLD	(UV-) Erasable Programmable Logic Device	(UV-) Löschbarer programmierbarer Logikbaustein
• FPAD	Field Programmable Address Decoder	Frei programmierbarer Adreßdekoder
• FPAL	Field Programmable Array Logic	Frei programmierbare Logikmatrix
• FPGA	Field Programmable Gate Array	Frei programmierbare Gattermatrix
• FPLA	Field Programmable Logic Array	Frei programmierbare Logikmatrix
• FPLD	Field Programmable Logic Device	Frei programmierbarer Logikbaustein
• FPLS	Field Programmable Logic Sequencer	Frei programmierbare sequentielle Logik
• FPML	Field Programmable Macro Logic	Frei programmierbare Makrologik
• GAL	Generic Array Logic	Universelle Matrix-Logik
• HAL	Hard Array Logic	Vom Hersteller programmiertes PAL
• IFL	Integrated Fuse Logic	Integrierte Logik mit Trennstellen
• LCA	Logic Cell Array	Logikzellen-Matrix
• PAL	Programmable Array Logic	Programmierbare Logik-Matrix
• PLA	Programmable Logic Array	Programmierbare Logikmatrix
• PLD	Programmable Logic Device	Programmierbarer Logikbaustein
• PLE	Programmable Logic Element	Programmierbares Logikelement
• PLS	Programmable Logic Sequencer	Programmierbare sequentielle Logik
• PML	Programmable Macro Logic	Programmierbare Makrologik

WinCUPL Sproget

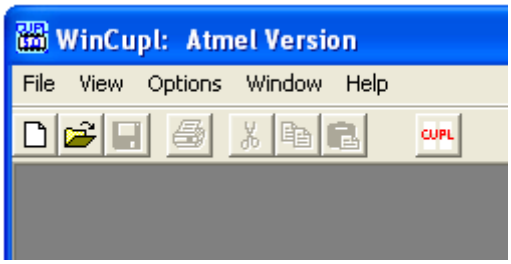
Windows Universal Compiler for Programmable Logic (CUPL),

Programmet WINCUPL bruges til at designe GAL-kredse, og skabe en JEDEC-fil (*.JED) der så efterfølgende skal brændes over i IC-en.

OBS. Vigtigt. Undlad altid at bruge æ, ø og å i amerikanske programmer. Dette gælder både i sti-navne, i tekst, der skrives ind i programmet, ja selv i kommentarer kan det være et problem !!

Start WinCUPL.

WinCUPL Main window.



Start et nyt projekt.

Vælg: **File / New Project**

Herefter starter en wizard, der hjælper med at skabe en del af kildeteksten.

Dvs. program-hoved og dele af teksten i Body.

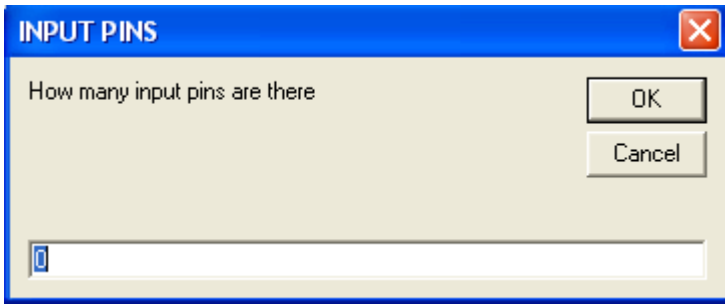
Men man kan også sagtens selv skrive "hovedet" i kilde-filen.

Indtast fx det viste i felterne til højre.

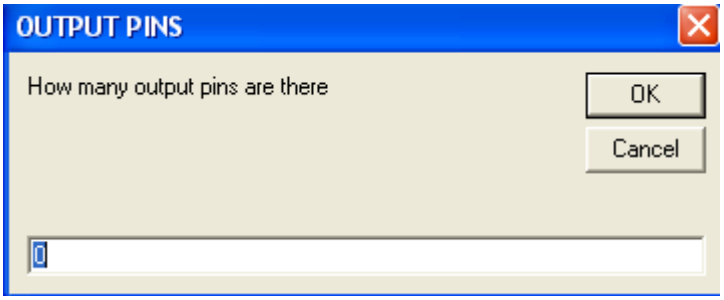
Husk at angive Device-typen !!!!

Indtast antal inputs, der skal bruges i det færdige GAL-design, og Tryk OK

Det kan dog altid ændres senere !!



Og angiv antal outputs.

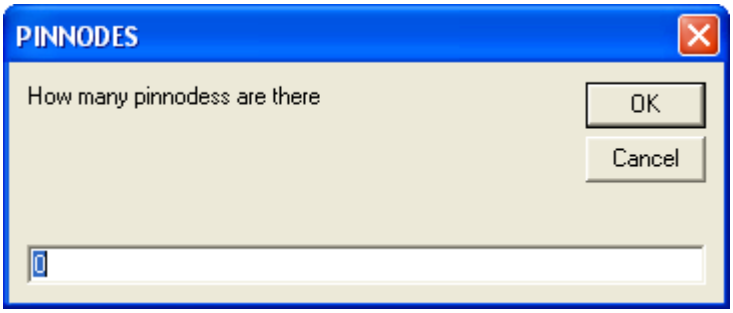


Pinnodes kan opfattes som outputs for ligninger, der ikke skal føres ud til en output-pin.

Dette kan ikke bruges i 16V8.

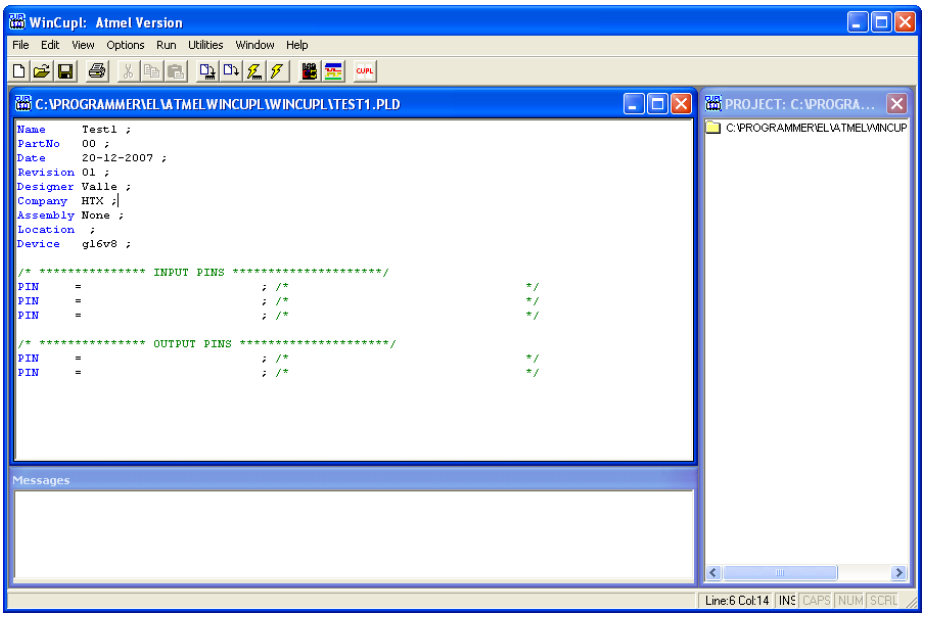
"*In Gal16V8, there are no buried nodes*"

Vælg derfor 0.



Nu genererer WINCUPL en tekstside fx med følgende tekst allerede skrevet.

Dvs. at de ovenstående 3 indtastninger blot fører til lidt hjælp til at generere noget af kildeteksten til designet.



Bemærk:

/* og */ omkranser kommentarer. Kommentarerne kan strække sig over flere linjer.

Ovenfor ses, at teksten er opdelt i 3 dele. De kan kaldes følgende:

En Header-del	Her defineres forskellige data, fx firma, programmør, og IC-navn.
En Declaration-del	Her defineres inputs og outputs med navne, fx A, B osv.

En Body-del	Her defineres de boolske ligninger for output. De mangler endnu i billedet herover.

Delene er adskilt i teksten af en kommentarlinje for overskuelighedens skyld.

Definering af Input pin navne:

For at der kan skrives boolske ligninger for outputs, skal IC'ens pinnumre altid have tildelt et navn, fx et bogstav og evt. et tal.

Her følger et par eksempler på tildeling af navne til input-pins:

```
/*** INPUT PINS ***/

PIN 1 = A ;                  /* Pin 1 får navnet A */
PIN 2 = B ;                  /* her evt. kommentarer */
PIN [2..5] = [B3..0];
PIN [2..6] = [A0..2,A4,A5];
Pin [2..5] = [b, c, d, e];  /* pin 2=b, pin 3=c, pin 4=d, pin5=e*/
Pin [6,8] = [f,g]; /* pin 6 = f og pin 8 = g.*/
```

Bemærk: Hver linje afsluttes med semikolon.

Definering af Output pin navne

Også Outputs skal defineres med navne:

Eksempler på tildeling af output pin-navne:

```
/*** OUTPUT PINS ***/

PIN 12 = F1;

PIN [12,19] = [F1..2] ;     ( pin 12 skal hedde F1, pin 19 F2 )

PIN [13,14,15,16,17,18] = [Y0,Y1,Y2,Y3,Y4,Y5];

PIN [13..18] = [Y5..0];
```

Hver statement-linje skal slutte med et semi-colon.

OUTPUT-Ligninger:

```
/** Equations **/
```

Nu skal der defineres boolske ligninger for outputs.

De defineres som funktioner af inputs.

Bemærk de specielle tegn, der bruges i WinCUPL for at angive AND, OR osv.

Description	Example	Operator	Precedence
NOT	!A	!	1
AND	A&B	&	2
OR	A#B	#	3
X-OR	A$B	$	4

"Værdien" eller " Størst kompetence" af operatorerne er angivet som " Precedence "

Herudover kan man også bruge parenteser.

Et par eksempler :

Y0 = A0 & B0;
Y1 = A1 # B1;
Y2 = A2 $ B2;
Y3 = !B3;
Y4 = !(A4 & B4);
Y5 = !(A5 # B5);
A = !x&y # x&!y&z;
B = !x&!y&!z # x&!y # y&z;

Her følger et færdigt eksempel, taget fra et af de medfølgende eksempler i WinCUPL-programmet:

```
Name            Gates;
Partno          CA0001;
Revision        04;
Date            9/12/89;
Designer        G. Woolhiser;
Company         Logical Devices, Inc.;
Location        None;
Assembly        None;
Device          g16v8a;

/*****************************************************************/
/*                                                               */
/*      This is a example to demonstrate how CUPL                */
/*      compiles simple gates.                                   */
/*                                                               */
/*****************************************************************/

/*
```

```
 * Inputs:  define inputs to build simple gates from
 */

Pin 2 =  a;
Pin 3 =  b;

/*
 * Outputs:  define outputs as active HI levels
 *
 */

Pin 12 = inva;
Pin 13 = invb;
Pin 14 = and;
Pin 15 = nand;
Pin 16 = or;
Pin 17 = nor;
Pin 18 = xor;
Pin 19 = xnor;

/*
 * Logic:  examples of simple gates expressed in CUPL
 */

inva = !a;              /* inverters */
invb = !b;
and  = a & b;           /* and gate */
nand = !(a & b);        /* nand gate */
or   = a # b;           /* or gate */
nor  = !(a # b);        /* nor gate */
xor  = a $ b;           /* exclusive or gate */
xnor = !(a $ b);        /* exclusive nor gate */
```

Compilerings-procedure:

For at få kompileringen af kildefilen til at virke, er der et par switche, der skal sættes:

Først skal det sikres, at der er angivet hvilken kreds, der arbejdes med. Den kreds, vi har er en GAL16V8.

Det kan ske på 2 måder:

Enten i kildeteksten i header'en, under device.

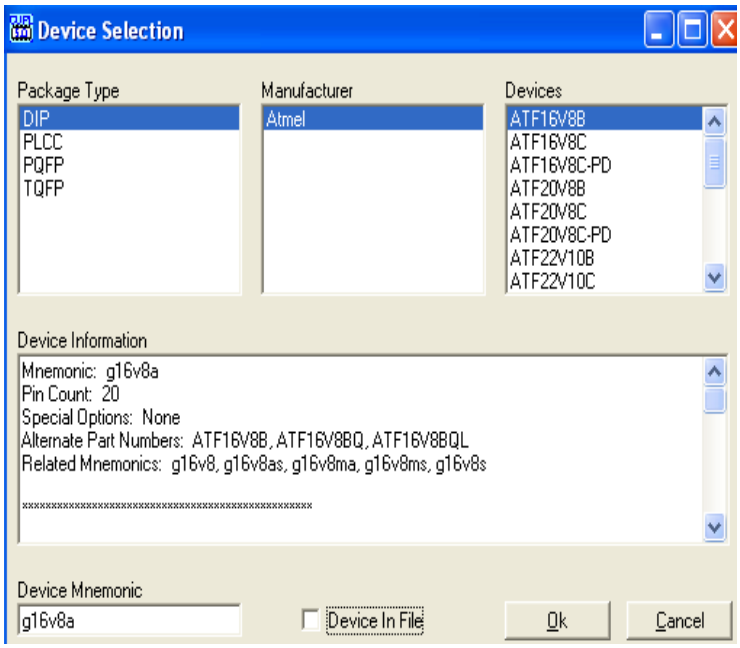
```
Device     g16v8a;
```

Eller alternativt ved at vælge

Option \ Devices

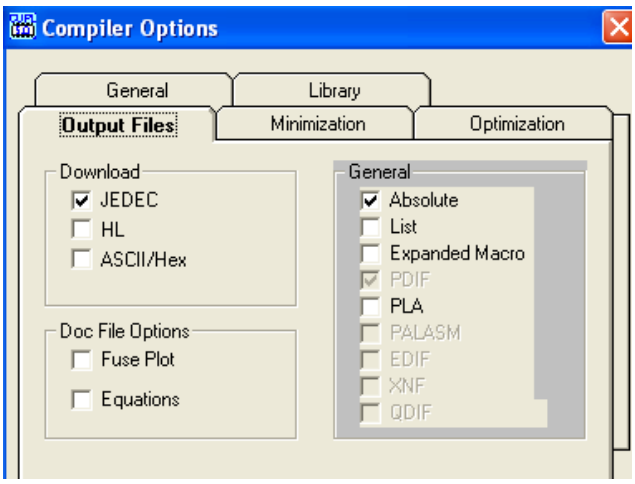
Vælg en ATF16V8B, som svarer til en GAL16V8

Bruges denne metode, skal fluebenet for neden i "Device In File" fjernes!



Det er tillige nødvendigt at gå ind i menuen Option / Compiler.

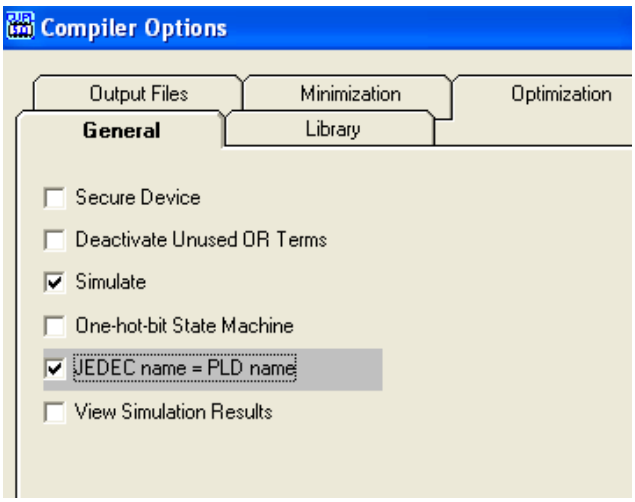
Her vises Default fanebladet ”Output Files”



Vælg fanebladet General og sæt flueben i

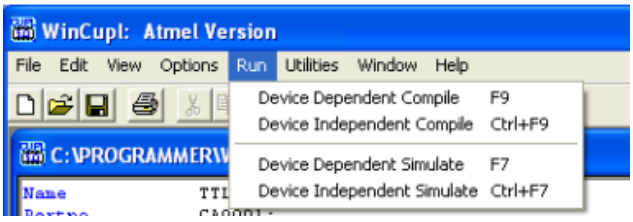
JEDEC name = PLD name.

Ellers bliver der ikke genereret en ”.JED-fil”.



Nu kan der Compileres.

Vælg Run \ Device Dependent Compile.

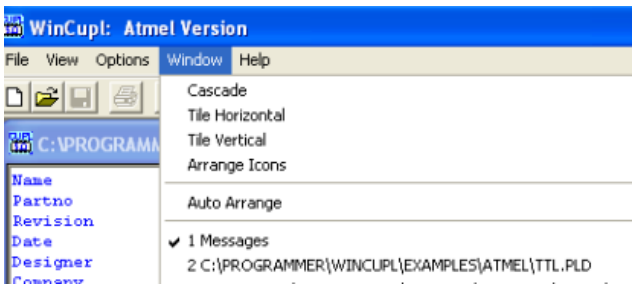


Herefter skulle der gerne være skabt en ”.Jed” fil i /Wincupl-mappen på PC-en.

Det er denne fil, der skal brændes over i GAL-kredsen.

Er der fejl ved compileringen, vil det være vist i LST-filen, og evt. i en .DOC-fil.

Ellers se i / Windows / Message-vinduet



Simulering af det design, man har lavet.

I WinCUPL er det muligt, at simulere sine ligninger. Men det er ret besværlig. Det springer vi over!

Brænding af GAL16V8

Til at brænde Jedec-filen over i GAL-kredsen, bruger vi et specielt brænderudstyr. Det er samme brænder, Leaper 48, der kan bruges til at programmere Microcontrollere.

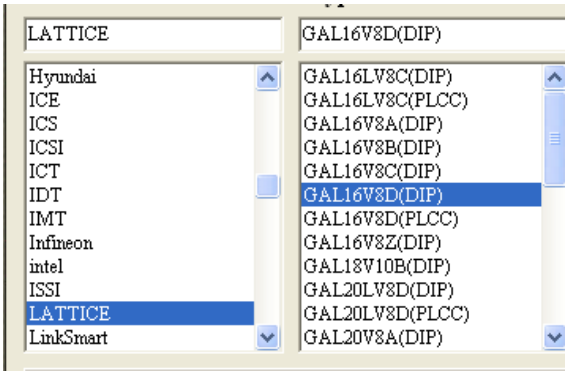
Skærmbilledet til brænderprogrammet ser således ud:

Til brænderen hører en mængde drivere. En driver til hver type programmerbar kreds, den kan håndtere.

Den rigtige driver vælges ved at klikke på ”Type”.

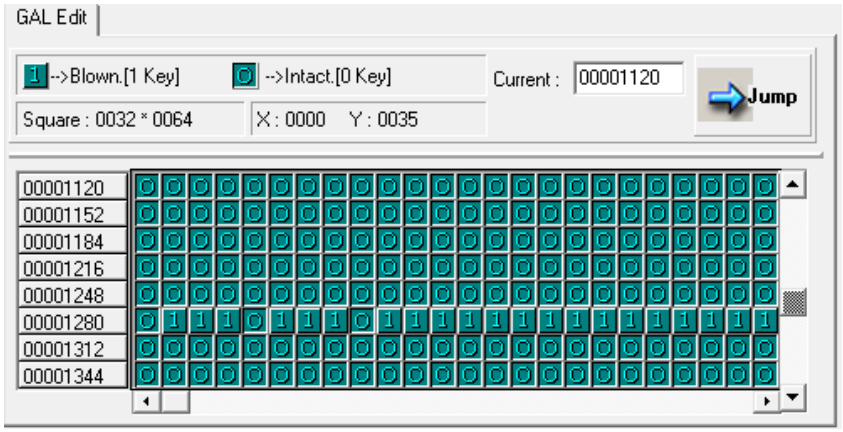
Der vælges

Lattic / GAL16V8D(DIP).



Nu kan Jedec-filen læses ind i brænderprogrammet.

Vælg ”Load”, og find filen. Den hedder xxx.JED.



I brænderprogrammet er der mulighed for at editere i fuse-mappen.

Men det skal man nok lade være med !!

Her er vist lidt af en fuse-map.

Den GAL-kreds, der ønskes programmeret, isættes brænderudstyret, som vist på skitsen på udstyret.

Løft håndtaget, og isæt IC-en, altid helt fra bunden, og med IC-ens Pin 1-markering opad.

Herefter startes programmeringen med den grønne pil.

Og så kan kredsen afprøves. Husk, det kun er en 5-volts kreds.

/**/

Opgaver til GAL-kredse.

Opgave 1: **Gate-kombinationer:**

Lav en fumlebrædt-øvelse. Afprøv forskellige gate-kombinationer.
Monter en lysdiode på udgangen, - husk at beregne formodstande.
Husk Pull Down modstande på indgange, så de ikke svæver.

Kald fx 3 indgange A, B og C.
Kald fx udgange for Q1, Q2, Q3 osv.

Konfigurer kredsen så:

$$Q1 = \bar{A}$$
$$Q2 = AB$$
$$Q3 = A \cdot B \cdot C,$$
$$Q4 = A + B + C$$
$$Q5 = \overline{A \cdot B + AC}$$

Opgave 2: **Tæller og 7-segment:**

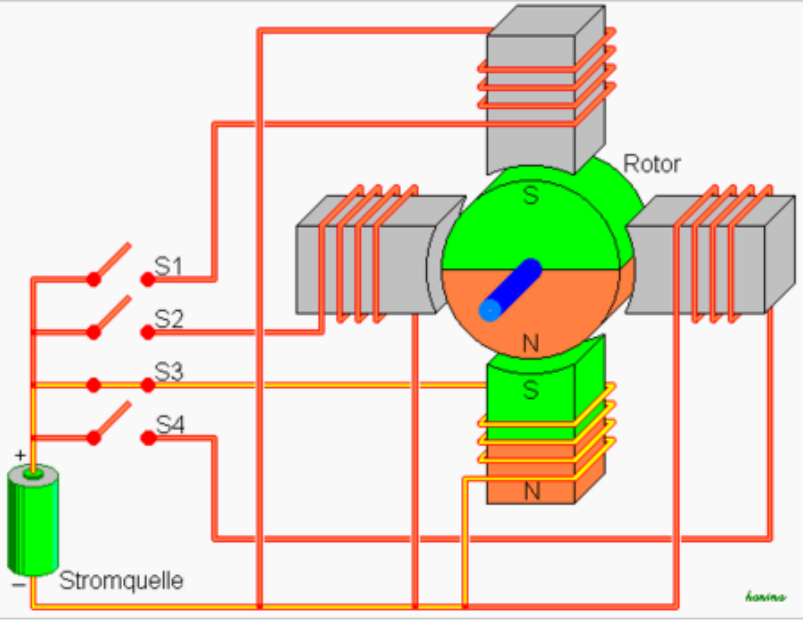
Opbyg en 4093 Oscillator, eller gør brug af en generator.

Opbyg et kredsløb med en 4518 tæller og brug en GAL-kreds til at konvertere det binære output fra tælleren til et 7-segment.

Hvis der bruges en 4520 tæller skal 7-segmentet kunne vise fra 0 til 9 og A til F.

Husk formodstande.
Se Orcad sim af Nand-gate osc.
Obs: Lav selv datablad for 7-segmenter

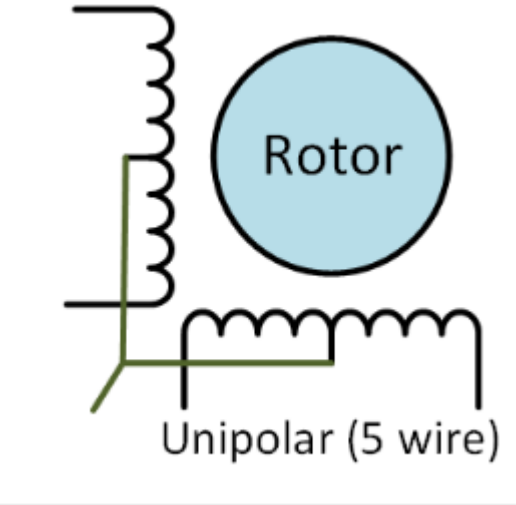
Lige igen lidt om Stepmotor:

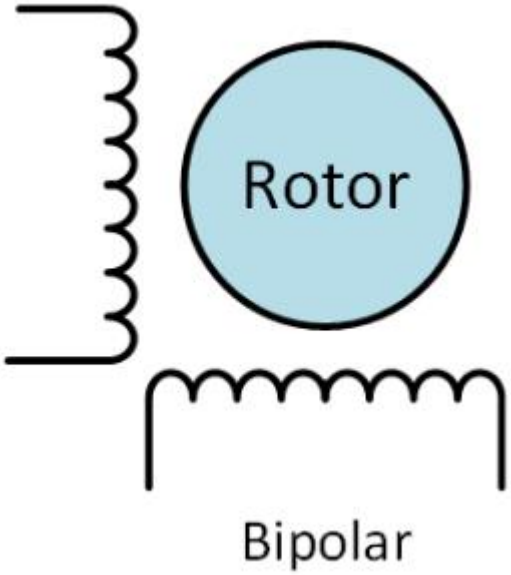


Principdiagram for stepmotor

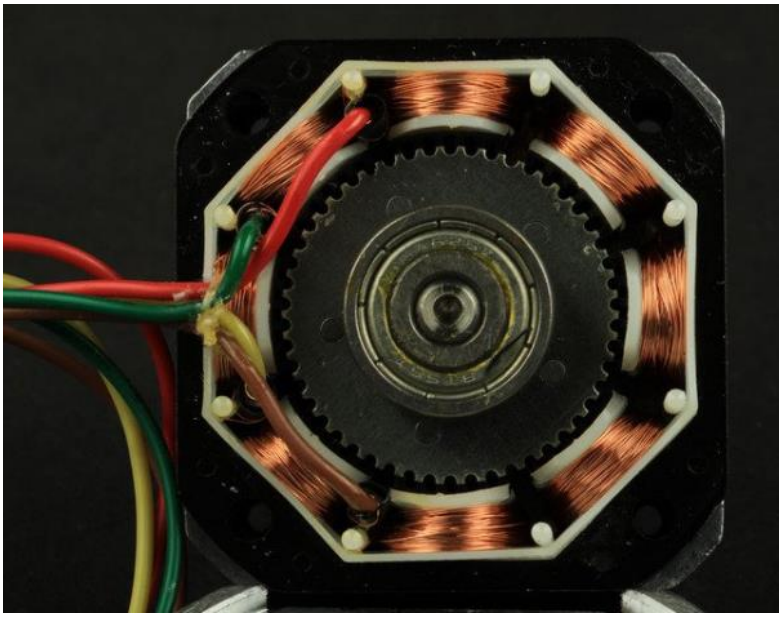
Det er bare ikke rigtigt. Ville jo betyde at for hver clockpuls ville stepmotoren dreje 90 grader.

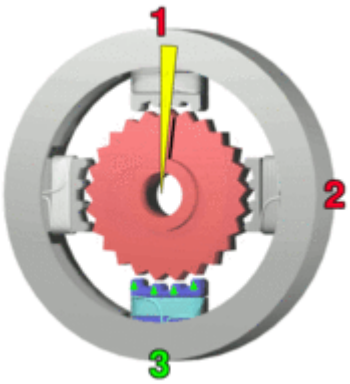
Der er basale viklingsmetoder for stepmotorer:





Et billede af hvordan indmaden i en stepmotor er lavet med tænder.

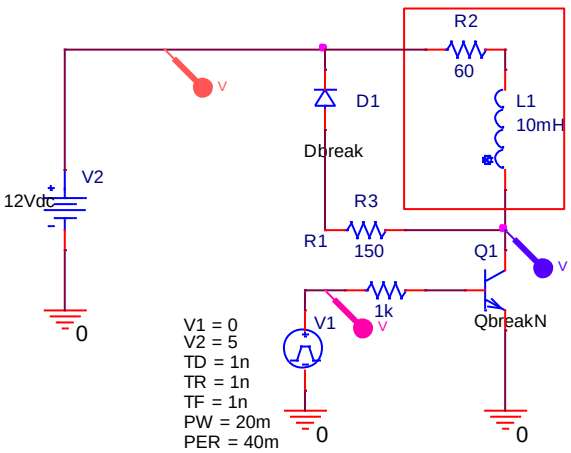




Rigtigt princip !!

Se: https://en.wikipedia.org/wiki/Stepper_motor

Se på spolekollaps



Opgave 3: **Stepmotor omløbs-reversering:**

Brug en 4093 oscillator til at give pulser til en 4017 tæller.
Test med lysdioder.

Brug tællerens output til at styre en forstærkerblok bestående af switch-transistorer.

Kobl en stepmotor på. Husk Freewheel diode.

Se på spolekollaps. Hvordan kan spolens switch off tid forøges ??

Se Orcad simulation !!

Design nu en GAL som omformer så man ved tryk på en switch kan reversere stepmotorens omløbsretningen.

Opgave 4: **Indbygget sekvensgenerator til Stepmotor**

Design en GAL-kreds med indbygget sekvensgenerator til at styre en stepmotor.
Herved kan man spare tælleren 4017!

Der skal både være indgange til Reset og til Direction.

Obs: Asyncron Reset og Syncron Preset findes ikke i 16V8, først i 22V8 !!

Derfor skal der tages hensyn hertil i ligningerne.

Obs. Reset-funktionen skal tvinge 1 og kun 1 udgang høj

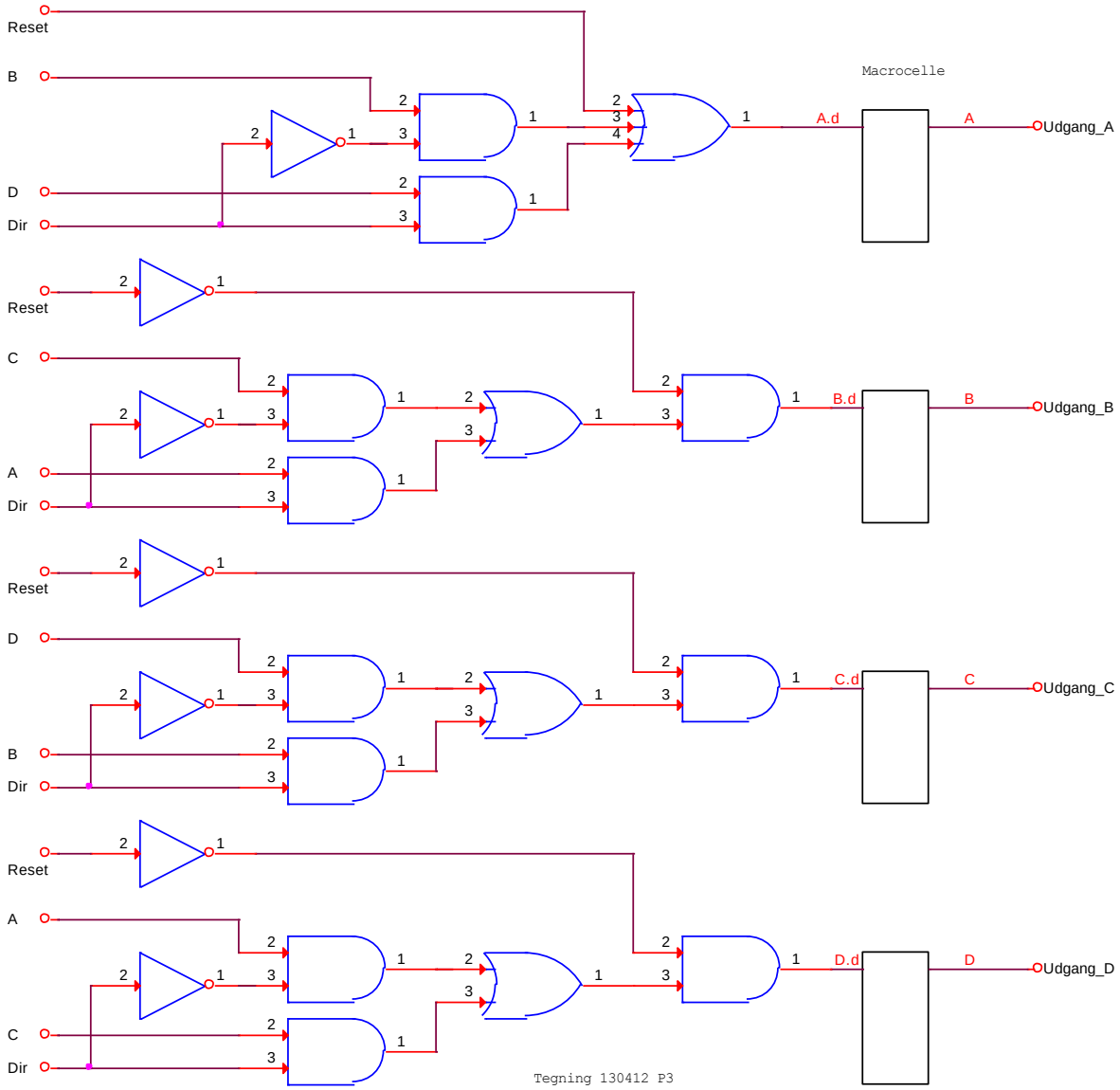
Evt. kan man også indbygge en oscillator i GAL-kredsen. (med hysterese)

Obs: GAL16V8 er udgåede

The most suitable replacements for the GAL16V8 from Atmel are the ATF16V8B/BQL and ATF16V8C/CZ. They are pin-to-pin and JEDEC fusemap compatible. Their AC and DC characteristics are very similar but there can be a few minor differences.

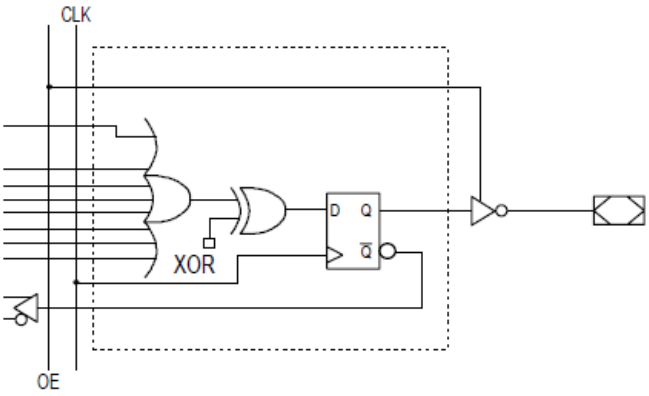
Kilde: http://atmel.force.com/support/articles/en_US/FAQ/GAL16V8-replacement

Diagrammet til opgave 4 kunne se således ud:



Husk at ben 11 /OE skal stelles!

Overvej Clock-signalet !!



Registered Configuration for Registered Mode

- SYN=0.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this output configuration.
- Pin 1 controls common CLK for the registered outputs.
- Pin 11 controls common $\overline{OE}$ for the registered outputs.
- Pin 1 & Pin 11 are permanently configured as CLK & $\overline{OE}$.

Kode til at styre en stepmotorer, hvor GAL-kredsen selv genererer sekvensen. Med reset, og direction-input!

Bemærk, at pin 1 er permanent clock, & Pin 11 er /(OE)

```
Device   g16v8 ;

/* *************** INPUT PINS *********************/
Pin 1 = clk ;
PIN  2   = Dir ;
PIN  3   = Res ;

/* *************** OUTPUT PINS *********************/

PIN  18   = A ;
PIN  13   = B ;
PIN  14   = C;
PIN  15   = D;

/* *************** Equations *********************/

A.d = D&Dir # B&!Dir # Res;       /* Set  */
B.d = (A&Dir # C&!Dir) & !Res  ;
C.d = (B&Dir # D&!Dir) & !Res ;
D.d = (C&Dir # A &!Dir) & !Res ; /*  For at resette */

/*  Husk, at OE på ben 11 skal stelles */
```

Overvej hurtigere kollaps af magnetfeltet.

Overvej Power-on reset / Preset.