LEMBAR PERSETUJUAN

PERENCANAAN DAN PEMBUATAN DOWNLOADER IC GAL16V8 DAN 22V10 BERBASISKAN VHDL

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat Memperoleh Gelar Sarjana Teknik

Disusun oleh :

MUHAMMAD TOHIRON

00.17.175

Mengetahui	Diperiksa Dan Disetujui
Ketua Jurusan Teknik Elektro S-1	Dosen Pembimbing
Ir. F. Yudi Limpraptono, MT	**Joseph Dedy Irawan, ST MT**
NIP Y.1039500274	NIP 132315178

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
KONSENTRASI TEKNIK ELEKTRONIKA S-1

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	: Muhammad Tohiron
NIM	: 00.17.175
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika
Judul Skripsi	: Perencanaan dan Pembuatan Downloader IC GAL 16V8 dan 22V10 Berbasiskan VHDL

Dipertahankan Dihadapan Tim Penguji Skripsi Jenjang Strata Satu (S-1) Pada :

Hari : Sabtu

Tanggal : 18 Maret 2006

Dengan Nilai : 79,6 (B+)

Panitia Ujian Skripsi

Ketua

(Ir. Mochtar Asroni, MSME)
NIP. Y.1018100036

Sekretaris

(Ir. F. Yudi Limpraptono, MT)
NIP. Y.1039500274

Anggota Penguji

Penguji I

Ir. M. Luqman, MS

Penguji II

Ir. Eko Nurcahyo

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Atas rahmat ALLAH SWT laporan skripsi ini dapat terselesaikan dengan baik, rasa syukur dan ALHAMDULILLAH ku persembahkan hanya kepada-MU

. . . . .sebuah karya seorang anak manusia . . . . .
. . . .kupersembahkan untuk. . . .

Ayah dan Ibu yang selalu menyayangiku, terima kasih atas semua DO'A dan Cintanya s'Moga banyak rizki dan Bahagia dunia akhiratnya. . . AMIEN

Semua keluarga besarku dan saudara2x ku semua, terima kasih atas dukungan kalian semua, akhirnya perjuanganku selama lima tahun setengah ini dapat tercapai. Buat Riska, cepetan Gede dan jangan pernah nakal lagi

******* Kepada Teman 2x yang memberikan dukungan and gangguan *******

Shodiq (yang telah setia menemaniku tidur selama aku di Ngalam) jangan pernah putus asa, jangan pernah berhenti untuk kuliah, berjuanglah terus supaya kamu cepat wisuda.,
"master in master" downloader Syarifal terima kasih atas bantuannya selama ini, yang serius kursusnya, dan mudah2an dapet kerja
Pak KuN karena "JAS"mu aku bisa lulus,
Aries (Thanks for IRDAnya) kapan Loe LULUZ?,
DK(double K – Kacak Kelik) matur nuwun atas pinjaman JASnya (Walaupun kegedean),
mas CIPUT bisa nggak desember wisuda?,
temen-temenku yang udah lulus duluan (Kholis, Mahesa, Gendon, Febri, Habib, Udin, Yoyok)
Wahid (Memet) Lagi patah hati ya?
Koncoku di ELKA, khususnya angkatan 2000
Terima kasih juga buat temen-temen yang satu kelompok seminar hasil (Fadhil, Surya, Mumun, Sigit, Indra)

Special thank's to MAMA SANTY, terima kasih atas dukungan, pengorbananmu dan doa-doanya. Doanya MAMA selalu terkabul.
Tolong jaga "Pentika nPentiki" itu adalah milikku selamanya
Terima kasih juga sudah buat TYARA sama PEGGY jadi Lengket Banget. Mereka tidak dapat dipisahkan lagi.
Mereka itu sudah buat kita bahagia selamanya

http://iron-santy.tripod.com

ABSTRAK

Muhammad Tohiron, 00.17.175, *Perencanaan dan Pembuatan Downloader IC GAL 16V8 dan 22V10 berbasiskan VHDL.* Skripsi. Jurusan Teknik Elektro Konsentrasi Teknik Elektronika S1 Fakultas Teknologi Industri. Institut Teknologi Nasional Malang. Dibawah bimbingan Joseph Dedy Irawan, ST. MT.

Keyword : downloader, gal16v8, gal22v10

Skripsi ini bertujuan untuk merancang dan membuat perangkat keras pengisi dan penghapus IC yang akan digunakan untuk memprogram IC GAL 16V8 dan 22V10 yang diinterfacekan ke komputer menggunakan port parallel dengan menggunakan power supply dari luar dengan tegangan sebesar 12 Volt.
Alat ini dibuat dengan memanfaatkan IC GAL 22V10 sebagai IC master yang akan digunakan untuk mengatur programing voltage, strobe dan row addres dai IC slave, yaitu IC GAL 16V8 dan 22V10.
Hasil pengujian menunjukkan bahwa alat ini dapat menghapus dan mengisi IC GAL 16V8 dan 22V10 pada komputer dengan spesifikasi kompute Pentium II 300Mhz ,RAM 64M dan dengan menggunakan operating system Windows 98SE

KATA PENGANTAR

Dengan memanjatkan puji dan syukur kehadirat ALLAH SWT, karena limpahan rahmat serta karunia-NYA, penyusun telah menyelesaikan laporan skripsi dengan judul "PERENCANAAN DAN PEMBUATAN DOWNLOADER IC GAL 16V8 DAN 22V10 BERBASISKAN VHDL" ini. Selain itu diharapkan laporan skripsi ini bisa memberikan manfaat bagi teman-teman mahasiswa yang ingin memanfaatkan alat downloader ini.

Tak lupa pada kesempatan kali ini, penyusun ingin mengucapkan terima kasih dan penghargaan kepada :

1. Bapak Dr. Ir. Abraham Lomi, MSEE selaku rector ITN Malang.
2. Bapak Joseph Dedy Irawan ST, MT selaku dosen pembimbing skripsi.
3. Kedua orang tua serta keluarga besar penyusun, terima kasih atas dukungan moral dan spiritualnya.
4. Teman-teman semua yang telah banyak membantu dalam menyelesaikan laporan skripsi ini.

Penyusun telah berusaha semaksimal mungkin untuk menyelesaikan laporan skripsi ini dengan sebaik-baiknya, tetapi sebagai manusia yang memiliki kekurangan, penyusun menyadari bahwa laporan skripsi ini masih jauh dari sempurna. Maka penyusun sangat mengharapkan kritik dan saran yang bersifat membangun untuk kesempurnaan laporan skripsi ini lebih lanjut.

Malang, Maret 2006

Penyusun,

Muhammad Tohiron

00.17.175

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam perkembangan ilmu pengetahuan dan teknologi, khususnya dalam bidang elektronika dewasa ini, telah membawa perubahan dan kemajuan yang sangat pesat dalam peradaban dan kehidupan umat manusia. Begitu juga dengan perkembangan teknologi komponen elektronika itu sendiri. Misalnya, IC digital yang berbasis PLDs (Programable Logic Device), seperti GAL 16V8, 22V10, dan lain sebagainya.

Dengan teknologi VHDL (Very High Speed Integrated Circuit Deskription Language) yang merupakan suatu bahasa pemrograman tingkat tinggi yang digunakan untuk memprogram suatu IC digital berbasis PLDs (Programable Logic Devices), dimana terdiri dari sejumlah gerbang-gerbang yang dapat diprogram dengan cara memutuskan fuse yang terdapat pada IC tersebut.

Aplikasi IC GAL sebagai pengontrol utama sangat membantu dalam mengotomatisasi suatu sistem kontrol untuk mempermudah dan mempercepat proses kinerja suatu alat yang digunakan, sehingga dewasa ini sudah banyak desain kontrol yang memakai IC GAL sebagai pengontrol utama. Penerapan IC GAL untuk berbagai sistem kontrol tentunya memerlukan software yang berbeda, sesuai dengan alat yang akan dikontrrol oleh IC GAL. Salah satu IC GAL yang dapat digunakan sebagai pengontrol utama yaitu GAL16V8 dan GAL22V10, dimana IC GAL ini terdapat gerbang-gerbang logika yang dapat diprogram..

Memperhatikan kenyataan tersebut, maka muncul suatu ide atau gagasan untuk mewujudkan peralatan yang mampu mengisi program (software) pada IC GAL dengan program yang dapat diisi, dapat dihapus, dan dapat diisi kembali dengan program yang baru dengan menggunakan IC GAL22V10 sebagai control dan pengatur line fuse IC yang akan diisi. Dimana IC GAL 22V10 ini harus diisi terlebih dahulu dengan program yang dikehendaki. Dengan demikian akan mempermudah pemakai IC GAL ini dalam penerapan berbagai alat yang akan dikontrol.

1.2. Rumusan Masalah

Adapun permasalahan yangs edang dihadapi dalam penulisan skripsi ini adalah :

1. Bagaimana merancang rangkaian downloader IC GAL16V8 dan 22V10 dalam satu rangkaian (peletakan soket terpisah pada tiap IC).
2. Bagaimana proses pembuatan software yang dibuat.
3. Bagaimana perancangan IC GAL22V10 sebagai kontrol dan pengatur line fuses.

1.3. Tujuan

Adapun tujuan dari perencanaan dan pembuatan downloader ini adalah :

1. Merancang dan membuat rangkaian downloader IC GAL 16V8 dan 22V10.

2. Memanfaatkan teknologi VHDL sebagai pengatur control dan line fuses melalui port paralel.
3. Membuat rangkaian programming modes menggunakan IC GAL22V10 yang dapat diisi dengan progam yang diinginkan.

1.4. Batasan Masalah

Sehubungan dengan masalah yang dihadapi dalam pembuatan skripsi ini, permasalahan hanya dibatasi dengan tujuan untuk mencegah kemungkinan meluasnya masalah dan penyimpangan dari permasalahan. Pembatasan tersebut antara lain :

1. Perangkat keras IC GAL 16V8 dan 22V10
2. Paralel port printer sebagai jalur pengisian program antara PC dan IC GAL 16V8 dan 22V10.
3. Perangkat keras dan perangkat lunak pengisi IC GAL 22V10 sebagai pengatur control dan line fuses.
4. Perangkat keras dan perangkat lunak pengisi program IC GAL.

1.5. Metodologi

Dalam penulisan skripsi ini penulis manggunakan metode-metode yang sering digunakan. Adapun metode tersebut adalah sebagai berikut :

1. Kajian literature/referensi mengenai komponen-komponen yang digunakan dalam pembuatan alat.

2. Merancang dan membuat alat, kemudian mencoba dalam papan percobaan dan seterusnya merakit alat tersebut dalam PCB.
3. Penyusunan laporan skripsi.

1.6. Sistematika Pembahasan

Dalam pengerjaan laporan skripsi ini kami menggunakan sistematika sebagai berikut :

BAB I : PENDAHULUAN

Memuat latar belakang, tujuan pembuatan alat, batasan masalah, metodeologi penulisan dan sistematika pembahasan.

BAB II : LANDASAN TEORI

Pada landasan teori ini akan dibahas mengenai teori-teori yang mendasari pembuatan alat ini

BAB III : PERENCANAAN DAN PEMBUATAN ALAT

Membahas dan menjelaskan masalah perencanaan dan pembuatan perangkat keras maupun cara pengisian IC GAL 16V8 dan 22V10. Dan perancangan serta pengisian IC GAL 22V10 sebagai rangkaian format programming modes beserta cara kerja rangkaian.

BAB IV : PENGUJIAN ALAT

Memuat hasil pengujian alat IC GAL 16V8 dan 22V10. Pengujian dilakukan dengan menguji tiap blok rangkaian secara keseluruhan

BAB V : PENUTUP

Berisi tentang kesimpulan dan saran perancangan dan pembuatan alat serta kemungkinan pengembangan maupun aplikasi-aplikasi yang dilakukan pada alat yang dirancang bangun

BAB II
LANDASAN TEORI

2.1. Very High Speed Integrated Circuits Hardware Laguage (VHDL)

Dalam merancang suatu sistem digital yang besar dan kompleks dengan cara manual akan mengakibatkan banyak kesulitan. Perancangan secara manual yang dimaksudkan adalah perancangan dengan menggunakan kertas dan pensil, kemudian dibuat suatu prototipe dan setelah itu diuji apakah hasil rancangan tadi dapat bekerja dengan baik atau tidak. Untuk mengatasi hal tersebut diperlukan Computer Aided Design (CAD). Salah satu kegunaan dari CAD adalah merancang dan menganalisa sistem digital yang dibuat. Untuk melakukan fungsi dan kegunaannya tersebut masing-masing CAD menggunakan Hardware Deskription Language (HDL) yang berbeda-beda. Karena HDL yang digunakan berbeda-beda tersebut maka tiap-tiap CAD memiliki kelebihan dan kekurangannya sendiri-sendiri. Dan dewasa ini CAD mulai menggunakan VHDL (Very High Speed Integrated Circuits Hardware Description Laguage) untuk merancang suatu sistem digital.

2.2. Pengisi Dan Penghapusan Program IC GAL 16V8 dan 22V10

Dengan menggunakan alat pengisi dan penghapus program IC GAL khususnya GAL16V8 dan GAL22V10 ini, maka akan memudahkan bagi pengguna IC GAL16V8 dan GAL22V10 untuk bereksperimen dengan program yang berbeda – beda sesuai dengan yang diinginkan. Penggunaan IC GAL16V8

dan GAL22V10 menjadi lebih praktis karena proses pengisian dan penghapusan dapat dilakukan dengan cepat tanpa harus pergi ketempat – tempat yang menyediakan fasilitas pengisian pemrograman. Adapun media yang digunakan untuk pengisian software adalah Personal Computer, dan untuk menghubungkan PC dengan peralatan ini dipergunakan jalur pararel port.

2.3. Port Pararel

Untuk menghubungkan alat yang lain dengan komputer, maka perlu diberi tanbahan berupa port pararel dan port serial. Pada port pararel data dikirim dalam format pararel (sekaligus 8 bit) sehingga data dapat dikirim dengan cepat. Namun diperlukan banyak kabel dan tidak dapat dipakai untuk jarak jauh.

Karena alat tidak akan diletakkan jauh dari komputer, maka kita gunakan port pararel saja. Disamping lebih cepat, pemrograman port pararel lebih mudah bila dibandingkan dengan pemrograman port serial.

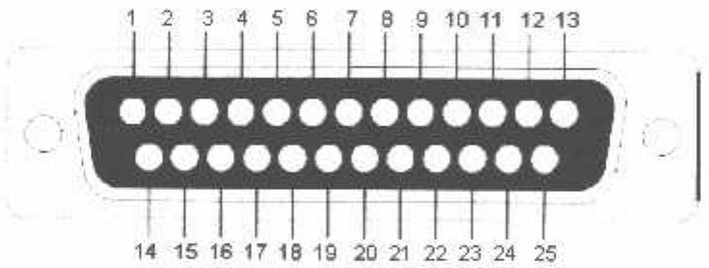
Port pararel biasanya terletak pada video adapter atau di multi I/O card. Port pararel diberi nama LPT0, LPT1, dan LPT2 yang memiliki alamat sendiri – sendiri.

LPT1 atau port printer sebenarnya terdiri dari tiga bagian besar yang masing –masing diberi nama sesuai dengan tugasnya dalam melaksanakan tugas percetakan, tiga bagian tersebut adalah DP, PC, dan PS. DP (Data Port) digunakan oleh komputer dalam hal ini CPU untuk mengirimkan data biner dari karakter (dalam standart ASCII) yang harus dicetak oleh printer. PC mempunyai tugas mengirimkan kode – kode pengontrol dari CPU ke printer, misalnya printer

kehabisan kertas atau kertas telah keluar maka suatu kode status harus dikirimkan ke CPU agar pengiriman data karakter dihentikan untuk memberikan kesempatan memasukkan kertas.

DP, PC, dan PS sebenarnya adalah port – port 8 bit namun hanya DP yang benar – benar port 8 bit. Sedangkan untuk PC, PS hanya beberapa bit saja yang dibutuhkan dalam proses percetakan maka bit – bit yang lain tidak digunakan yang berarti hanya beberapa bit dari port – port ini yang dapat kita manfaatkan untuk keperluan pengontrolan nanti. Perlu diingat pula bahwa DP dan PC adalah port baca/tulis (read / write) sedangakan PS adalah port baca saja (read only).

Slot DB-25 dapat dilihat dari gamber dibawah ini :



Gambar 2.1 Konfigurasi Slot Db-25 Male [1]

[1] Interfacing Port Paralel dan Port Serial dengan Visual Basic 6.0 Hal.50

Tabel 2.1 Alamat Register Port Pada Lpt1 [2]

LPTI	ALAMAT REGISTER
Data Port (DP)	378H
Printer Control (PC)	379H
Printer Status (PS)	37AH

Definisi bit – bit port control (PC) adalah sebagai berikut:

Tabel 2.2 Definisi Bit-Bit Printer Control [3]

Printer Control	Nama	Sifat
PC-0	Strobe	Inverting
PC-1	Autofeed	Inverting
PC-2	Init	normal
PC-3	Select In	Inverting
PC-4	IRQ-7 Enable	normal
PC-5 .... PC-7	Tidak Dipakai	

Sifat PC yang inverting, berarti bila logika PC tinggi, pada konektor DB25 akan berlogika rendah dan demikian juga sebaliknya. Definisi bit – bit dalam port status (PS) adalah sebagai berikut:

Tabel 2.3 Definisi Bit-Bit Printer Status [4]

Printer Status	Nama	Sifat
PS-0...PS-2	Tidak Dipakai	
PS-3	Error	Normal
PS-4	Select	Normal
PS-5	Paper End	Normal
PS-6	Acknowledge	Normal
PS-7	Budy	inverting

[2], [3], [4] Bereksperimen Dengan Mikrokontroler 8031 Hal.116

Letak Data Port (DP), Port Status (PS), dan Port Control (PC) pada konektor DB-25 adalah sebagai berikut:

Tabel 2.4 Konfigurasi Letak Pin Pada Db-25 [5]

Nama	Letak Pin Pada DB-25
DP-0	2
DP-1	3
DP-2	4
DP-3	5
DP-4	6
DP-5	7
DP-6	8
DP-7	9
PS-3	15
PS-4	13
PS-5	12
PS-6	10
PS-7	11
PC-0	1
PC-1	14
PC-2	16
PC-3	17
Ground	18 – 25

[5] Bereksperimen Dengan Mikrokontroler 8031 Hal.116

2.4. IC GAL 16V8

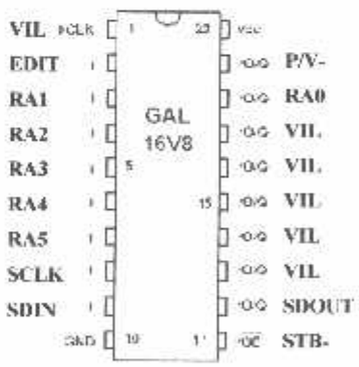
Di pasaran terdapat beberapa jenis IC GAL yang diproduksi oleh beberapa pabrik seperti AMD (Advance Micro Devaice), Cypress Semiconductor, Lattice Semiconductor, NS (National Semiconductor), Samsung , dan lainnya. Dalam penyusunan skripsi ini penulis menggunakan IC GAL keluaran Lattice dengan type GAL 16v8.

Gal 16v8 memiliki beberapa spesifikasi antara lain :

- GAL 16V8 merupakan IC SPLD (Simple Programing Logic Device) yang terdiri dari EECMOS PLD dengan kinerja yang tinggi.
- Terdiri dari 16 input dan 8 output 1 jalur clock 1 buah OE yang dapat dikombinasi dengan gerbang-gerbang logic dan flip-flop melalui software yang di programkan ke GAL secara efisien.

2.4.1. Konfigurasi Pin – Pin IC GAL 16V8

IC GAL 16V8 terdiri atas 20 pin, dengan konfigurasi sebagai berikut :



Gambar 2.2 Konfigurasi Pin – Pin Ic Gal 16v8 [6]

[6] Lattice Semiconductor, Data Sheet GAL16V8 Hal.1

Fungsi dari tiap – tiap pin tersebut adalah:

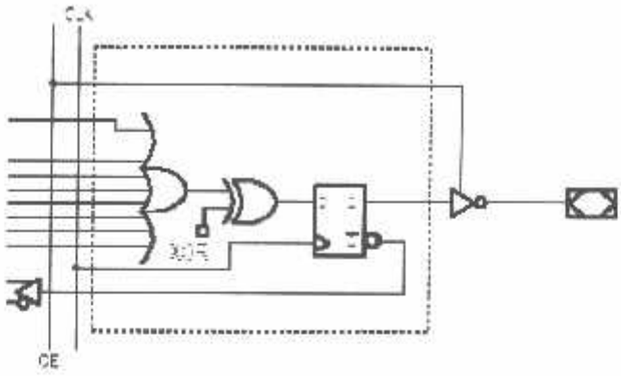
1. Pin 1sebagai clock GAL.
2. Pin 2 sampai pin 9 sebagai jalur input.
3. Pin 10 sebagai ground.
4. Pin 11 adalah jalur input dan OE
5. Pin 12 sampai pin 19 sebagai jalur input output.
6. Pin 20 sebagai Vcc.

2.4.2. Output Logic Macrocell (OLMC) IC GAL 16V8

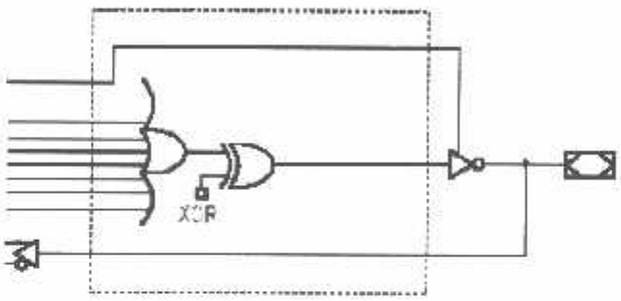
Pada IC GAL 16V8 terdapat tiga macam mode konfigurasi OLMC. Yaitu mode simple, complex dan registered. Dua bit SYN dan AC0, mengendalikan mode konfigurasi untuk semua macrocells. Bit XOR dari tiap-tiap macrocells mengendalikan polaritas dari outputan ketiga mode, sedangkan bit AC1 dari tiap macrocell mengendalikan konfigurasi input/output.

2.4.2.1. Mode Registered

Pada registered mode, macrocell diatur sebagai output atau fungsi I/O. Konfigurasi arsitektur yang tersedia dalam mode ini sama dengan 16R8 dan 16RP8 dari tipe PAL dengan berbagai perubahan polaritas, I/O dan penempatan register. Beberapa macrocell dapat diatur, kecuali pin 1 (clock) dan pin 11 (OE). Dalam mode registered pin ini tidak dapat diatur sebagai input.

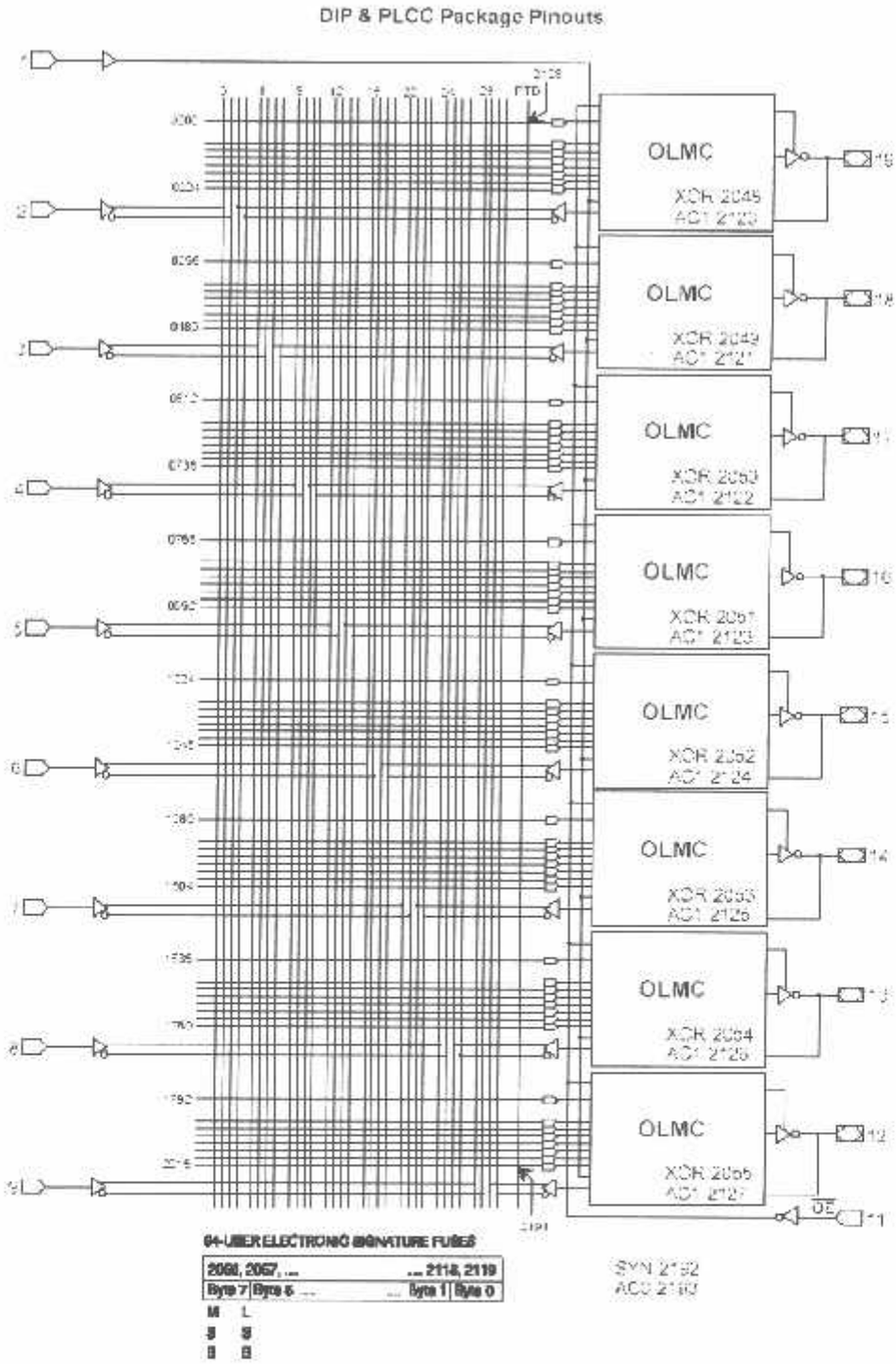


Gambar 2.3 Bentuk Registered Untuk Mode Registered [7]



Gambar 2.4 Bentuk Combinatorial Untuk Mode Registered [8]

[7], [8] Lattice Semiconductor, Data Sheet GAL16V8 Hal.5

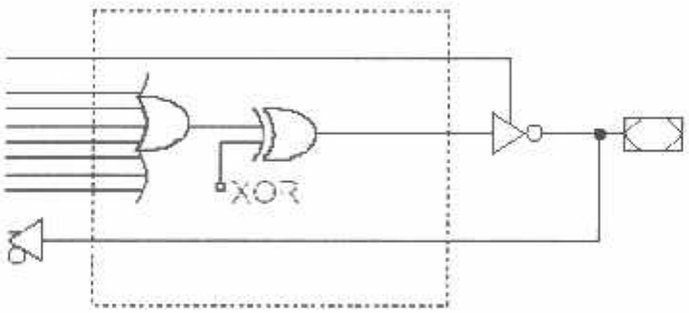


Gambar 2.5 Diagram Logika Mode Registered [9]

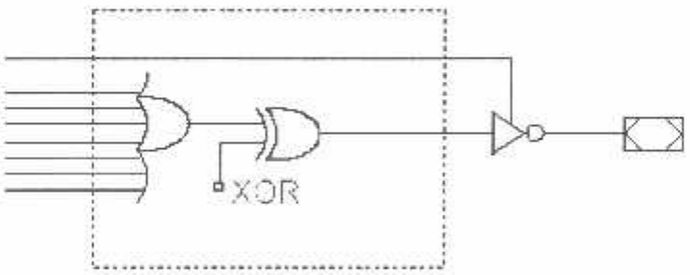
[9] Lattice Semiconductor, Data Sheet GAL16V8 Hal.6

2.4.2.2. Mode Complex

Pada mode complex, macrocell yang diatur hanya output atau fungsi I/O. Bentuk wujud arsitektur yang ada dalam mode ini seperti pada umumnya IC 16I.8 dan 16P8 dari tipe PAL dengan polaritas programable pada setiap macrocellnya. Pada input/output yang ada dapat di implementasikan sebagai fungsi I/O. Dua macrocell paling luar (pin12 dan 19) tidak mempunyai kapasitas sebagai input.

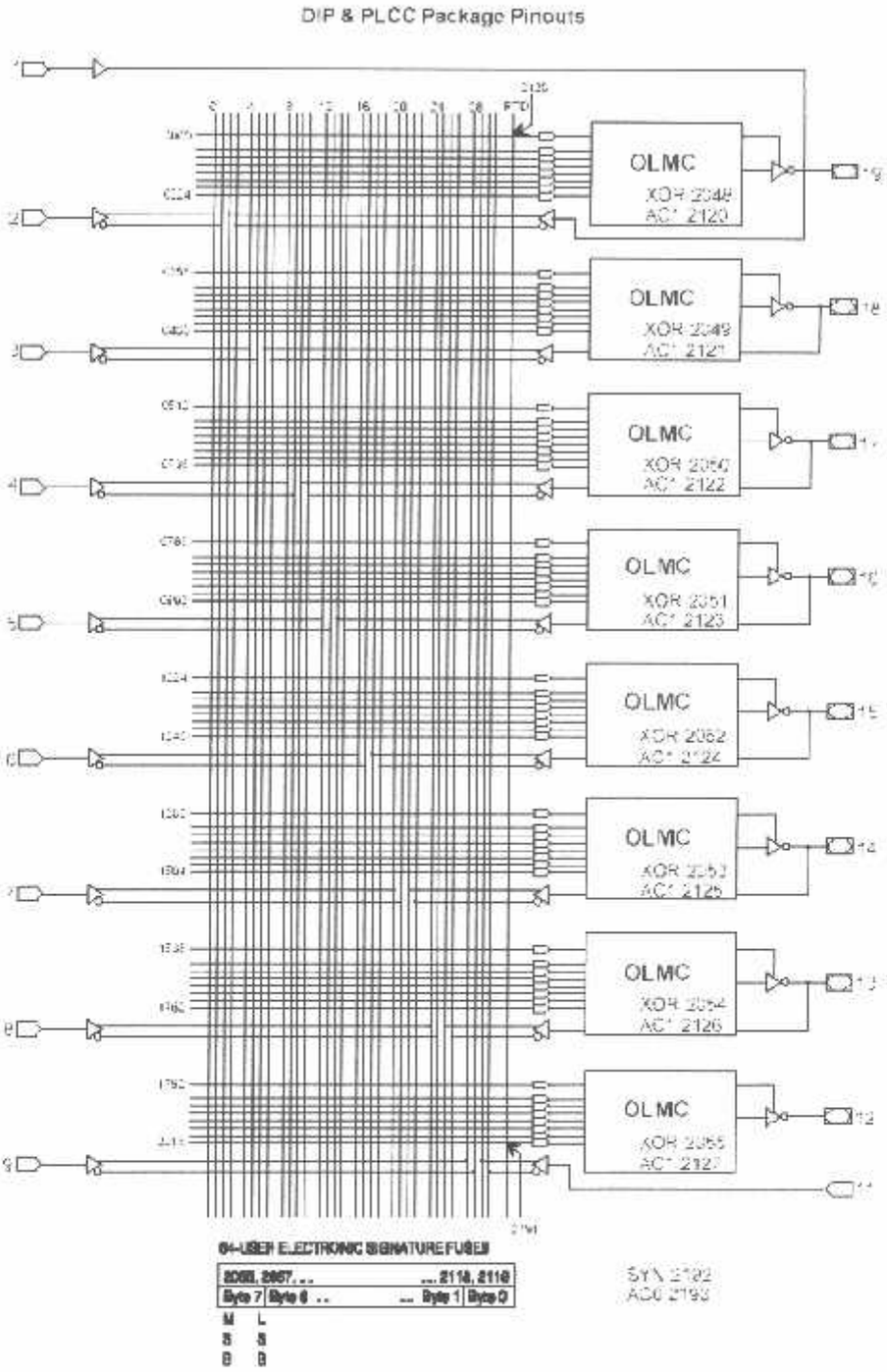


Gambar 2.6 Bentuk I/O Combinatorial Untuk Mode Complex [10]



Gambar 2.7 Bentuk Output Combinatorial Untuk Mode Complex [11]

[10], [11] Lattice Semiconductor, Data Sheet GAL16V8 Hal.7

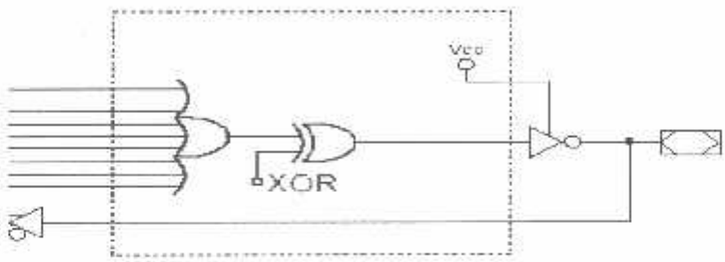


Gambar 2.8 Diagram Logika Mode Complex [12]

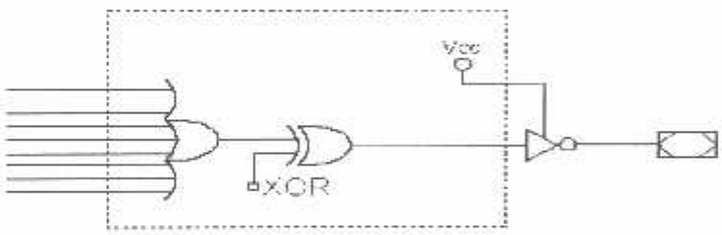
[12] Lattice Semiconductor, Data Sheet GAL16V8 Hal.8

2.4.2.3. Mode Simple

Pada mode simple, macrocell diatur sebagai input atau output combinatorial yang selalu aktif. Bentuk wujud yang ada dalam mode ini adalah sama dengan IC 10L8 dan 12P6 dari tipe PAL, dengan banyak perubahan pada polaritas outputan atau pada pilihan inputan. Pada pin 1 dan 11 selalu tersedia data input ke dalam array AND. Dua pusat macrocell (pin15 dan 16) tidak dapat digunakan sebagai pin I/O atau inputan dan hanya sebagai output saja.



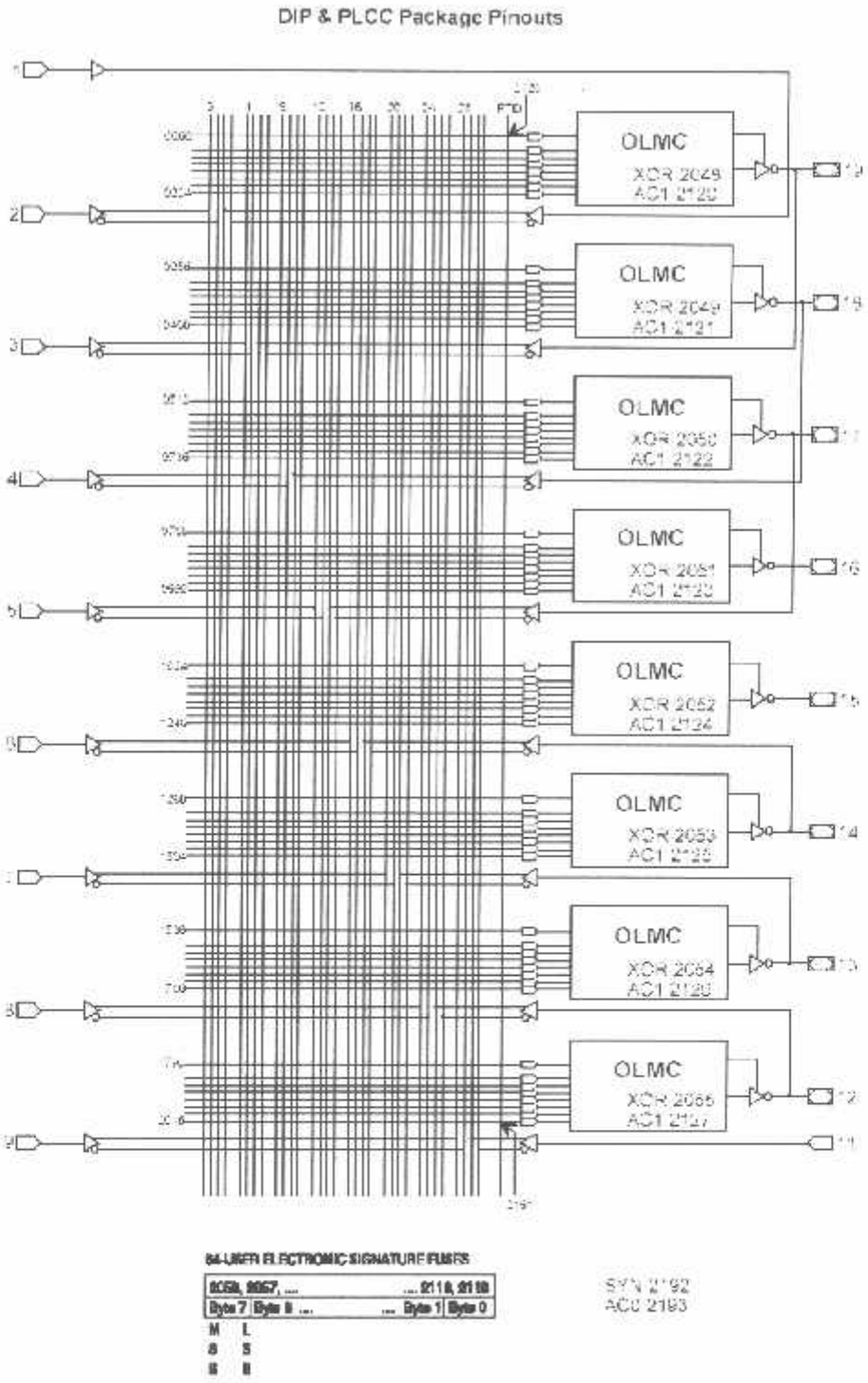
Gambar 2.9 Bentuk Output Combinatorial Dengan Feedback untuk Mode Simple [13]



Gambar 2.10 Bentuk Output Combinatorial untuk Mode Simple [14]

Gambar 2.11 Bentuk Inputan Yang Dipakai untuk Mode Simple [15]

[13], [14], [15] Lattice Semiconductor, Data Sheet GAL16V8 Hal.9



Gambar 2.12 Diagram Logika Mode Simple [16]

[16] Lattice Semiconductor, Data Sheet GAL16V8 Hal.10

2.5. IC GAL 22V10

GAL (Generic Array Logic) merupakan array gerbang logika (array of logic gate) yang dapat diprogram oleh pemakai. GAL termasuk kelompok PLD yang gerbang AND-nya dapat diprogram tapi gerbang OR-nya tidak dapat diprogram (programable AND-fixed OR). GAL juga termasuk dalam EEPLD (Electrically Erasable PLD), yaitu PLD yang dapat dihapus secara listrik. Pada GAL penghapusan ini terjadi dalam skala waktu milisekon (ms) atau lebih tepatnya sekitar 50 milisekon. Ada beberapa keuntungan jika kita menggunakan komponen GAL ini, seperti:

- GAL difabrikasi dengan teknologi Very High Speed E2CMOS (Electrically Erasable CMOS) yang mempunyai kemampuan test dan kandalan tinggi.
- Konsumsi daya yang rendah.
- Mempunyai Output Logic Macrocell (OLMC) sehingga perancang dapat membuat konfigurasi output yang diinginkan.
- Dalam sistem penghapusannya jauh lebih cepat bila dibandingkan dengan EPLD yang menggunakan sinar ultra violet. EPLD dengan teknologi UVCMOS (Ultra Violet CMOS) akan membutuhkan waktu hapus sekitar 20 menit, sedangkan untuk GAL hanya membutuhkan waktu hapus sekitar 50 milisekon.

Terdapat beberapa jenis GAL sesuai dengan kapasitas dari susunan OLMC-nya, yaitu: GAL16V8, GAL 20V8, GAL 22V10, GAL 6001, dan masih ada beberapa kompnen GAL yang lain. Perbedaan mendasar dari komponen-

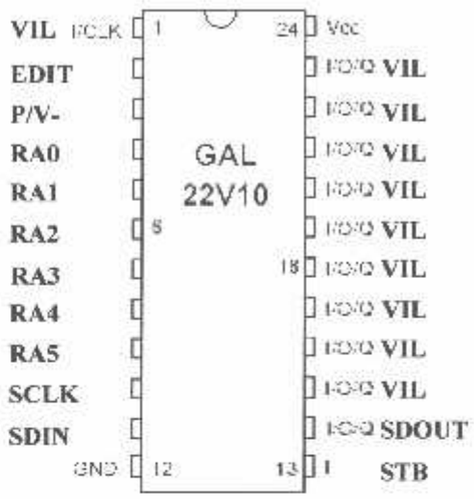
komponen GAL ini terletak pada jumlah pin yang tersedia (baik yang digunakan sebagai input atau yang digunakan sebagai I/O, banyaknya susunan gerbang AND serta jumlah dari OLMC yang disediakan.

Didalam GAL 22V10 terdapat OLMC yang dapat diprogram dan juga tersedia macam-macam distribusi product term. Masing-masing macrocell dapat secara sendiri dengan cara memprogram keadaan konfigurasi bit-bit. Polaritas dari output macrocell juga dapat dipilih dengan kontrol polaritas. Macrocell yang programable juga mempunyai feddback yang dapat digunakan sebagai input, feedback ini dapat dilangsungkan dari register atau dari buffer I/O.

Komponen GAL 22V10 ini sama dengan komponen PALCE 22V10 yang banyak di jual di pasaran, baik itu jumlah macrocellnya maupun jumlah pin-pin I/O yang disediakan. Inovasi lain dari 22V10 selain macrocell adalah disediakannya distribusi product term yang bervariasi. GAL 22V10 memberikan fasilitas 22 input dan 10 output, dan didalam 22V10 juga terdapat 10 macrocell, dimana masing-masing macrocell mempunyai atau dilengkapi dengan multiplexer. Pada umumnya waktu delay propagasi (propagation delay time), atau sejumlah waktu yang diperlukan agar output kombinasional menjadi valid setelah input diberikan ke pin-pin IC dari GAL 22V10 adalah tidak lebih dari 4 nano second untuk versi standar.

2.5.1. Konfigurasi Pin – Pin IC GAL 22V10

IC GAL 22V10 terdiri atas 24 pin, dengan konfigurasi sebagai berikut :



Gambar 2.13 Konfigurasi Pin – Pin IC GAL 22V10 [17]

Fungsi dari tiap – tiap pin tersebut adalah:

1 Pin 1sebagai clock GAL.

2 Pin 2 sampai pin 11 sebagai jalur input.

3 Pin 12 scbagai ground.

4 Pin 13 adalah jalur input.

5 Pin 14 sampai pin 23 sebagai jalur input output.

6 Pin 24 sebagai Vcc.

[17] Lattice Semiconductor, Data Sheet GAL22V10 Hal.1

2.5.2. Output Logic Macrocell (OLMC) IC GAL 22V10

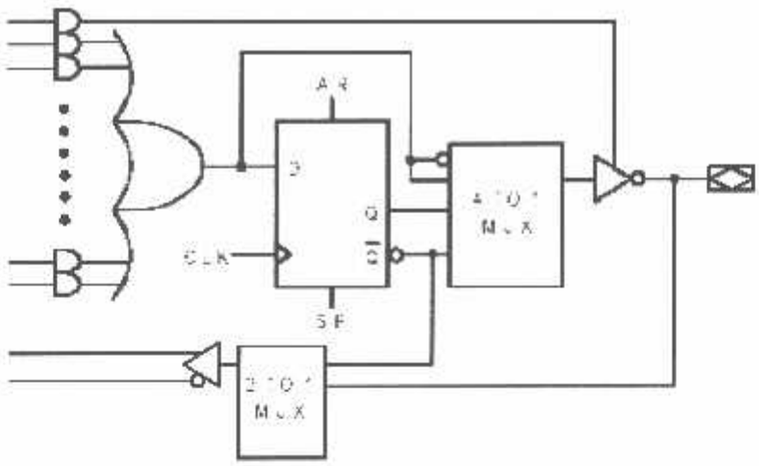
GAL 22V10 memiliki beberapa variabel product terms per OLMC. Dari 10 yang tersedia, dua OLMC memiliki akses ke delapan product terms (pin 14 dan pin 23), dua memiliki 10 product terms (pin15 dan pin 22), dua memiliki 12 product terms (pin 16 dan pin 21), dua memiliki 14 product terms (pin 17 dan pin 20), dan dua OLMC memiliki 16 product terms (pin 18 dan pin 19). Sebagai tambahan pada product terms tersedia sebagai logika, setiap OLMC memiliki tambahan product terms yang didedikasikan untuk output kontrol enable.

Polaritas output masing-masing OLMC dapat diprogram sendiri-sendiri, dijadikan benar atau sebaliknya, mode combinaorial atau mode registered. Ini mengizinkan setiap output secara individu dikonfigurasi menjadi aktif high atau aktif low.

GAL 22V10 memiliki product terms untuk Asynchronous Reset (AR) dan product terms untuk synchronous Preset (SP). Kedua product terms tersebut menjadi bagian dari OLMC. Asynchronous Reset mengeset semua register menjadi nol.

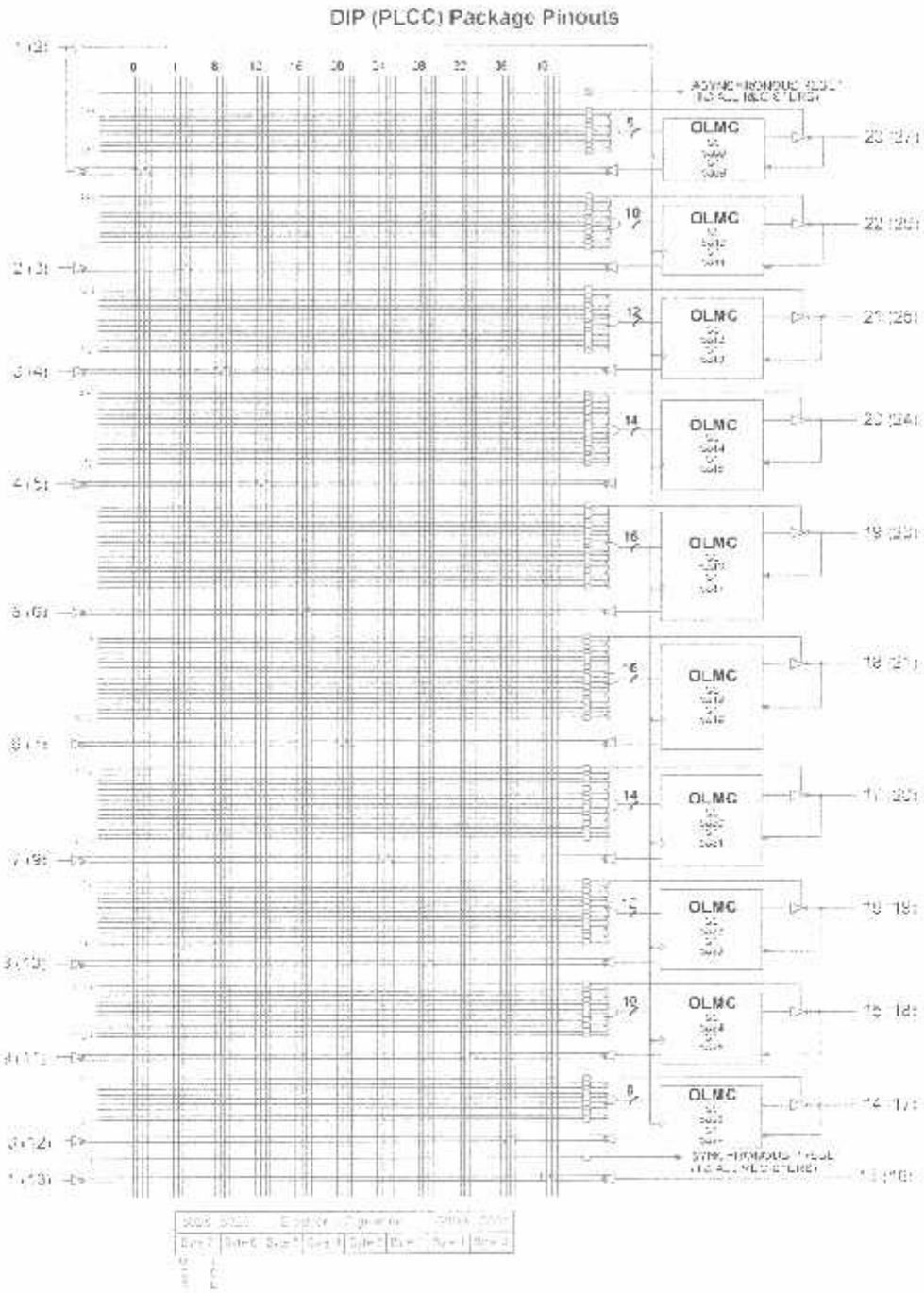
Synchronous Preset mengeset semua register menjadi logika satu saat clock berubah dari nol ke satu.

Diagram Output Logic Macrocell (OLMC) dari IC yang digunakan dapat dilihat pada gambar berikut :



Gambar 2.14 Output Logic Macrocell (OLMC) IC GAL 22V10 [18]

[18] Lattice Semiconductor, Data Sheet GAL22V10 Hal.4



Gambar 2.15 Diagram Logika IC GAL 22V10 [19]

[19] Lattice Semiconductor, Data Sheet GAL22V10 Hal.6

2.6. Implementasi VHDL Pada IC

Gambar schematic yang telah dibuat di schematic editor dari Protel 99se, diubah (compile) menjadi file jedec. File ini yang nantinya akan dituliskan pada IC-IC yang akan diprogram. Maka file jedec ini diprogramkan pada IC jenis PAL, GAL, FPGA, CPLD,pAsic dan lain-lain.

Pada perancangan ini digunakan IC GAL22V10 dari Lattice Semiconductor yang merupakan salah satu jenis IC SPLD, sebagai pengontrol line fuses IC yang akan diprogram.

2.7. Programming Algoritma

Untuk memprogram IC GAL 16V8 dan 22V10, Urutannya adalah sebagai berikut :

1. Memasukkan Chip (IC GAL 16V8 atau 22V10)
2. Powe-up
3. Inisialisasi
4. Read (Gal 16V8) pada 12 Volt di EDIT (Pin 2)
5. Write, Erase, Security fuse (Gal 16V8) pada VPP volt di EDIT (Pin 2)
6. Read (Gal 22V10) pada 12 Volt di EDIT (Pin 2)
7. Write, Erase, Security fuse (Gal 22V10) pada VPP volt di EDIT (Pin 2)
8. Power-Down

2.7.1. Memasukkan Chip (IC GAL 16V8 atau 22V10)

Sebelum memasukkan IC GAL16V8 atau 22V10 ke soketnya, catu daya harus dalam keadaan mati. Kemudian masukkan IC yang akan diprogram ke dalam soketnya.

2.7.2. Power-Up

Sebelum IC GAL diprogram, kita harus menentukan tegangan dan tegangan pemrograman IC GAL tersebut. Kita harus menghubungkan VCC terlebih dahulu, yang mana beberapa pin input dan output akan bekerja sebagai outputan. Kita harus memberikan logika yang valid terhadap inputan, karena kita tidak dapat menentukan I/O mana yang benar-benar bekerja sebagai inputan, kita harus menentukan level resistor pull-up dan pull-down. Oleh karena itu kita menghubungkan resistor pull-down ke semua pin VIL dan resistor pull-up ke pin yang lainnya. Pada mode pemrograman, tegangan 12 volt akan dihubungkan ke pin EDIT (pin 2).

2.7.3. Inisialisasi

Set semua pin GAL (kecuali GND, VCC, EDIT dan SDOUT) ke logika Low.kecuali STB diset ke High.

2.7.4. Read (16V8)

Set P/V ke Low, mengeset RA0-RA5 dengan pola yang diinginkan, Atur STB selama 1-10μs ke Low dan kembalikan lagi ke High.Baca bit dari SDOUT,

atur SCLK ke High selama 1-10μs dan kembalikan lagi ke Low, dan baca bit berikutnya sampai semua bari dari line ini dibaca.

2.7.5. Read (22V10)

Set P/V ke Low, atur RA0-RA5 sesuai dengan pola yang diinginkan. Set bit (Low untuk fuses, salain itu High atau Low jika diperlukan untuk alamat) ke SDIN. Set SCLK ke High selama 1-10μs dan kembalikan ke Low dan transfer bit berikutnya sampai semua bit dari line ini ditransfer. Atur STB ke Low selama 1-10μs dan kembalikan lagi ke High. Baca bit dari SDOUT. Set SCLK ke High dan kembalikan lagi ke Low dan baca bit selanjutnya sampai semua bit dari line ini di baca.

2.7.6. Write, Erase, Security Fuse (16V8)

Set P/V ke High, atur RA0-RA5 sesuai dengan pola yang diinginkan. Aplikasikan bit pertama ke SDIN, atur SCLK ke High selama 1-10μs dan kembalikan lagi ke Low, dan transfer bit selanjutnya sampai semua bit dari line ini telah di transfer. Set STB ke Low dan tunggu untuk programming pulse time sebelum kita mengatur kembali ke High lagi. Set kembali P/V ke Low di akhir operasi.

2.7.7. Write, Erase, Security Fuse (22V10)

Set P/V ke High, atur RA0-RA5 sesuai dengan pola yang diinginkan. Set bit ke SDIN, atur SCLK ke High selama beberapa waktu dan kembalikan ke

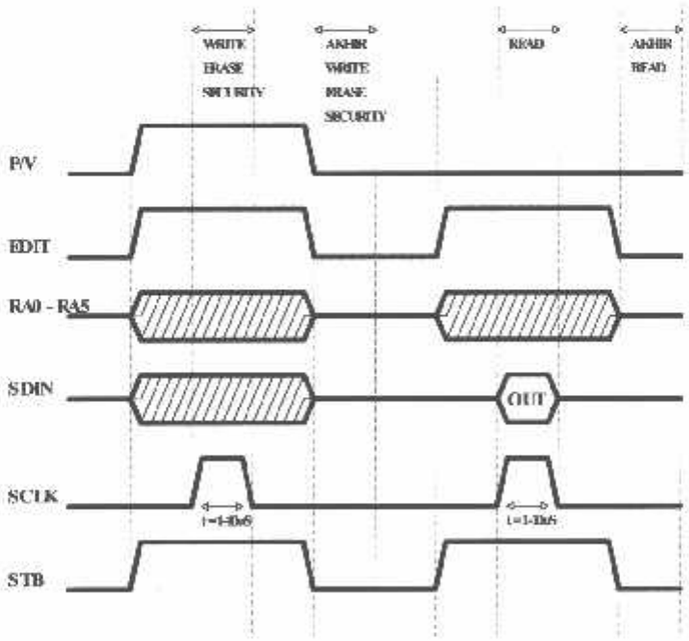
Low, dan transfer bit selanjutnya sampai semua bit dari line ini telah ditransfer. Set SDIN ke Low, dan set STB ke Low dan tunggu untuk programming pulse time sebelum kita mengatur kembali ke High lagi. Set kembali P/V ke Low di akhir operasi.

2.7.8. Power Down

Sebelum kita mencabut IC yang telah selesai diprogam, catu daya harus dalam keadaan mati.

2.8. Timing Diagram

Timing diagram dari proses penulisan, penghapusan, security dan pembacaaan IC GAL adalah sebagai berikut :



Gambar 2.16 Timing Diagram Pemrograman

BAB III
PERENCANAAN DAN PEMBUATAN ALAT

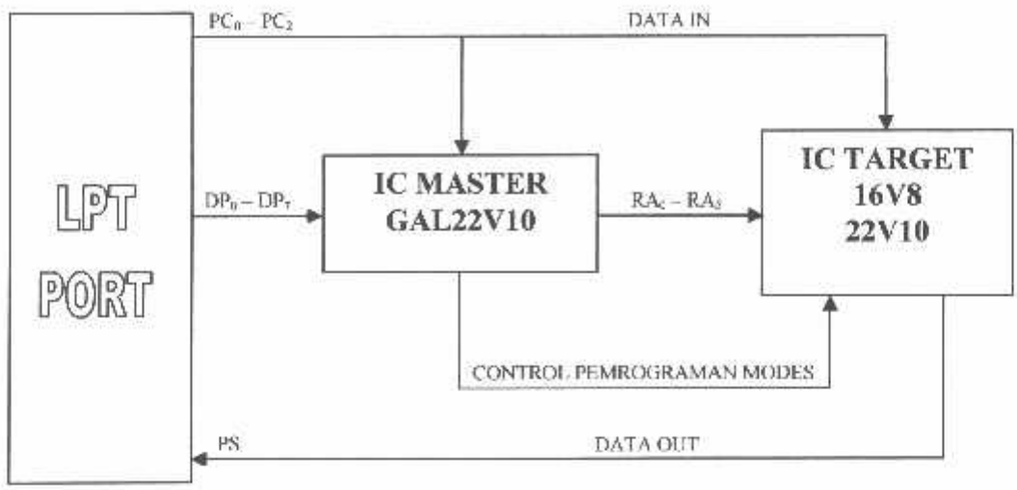
3.1. Pendahuluan

Pada bab ini akan dibahas mengenai perencanaan dan pembuatan alat yang dibagi dua bagian yaitu : bagian perencanaan dan pembuatan perangkat keras dan bagian perencanaan dan pembuatan perangkat lunak.

Proses pembuatan alat downloader untuk IC GAL16V8 dan IC GAL22V10 ini meliputi perancangan hadware dan pembuatan PCB. Sedangkan IC GAL 16V8 dan 22V10 merupakan target atau sasaran dari pengisian maupun penghapusan program yang nantinya akan dibuat. Program dibuat dalam bahasa VHDL pada PC (Personal Computer) kemudian ditransfer ke hardware. Adapun jalur yang digunakan adalah melalui pararel port (LPT1).

3.2. Perencanaan Dan Pembuatan Perangkat Keras

Pada perencanaan dan pembuatan downloader untuk IC GAL 16V8 dan IC GAL 22V10 ini, targetnya adalah IC GAL 16V8 dan IC GAL 22V10 itu sendiri. Adapun blok diagramnya adalah sebagi berikut :



Gambar 3.1 Blok Diagram Alat

Gambar diatas menunjukkan bahwa IC master GAL22V10 sebagai control pemerograman modes dan pengatur line fuses IC target. Sedangkan IC GAL 16V8 dan IC GAL 22V10 adalah merupakan target dari proses pengisian dan penghapusan program yang akan dibuat, sedangkan piranti yang lain sebagai penunjang. Adapun fungsi dari masing – masing blok dapat dijelaskan sebagai berikut :

1. LPT PORT

   LPT port merupakan interface computer yang digunakan sebagai antarmuka dengan piranti lain yang memiliki tiga saluran antara lain DP, PC dan PS. Dengan DP memiliki 8 port pinout sebagai saluran data (I/O), dan PC memiliki 4 pinout sebagai saluran (I/O), serta PS memiliki 8 pinout sebagai saluran (I) yang akan digunakan sebagai saluran data dan control.

2. IC 22V10

   Di dalam IC GAL terdapat Line Fuses yang terdiri dari baris dan kolom, IC master 22V10 disini berfungsi sebagai pengatur Line

Fuses IC GAL 16V8 dan 22V10 yang sebelumnya di program terlebih dahulu.

3. IC GAL 16V8 dan 22V10

   IC Slave GAL 16V8 dan 22V10 merupakan IC target yang akan diisi program.

3.2.1. Prinsip Kerja Rangkaian

Prinsip kerja dari rangkaian yaitu adalah mengatur transfer atau downloading data dari desktop PC ke IC GAL16V8 dan 22V10 slave melalui port paralel, dengan memakai power supply 12 volt dari adaptor

3.2.2. Rangkaian IC GAL 22V10 Master

Rangkaian IC GAL 22V10 master dirancang agar IC master dapat berkomunikasi dengan komputer melalui port paalel, sehingga IC GAL 22V10 dapat melakukan pengaturan kontrol serial downloading, jadi semua descriptor yang dibutuhkan dan diminta oleh host (CPU) diprogram dulu didalam IC master, adapun dalam rangkaian Untuk deskripsi pin-pin yang digunakan adalah sebagai berikut:

1. Pin 2 sampai dengan Pin7

   Digunakan untuk mengatur Row Adrress IC slave GAL 16V8 dan 22V10.

2. Pin 8

   Digunakan untuk mengatur progaming voltage IC slave GAL 16V8 dan 22V10.

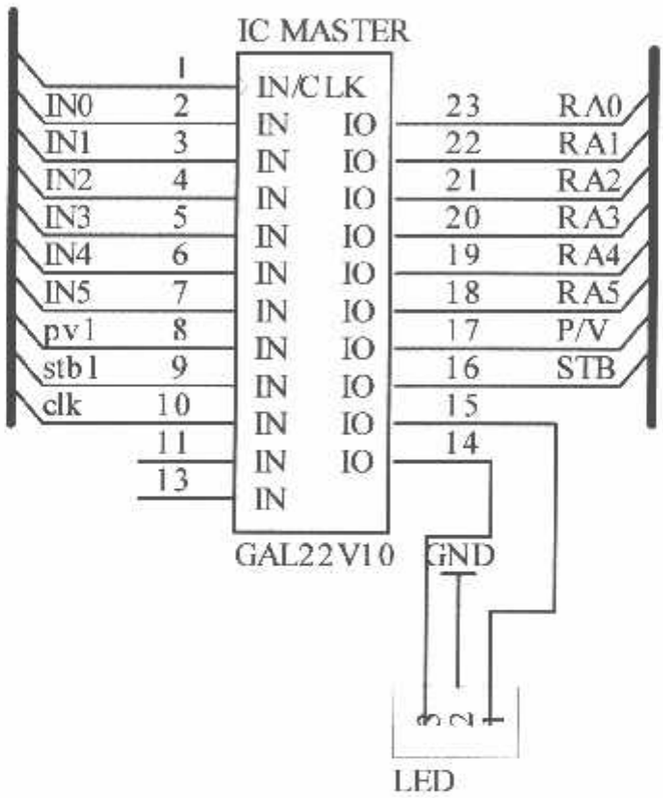
3. Pin 9

   Digunakan untuk mengatur Strobe IC slave GAL 16V8 dan 22V10.

4. Pin 10

Digunakan untuk mengatur clock

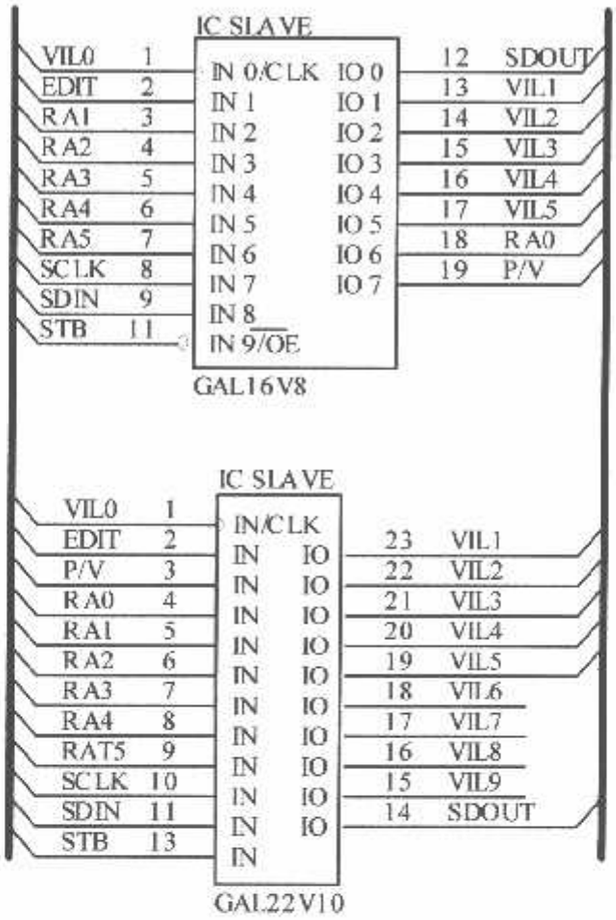
Untuk rangkaian IC masternya adalah sebagai berikut :



Gambar 3.2 Rangkaian IC GAL 22V10 Master

3.2.3. Rangkaian dan Konfigurasi pin IC GAL 16v8 Dan 22V10 Slave

Agar dapat melakukan serial downloading, GAL 16V8 dan 22V10 slave tidak membutuhkan komponen tambahan, selain dari IC master sebagai pengatur serial downloading, IC GAL 16V8 dan 22V10 slave mempunyai konfigurasi dan rangkaian sebagai berikut :



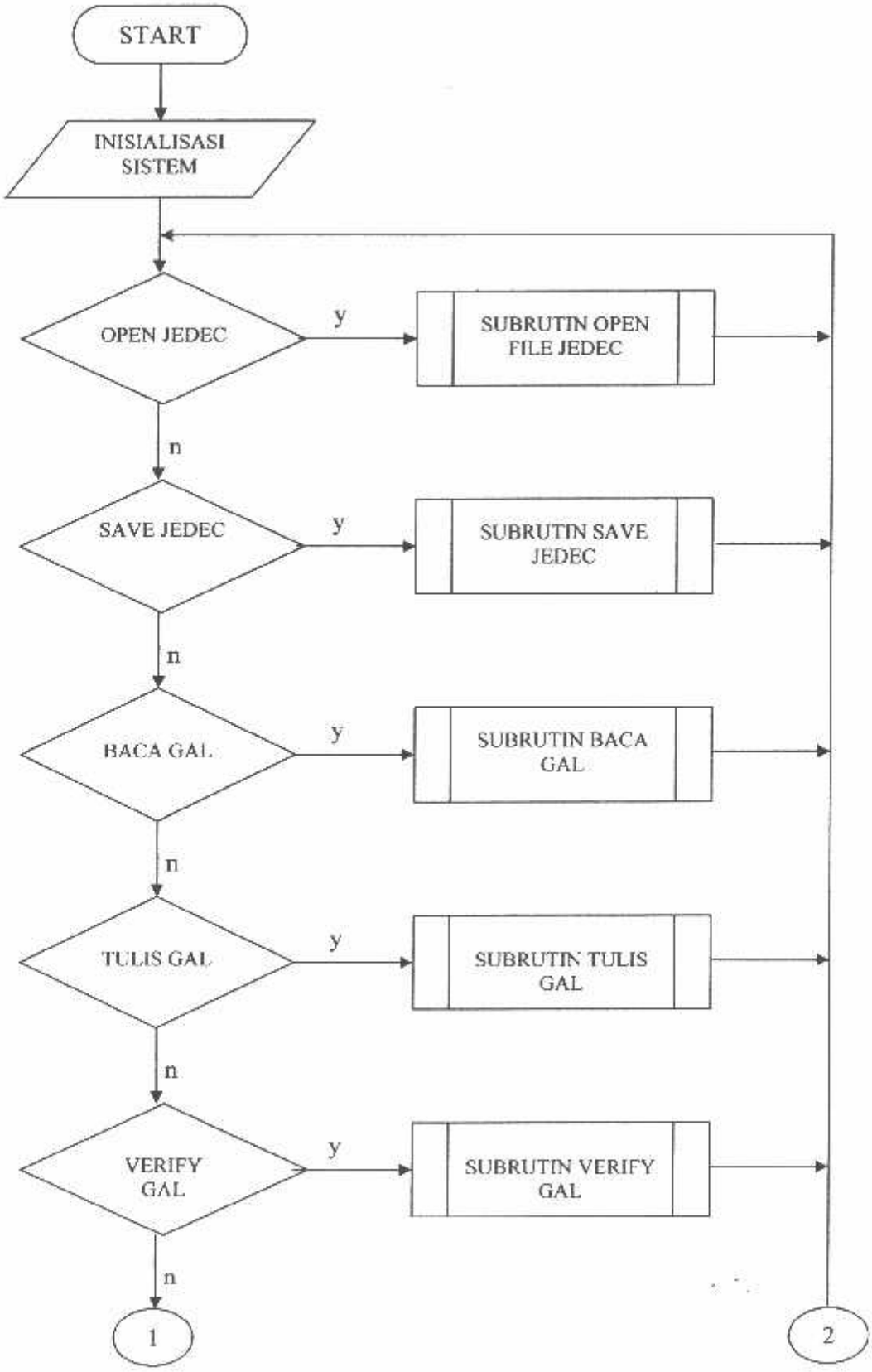
Gambar 3.3 Rangkaian IC GAL Slave

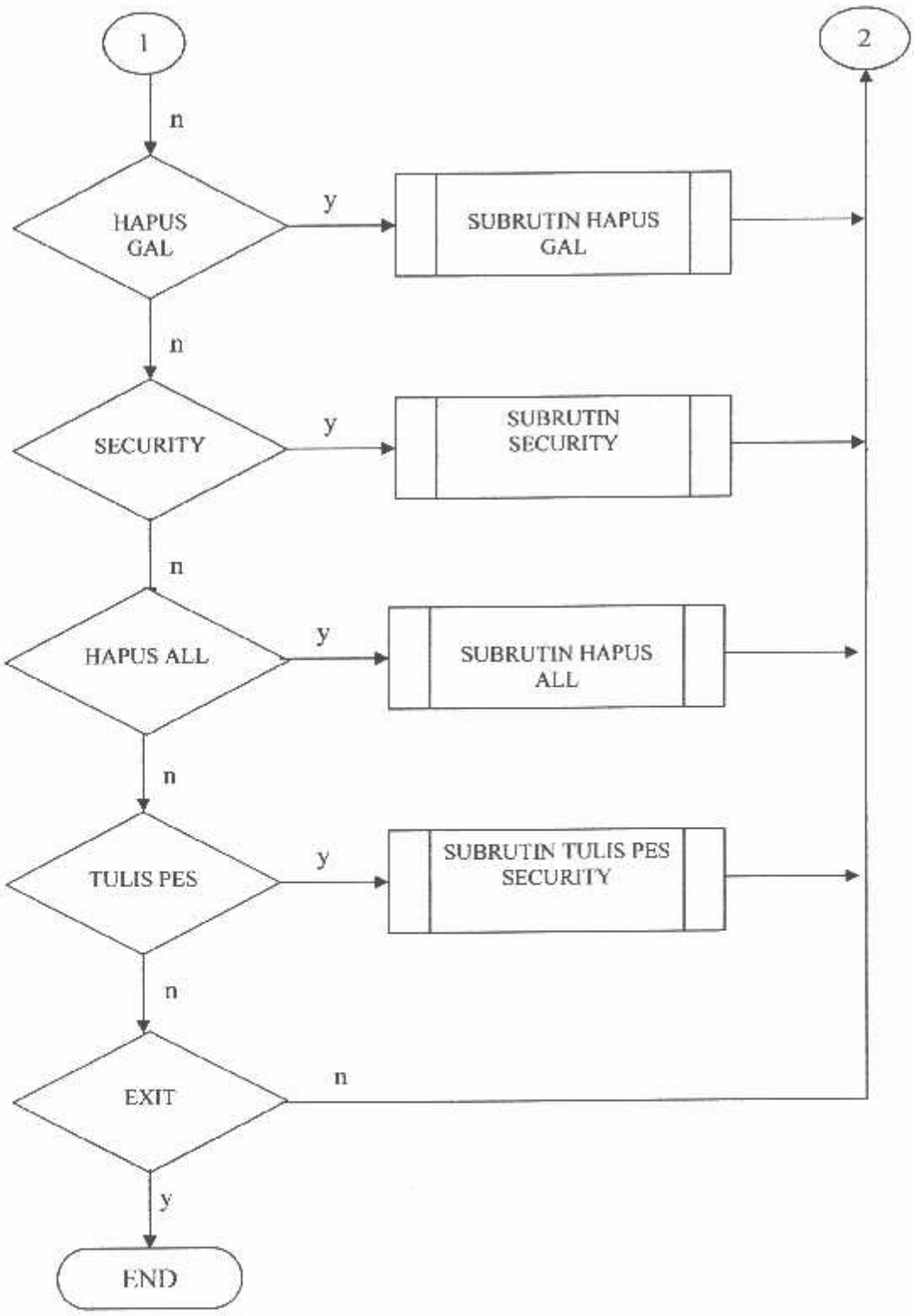
Dalam rangkaian IC slave ini tidak ada komponen tambahan sama sekali, jadi pin IC langsung dihubungkan dengan pin IC master.

3.3. Flowchart

Adapun diagram alir kerja dari alat yang telah dibuat adalah sebagai berikut:

Flowchart Program Pada PC




1
2
n
HAPUS GAL
y
SUBRUTIN HAPUS GAL
n
SECURITY
y
SUBRUTIN SECURITY
n
HAPUS ALL
y
SUBRUTIN HAPUS ALL
n
TULIS PES
y
SUBRUTIN TULIS PES SECURITY
n
EXIT
n
y
END

3.4. Implementasi VHDL Pada IC

Gambar schematic yang telah dibuat di schematic editor dari Protel 99se, diubah (compile) menjadi file jedec. File ini yang nantinya akan dituliskan pada IC-IC yang akan diprogram. Maka file jedec ini diprogramkan pada IC jenis PAL, GAL, FPGA, CPLD,pAsic dan lain-lain.

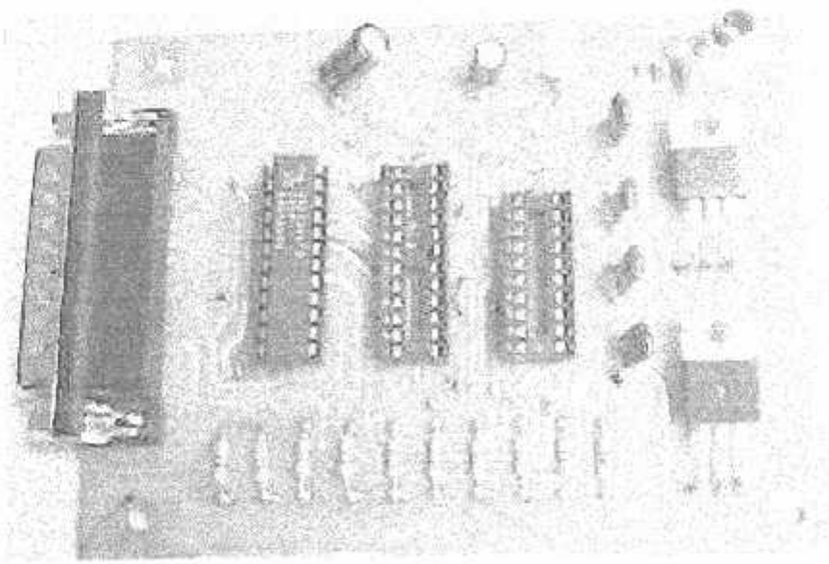
Pada perancangan ini digunakan IC GAL22V10 dari Lattice Semiconductor yang merupakan salah satu jenis IC SPLD, sebagai pengontrol line fuses IC yang akan diprogram.

BAB IV
PENGUJIAN ALAT

4.1. Pendahuluan

Pada bab ini akan dibahas mengenai pengujian tentang alat yang telah dibuat. Hal ini dilakukan untuk mengetahui kekurangan unjuk kerja dari sistem yang telah dibuat. Pengujian dilakukan dengan cara melakukan tahapan – tahapan proses pengisian program pada IC GAL 16V8 dan 22V10 mulai dari pengambilan data dari komputer, sampai dengan membaca kembali data pengisian program apakah data yang dikirim sesuai dengan data yang nantinya telah disimpan dalam IC GAL. Selanjutnya pengujian juga dilakukan dengan membuat salah satu program yang hasilnya bisa ditampilkan pada rangkaian LED.

Adapun gambar dari alat yang dibuat adalah sebagai berikut :



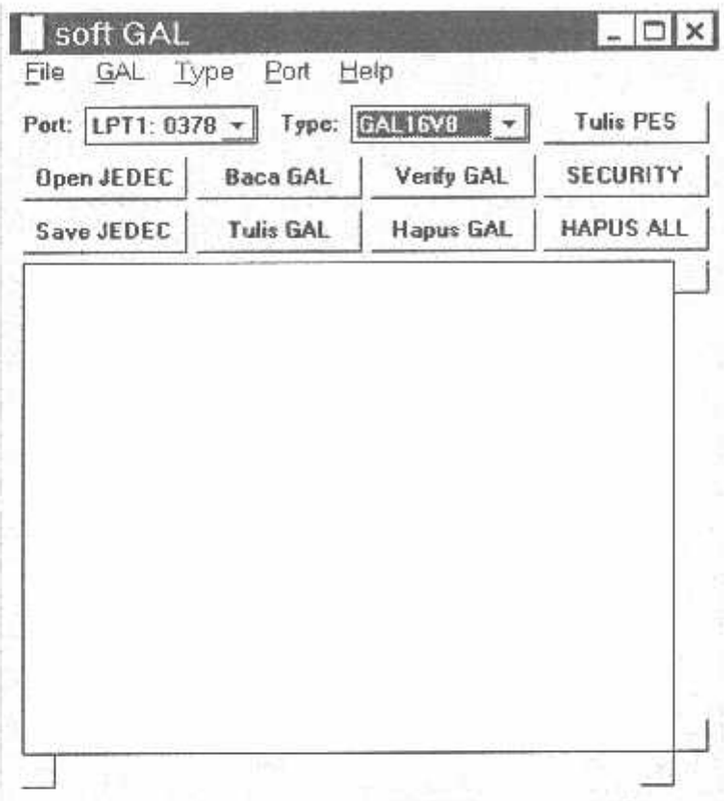
Gambar 4.1 Alat Downloader Ic Gal 16v8 Dan 22v10

4.2. Perangkat Lunak

Pengujian pertama kali dilakukan terhadap perangkat lunak yang telah dibuat yang berisikan semua kebutuhan yang diperlukan dalam pengembangan IC GAL khususnya GAL 16V8 dan 22V10. Program telah dibuat dalam bahasa pemrograman Delphi dan telah menyediakan fasilitas – fasilitas yang akan dibutuhkan dalam proses pemrograman. Adapun fasilitas program tersebut terdiri atas:

- Label Open JEDEC digunakan untuk membuka file berekstensi *.jed yang akan diisikan ke IC.
- Label Tulis GAL digunakan untuk mengisikan program data yang telah kita ambil dari komputer.
- Label Baca GAL dipergunakan untuk membaca data dari IC.
- Label Verify GAL digunakan untuk mengecek kondisi IC GAL rusak apa tidak, juga digunakan untuk mengetahui jika PES IC GAL ilegal atau tidak dikenali.
- Label Save JEDEC digunakan untuk menyimpan data yang telah kit baca dari IC ke dalam komputer dengan eksitensi *.jed.
- Label Hapus GAL digunakan untuk menghapus isi dari IC GAL.
- Label Security digunakan untuk melakukan penguncian data agar tidak dapat dibaca kembali.
- Label Hapus All digunakan untuk menghapus dari IC GAL dan juga menghapus PES IC GAL.
- Tulis PES digunakan untuk mengganti PES dari IC GAL.

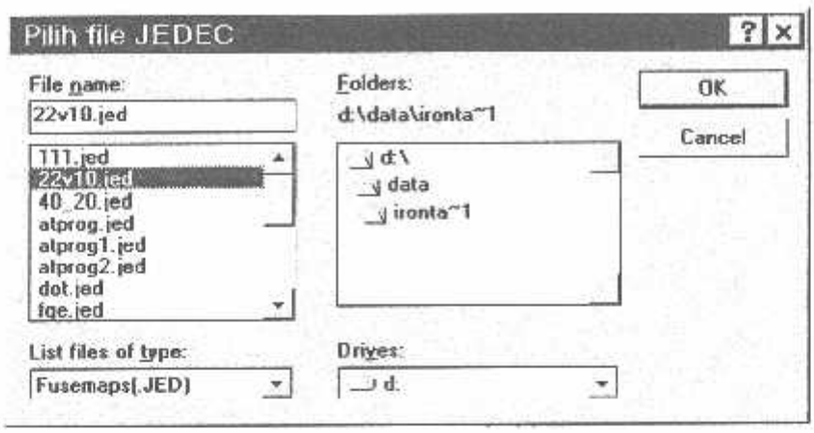
Adapun tampilan menu utama dalam proses ini adalah sebagai berikut:



Gambar 4.2 Menu Utama Downloader IC GAL 16V8 Dan 22V10

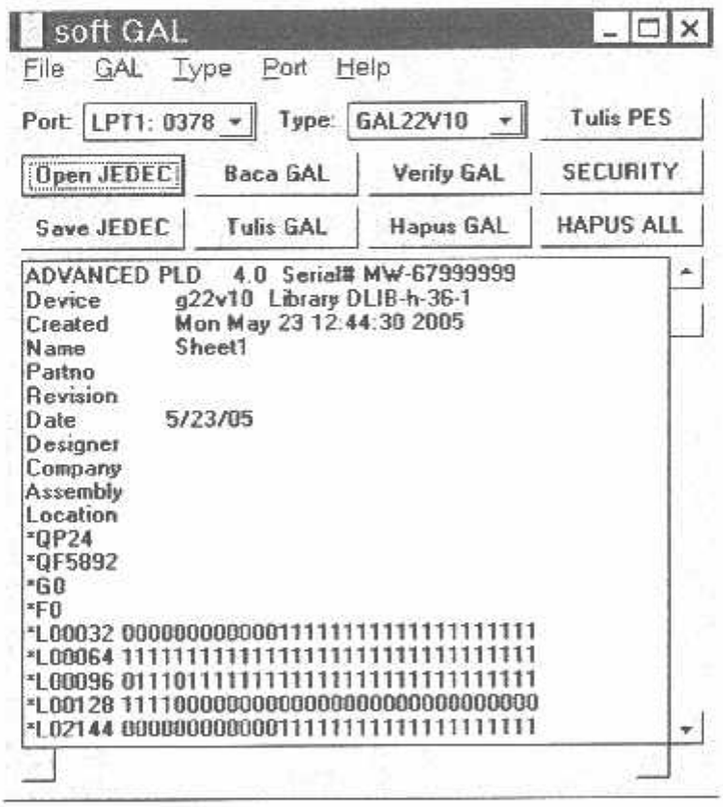
4.2.1. Menjalankan Menu Open JEDEC

Menu ini berfungsi untuk mengambil nama file jedec yang nantinya akan diisikan kedalam chip GAL 16V8 dan 22V10. Menu ini diaktifkan dengan menekan label Open JEDEC yang terdapat pada menu utama program alat.



Gambar 4.3 Tampilan Pada Saat Menu Open Jedec Diaktifkan

Pada saat tampilan Open JEDEC diaktifkan maka data akan terbaca pada tampilan menu utama data yang telah kita ambil.



Gambar 4.4 Tampilan Pengambilan Data Dari Komputer

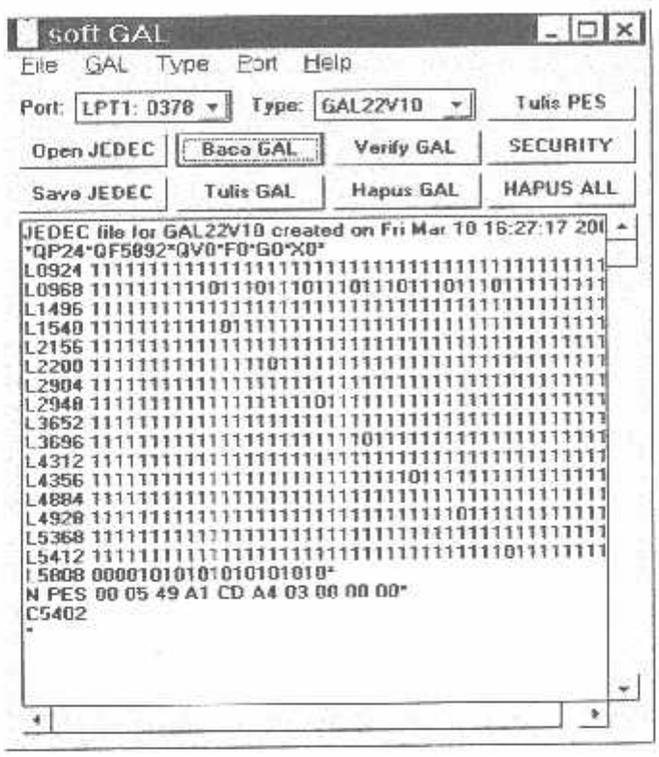
4.2.2. Menjalankan Menu Tulis GAL

Menu ini berfungsi untuk mengisikan data yang telah diambil dari komputer ke IC GAL 16V8 atau 22V10 yang terbaca pada menu utama. Adapun tampilan pada saat akan menulis ke IC GAL adalah sebagai berikut:

Gambar 4.5 Tampilan Pada Saat Akan Menulis Ke IC GAL

4.2.3. Menjalankan Menu Baca GAL

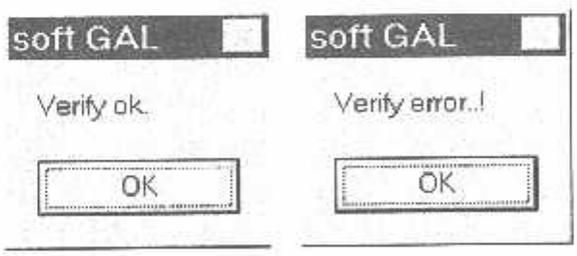
Menu ini berfungsi untuk membaca isi dari chip IC GAL 16V8 dan 22V10 yang akan kita isikan program ataupun telah kita isikan program. Untuk menjalankan menu ini kita cukup menekan label Baca GAL untuk membaca file yang sudah diisikan ke dalam chip.



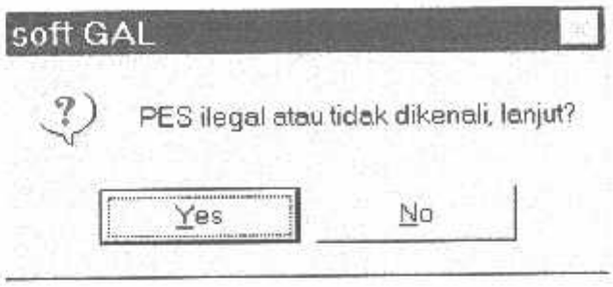
Gambar 4.6 Tampilan Setelah Menu Baca GAL Diaktifkan

4.2.4. Menjalan kan Menu Verify GAL

Menu ini berfungsi untuk mengetahui kondisi IC GAL dalam keadaan baik apa tidak. Menu verify GAL juga berfungsi untuk mengetahui PES IC GAL ilegal atau tidak dikenali yang ditunjukkan dengan keluanya message box.



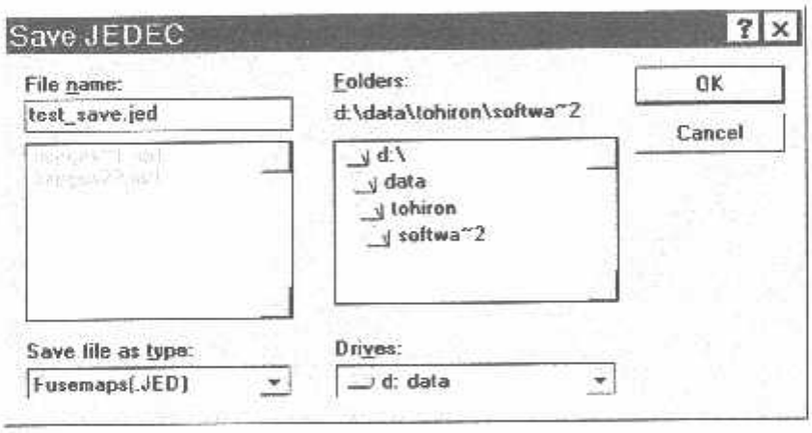
Gambar 4.7 Tampilan Saat Menu Verify GAL Diaktifkan



Gambar 4.8 Tampilan Pes IC GAL Ilegal Atau Tidak Dikenali

4.2.5. Menjalankan Menu Save JEDEC

Menu ini berfungsi untuk menyimpan data yang telah kita baca dari IC ke komputer. Untuk mengaktifkannya dengan menekan label Save JEDEC. Adapun tampilan pada saat menu ini diaktifkan adalah sebagai berikut:



Gambar 4.9 Tampilan Pada Saat Menu Save GAL Diaktifkan

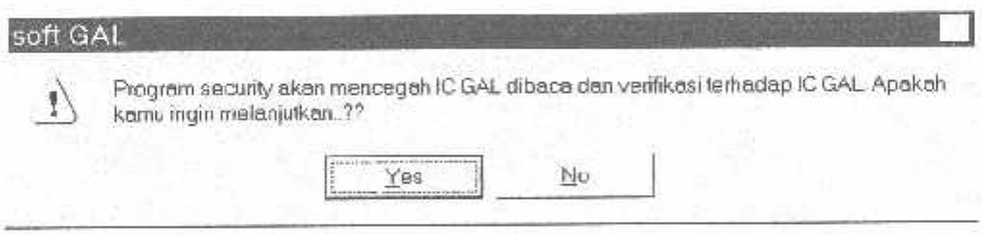
4.2.6. Menjalankan Menu Hapus GAL

Menu ini berfungsi untuk menghapus isi dari chip GAL. Penghapusan ini dilakukan agar chip bisa diisi program yang baru. Untuk mengaktifkannya kita menekan label Hapus GAL. Adapun tampilan pada saat menu ini diaktifkan adalah sebagai berikut:

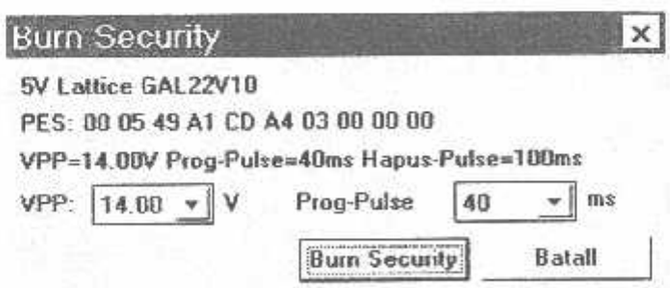
Gambar 4.10 Tampilan Pada Saat Akan Menghapus IC GAL

4.2.7. Menjalankan Menu Security

Security berfungsi untuk mengunci program yang telah kita buat agar tidak bisa dibaca oleh orang lain. Software telah mengatur cara kerjanya dan kita cukup meng-klik label Secuity yang tersedia pada menu utama. Selanjutnya akan muncul pesan akibat dari proses pengaktifan menu security. Adapun tampilan pada saat menu ini diaktifkan adalah sebagai berikut:



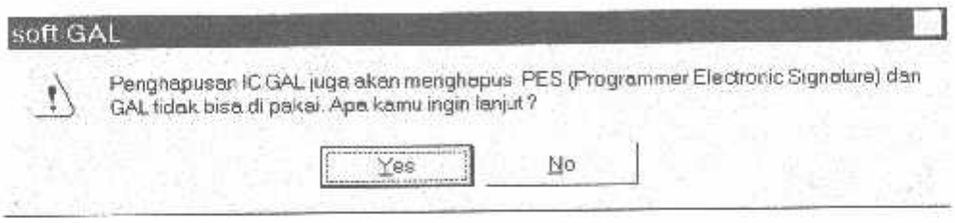
Gambar 4.11 Pesan Sebelum Proses Security Diaktifkan



Gambar 4.12 Tampilan Pada Saat Akan Mengunci IC GAL

4.2.8. Menjalankan Menu Hapus ALL

Menu ini berfungsi untuk menghapus isi dari chip GAL dan juga akan menghapus PES IC GAL. Jika penghapusan ini dilakukan, maka IC GAL tidak akan bisa diprogram lagi sebelum PESnya diisi/dimasukkan.

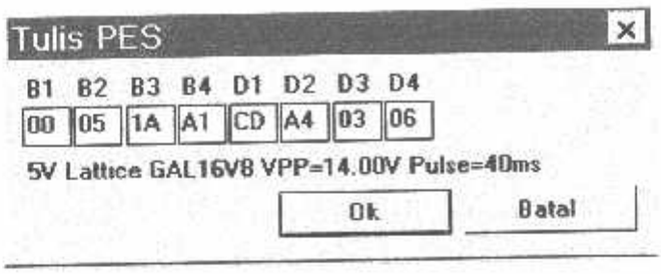


Gambar 4.13 Pesan Sebelum Proses Hapus All Diaktifkan

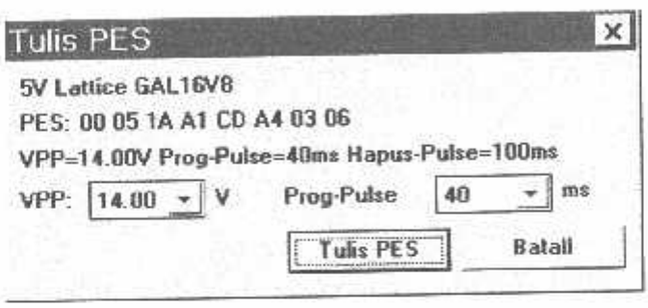
Gambar 4.14 Tampilan Pada Saat Akan Menghapus All GAL

4.2.9. Menjalankan Menu Tulis PES

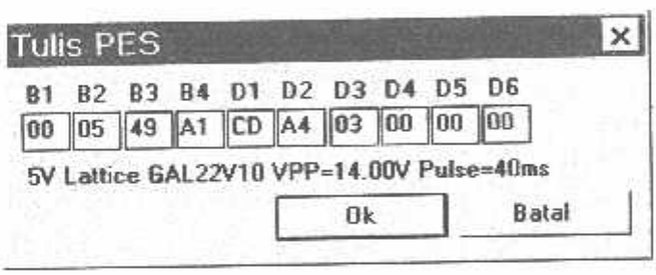
Menu ini berfungsi untuk mengetahui dan mengubah Programable Electronic Signature (PES) IC GAL. Adapun tampilan pada saat Menu Tulis PES diaktifkan adalah sebagai berikut :



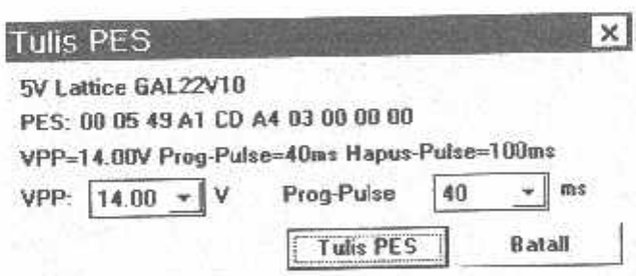
Gambar 4.15 Proses Memasukkan PES IC GAL 16V8



Gambar 4.16 Proses Penulisan PES IC GAL 16V8



Gambar 4.17 Proses Memasukkan PES IC GAL 22V10



Gambar 4.18 Proses Penulisan PES IC GAL 22V10

4.3. Perangkat Keras Alat

Untuk menjalankan peralatan ini diperlukan catu daya yang sesuai dengan peralatan ini. Catu daya yang dipakai adalah catu daya yang mempunyai keluaran 5V sampai 25V.

Langkah – langkah yang harus dilakukan adalah sebagai berikut:

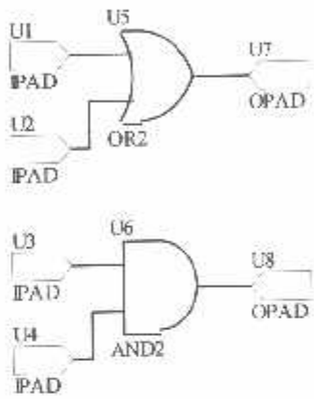
- Hubungkan perangkat keras alat dengan komputer melalui port DB25.
- Hubungkan peralatan dengan Power Supply.
- Aktifkan label Verify GAL untuk mengetahui IC sudah siap diisi apa belum, bisa juga untuk mengetahui apakah IC GAL rusak apa tidak.
- Aktifkan label Baca GAL apakah IC tersebut ada isinya atau tidak.
- Jika IC ada isinya maka aktifkan label Hapus GAL untuk menghapus isi dari IC GAL.
- Aktifkan label Open GAL untuk mengambil file yang berekstensi *.JED.
- Setelah data yang diambil terbaca pada menu utama maka jalankan proses pengisian dengan mengaktifkan label Tulis GAL.
- Kemudian agar program yang diisikan tidak bisa dibaca oleh orang lain maka aktifkan label Security.
- Chip yang telah diisi program telah bisa dihubungkan dengan piranti yang hendak dijalankan.

4.4. Langkah-Langkah membuat file *.JED Di Protel 99SE

Pada perancangan downloader ini digunakan software protel 99SE dengan service pack 5 keluaran dari Protel Internasional Limited untuk membuat file

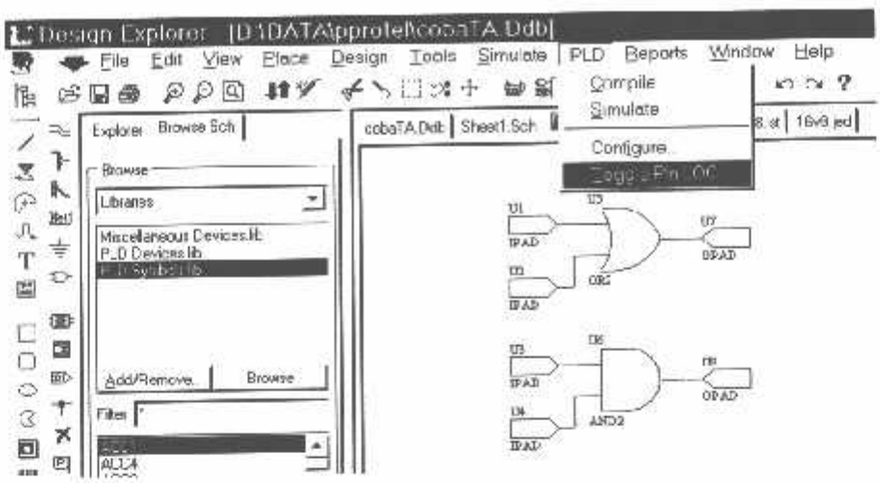
*.jed. File dengan eksitensi jed inilah yang akan diisikan kedalam IC yang akan diprogram.

Langkah-langkah dalam membuat file *.jed di protel dimulai dari membuat gerbang-gerbang yang akan diisikan ke dalam IC dan juga menentukan ipad maupun opadnya,



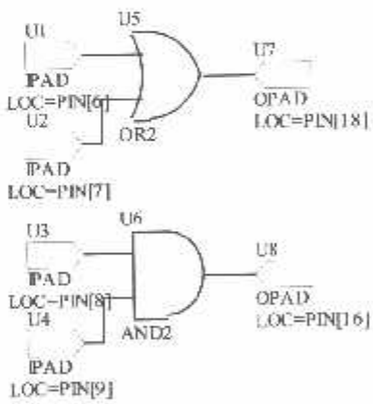
Gambar 4.19 Rangkaian Sebuah Gerbang sederhana

setelah itu kita harus menentukan pin LOCnya. Pin LOC adalah pin inputan maupun pin outputan. Dengan menekan tombol PLD yang ada di toolbar Protel 99se.



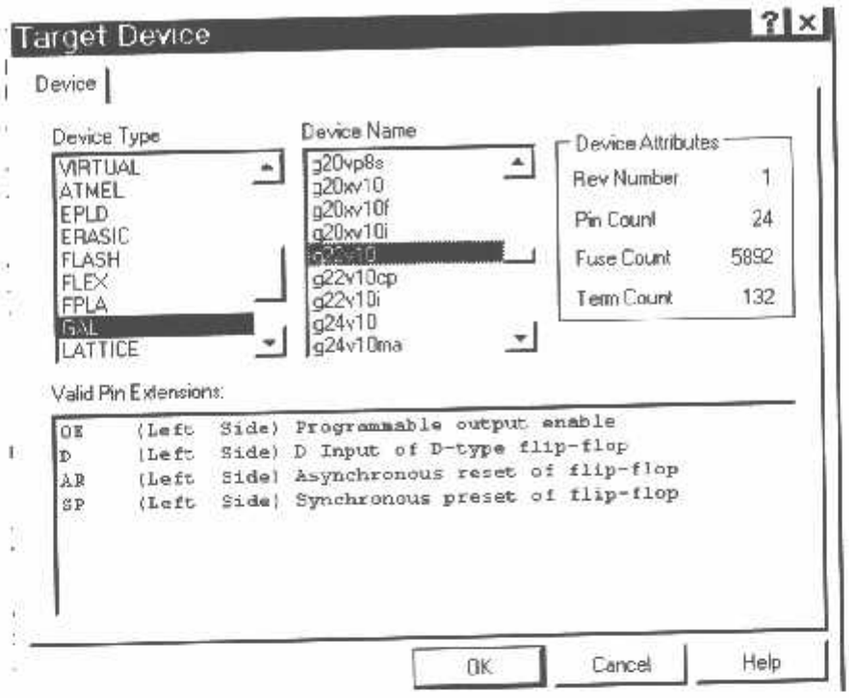
Gambar 4.20 Menentukan Pin LOC Rangkaian

Setelah itu kita harus memasukkan pin LOCnya. Pin LOC ipad adalah pin input dari IC yang akan diisi dan Pin LOC opad adalah pin output IC.



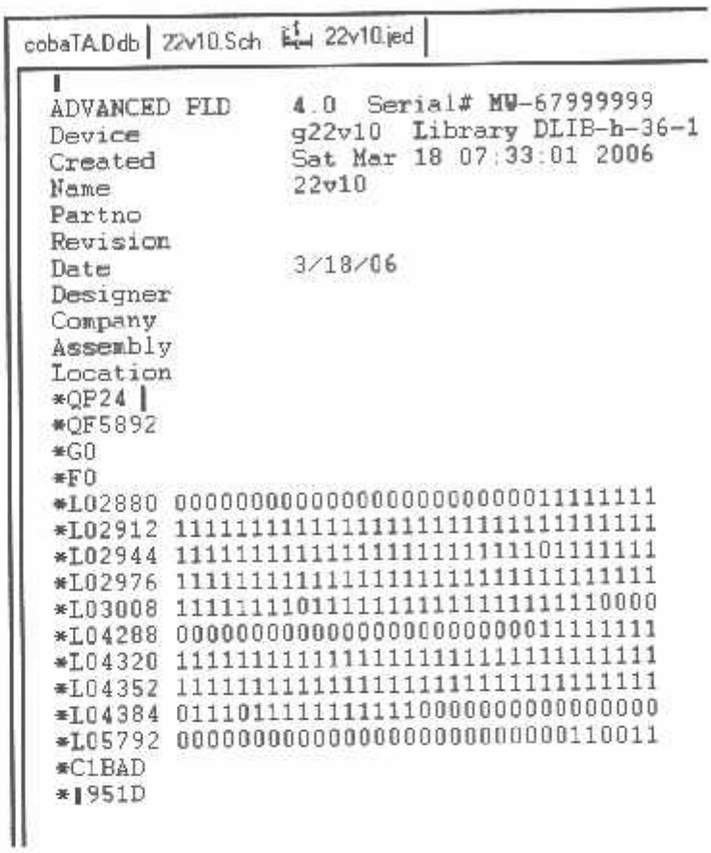
Gambar 4.21 Pengesetan Pin LOC

Setelah pin LOC sudah ditentukan maka kita harus mengkonfigurasinya ke IC GAL16V8 atau 22V10,



Gambar 4.22 Mengkonfigurasi Rangkaian Kc Dalam Ic Gal 22v10

setelah itu baru di compile, dan hasil compilan yang bercksitensi *.jed ini yang akan diisikan kedalam IC.



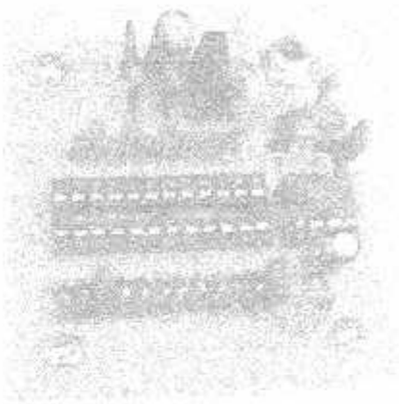
Gambar 4.23 Hasil Compilan Dengan Eksitensi *.Jed

4.5. Pengujian Dengan Mempergunakan LED

Untuk membuktikan bahwa alat dan program telah berfungsi dengan baik, kita harus membuktikannya dengan memberikan salah satu contoh pengisian program yang diinterfacekan dengan peralatan luar. Dalam hal ini kita mempergunakan LED sebagai media untuk menampilkan contoh pengisian programnya.

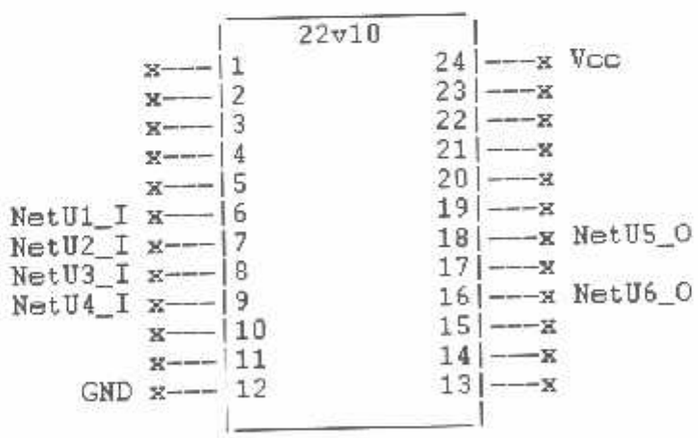
Pada rangkaian seperti gambar 4.19 diatas adalah menggunakan logika gerbang OR dan AND. Pada Logika OR, jika inputannya salah satunya high atau keduanya high maka outputannya akan high, dan jika kedua inputannya low maka

outputannya akan sama dengan low. Sedangkan pada logika gerbang AND, jika inputannya salah satunya Low atau kedua inputannya Low maka outputannya akan Low. Dan jika kedua inputannya berlogika high maka outputannya akan high.



Gambar 4.24 Simulator Dengan Menggunakan LED

Jika pada gambar 4.21 diisikan kedalam IC GAL 22V10 maka gambar rangkaiannya adalah sebagai berikut :



Gambar 4.25 Bentuk Rangkaian Gal22v10 Setelah Diisi Dengan File *.Jed

Gambar diatas jika disimulasikan dengan LED maka LED di pin no.16 dan no.18 akan menyala. Jika pin no.6 dan no.7 digroundkan (Low) maka LED di pin no.16 akan mati. Begitu juga jika pin no.8 atau pin no.9, atau juga jika kedua pin tersebut digroundkan maka LED di pin no.18 akan mati.

BAB V
PENUTUP

5.1. Kesimpulan

Setelah melakukan perencanaan dan pembuatan serta pengujian alat downloader IC GAL16V8 dan 22V10 ini, maka dapat ditarik kesimpulan sebagai berikut :

1. Alat Downloader IC GAL16V8 dan 22V10 ini berfungsi dengan baik karena program yang kita buat sesuai dengan yang kita inginkan.
2. Alat ini juga meminimumkan komponen pendukung pada rangkain mode format pemrograman dengan menggunakan IC GAL22V10 yang dapat diprogram sesuai dengan yang kita inginkan.
3. Dengan menggunakan alat ini kita dapat dengan mudah melakukan eksperimen dengan bermacam - macam program dan dapat langsung menginterfacekannya dengan peralatan, baik untuk IC GAL16V8 maupun IC GAL 22V10.
4. Alat Downloader ini dapat menghapus dan mengisi IC GAL 16V8 dan 22V10, alat ini bekerja pada power supply DC 12V. Dengan spesifikasi komputer yang digunakan saat pengujian alat adalah Pentium II 300MHz, RAM 64 dan dengan menggunakan operating sistem Windows 98SE.
5. Untuk membuat file *.jed bisa menggunakan software protel 99se dari Protel Internasional Limited.

Daftar Pustaka

Dwi Sutadi, **I/O Bus & Motherboard,** Andi Yogyakarta 2003.

Moh. Ibnu Malik & Anistardi, **Bereksperimen dengan Mikrokontroller 8031**, PT.Elex Media Komputindo, Gramedia Jakarta.

Hafindo Electronic & Education, **Pelatihan Pemprograman Delphi dan Aplikasi pada Teknik Interface**.

Retna Prasetia & Catur Edi Widodo, **Interfacing Port Paralel dan Port Serial Komputer Dengan Viisual Basic 6.0**, Penerbit Andi Yogyakarta 2004

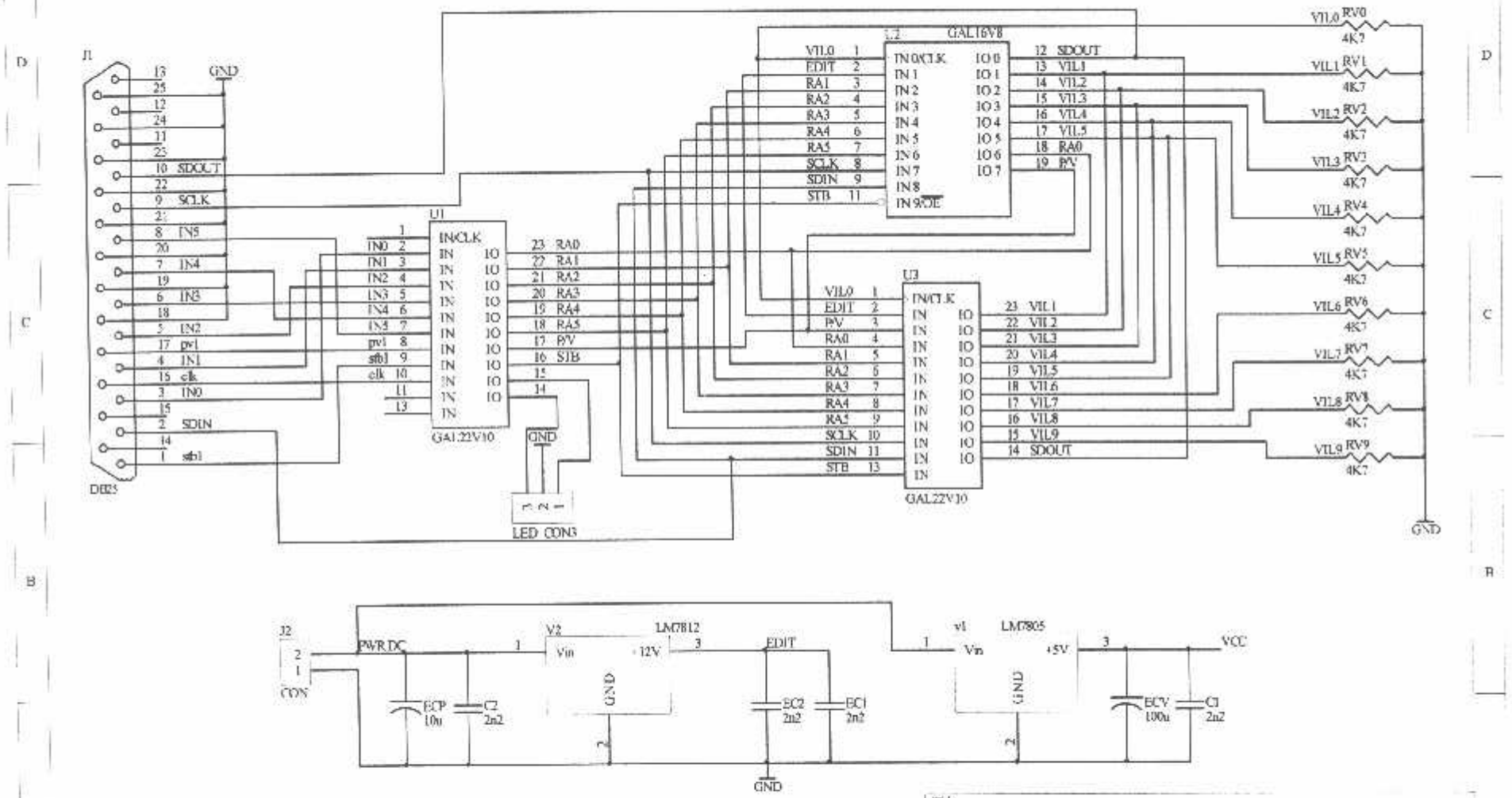
http://www.geocities.com/ResearchTriangle/ Forum/8070/galblast.htm

http://www.geocities.com/mwinterhoff/

http://www.latticesemi.com

http://www.beyondlogic.org/spp

LAMPIRAN



Title	DOWNLOADER IC GAL 16V8 DAN 22V10 BERBASISKAN VHDL	
Size: B	Number: 0017175	Revision:
Date:	13-Mar-2006	Sheet of
File:	D:\DATA\tohiron\skripsi2\Skripsi_iron2.Ddb	Drawn By: MUHAMMAD TOHIRON

TABEL KEBENARAN IC GAL MASTER

PV	STB	CNT	L1	L2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

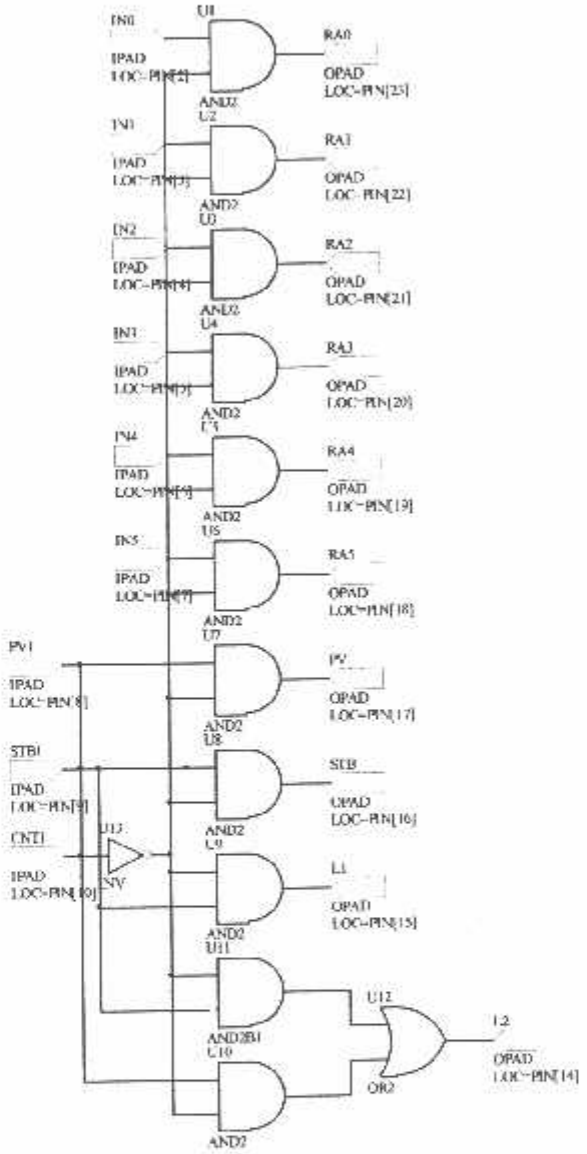
PETA KARNOUGH

	00	01	11	10
0	0	0	1	0
1	0	0	1	0

L1 = STB . CNT

	00	01	11	10
0	0	1	0	0
1	0	1	1	0

L1 = $\overline{STB}$.CNT + PV.CNT



Title: ISE GAL MASTER			
Size: D	Number: 001175		Revision:
Date: 13-Mar-2006	Sheet of	Drawn By:	MUHAMMAD TOHIRON
File: D:\DATA\tohiron\skripsi2\Skripsi_iron2.Dsn			

```
/ soft GAL

include <windows.h>
include <stdio.h>
include <ver.h>
include <stdlib.h>
include <stdarg.h>
include <mmsystem.h>
include <dos.h>
include <time.h>
include <ctype.h>
include <string.h>
include <commdlg.h>
include <shellapi.h>

define LATTICE 0xA1
define NATIONAL 0x8F
define SGSTHOMSON 0x20

tatic char *caption[]={"Baca GAL","Verify GAL","Baca PES","Test SCLK","Tulis
AL","Hapus GAL","Hapus ALL","Burn Security","Tulis PES","VPP Test"};
define BACAGAL 0
define VERIFYGAL 1
define BACAPES 2
define SCLKTEST 3
define TULISGAL 4
define HAPUSGAL 5
define HAPUSALL 6
define BURNSECURITY 7
define TULISPES 8
define VPPTEST 9

define BACAVPP 48 // 12V

ypedef enum { UNKNOWN,GAL16V8,GAL22V10} GALTYPE;

tatic int cfg16V8[]=


128,2129,2130,2131,2132,2133,2134,2135,2136,2137,2138,2139,2140,2141,2142,2143,2144,
145,2146,2147,2148,2149,2150,2151,2152,2153,2154,2155,2156,2157,2158,2159,
   2048,2049,2050,2051,
   2193,
   2120,2121,2122,2123,2124,2125,2126,2127,
   2192,
   2052,2053,2054,2055,

160,2161,2162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2175,2176,
177,2178,2179,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191
;
tatic int cfg16V8AB[]=

   2048,2049,2050,2051,
   2193,
   2120,2121,2122,2123,

128,2129,2130,2131,2132,2133,2134,2135,2136,2137,2138,2139,2140,2141,2142,2143,2144,
145,2146,2147,2148,2149,2150,2151,2152,2153,2154,2155,2156,2157,2158,2159,2160,2161,
162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2175,2176,2177,2178,
179,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191,
   2124,2125,2126,2127,
   2192,
   2052,2053,2054,2055
;
```

```
static int cfg22V10[]=

    5809,5808,
    5811,5810,

    5813,5812,
    5815,5814,
    5817,5816,
    5819,5818,
    5821,5820,
    5823,5822,
    5825,5824,
    5827,5826
;
static struct

    GALTYPE type;
    unsigned char id0,id1;
    char *name;
    int fuses;
    int pins;
    int rows;
    int bits;
    int uesrow;
    int uesfuse;
    int uesbytes;
    int eraserow;
    int eraseallrow;
    int pesrow;
    int pesbytes;
    int cfgrow;
    int *cfg;
    int cfgbits;

alinfo[]=

    {UNKNOWN,  0x00,0x00,"unknown",     0, 0, 0,  0, 0,   0,0, 0, 0, 0, 8, 0,NULL,0},
    {GAL16V8,  0x00,0x1A,"GAL16V8",   2194,20,32, 64,32,2056,8,63,54,58,
,60,cfg16V8AB,sizeof(cfg16V8AB)/sizeof(int)},
    {GAL22V10, 0x48,0x49,"GAL22V10",
892,24,44,132,44,5828,8,61,60,58,10,16,cfg22V10,sizeof(cfg22V10)/sizeof(int)},
;

tatic HANDLE hInstance;
tatic HICON appicon;
tatic int vppmul=128;
tatic char progname[]="GALBlast";
tatic short far *lptbase=(short far *)0x00400008L;
tatic short lptport[3];
tatic int duration[16]={1,2,5,10,20,30,40,50,60,70,80,90,100,200,0,0};
tatic int lpt=0,security=0,erase=0,pulse=0,vpp=0;
tatic GALTYPE gal;
tatic char fusemap[10000];
tatic char backup[10000];
tatic unsigned short checksum;
tatic int pesread=0;
tatic unsigned char pes[12],pesbackup[12];
tatic char buffer[16348];
tatic unsigned char port;
tatic unsigned char control=0x04;
tatic int lasttick;

tatic void Debug(char format[],...)
```

```
    static char butter[1024];
    va_list arg;

    va_start(arg,format);
    wvsprintf(buffer,format,arg);
    va_end(arg);
    OutputDebugString(buffer);
}

static int Message(HWND wnd,LPSTR format,LPSTR caption,int flags,...)
{
    static char buffer[1024];
    va_list arg;

    va_start(arg,flags);
    wvsprintf(buffer,format,arg);
    va_end(arg);
    if(!caption) caption=progname;
    MessageBeep(flags&MB_ICONMASK);
    return MessageBox(wnd,buffer,caption,flags);
}

static void WaitCursor(BOOL on)
{
    POINT p;
    HWND w;
    HCURSOR h;

    if(on)
    {
        SetCursor(LoadCursor(0,IDC_WAIT));
    }
    else
    {
        GetCursorPos(&p);
      w=WindowFromPoint(p);
        h=w?GetClassWord(w,GCW_HCURSOR):0;
      SetCursor(h?h:LoadCursor(0,IDC_ARROW));
    }
}

static unsigned short CheckSum(int n)
{
    unsigned short c,e;
    long a;
    int i;

    c=e=0;
    a=0;
    for(i=0;i<n;i++)
    {
        e++;
        if(e==9)
      {
            e=1;
          a+=c;
            c=0;
      }
        c>>=1;
      if(fusemap[i]) c+=0x80;
    }
    return (unsigned short)((c>>(8-e))+a);
```

```
static int ParseFuseMap(HWND wnd,char *ptr)
{
    int i,n,type,checksumpos,address,pins,lastfuse,state; // 0=outside JEDEC,
1=skipping comment or unknown, 2=read command

    state=0;
    security=0;
    checksum=0;
    checksumpos=0;
    pins=0;
    lastfuse=0;
    for(n=0;ptr[n];n++)
    {
      if(ptr[n]=='*') state=2; // else ignored
      else switch(state)
      {
      case 2:
          if(!isspace(ptr[n])) switch(ptr[n])
          {
          case 'L':
            address=0;
            state=3;
            break;
          case 'F':
            state=5;
            break;
          case 'G':
                state=13;
            break;
          case 'Q':
            state=7;
            break;
          case 'C':
            checksumpos=n;
            state=14;
                break;
          default:
            state=1;
          }
          break;
      case 3:
          if(!isdigit(ptr[n])) return n;
          address=ptr[n]-'0';
          state=4;
          break;
      case 4:
          if(isspace(ptr[n]))
          {
            state=6;
          }
          else if(isdigit(ptr[n]))
          {
            address=10*address+(ptr[n]-'0');
          }
          else
          {
            return n;
          }
          break;
      case 5:
          if(isspace(ptr[n])) break; // ignored
          if(ptr[n]=='0'||ptr[n]=='1')
          {
            memset(fusemap,ptr[n]-'0',sizeof(fusemap));
```

```
    }
    else
    {
          return n;
    }

    state=1;
    break;
case 6:
    if(isspace(ptr[n])) break; // ignored
    if(ptr[n]=='0'||ptr[n]=='1')
    {
      fusemap[address--]=ptr[n]-'0';
    }
    else
    {
      return n;
    }
    break;
case 7:
    if(isspace(ptr[n])) break; // ignored
      if(ptr[n]=='P')
    {
      pins=0;
          state=8;
      }
      else if(ptr[n]=='F')
    {
      lastfuse=0;
      state=9;
    }
    else state=2;
      break;
case 8:
    if(isspace(ptr[n])) break; // ignored
    if(!isdigit(ptr[n])) return n;
    pins=ptr[n]-'0';
      state=10;
    break;
case 9:
    if(isspace(ptr[n])) break; // ignored
      if(!isdigit(ptr[n])) return n;
    lastfuse=ptr[n]-'0';
    state=11;
      break;
  case 10:
      if(isdigit(ptr[n]))
    {
          pins=10*pins+(ptr[n]-'0');
    }
    else if(isspace(ptr[n]))
    {
        state=12;
    }
    else return n;
      break;
case 11:
      if(isdigit(ptr[n]))
    {
      lastfuse=10*lastfuse+(ptr[n]-'0');
    }
      else if(isspace(ptr[n]))
    {
       state=12;
```

```
        }
        else return n;
        break;
   case 12:
        if(!isspace(ptr[n])) return n;
       break;
   case 13:
       if(isspace(ptr[n])) break; // ignored
       if(ptr[n]=='0'||ptr[n]=='1')
       {
         security=ptr[n]-'0';
       }
       else
       {
         return n;
       }
       state=1;
       break;
    case 14:
       if(isspace(ptr[n])) break; // ignored
         if(isdigit(ptr[n]))
         {
         checksum=ptr[n]-'0';
       }
         else if(toupper(ptr[n])>='A'&&toupper(ptr[n])<='F')
       {
         checksum=toupper(ptr[n])-'A'+10;
       }
         else return n;
       state=15;
       break;
    case 15:
       if(isdigit(ptr[n]))
         {
         checksum=16*checksum+ptr[n]-'0';
       }
       else if(toupper(ptr[n])>='A'&&toupper(ptr[n])<='F')
         {
         checksum=16*checksum+toupper(ptr[n])-'A'+10;
       }
         else if(isspace(ptr[n]))
         {
             state=2;
       }
       else return n;
       break;
    }
  }
  if(lastfuse||pins)
  {
    if(checksum&&checksum!=CheckSum(lastfuse))
    {
        if(Message(wnd,"Checksum given %04X calculated
)4X",NULL,MB_OKCANCEL,checksum,CheckSum(lastfuse))==IDCANCEL)
        {
          return checksumpos;
        }
    }
    for(type=0,i=1;i<sizeof(galinfo)/sizeof(galinfo[0]);i--)
    {
:((lastfuse==0||galinfo[i].fuses==lastfuse||galinfo[i].uesfuse==lastfuse&&galinfo[i]
uesfuse+8*galinfo[i].uesbytes==galinfo[i].fuses)
        && (pins==0||galinfo[i].pins==pins||galinfo[i].pins==24&&pins==28))
```

```
        {
          if(gal==0)
          {
              type-i;
              break;
          }
          else if(!type)
              {
              type=i;
          }
        }
    }
    if(gal!=type)
    {
        gal=type;
        SendDlgItemMessage(wnd,101,CB_SETCURSEL,gal-1,0L);
        SendMessage(wnd,WM_COMMAND,101,MAKELONG(0,CBN_SELCHANGE));
    }
  }
  return n;

tatic BOOL CheckJEDEC(HWND wnd)

  int position;

  position=ParseFuseMap(wnd,buffer);
  if(position==strlen(buffer)) return TRUE;
  Message(wnd,"Error in JEDEC",NULL,MB_OK|MB_ICONEXCLAMATION);
  SendDlgItemMessage(wnd,102,EM_SETSEL,0,MAKELONG(position,position));
  SendMessage(wnd,WM_NEXTDLGCTL,GetDlgItem(wnd,102),TRUE);
  return FALSE;

:atic void Delay(int msec)

  long start=timeGetTime();
  while(timeGetTime()<start+msec) ;

>id output(int port,unsigned char value)

  int last1,last2;

  last1=inportb(0x40);

  last2=inportb(0x40);
  while(last1==inportb(0x40)&&last2==inportb(0x40)) ;
  outportb(port,value);

isigned char input(int port)

  int last1,last2;

  last1=inportb(0x40);
  last2=inportb(0x40);
  while(last1==inportb(0x40)&&last2==inportb(0x40)) ;
  return inportb(port);

 set/reset individual pins of LPT port */
atic void SetSTROBE(BOOL on)
```

```
  output(lptport[lpt]+2,control=(on?control&~0x01:control|0x01));

atic void SetFEED(BOOL on)

  output(lptport[lpt]+2,control=(on?control&~0x02:control|0x02));

atic void SetINIT(BOOL on)

  output(lptport[lpt]-2,control=(on?control|0x04:control&~0x04));

atic void SetSEL(BOOL on)

  output(lptport[lpt]+2,control=(on?control&~0x08:control|0x08));

atic void SetRow(int row)

  port=(port&0x81)|(row<<1);
  output(lptport[lpt],port);

tic BOOL GetACK(void)

  return (input(lptport[lpt]+1)&0x40)!=0;

 set/reset individual pins of GAL */
atic void SetVCC(BOOL on)

  SetINIT(!on);

atic void SetVPP(int volt)

  // 49 -> 0x80 -> 12V, volt in
  output(lptport[lpt],(unsigned char)MulDiv(volt,vppmul,48));
  SetFEED(0);
  SetFEED(1);
  Delay(2); // according to Bram L.L.
  output(lptport[lpt],port);

atic void SetSDIN(BOOL on)

  if(on)
  {
    port|=0x01;
  }
  else
  {
    port&=0xFE;
  }
  output(lptport[lpt],port);

tic void SetSTB(BOOL on)

  SetSTROBE(on);

tic void SetPV(BOOL on)
```

```
   SetSEL(on);

tatic void SetSCLK(BOOL on)

   if(on)
   {
     port|=0x80;
   }
   else
   {
     port&=0x7F;
   }
   output(lptport[lpt],port);


atic BOOL GetSDOUT(void)

   return GetACK();


atic void TurnOff(void)

   Delay(100);
   SetRow(0x3F);
   SetSDIN(1);
   // SetSCLK(1);
   // SetPV(1);
   SetVPP(0); // turn off VPP
   Delay(2);
   SetVCC(0); // turn off VCC and drivers
   WaitCursor(FALSE);


OL WINAPI GrayDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

   RECT r;
   PAINTSTRUCT ps;

   switch(msg)
   {
   case WM_INITDIALOG:
     return TRUE;
   case WM_COMMAND:
     switch(LOWORD(wParam))
     {
     case IDCANCEL:
         EndDialog(wnd,FALSE);
         return TRUE;
     }
     break;
    case WM_CTLCOLOR:
     if(HIWORD(lParam)==CTLCOLOR_STATIC)
     {
         SetBkMode((HDC)wParam,TRANSPARENT);
         return GetStockObject(LTGRAY_BRUSH);
     }
     return FALSE;
    case WM_ERASEBKGND:
     if(IsIconic(wnd)) return TRUE;
     GetClientRect(wnd,&r);
     FillRect((HDC)wParam,&r,GetStockObject(LTGRAY_BRUSH));
     SetWindowLong(wnd,DWL_MSGRESULT,1);
```

```
    return TRUE;
  case WM_PAINT:
    if(IsIconic(wnd))
    {
        BeginPaint(wnd,&ps);
        DefWindowProc(wnd,WM_ICONERASEBKGND,(WORD)ps.hdc,0L);
        DrawIcon(ps.hdc,0,0,appicon);
        EndPaint(wnd,&ps);
        return TRUE;
    }
    return FALSE;
  case WM_QUERYDRAGICON:
    return appicon;
  }
  return FALSE;

OOL WINAPI _export AssureDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

  int i,j,n;
  char buffer[128];
  static int useerase;

  switch(msg)
  {
  case WM_INITDIALOG:
    SetWindowText(wnd,caption[lParam]);
      SetDlgItemText(wnd,IDOK,caption[lParam]);
      if(pes[1]&0x10)
      {
          strcpy(buffer,"3.3V");
    }
      else
      {
          strcpy(buffer,"5V");
      }
      switch(pes[3])
      {
      case LATTICE:
          strcat(buffer," Lattice ");
          break;
      case NATIONAL:
          strcat(buffer," National ");
          break;
      case SGSTHOMSON:
          strcat(buffer," ST ");
        break;
      default:
          strcat(buffer," Unknown ");
    }
      strcat(buffer,galinfo[gal].name);
      SetDlgItemText(wnd,100,buffer);
    j=wsprintf(buffer,"PES:");
      for(i=0;i<galinfo[gal].pesbytes;i++)
      {
        j+=wsprintf(buffer+j," %02X",pes[i]);
      }
      SetDlgItemText(wnd,103,buffer);
    wsprintf(buffer,"VPP=%d.%02dV Prog-Pulse=%dms Hapus-
lse=%dms",vpp/4,(vpp%4)*25,pulse,hapus);
      SetDlgItemText(wnd,104,buffer);
    n--1;
      for(i=48;i<83;i++)
      {
```

```
        wsprintf(buffer,"%d.%02d",i/4,(i%4)*25);
       j=(int)SendDlgItemMessage(wnd,101,CB_ADDSTRING,0,(LPARAM)(LPSTR)buffer);
       SendDlgItemMessage(wnd,101,CB_SETITEMDATA,j,i);
       if(i==vpp) n=j;
     }
   SendDlgItemMessage(wnd,101,CB_SETCURSEL,n,0L);
   n=-1;
     if(lParam==HAPUSGAL||lParam==HAPUSALL)
     {
         useerase=TRUE;
         SetDlgItemText(wnd,99,"Erase-Pulse");
         for(i=25;i<=800;i<<=1)
         {
             wsprintf(buffer,"%d",i/2);
=(int)SendDlgItemMessage(wnd,102,CB_ADDSTRING,0,(LPARAM)(LPSTR)buffer);
             SendDlgItemMessage(wnd,102,CB_SETITEMDATA,j,i/2);
             if(i/2==erase) n=j;
         }
     }
     else
     {
         useerase=FALSE;
         SetDlgItemText(wnd,99,"Prog-Pulse");
         for(i=0;i<14;i++)
         {
             wsprintf(buffer,"%d",duration[i]);
         j=(int)SendDlgItemMessage(wnd,102,CB_ADDSTRING,0,(LPARAM)(LPSTR)buffer);
             SendDlgItemMessage(wnd,102,CB_SETITEMDATA,j,duration[i]);
             if(duration[i]==pulse) n=j;
         }
     }
     SendDlgItemMessage(wnd,102,CB_SETCURSEL,n,0L);
     return TRUE;
  case WM_COMMAND:
     switch(LOWORD(wParam))
     {
     case IDOK:
       i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
       if(i==-1)
         {
             Message(wnd,"Select VPP first",progname,MB_ICONEXCLAMATION|MB_OK);
             SetFocus(GetDlgItem(wnd,101));
             return TRUE;
         }
         vpp=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
         i=(int)SendDlgItemMessage(wnd,102,CB_GETCURSEL,0,0L);
       if(i==-1)
         {
             GetDlgItemText(wnd,99,buffer,sizeof(buffer));
             Message(wnd,"Select %s
irst",progname,MB_ICONEXCLAMATION|MB_OK,(LPSTR)buffer);
             SetFocus(GetDlgItem(wnd,102));
             return TRUE;
         }
         if(useerase)
         {
             erase=(int)SendDlgItemMessage(wnd,102,CB_GETITEMDATA,i,0L);
         }
         else
         {
         pulse=(int)SendDlgItemMessage(wnd,102,CB_GETITEMDATA,i,0L);
         }
         EndDialog(wnd,1);
```

```
            return TRUE;
        case IDCANCEL:
            EndDialog(wnd,0);
          return TRUE;
        }
    }
    return GrayDlgProc(wnd,msg,wParam,lParam);


atic BOOL TurnOn(HWND wnd,int mode)

    BOOL writeorerase;

(mode==TULISGAL||mode==HAPUSGAL||mode==HAPUSALL||mode==BURNSECURITY||mode==TULISPES
mode==VPPTEST)
    {
        if(DialogBoxParam(hInstance,MAKEINTRESOURCE(4),wnd,AssureDlgProc,mode)!=1)
turn FALSE;
        writeorerase=TRUE;
    }
    else
    {
        writeorerase=FALSE;
    }
    WaitCursor(TRUE);
    SetVPP(0); // assure VPP is 0
    SetSTB(1); // set all (disabled) outputs to H (high impedance)
    SetSCLK(1);
    SetSDIN(1);
    SetPV(1);
    SetRow(0x3F);
    SetVCC(1); // turn on VCC and drivers
    Delay(100);
    SetVPP(writeorerase?vpp:READVPP); // turn on VPP (readout at 12V)
    SetSCLK(0); // now that all pins are in programming mode
    SetPV(0); // bring critical pins down to L
    Delay(10);
    if(writeorerase)
    {
      SetPV(1);
      Delay(10);
    }
    return TRUE;


  atic BOOL ReceiveBit(void)

    BOOL bit;

    bit=GetSDOUT();
    SetSCLK(1);
    SetSCLK(0);
    return bit;


  atic void DiscardBits(int n)

    while(n-->0) ReceiveBit();


  atic void SendBit(int bit)

    SetSDIN(bit);
```

```
    SetSCLK(1);
    SetSCLK(0);

tatic void SendBits(int n,int bit)

    while(n-->0) SendBit(bit);

tatic void SendAddress(int n,int row)

    while(n-->0)
    {
      SendBit(row&1);
      row>>=1;
    }

tatic void Strobe(int msec)

    Delay(2);
    SetSTB(0);
    Delay(msec);
    SetSTB(1);
    Delay(2);

:atic void StrobeRow(int row)

    switch(gal)
    {
    case GAL16V8:
      SetRow(row);
      break;
    default:
      SetRow(0);
      SendBits(galinfo[gal].bits,0);
      SendAddress(6,row);
    }
    Strobe(2);

:atic BOOL GetSetup(HWND wnd)

    int i;

    i=(int)SendDlgItemMessage(wnd,100,CB_GETCURSEL,0,0L);
    if(i==-1)
    {
      Message(wnd,"Select LPT port first",NULL,MB_OK|MB_ICONEXCLAMATION);
      return FALSE;
    }
    lpt=(int)SendDlgItemMessage(wnd,100,CB_GETITEMDATA,i,0L);
    i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
    if(i==-1)
    {
      Message(wnd,"Select a GAL type first",NULL,MB_OK|MB_ICONEXCLAMATION);
      return FALSE;
    }
    gal=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
    return TRUE;

:atic void GetPES(char pes[])
```

```
  int bitmask,byte;

  StrobeRow(galinfo[gal].pesrow);
  for(byte=0;byte<galinfo[gal].pesbytes;byte++)
  {
    pes[byte]=0;
    for(bitmask=0x1;bitmask<=0x80;bitmask<<=1)
    {
        if(ReceiveBit()) pes[byte]|=bitmask;
    }
  }


atic void BacaPES(HWND wnd,char pes[])

  TurnOn(wnd,BACAPES);
  GetPES(pes);
  TurnOff();


atic void TulisPES(HWND wnd,char pes[])

  int byte,bitmask;

  if(TurnOn(wnd,TULISPES))
  {
      switch(gal)
      {
      case GAL16V8:
          SetRow(galinfo[gal].pesrow);
          for(byte=0;byte<8;byte++) for(bitmask=0x01;bitmask<=0x80;bitmask<<=1)
ndBit((pes[byte]&bitmask)!=0);
          break;
      default:
          SetRow(0);
          for(byte=0;byte<galinfo[gal].pesbytes;byte++)
r(bitmask=0x01;bitmask<=0x80;bitmask<<=1) SendBit((pes[byte]&bitmask)!=0);
          if(galinfo[gal].pesbytes*8<galinfo[gal].bits) SendBits(galinfo[gal].bits-
linfo[gal].pesbytes*8,0);
          SendAddress(6,galinfo[gal].pesrow);
          SetSDIN(0);
      }
      Strobe(pulse);
    TurnOff();
  }


atic void TulisGAL(HWND wnd,char *fuses)

  int row,bit;

  if(TurnOn(wnd,TulisGAL))
  {
      switch(gal)
      {
      case GAL16V8:
      case GAL20V8:
          for(row=0;row<galinfo[gal].rows;row++)
          {
              SetRow(row);
              for(bit=0;bit<galinfo[gal].bits;bit++)
              {
                  SendBit(fuses[galinfo[gal].rows*bit+row]);
```

```
        }
        Strobe(pulse);
    }
    // UES
    SetRow(galinfo[gal].uesrow);
    for(bit=0;bit<64;bit++)
    {
        SendBit(fuses[galinfo[gal].uesfuse+bit]);
    }
    Strobe(pulse);
    // CFG
    SetRow(galinfo[gal].cfgrow);
    for(bit=0;bit<82;bit++)
    {
        switch(pes[2])
        {
        case 0x00:
            SendBit(fuses[cfg16V8[bit]]);
            break;
        case 0x1A:
            SendBit(fuses[cfg16V8AB[bit]]);
            break;
        }
    }

    // UES
    SendBits(20,0);
    for(bit=0;bit<72;bit++) SendBit(fuses[galinfo[gal].uesfuse+bit]);
    SendBits(3,0);
    SendBit(1);
    SendAddress(7,galinfo[gal].uesrow);
    SendBits(16,0);
    SetSDIN(0);
    Strobe(pulse);

    // CFG
    SetRow(galinfo[gal].cfgrow);
    for(bit=0;bit<galinfo[gal].cfgbits;bit++)
    {
        SendBit(fuses[galinfo[gal].cfg[bit]]);
    }
    SetSDIN(0);
    Strobe(pulse);

    break;
default:
    SetRow(0);
    for(row=0;row<galinfo[gal].rows;row++)
    {
        for(bit=0;bit<galinfo[gal].bits;bit++)
        {
            SendBit(fuses[galinfo[gal].rows*bit+row]);
        }
        SendAddress(6,row);
        SetSDIN(0);
        Strobe(pulse);
    }
    // UES
    for(bit=0;bit<galinfo[gal].uesbytes*8;bit++)
    {
        SendBit(fuses[galinfo[gal].uesfuse+bit]);
    }
    if(galinfo[gal].uesbytes*8<galinfo[gal].bits)
    {
```

```
            SendBits(galinfo[gal].bits-galinfo[gal].uesbytes*8,0);
        }
        SendAddress(6,galinfo[gal].uesrow);
        SetSDIN(0);
        Strobe(pulse);
        // CFG
        SetRow(galinfo[gal].cfgrow);
        for(bit=0;bit<galinfo[gal].cfgbits;bit++)
        {
            SendBit(fuses[galinfo[gal].cfg[bit]]);
        }
        SetSDIN(0);
        Strobe(pulse);
    }
  TurnOff();
}

atic void BacaGAL(HWND wnd,char *fuses)

  int row,bit;

  TurnOn(wnd,BacaGAL);
  switch(gal)
  {
  case GAL16V8:
    for(row=0;row<galinfo[gal].rows;row++)
    {
        StrobeRow(row);
        for(bit=0;bit<galinfo[gal].bits;bit++)
        {
          fuses[galinfo[gal].rows*bit+row]=ReceiveBit();
        }
    }
    // UES
    StrobeRow(galinfo[gal].uesrow);
    for(bit=0;bit<64;bit++)
    {
        fuses[galinfo[gal].uesfuse+bit]=ReceiveBit();
    }
    // CFG
    StrobeRow(galinfo[gal].cfgrow);
    for(bit=0;bit<82;bit++)
    {
        switch(pes[2])
        {
        case 0x00:
          fuses[cfg16V8[bit]]=ReceiveBit();
          break;
        case 0x1A:
          fuses[cfg16V8AB[bit]]=ReceiveBit();
          break;
        }
    }

    // UES
    StrobeRow(galinfo[gal].uesrow);
    DiscardBits(20);
    for(bit=0;bit<72;bit++) fuses[galinfo[gal].uesfuse+bit]=ReceiveBit();
    // CFG
    SetRow(galinfo[gal].cfgrow);
    Strobe(2);
    for(bit=0;bit<galinfo[gal].cfgbits;bit++)
ses[galinfo[gal].cfg[bit]]=ReceiveBit();
```

```
    break;
  default:
    for(row=0;row<galinfo[gal].rows;row++)
    {
        StrobeRow(row);
        for(bit=0;bit<galinfo[gal].bits;bit++)
        {
          fuses[galinfo[gal].rows*bit+row]=ReceiveBit();
        }
    }
    // UES
    StrobeRow(galinfo[gal].uesrow);
    for(bit=0;bit<galinfo[gal].uesbytes*8;bit++)
    {
        fuses[galinfo[gal].uesfuse+bit]=ReceiveBit();
    }
    // CFG
    SetRow(galinfo[gal].cfgrow);
    Strobe(2);
    for(bit=0;bit<galinfo[gal].cfgbits;bit++)
    {
        fuses[galinfo[gal].cfg[bit]]=ReceiveBit();
    }
  }
  TurnOff();


atic void HapusGAL(HWND wnd,int gal)

  if(TurnOn(wnd,HapusGAL))
  {
      SetRow(galinfo[gal].eraserow);
    if(gal==GAL16V8) SendBit(1);
      Strobe(erase);
    TurnOff();
  }


atic void EraseWholeGAL(HWND wnd,int gal)

  int i;

  if(TurnOn(wnd,HapusALL))
  {
      SetRow(galinfo[gal].eraseallrow);
    if(gal==GAL16V8) SendBit(1);
      Strobe(erase);
    TurnOff();
  }


atic void BurnSecurity(HWND wnd,int gal)

  if(TurnOn(wnd,BURNSECURITY))
  {
      switch(gal)
      {
      case GAL16V8:
          SetRow(61);
          SendBit(1);
          break;
       default:
          SetRow(0);
```

```
        switch(gal)

        {
        case GAL22V10:
            SendBits(132,0);
            break;
        }
        SendAddress(6,61);
        SetSDIN(0);
      }
      Strobe(pulse);
    TurnOff();
  }

tatic void FormatJEDEC(int gal,char *fuses,char *buffer)

  int i,j,k,n;
  int unused,start;
  time_t now;
  BOOL unquote;
  unsigned char ch;

  time(&now);
  n=wsprintf(buffer,"JEDEC file for %s created on
",(LPSTR)galinfo[gal].name,(LPSTR)asctime(localtime(&now))-1;

=wsprintf(buffer+n,"\r\n*QP%d*QF%d*QV0*F0*G0*X0*\r\n",galinfo[gal].pins,galinfo[gal
fuses);
  if(gal==GAL6001||gal==GAL6002)
  {
    for(i=k=0;i<64;i++)
    {
        start=n;
        unused=TRUE;
        n+=wsprintf(buffer+n,"L%04d ",k);
        for(j=0;j<114;j++)
        {
          if(fuses[k]) unused=FALSE;
          buffer[n++]='0'-fuses[k++];
        }
        n+=wsprintf(buffer+n,"*\r\n");
        if(unused) n=start;
    }
    for(i=0;i<11;i++)
    {
        start=n;
        unused=TRUE;
        n+=wsprintf(buffer+n,"L%04d ",k);
        for(j=0;j<78;j++)
        {
          if(fuses[k]) unused=FALSE;
          buffer[n++]='0'+fuses[k++];
        }
        n+=wsprintf(buffer+n,"*\r\n");
        if(unused) n=start;
    }
  }
  else for(i=k=0;i<galinfo[gal].bits;i++)
  {
    start=n;

    unused=TRUE;
    n+=wsprintf(buffer+n,"L%04d ",k);
```

```
  for(j=0;j<galinfo[gal].rows;j++)
  {
      if(fuses[k]) unused=FALSE;
      buffer[n++]='0'+fuses[k++];
  }
  n+=wsprintf(buffer+n,"*\r\n");
  if(unused) n=start;
}
if(k<galinfo[gal].uesfuse)
{
  start=n;
  unused=TRUE;
  n+=wsprintf(buffer+n,"L%04d ",k);
  while(k<galinfo[gal].uesfuse)
  {
      if(fuses[k]) unused=FALSE;
      buffer[n++]='0'+fuses[k++];
  }
  n+=wsprintf(buffer+n,"*\r\n");
  if(unused) n=start;
}
start=n;
unused=TRUE;
n+=wsprintf(buffer+n,"N UES");
unquote=TRUE;
for(j=0;j<galinfo[gal].uesbytes;j++)
{
  ch=0;
  for(i=0;i<8;i++) if(fuses[k+8*j+i]) ch|=1<<i;
  if(isprint(ch))
  {
      if(unquote)
      {
        buffer[n++]=' ';
        buffer[n++]='"';
        unquote=FALSE;
      }
      buffer[n++]=ch;
  }
  else
  {
      if(!unquote)
      {
        buffer[n++]='"';
        unquote=TRUE;
      }
      n+=wsprintf(buffer+n," %02X",ch);
  }
}
n+=wsprintf(buffer+n,"\"*\r\nL%04d "!unquote,k);
for(j=0;j<8*galinfo[gal].uesbytes;j++)
{
  if(fuses[k]) unused=FALSE;
  buffer[n++]='0'+fuses[k++];
}
n+=wsprintf(buffer+n,"*\r\n");
if(unused) n=start;
if(k<galinfo[gal].fuses)
{
  start=n;
  unused=TRUE;
  n+=wsprintf(buffer+n,"L%04d ",k);
  while(k<galinfo[gal].fuses)
  {
```

```
        if(fuses[k]) unused=FALSE;
        buffer[n++]='0'+fuses[k++];
    }
    n+=wsprintf(buffer+n,"*\r\n");
    if(unused) n=start;
  }
  n+=wsprintf(buffer+n,"N PES");
  for(i=0;i<galinfo[gal].pesbytes;i++)
  {
    n+=wsprintf(buffer+n," %02X",pes[i]);
  }
  n+=wsprintf(buffer+n,"*\r\nC%04X\r\n*",CheckSum(galinfo[gal].fuses));
  buffer[n]='\0';

tatic BOOL programmable(char *buffer,char *fuses,int bytes)

  while(bytes-->0)
  {
    if(~*buffer&*fuses) return FALSE;
    fuses++;
    buffer++;
  }
  return TRUE;

tatic BOOL erased(char *buffer,int bytes)

  while(bytes-->0) if(*buffer++!=1) return FALSE;
  return TRUE;

tatic char *compare(char *buffer,char *ptr,int bytes)

  while(bytes-->0)
  {
      if(*buffer!=*ptr) return buffer;
      buffer++;
    ptr++;
  }
  return NULL;

tatic int hex(char *buffer)

  char *end;
  unsigned long value;

  value=strtoul(buffer,&end,16);
  if(value>255||*end!='\0') return -1;
  return (int)value;

OL WINAPI AboutDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

  DWORD size,handle;
  UINT len;
  char buffer[_MAX_PATH];
  void FAR *ptr;
  char *mem;

  switch(msg)
  {
  case WM_INITDIALOG:
```

```
    GetModuleFileName(hInstance,buffer,sizeof(buffer));
    size=GetFileVersionInfoSize(buffer,&handle);
    if(size)
    {
        mem=malloc(size);
        if(mem)
        {
          if(GetFileVersionInfo(buffer,0L,size,mem))
          {

f(VerQueryValue(mem,"\\StringFileInfo\\040904E4\\FileDescription",&ptr,&len)&&len&&p
r)
                {
                  SetDlgItemText(wnd,100,ptr);
                }
f(VerQueryValue(mem,"\\StringFileInfo\\040904E4\\FileVersion",&ptr,&len)&&len&&ptr)
                {
                  SetDlgItemText(wnd,101,ptr);
                }
f(VerQueryValue(mem,"\\StringFileInfo\\040904E4\\LegalCopyright",&ptr,&len)&&len&&pt
)
                {
                  SetDlgItemText(wnd,102,ptr);
                }
f(VerQueryValue(mem,"\\StringFileInfo\\040904E4\\CompanyName",&ptr,&len)&&len&&ptr)
                {
                  SetDlgItemText(wnd,103,ptr);
                }
            }
            free(mem);
          }
      }
      break;
    }
    return GrayDlgProc(wnd,msg,wParam,lParam);


tatic void ParsePES(void)

    int algo=pes[1]&0x0F;

    if(algo==5)
    {
      erase=(25<<((pes[4]>>2)&7))/2;
      pulse=duration[((((unsigned)pes[5]<<8)|pes[4])>>5)&15];
      vpp=2*((pes[5]>>1)&31)+20;
/         vread=2*(((((unsigned)pes[6]<<8)|pes[5])>>6)&31)+20;
    }
    else switch(gal)
    {
    case GAL16V8:
      erase=100;
      goto more;
    more:
      switch(algo)
      {
      case 0:
          vpp=63; // 15.75V
          pulse=100;
          break;
```

```
  case 1:
      vpp=63; // 15.75V
      pulse=80;
      break;
  case 2:
      vpp=66; // 16.5V
      pulse=10;
      break;
  case 3:
      vpp=pes[3]==NATIONAL?60:58; // 15/14.5V
      pulse=40;
      break;
  case 4:
      vpp=56; // 14V
      pulse=100;
      break;
  }
  break;
default:
  erase=pes[3]==NATIONAL?80:100;
  switch(algo)
  {
  case 0:
      vpp=66; // 16.5V
      pulse=10;
      break;
  case 1:
      vpp=63; // 15.75V
      pulse=100;
      break;
  case 2:
      vpp=pes[3]==NATIONAL?60:58; // 15/14.5V
      pulse=40;
      break;
  case 3:
      vpp=56; // 14V
      pulse=100;
      break;
  }
  break;
}

OOL WINAPI ProgDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

  int i;
  char buffer[128];

  switch(msg)
  {
  case WM_INITDIALOG:
    for(i=0;i<galinfo[gal].pesbytes;i++)
    {
        wsprintf(buffer,"%02X",(unsigned char)pes[i]);
        SetDlgItemText(wnd,110+i,buffer);
    }
    while(i<12)
    {
        ShowWindow(GetDlgItem(wnd,200+i),SW_HIDE);
        ShowWindow(GetDlgItem(wnd,110+i++),SW_HIDE);
    }
    SendMessage(wnd,WM_COMMAND,101,0L);
    return TRUE;
  case WM_COMMAND:
```

```
switch(LOWORD(wParam))
{
case IDOK:
    EndDialog(wnd,TRUE);
    return TRUE;
case IDCANCEL:
    EndDialog(wnd,FALSE);
    return TRUE;
default:
    for(i=0;i<galinfo[gal].pesbytes;i++)
    {
      if(GetDlgItemText(wnd,110+i,buffer,sizeof(buffer))%3==0||hex(buffer)==-1)
      {
          EnableWindow(GetDlgItem(wnd,IDOK),FALSE);
          return TRUE;
      }
    }
    EnableWindow(GetDlgItem(wnd,IDOK),TRUE);
    for(i=0;i<galinfo[gal].pesbytes;i++)
    {
      GetDlgItemText(wnd,110+i,buffer,sizeof(buffer));
      pes[i]=hex(buffer);
    }
      if(pes[1]&0x10)
      {
          strcpy(buffer,"3.3V");
      }
      else
      {
          strcpy(buffer,"5V");
      }
     switch(pes[3])
     {
     case LATTICE:
       strcat(buffer," Lattice ");
       break;
     case NATIONAL:
       strcat(buffer," National ");
       break;
     case SGSTHOMSON:
       strcat(buffer," ST Microsystems ");
       break;
     default:
       strcpy(buffer," Unknown ");
     }
     strcat(buffer,galinfo[gal].name);
     ParsePES();
     i=strlen(buffer);
     wsprintf(buffer+i," VPP=%d.%02dV Pulse=%dms
apus=%dms",vpp/4,(vpp%4)*25,pulse,Hapus);
     SetDlgItemText(wnd,100,buffer);
     return TRUE;
  }
}
return GrayDlgProc(wnd,msg,wParam,lParam);

tatic BOOL TestProperGAL(HWND wnd)

int type;

if(!GetSetup(wnd)) return FALSE;
ReadPES(wnd,pes);
for(type=sizeof(galinfo)/sizeof(galinfo[0]);type;type--)
```

```
  {
    if(pes[2]==galinfo[type].id0||pes[2]==galinfo[type].id1) break;
  }
  if(type==-0)
  {
    pesread=0;
    if(Message(wnd,"PES Ilegal atau tidak dikenali, Lanjut
",NULL,MB_YESNO|MB_ICONQUESTION)==IDNO) return FALSE;
  }
  else
  {
    pesread=type;
    if(type!=gal)
    {
        switch(Message(wnd,"PES mengindikasikan type GAL tidak sesuai dengan yang
ipilih, Ubah Tipe GAL?",NULL,MB_YESNOCANCEL|MB_ICONQUESTION)==IDCANCEL)
        {
        case IDCANCEL:
          return FALSE;
        case IDYES:
          gal=type;
        }
    }
  }
  ParsePES();
  return TRUE;

OL WINAPI  export VoltDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

  char temp[6];
  double v;
  char *ptr;

  switch(msg)
  {
  case WM_INITDIALOG:
      vppmul=128;
      SendDlgItemMessage(wnd,101,EM_LIMITTEXT,0,sizeof(temp));
      TurnOn(wnd,FALSE);
      return TRUE;
  case WM_COMMAND:
      switch(wParam)
      {
      case IDOK:
          if(GetDlgItemText(wnd,101,temp,sizeof(temp))>0)
          {
              ptr=strchr(temp,',');
              if(ptr) *ptr='.';
              v=strtod(temp,&ptr);
              if(*ptr!='\0'||v<1.0||v>30.0)
              {
                  Message(wnd,"Invalid value",progname,MB_ICONEXCLAMATION|MB_OK);
              SendDlgItemMessage(wnd,101,EM_SETSEL,0,MAKELONG(0,32767));
                  SetFocus(GetDlgItem(wnd,101));
                  return TRUE;
              }
              vppmul=(int)((12.0*128)/v+0.5);
              wsprintf(temp,"%d",vppmul);
              WriteProfileString(progname,"MulDiv",temp);
          }
          /* fall thru */
      case IDCANCEL:
          TurnOff();
```

```
            EndDialog(wnd,0);
            return TRUE;
        }
        break;
    }
    return GrayDlgProc(wnd,msg,wParam,lParam);

)OL WINAPI GALDlgProc(HWND wnd,unsigned msg,WPARAM wParam,LPARAM lParam)

    int i,j,l;
    OPENFILENAME ofn;
    WINDOWPLACEMENT wp;
    char *ptr;
    RECT r;
    static RECT editrect;
    int handle;
    char temp[40];


    switch(msg)
    {
    case WM_INITDIALOG:
      GetClientRect(wnd,&r);
      GetWindowRect(GetDlgItem(wnd,102),&editrect);
      ScreenToClient(wnd,(LPPOINT)&editrect.left);
      ScreenToClient(wnd,(LPPOINT)&editrect.right);
      SetRect(&editrect,editrect.left-r.left,editrect.top-r.top,r.right-
(itrect.right,r.bottom-editrect.bottom);
      if(GetProfileString(progname,"Window","",temp,sizeof(temp))>0)
      {
          sscanf(temp,"%u%u%u%u",&r.left,&r.top,&r.right,&r.bottom);
          MoveWindow(wnd,r.left,r.top,r.right-r.left,r.bottom-r.top,FALSE);
      }
      lpt=GetProfileInt(progname,"Port",1);
      for(i=0;i<3;i++)
      {
          wsprintf(buffer,"LPT%d",i+1);
          if(GetProfileString(progname,buffer,"",buffer,sizeof(buffer))>0)
          {
            lptport[i]=strtoul(buffer,NULL,0);
          }
          else
          {
            lptport[i]=lptbase[i];
          }
          if(lptport[i])
          {
            wsprintf(buffer,"LPT%d: %04x",i+1,lptport[i]);
            j=(int)SendDlgItemMessage(wnd,100,CB_ADDSTRING,0,(LPARAM)(LPSTR)buffer);
            SendDlgItemMessage(wnd,100,CB_SETITEMDATA,j,i);
            if(i==lpt) SendDlgItemMessage(wnd,100,CB_SETCURSEL,j,0L);
          }
      }
      SendMessage(wnd,WM_COMMAND,100,MAKELONG(0,CBN_SELCHANGE));
      TurnOff();
      for(i=1;i<sizeof(galinfo)/sizeof(galinfo[0]);i++)
      {

(int)SendDlgItemMessage(wnd,101,CB_ADDSTRING,0,(LPARAM)(LPSTR)galinfo[i].name),
          SendDlgItemMessage(wnd,101,CB_SETITEMDATA,j,galinfo[i].type);
      }
      return TRUE;
    case WM_DESTROY:
```

```
      wp.length=sizeof(wp);
      GetWindowPlacement(wnd,&wp);
      sprintf(temp,"%u %u %u
%u",wp.rcNormalPosition.left,wp.rcNormalPosition.top,wp.rcNormalPosition.right,wp.rcN
ormalPosition.bottom);
      WriteProfileString(progname,"Window",temp);
        break;
    case WM_SIZE:
      GetClientRect(wnd,&r);
      MoveWindow(GetDlgItem(wnd,102),r.left+editrect.left,r.top+editrect.top,r.right-
editrect.right-(r.left+editrect.left),r.bottom-editrect.bottom-
(r.top+editrect.top),TRUE);
      break;
    case WM_COMMAND:
      switch(LOWORD(wParam))
      {
      case IDCANCEL:
            if(SendDlgItemMessage(wnd,102,EM_GETMODIFY,0,0L))
            {
                switch(Message(wnd,"Memasukkan JEDEC fuse map, simpan ?",NULL,
                MB_ICONQUESTION|MB_YESNOCANCEL))
                {
                case IDCANCEL:
                    return 0L;
                case IDYES:
                    if(SendMessage(wnd,WM_COMMAND,4,0L)==0) return 0L;
                }
            }
          EndDialog(wnd,0);
          return TRUE;
       case 3: // Open
          memset(&ofn,0,sizeof(OPENFILENAME));
          buffer[0]='\0';
          ofn.lStructSize=sizeof(OPENFILENAME);
          ofn.hwndOwner=wnd;
          ofn.hInstance=hInstance;
          ofn.lpstrFilter="Fusemaps(.JED)\0*.jed\0All Files\0*.*\0";
          ofn.nFilterIndex=1L;
          ofn.lpstrFile=buffer;
          ofn.lpstrDefExt="JED";
            ofn.lpstrTitle="Load JEDEC";
          ofn.nMaxFile=sizeof(buffer);
          ofn.Flags=OFN_FILEMUSTEXIST|OFN_PATHMUSTEXIST|OFN_HIDEREADONLY;
          if(GetOpenFileName(&ofn))
          {
            handle=_lopen(buffer,OF_READ);
            if(handle==-1)
            {
                Message(wnd,"Can not read from file",NULL,MB_OK|MB_ICONEXCLAMATION);
            }
            else
            {
                i=_lread(handle,buffer,sizeof(buffer));
                buffer[i]='\0';
                ptr=strchr(buffer,'\2');
                if(ptr)
                {
                  memmove(buffer,ptr+1,i-(ptr-buffer));
                    ptr=strchr(buffer,'\3');
                    if(ptr) *ptr='\0';
                }
                SetDlgItemText(wnd,102,buffer);
                _lclose(handle);
                    SendMessage(wnd,WM_COMMAND,102,MAKELONG(0,EN_CHANGE));
```

```
            i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
            if(i==-1) gal=0; else
al=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
            CheckJEDEC(wnd);
          }
        }
        break;
    case 4: //  Save
        if(SendDlgItemMessage(wnd,102,WM_GETTEXTLENGTH,0,0L)<1)
        {
            Message(wnd,"Nothing to save",progname,MB_ICONEXCLAMATION|MB_OK);
            return 0L;
        }
        memset(&ofn,0,sizeof(OPENFILENAME));
        buffer[0]='\0';
        ofn.lStructSize=sizeof(OPENFILENAME);
          ofn.lpstrTitle="Save JEDEC";
        ofn.hwndOwner=wnd;
        ofn.hInstance=hInstance;
        ofn.lpstrFilter="Fusemaps(.JED)\0*.jed\0All Files\0*.*\0";
        ofn.nFilterIndex=1L;
        ofn.lpstrFile=buffer;
        ofn.lpstrDefExt="JED";
        ofn.nMaxFile=sizeof(buffer);
        ofn.Flags=OFN_PATHMUSTEXIST|OFN_PATHMUSTEXIST|OFN_HIDEREADONLY;
        if(GetSaveFileName(&ofn))
        {
          handle=_lcreat(buffer,0);
          if(handle==-1)
          {
              Message(wnd,"File tidak bisa di buat",NULL,MB_OK|MB_ICONEXCLAMATION);
          }
          else
          {
              buffer[0]='\2';
              i=GetDlgItemText(wnd,102,buffer+1,sizeof(buffer)-5)+1;
              for(l=0,j=1;j<i;j++) l+=buffer[j];
              i+=wsprintf(buffer+i,"\3%04X",l+5);
              _lwrite(handle,buffer,i);
              _lclose(handle);
                  SendDlgItemMessage(wnd,102,EM_SETMODIFY,0,0L);
                  return TRUE;
          }
        }
        break;
    case 5: // Baca GAL
        if(!TestProperGAL(wnd)) return TRUE;
        ReadGAL(wnd,fusemap);
        FormatJEDEC(gal,fusemap,buffer);
        SetDlgItemText(wnd,102,buffer);
        SendMessage(wnd,WM_COMMAND,102,MAKELONG(0,EN_CHANGE));
        return TRUE;
    case 6: // Tulis GAL
        i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
        if(i==-1) gal=0; else
al=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
        if(GetDlgItemText(wnd,102,buffer,sizeof(buffer))<1)
          {
              MessageBox(wnd,"Open file JEDECnya
alu",progname,MB_ICONEXCLAMATION|MB_OK);
              return 0L;
          }
        if(!CheckJEDEC(wnd)) return TRUE;
        if(!TestProperGAL(wnd)) return TRUE;
```

```
        WriteGAL(wnd,fusemap);
        if(security)
        {
          if(Message(wnd,"Program security akan mencegah IC GAL dibaca dan
erifikasi terhadap IC GAL. Apakah kamu ingi melanjutkan
",progname,MB_ICONEXCLAMATION|MB_YESNO)!=IDYES) return TRUE;
          BurnSecurity(wnd,gal);
        }
        return TRUE;
      case 12: // Verify GAL
        i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
        if(i==-1) gal=0; else
el=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
        if(GetDlgItemText(wnd,102,buffer,sizeof(buffer))<1)
          {
              MessageBox(wnd,"Open file JEDECnya
ulu",progname,MB_ICONEXCLAMATION|MB_OK);
              return 0L;
          }
        if(!CheckJEDEC(wnd)) return TRUE;
        if(!TestProperGAL(wnd)) return TRUE;
        ReadGAL(wnd,backup);
        if(memcmp(backup,fusemap,galinfo[gal].fuses)==0)
        {
          Message(wnd,"Verify ok.",NULL,MB_OK);
        }
        else
        {
          Message(wnd,"Verify failed !",NULL,MB_OK);
        }
        return TRUE;
      case 7: // Hapus GAL
        if(!TestProperGAL(wnd)) return TRUE;
        if(pes[1]&0x80) if(Message(wnd,"GAL is a MASTER, HAPUS GAL
",NULL,MB_ICONSTOP|MB_YESNO)!=IDYES) return TRUE;
        EraseGAL(wnd,gal);
        return TRUE;
      case 8: // Tulis PES
        if(!GetSetup(wnd)) return FALSE;
        if(DialogBox(hInstance,MAKEINTRESOURCE(2),wnd,ProgDlgProc))
        {
          WritePES(wnd,pes);
        }
        return TRUE;
      case 10: // SECURITY
        if(!TestProperGAL(wnd)) return TRUE;
        if(Message(wnd,"Program security akan mencegah IC GAL dibaca dan verifikasi
erhadap IC GAL. Apakah kamu ingi melanjutkan
",progname,MB_ICONEXCLAMATION|MB_YESNO)!=IDYES) return TRUE;
        if(pes[1]&0x80) if(Message(wnd,"GAL is a MASTER, still burn security fuse
",NULL,MB_ICONSTOP|MB_YESNO)!=IDYES) return TRUE;
        BurnSecurity(wnd,gal);
        return TRUE;
      case 11: // HAPUS ALL
        if(!TestProperGAL(wnd)) return TRUE;
        if(Message(wnd,"Penghapusan IC GAL juga akan menghapus PES (Programmer
lectronic Signature) dan GAL tidak bisa dipakai. Apakah kamu ingin Lanjut
",progname,MB_ICONEXCLAMATION|MB_YESNO)!=IDYES) return TRUE;
        if(pes[1]&0x80) if(Message(wnd,"GAL is a MASTER, Menghapus semua IC GAL
",NULL,MB_ICONSTOP|MB_YESNO)!=IDYES) return TRUE;
        EraseWholeGAL(wnd,gal);
        return TRUE;
      case 100:
        if(HIWORD(lParam)==CBN_SELCHANGE)
```

```
    {
      i=(int)SendDlgItemMessage(wnd,100,CB_GETCURSEL,0,0L);
      if(i!=-1)
      {
          lpt=(int)SendDlgItemMessage(wnd,100,CB_GETITEMDATA,i,0L);
          itoa(lpt,buffer,10);
          WriteProfileString(progname,"Port",buffer);
          TurnOff();
      }
    }
    return TRUE;
case 101:
    if(HIWORD(lParam)==CBN_SELCHANGE)
    {
      SendDlgItemMessage(wnd,103,CB_SETCURSEL,-1,0L);
      SendDlgItemMessage(wnd,104,CB_SETCURSEL,-1,0L);
      SendDlgItemMessage(wnd,105,CB_SETCURSEL,-1,0L);
    }
    return TRUE;
case 900:
    i=GetModuleFileName(hInstance,buffer,sizeof(buffer));
    strcpy(buffer+i-3,"htm");
    if(ShellExecute(wnd,"open",buffer,NULL,NULL,SW_SHOW)<32)
    {
      MessageBox(wnd,buffer,"ShellExecute failed",MB_OK|MB_ICONEXCLAMATION);
    }
    return TRUE;
case 999:
    return DialogBox(hInstance,MAKEINTRESOURCE(3),wnd,AboutDlgProc);
case 200:
case 201:
case 202:
case 203:
case 204:
case 205:
case 206:
case 207:
case 208:
case 209:
    SendDlgItemMessage(wnd,101,CB_SETCURSEL,LOWORD(wParam)-200,0L);
    return TRUE;
case 300:
case 301:
case 302:
    SendDlgItemMessage(wnd,100,CB_SETCURSEL,LOWORD(wParam)-300,0L);
    return TRUE;
  case 310:
      DialogBox(hInstance,MAKEINTRESOURCE(5),wnd,VolDlgProc);
      return TRUE;
  }
 break;
case WM_INITMENU:
  i=(int)SendDlgItemMessage(wnd,100,CB_GETCURSEL,0,0L);
  if(i==-1) lpt=-1; else
t=(int)SendDlgItemMessage(wnd,100,CB_GETITEMDATA,i,0L);
  CheckMenuItem((HMENU)wParam,300,lpt==0?MF_CHECKED:MF_UNCHECKED);
  CheckMenuItem((HMENU)wParam,301,lpt==1?MF_CHECKED:MF_UNCHECKED);
  CheckMenuItem((HMENU)wParam,302,lpt==2?MF_CHECKED:MF_UNCHECKED);
  EnableMenuItem((HMENU)wParam,300,lptport[0]?MF_ENABLED:MF_GRAYED);
  EnableMenuItem((HMENU)wParam,301,lptport[1]?MF_ENABLED:MF_GRAYED);
  EnableMenuItem((HMENU)wParam,302,lptport[2]?MF_ENABLED:MF_GRAYED);
  i=(int)SendDlgItemMessage(wnd,101,CB_GETCURSEL,0,0L);
  if(i==-1) gal=0; else gal=(int)SendDlgItemMessage(wnd,101,CB_GETITEMDATA,i,0L);
  CheckMenuItem((HMENU)wParam,200,gal==1?MF_CHECKED:MF_UNCHECKED);
```

```
      CheckMenuItem((HMENU)wParam,205,gal==2?MF_CHECKED:MF_UNCHECKED);
      return 0L;
    }
    return GrayDlgProc(wnd,msg,wParam,lParam);
}

int PASCAL WinMain(HANDLE hInst,HANDLE hPrev,LPSTR lpCmdLine,int nCmdShow)
{
    hInstance=hInst;
    vppmul=GetProfileInt(progname,"MulDiv",128);
    appicon=LoadIcon(hInst,MAKEINTRESOURCE(1));
    return DialogBox(hInst,MAKEINTRESOURCE(1),0,GALDlgProc);

}
```

GAL16V8

High Performance E²CMOS PLD Generic Array Logic™

Features

HIGH PERFORMANCE E²CMOS® TECHNOLOGY
- 3.5 ns Maximum Propagation Delay
- Fmax = 250 MHz
- 3.0 ns Maximum from Clock Input to Data Output
- UltraMOS® Advanced CMOS Technology

50% to 75% REDUCTION IN POWER FROM BIPOLAR
- 75mA Typ Icc on Low Power Device
- 45mA Typ Icc on Quarter Power Device

ACTIVE PULL-UPS ON ALL PINS

E² CELL TECHNOLOGY
- Reconfigurable Logic
- Reprogrammable Cells
- 100% Tested/100% Yields
- High Speed Electrical Erasure (<100ms)
- 20 Year Data Retention

EIGHT OUTPUT LOGIC MACROCELLS
- Maximum Flexibility for Complex Logic Designs
- Programmable Output Polarity
- Also Emulates 20-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility

PRELOAD AND POWER-ON RESET OF ALL REGISTERS
- 100% Functional Testability

APPLICATIONS INCLUDE:
- DMA Control
- State Machine Control
- High Speed Graphics Processing
- Standard Logic Speed Upgrade

ELECTRONIC SIGNATURE FOR IDENTIFICATION

LEAD-FREE PACKAGE OPTIONS

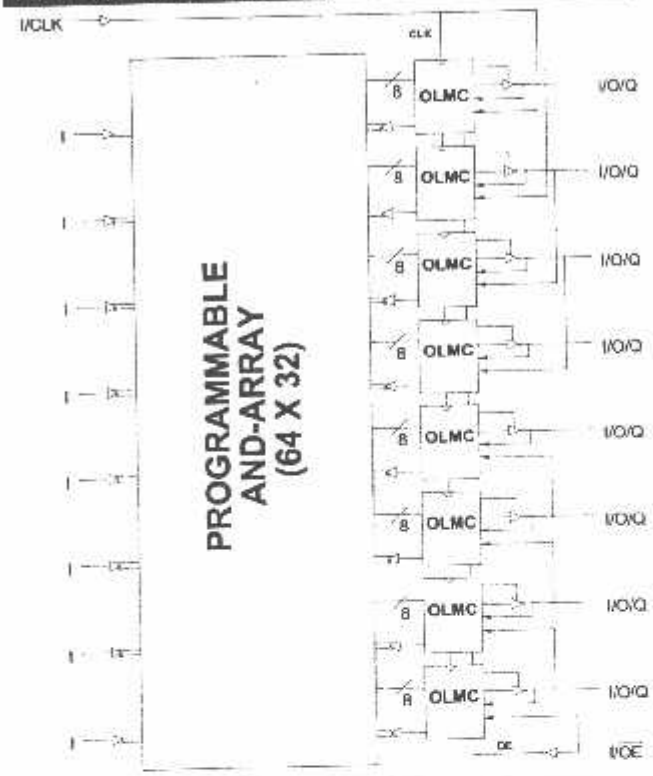
Description

The GAL16V8, at 3.5 ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

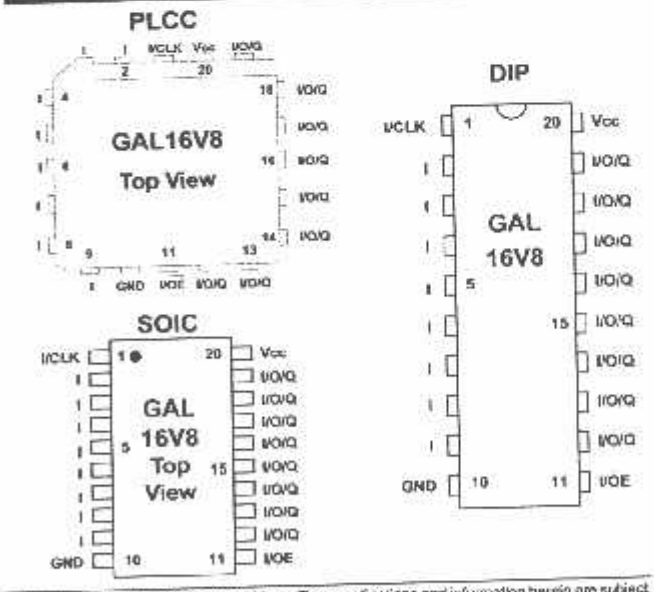
The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL16V8 are the PAL architectures listed in the table of the macrocell description section. GAL16V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Functional Block Diagram



Pin Configuration



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 268-8000; 1-800-LATTICE; FAX (503) 268-8556; http://www.latticesemi.com

August 2004

GAL16V8 Ordering Information

Conventional Packaging

Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
3.5	2.5	3.0	115	GAL16V8D-3LJ	20-Lead PLCC
5	3	4	115	GAL16V8D-5LJ	20-Lead PLCC
7.5	7	5	115	GAL16V8D-7LP	20-Pin Plastic DIP
			115	GAL16V8D-7LJ	20-Lead PLCC
			115	GAL16V8D-7LS	20-Pin SOIC
10	10	7	55	GAL16V8D-10QP	20-Pin Plastic DIP
			55	GAL16V8D-10QJ	20-Lead PLCC
			115	GAL16V8D-10LP	20-Pin Plastic DIP
			115	GAL16V8D-10LJ	20-Lead PLCC
			115	GAL16V8D-10LS	20-Pin SOIC
15	12	10	55	GAL16V8D-15QP	20-Pin Plastic DIP
			55	GAL16V8D-15QJ	20-Lead PLCC
			90	GAL16V8D-15LP	20-Pin Plastic DIP
			90	GAL16V8D-15LJ	20-Lead PLCC
			90	GAL16V8D-15LS	20-Pin SOIC
25	15	12	55	GAL16V8D-25QP	20-Pin Plastic DIP
			55	GAL16V8D-25QJ	20-Lead PLCC
			90	GAL16V8D-25LP	20-Pin Plastic DIP
			90	GAL16V8D-25LJ	20-Lead PLCC
			90	GAL16V8D-25LS	20-Pin SOIC

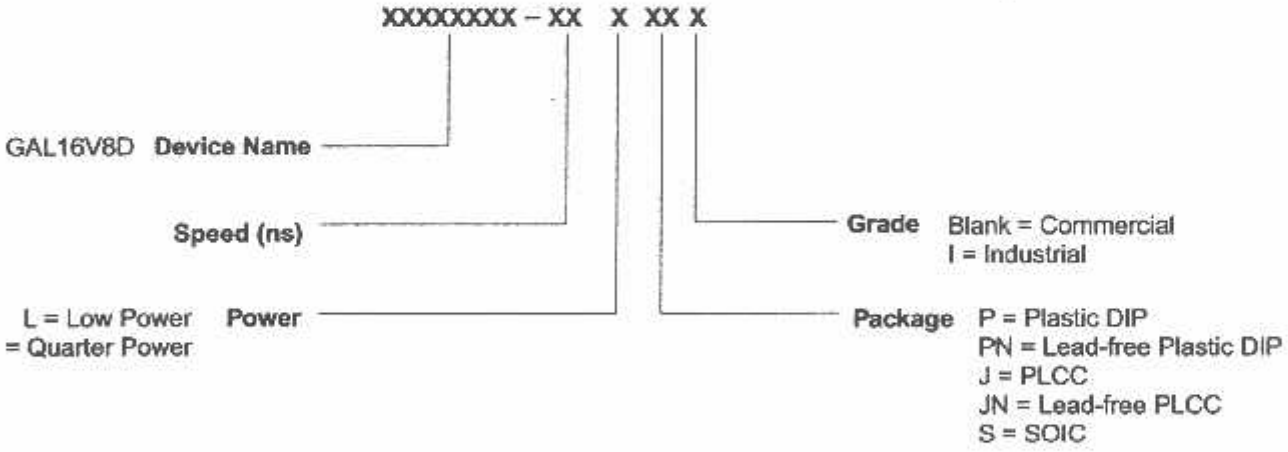
Industrial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
7.5	7	5	130	GAL16V8D-7LPI	20-Pin Plastic DIP
			130	GAL16V8D-7LJI	20-Lead PLCC
10	10	7	130	GAL16V8D-10LPI	20-Pin Plastic DIP
			130	GAL16V8D-10LJI	20-Lead PLCC
15	12	10	130	GAL16V8D-15LPI	20-Pin Plastic DIP
			130	GAL16V8D-15LJI	20-Lead PLCC
20	13	11	65	GAL16V8D-20QPI	20-Pin Plastic DIP
			65	GAL16V8D-20QJI	20-Lead PLCC
25	15	12	65	GAL16V8D-25QPI	20-Pin Plastic DIP
			65	GAL16V8D-25QJI	20-Lead PLCC
			130	GAL16V8D-25LPI	20-Pin Plastic DIP
			130	GAL16V8D-25LJI	20-Lead PLCC

Lead-Free Packaging
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
3.5	2.5	3.0	115	GAL16V8D-3LJN	Lead-Free 20-Lead PLCC
5	3	4	115	GAL16V8D-5LJN	Lead-Free 20-Lead PLCC
7.5	7	5	115	GAL16V8D-7LPN	Lead-Free 20-Pin Plastic DIP
			115	GAL16V8D-7LJN	Lead-Free 20-Leac PLCC
10	10	7	55	GAL16V8D-10QPN	Lead-Free 20-Pin Plastic DIP
			55	GAL16V8D-10QJN	Lead-Free 20-Lead PLCC
			115	GAL16V8D-10LPN	Lead-Free 20-Pin Plastic DIP
			115	GAL16V8D-10LJN	Lead Free 20-Lead PLCC
15	12	10	55	GAL16V8D-15QPN	Lead-Free 20-Pin Plastic DIP
			55	GAL16V8D-15QJN	Lead-Free 20-Lead PLCC
			90	GAL16V8D-15LPN	Lead-Free 20-Pin Plastic DIP
			90	GAL16V8D-15LJN	Lead-Free 20 Lead PLCC
25	15	12	55	GAL16V8D-25QPN	Lead-Free 20-Pin Plastic DIP
			55	GAL16V8D-25QJN	Lead-Free 20-Lead PLCC
			90	GAL16V8D-25LPN	Lead-Free 20-Pin Plastic DIP
			90	GAL16V8D-25LJN	Lead-Free 20-Lead PLCC

Part Number Description



Output Logic Macrocell (OLMC)

he following discussion pertains to configuring the output logic acrocell. It should be noted that actual implementation is accomlished by development software/hardware and is completely transarent to the user.

here are three global OLMC configuration modes possible: **imple, complex,** and **registered.** Details of each of these modes re illustrated in the following pages. Two global bits, SYN and C0, control the mode configuration for all macrocells. The XOR t of each macrocell controls the polarity of the output in any of the iree modes, while the AC1 bit of each of the macrocells controls ie input/output configuration. These two global and 16 individal architecture bits define all possible configurations in a GAL16V8 The information given on these architecture bits is only to give better understanding of the device. Compiler software will transarently set these architecture bits from the pin definitions, so the er should not need to directly manipulate these architecture bits.

ie following is a list of the PAL architectures that the GAL16V8 n emulate. It also shows the OLMC mode under which the AL16V8 emulates the PAL architecture.

PAL Architectures Emulated by GAL16V8	GAL16V8 Global OLMC Mode
16R8	Registered
16R6	Registered
16R4	Registered
16RP8	Registered
16RP6	Registered
16RP4	Registered
16L8	Complex
16H8	Complex
16P8	Complex
10L8	Simple
12L6	Simple
14L4	Simple
16L2	Simple
10H8	Simple
12H6	Simple
14H4	Simple
16H2	Simple
10P8	Simple
12P6	Simple
14P4	Simple
16P2	Simple

Compiler Support for OLMC

iftware compilers support the three different global OLMC modes different device types. These device types are listed in the table low. Most compilers have the ability to automatically select the vice type, generally based on the register usage and output able (OE) usage. Register usage on the device forces the softire to choose the registered mode. All combinatorial outputs with E controlled by the product term will force the software to choose complex mode. The software will choose the simple mode only ien all outputs are dedicated combinatorial without OE control. e different device types listed in the table can be used to override automatic device selection by the software. For further details, er to the compiler software manuals.

ien using compiler software to configure the device, the user ist pay special attention to the following restrictions in each mode. **egistered mode** pin 1 and pin 11 are permanently configured as clock and output enable, respectively. These pins cannot be configured as dedicated inputs in the registered mode.

In **complex mode** pin 1 and pin 11 become dedicated inputs and use the feedback paths of pin 19 and pin 12 respectively. Because of this feedback path usage, pin 19 and pin 12 do not have the feedback option in this mode.

In **simple mode** all feedback paths of the output pins are routed via the adjacent pins. In doing so, the two inner most pins (pins 15 and 16) will not have the feedback option as these pins are always configured as dedicated combinatorial output.

	Registered	Complex	Simple	Auto Mode Select
BEL	P16V8R	P16V8C	P16V8AS	P16V8
UPL	G16V8MS	G16V8MA	G16V8AS	G16V8
OG/iC	GAL16V8_R	GAL16V8_C7	GAL16V8_C8	GAL16V8
rCAD-PLD	"Registered"[1]	"Complex"[1]	"Simple"[1]	GAL16V8A
LDesigner	P16V8R[2]	P16V8C[2]	P16V8C[2]	P16V8A
ANGO-PLD	G16V8R	G16V8C	G16V8AS[3]	G16V8

Used with **Configuration** keyword.
Prior to Version 2.0 support.
Supported on Version 1.20 or later.

Registered Mode

the Registered mode, macrocells are configured as dedicated gistered outputs or as I/O functions.

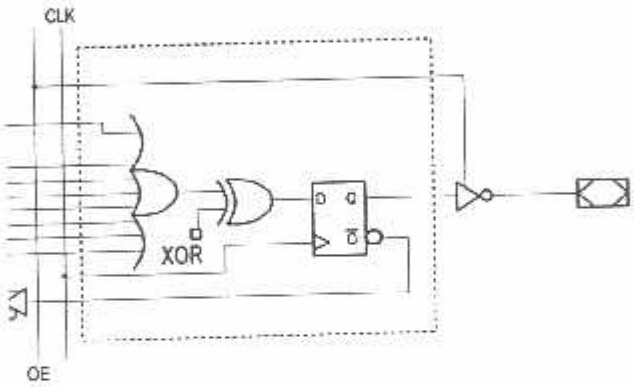
chitecture configurations available in this mode are similar to the mmon 16R8 and 16RP4 devices with various permutations of larity, I/O and register placement.

registered macrocells share common clock and output enable ntrol pins. Any macrocell can be configured as registered or I/ Up to eight registers or up to eight I/O's are possible in this mode.

Dedicated input or output functions can be implemented as subsets of the I/O function.

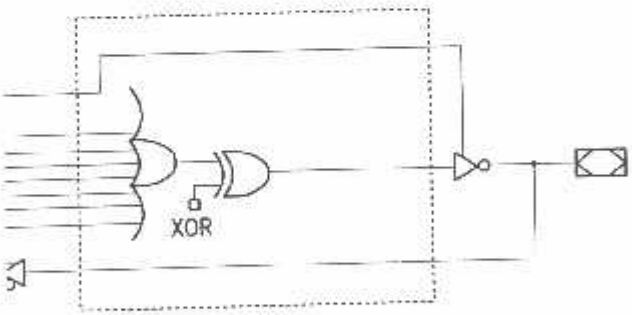
Registered outputs have eight product terms per output. I/O's have seven product terms per output.

The JEDEC fuse numbers, including the User Electronic Signature (UES) fuses and the Product Term Disable (PTD) fuses, are shown on the logic diagram on the following page.



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 & $\overline{OE}$ for registered output configuration.



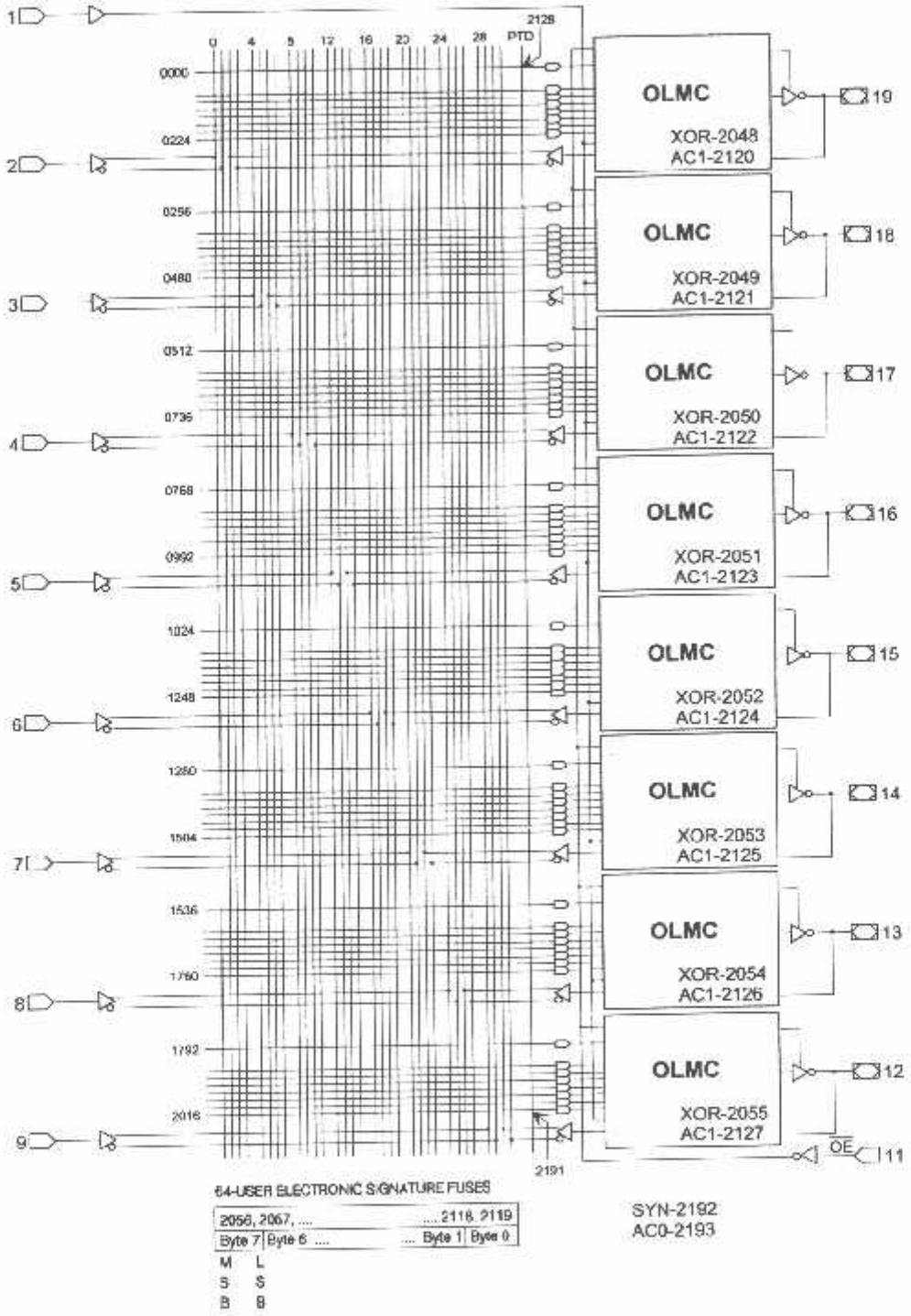
Combinatorial Configuration for Registered Mode

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

te: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Registered Mode Logic Diagram

DIP & PLCC Package Pinouts



Complex Mode

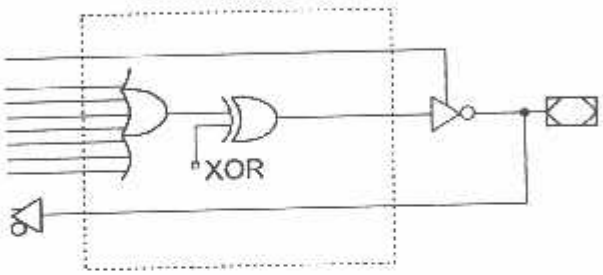
In the Complex mode, macrocells are configured as output only or I/O functions.

Architecture configurations available in this mode are similar to the common 16L8 and 16P8 devices with programmable polarity in each macrocell.

Up to six I/O's are possible in this mode. Dedicated inputs or outputs can be implemented as subsets of the I/O function. The two outer most macrocells (pins 12 & 19) do not have input capability. Designs requiring eight I/O's can be implemented in the Registered mode.

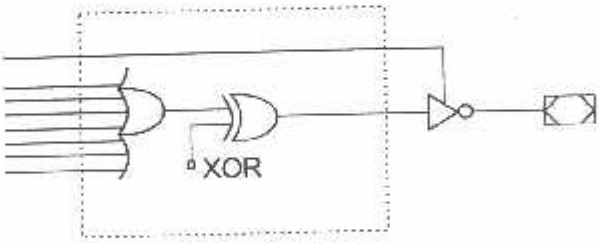
All macrocells have seven product terms per output. One product term is used for programmable output enable control. Pins 1 and 11 are always available as data inputs into the AND array.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram on the following page.



Combinatorial I/O Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 13 through Pin 18 are configured to this function.



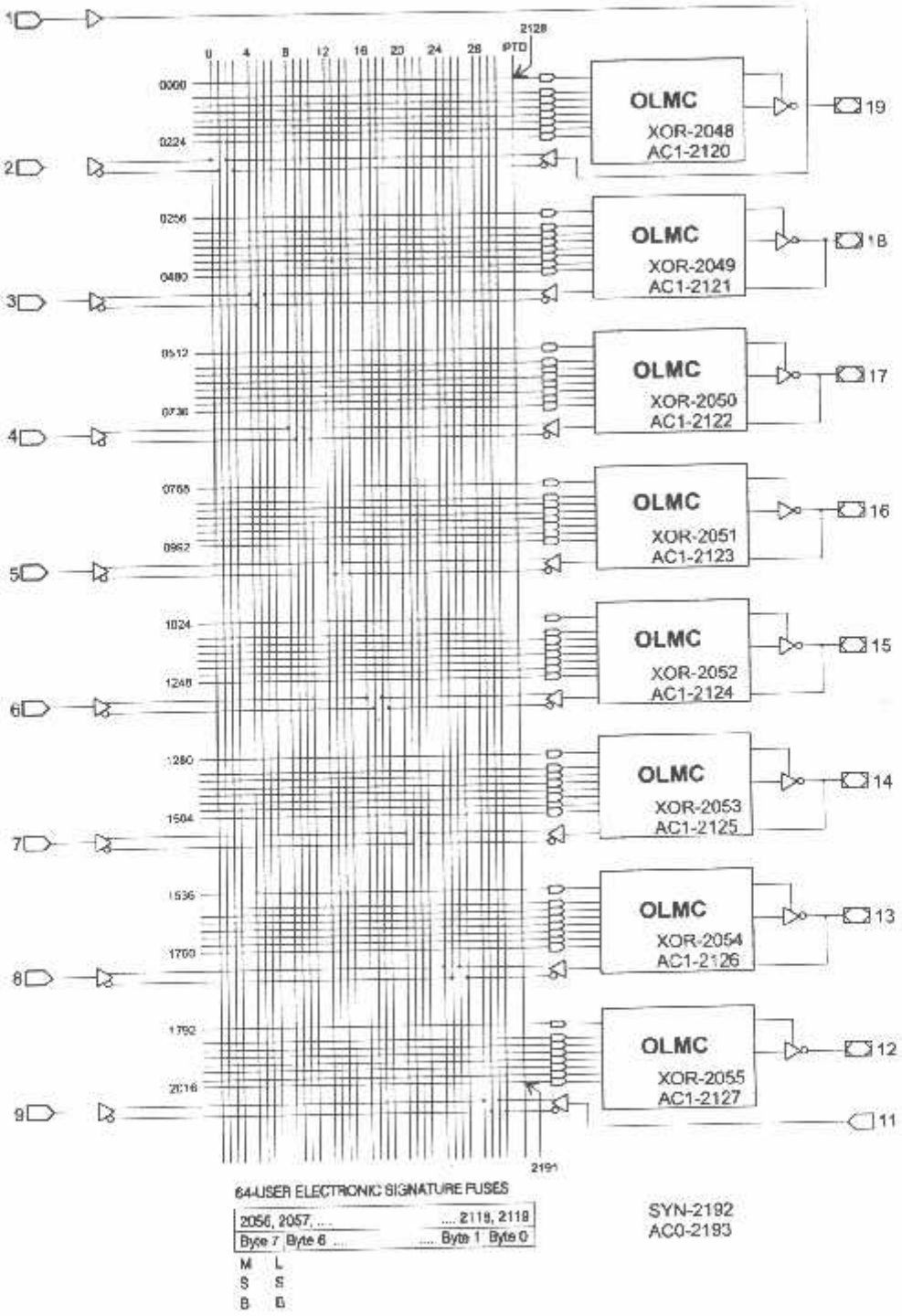
Combinatorial Output Configuration for Complex Mode

- SYN=1.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1.
- Pin 12 and Pin 19 are configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Complex Mode Logic Diagram

DIP & PLCC Package Pinouts



Simple Mode

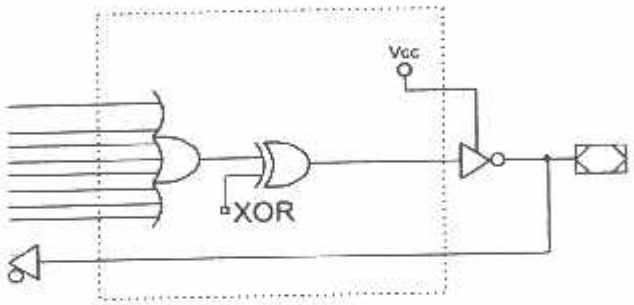
n the Simple mode, macrocells are configured as dedicated inputs or as dedicated, always active, combinatorial outputs.

Architecture configurations available in this mode are similar to the common 10L8 and 12P6 devices with many permutations of generic output polarity or input choices.

All outputs in the simple mode have a maximum of eight product terms that can control the logic. In addition, each output has programmable polarity.

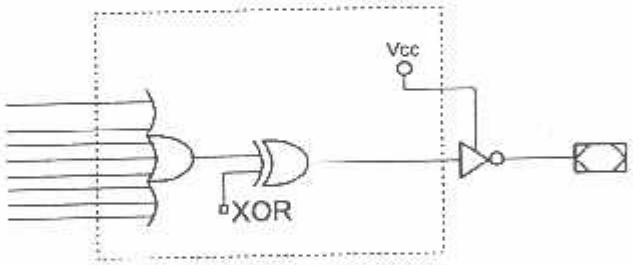
Pins 1 and 11 are always available as data inputs into the AND array. The center two macrocells (pins 15 & 16) cannot be used as input or I/O pins, and are only available as dedicated outputs.

The JEDEC fuse numbers including the UES fuses and PTD fuses are shown on the logic diagram.



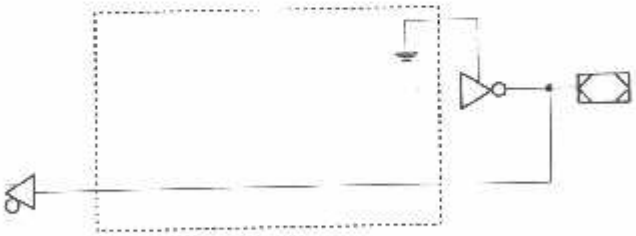
Combinatorial Output with Feedback Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.



Combinatorial Output Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this configuration.
- Pins 15 & 16 are permanently configured to this function.



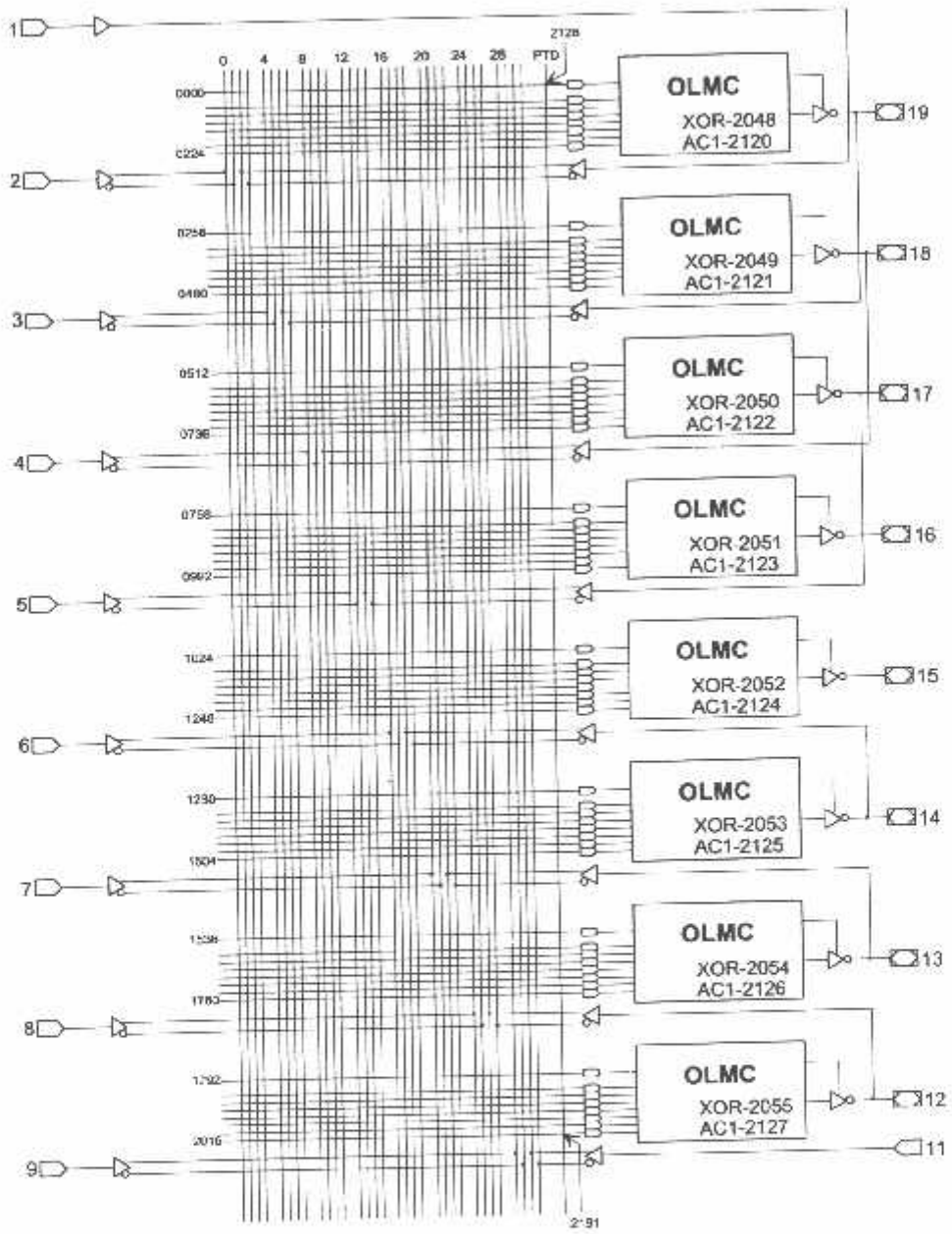
Dedicated Input Configuration for Simple Mode

- SYN=1.
- AC0=0.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this configuration.
- All OLMC **except** pins 15 & 16 can be configured to this function.

Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Simple Mode Logic Diagram

DIP & PLCC Package Pinouts



64-USER ELECTRONIC SIGNATURE FUSES

2056, 2057, ...	... 2118, 2119
Byte 7 Byte 8 ...	... Byte 1 Byte 0

MSB LSB

SYN-2192
AC0-2193

Absolute Maximum Ratings[1]

upply voltage V_{CC} ..... –0.5 to +7V
iput voltage applied ..... –2.5 to V_{CC} +1.0V
ff-state output voltage applied ..... –2.5 to V_{CC} +1.0V
torage Temperature ..... –65 to 150°C
mbient Temperature with
Power Applied ..... –55 to 125°C

Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied (while programming, follow the programming specifications).

Recommended Operating Conditions

Commercial Devices:
Ambient Temperature (T_A) ..... 0 to 75°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.75 to +5.25V

Industrial Devices:
Ambient Temperature (T_A) ..... –40 to 85°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.50 to +5.50V

DC Electrical Characteristics

Over Recommended Operating Conditions (Unless Otherwise Specified)

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.[3]	MAX.	UNITS
V_{IL}	Input Low Voltage		Vss – 0.5	—	0.8	V
V_{IH}	Input High Voltage		2.0	—	Vcc+1	V
I_{IL}[2]	Input or I/O Low Leakage Current	$0V \le V_{IN} \le V_{IL}$ (MAX.)	—	—	–100	µA
I_{IH}	Input or I/O High Leakage Current	$3.5V \le V_{IN} \le V_{CC}$	—	—	10	µA
V_{OL}	Output Low Voltage	I_{OL} = MAX. Vin = V_{IL} or V_{IH}	—	—	0.5	V
V_{OH}	Output High Voltage	I_{OH} = MAX. Vin = V_{IL} or V_{IH}	2.4	—	—	V
I_{OL}	Low Level Output Current	L-3/-5 & -7 (Ind. PLCC)	—	—	16	mA
		L-7 (Except Ind. PLCC)/-10/-15/-25 Q-10/-15/-20/-25	—	—	24	mA
I_{OH}	High Level Output Current		—	—	–3.2	mA
I_{OS}[2]	Output Short Circuit Current	V_{CC} = 5V V_{OUT} = 0.5V T_A = 25°C	–30	—	–150	mA

COMMERCIAL

I_{CC}	Operating Power Supply Current	V_{IL} = 0.5V V_{IH} = 3.0V f_{toggle} = 15MHz Outputs Open	L-3/-5/-7/-10	—	75	115	mA
			L-15/-25	—	75	90	mA
			Q-10/-15/-25	—	45	55	mA

INDUSTRIAL

I_{CC}	Operating Power Supply Current	V_{IL} = 0.5V V_{IH} = 3.0V f_{toggle} = 15MHz Outputs Open	L-7/-10/-15/-25	—	75	130	mA
			Q-20/-25	—	45	65	mA

The leakage current is due to the internal pull-up resistor on all pins. See **Input Buffer** section for more information.
One output at a time for a maximum duration of one second. Vout = 0.5V was selected to avoid test problems caused by tester ound degradation. Characterized but not 100% tested.
Typical values are at Vcc = 5V and T_A = 25 °C

AC Switching Characteristics

Over Recommended Operating Conditions

PARAMETER	TEST COND[1]	DESCRIPTION	COM -3		COM -5		COM / IND -7		UNITS
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{pd}	A	Input or I/O to Comb. Output	1	3.5	1	5	1	7.5	ns
t_{co}	A	Clock to Output Delay	1	3	1	4	1	5	ns
t_{cf}[2]	—	Clock to Feedback Delay	—	2.5	—	3	—	3	ns
t_{su}	—	Setup Time, Input or Feedback before Clock↑	2.5	—	3	—	5	—	ns
t_h	—	Hold Time, Input or Feedback after Clock↑	0	—	0	—	0	—	ns
f_{max}[3]	A	Maximum Clock Frequency with External Feedback, 1/(tsu + tco)	182	—	142.8	—	100	—	MHz
	A	Maximum Clock Frequency with Internal Feedback, 1/(tsu + tcf)	200	—	166	—	125	—	MHz
	A	Maximum Clock Frequency with No Feedback	250	—	166	—	125	—	MHz
t_{wh}	—	Clock Pulse Duration, High	2[4]	—	3[4]	—	4	—	ns
t_{wl}	—	Clock Pulse Duration, Low	2[4]	—	3[4]	—	4	—	ns
t_{en}	B	Input or I/O to Output Enabled	—	4.5	1	6	1	9	ns
	B	$\overline{OE}$ to Output Enabled	—	4.5	1	6	1	6	ns
t_{dis}	C	Input or I/O to Output Disabled	—	4.5	1	5	1	9	ns
	C	$\overline{OE}$ to Output Disabled	—	4.5	1	5	1	6	ns

1) Refer to **Switching Test Conditions** section.
2) Calculated from fmax with internal feedback. Refer to **fmax Descriptions** section.
3) Refer to **fmax Descriptions** section. Characterized but not 100% tested.
4) Characterized but not 100% tested.

Capacitance (T_A = 25 C, f = 1.0 MHz)

SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	8	pF	V_{CC} = 5.0V, V_I = 2.0V
$C_{I/O}$	I/O Capacitance	8	pF	V_{CC} = 5.0V, $V_{I/O}$ = 2.0V

*Characterized but not 100% tested.

AC Switching Characteristics

Over Recommended Operating Conditions

PARAM.	TEST COND[1].	DESCRIPTION	COM / IND -10		COM / IND -15		IND -20		COM / IND -25		UNITS
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
tpd	A	Input or I/O to Comb. Output	3	10	3	15	3	20	3	25	ns
tco	A	Clock to Output Delay	2	7	2	10	2	11	2	12	ns
tcf[2]	—	Clock to Feedback Delay	—	6	—	8	—	9	—	10	ns
tsu	—	Setup Time, Input or Fdbk before Clk↑	7.5	—	12	—	13	—	15	—	ns
th	—	Hold Time, Input or Fdbk after Clk↑	0	—	0	—	0	—	0	—	ns
fmax[3]	A	Maximum Clock Frequency with External Feedback, 1/(tsu + tco)	66.7	—	45.5	—	41.6	—	37	—	MHz
	A	Maximum Clock Frequency with Internal Feedback, 1/(tsu + tcf)	71.4	—	50	—	45.4	—	40	—	MHz
	A	Maximum Clock Frequency with No Feedback	83.3	—	62.5	—	50	—	41.6	—	MHz
twh	—	Clock Pulse Duration, High	6	—	8	—	10	—	12	—	ns
twl	—	Clock Pulse Duration, Low	6	—	8	—	10	—	12	—	ns
ten	B	Input or I/O to Output Enabled	1	10	—	15	—	18	—	20	ns
	B	$\overline{OE}$ to Output Enabled	1	10	—	15	—	18	—	20	ns
tdis	C	Input or I/O to Output Disabled	1	10	—	15	—	18	—	20	ns
	C	$\overline{OE}$ to Output Disabled	1	10	—	15	—	18	—	20	ns

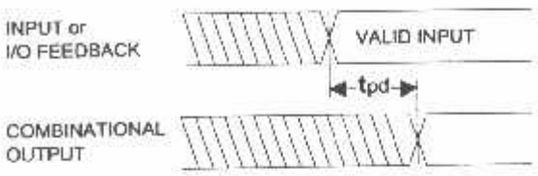
Refer to **Switching Test Conditions** section.
Calculated from fmax with internal feedback. Refer to **fmax Descriptions** section.
Refer to **fmax Descriptions** section. Characterized but not 100% tested.

Capacitance (T_A = 25°C, f = 1.0 MHz)

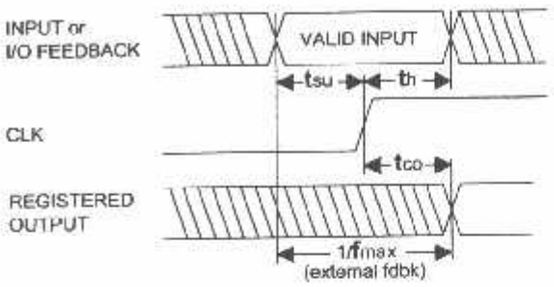
SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	8	pF	V_{CC} = 5.0V, V_I = 2.0V
$C_{I/O}$	I/O Capacitance	8	pF	V_{CC} = 5.0V, $V_{I/O}$ = 2.0V

*Characterized but not 100% tested.

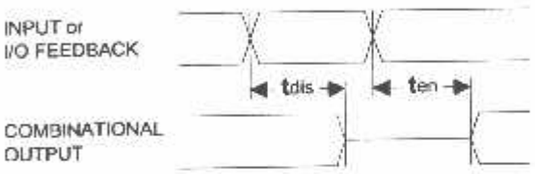
Switching Waveforms



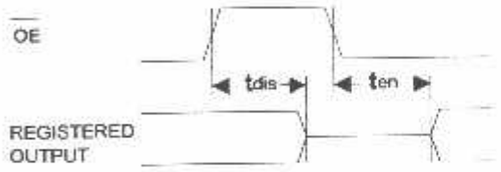
Combinatorial Output



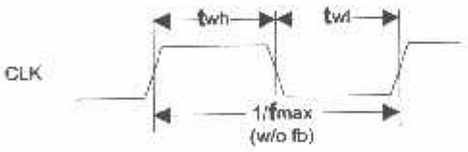
Registered Output



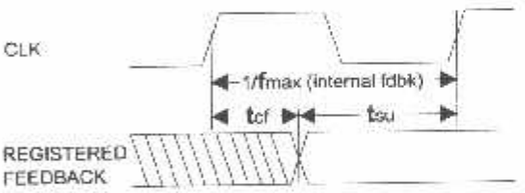
Input or I/O to Output Enable/Disable



$\overline{OE}$ to Output Enable/Disable

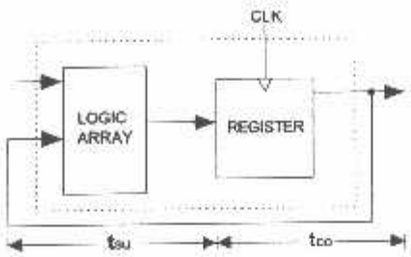


Clock Width



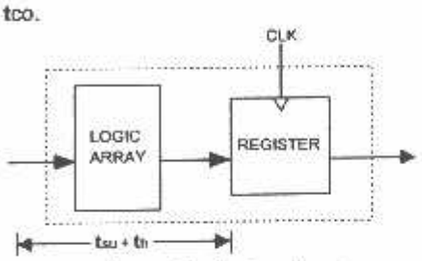
fmax with Feedback

fmax Descriptions



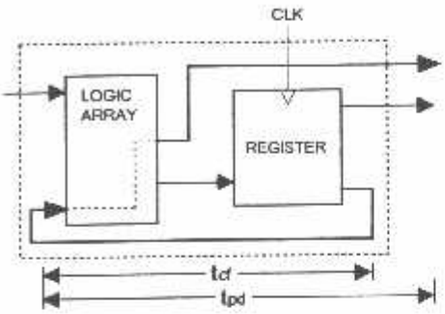
fmax with External Feedback 1/(tsu+tco)

ote: fmax with external feedback is calculated from measured :u and tco.



fmax with No Feedback

ote: fmax with no feedback may be less than 1/(twh + twl). This to allow for a clock duty cycle of other than 50%.



fmax with Internal Feedback 1/(tsu+tcf)

Note: tcf is a calculated value, derived by subtracting tsu from the period of fmax w/internal feedback (tcf = 1/fmax - tsu). The value of tcf is used primarily when calculating the delay from clocking a register to a combinatorial output (through registered feedback), as shown above. For example, the timing from clock to a combinatorial output is equal to tcf + tpd.

Switching Test Conditions

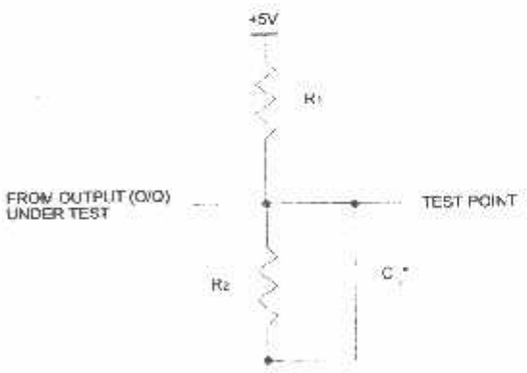
nput Pulse Levels		GND to 3.0V
nput Rise and Fall Times	GAL16V8D-10 (and slower)	2 – 3ns 10% – 90%
	GAL16V8D-3/-5/-7	1.5ns 10% – 90%
Input Timing Reference Levels		1.5V
Output Timing Reference Levels		1.5V
Output Load		See figure at right

Table 2-0003/16V8

state levels are measured 0.5V from eady-state active level.

AL16V8D (except -3) Output Load Conditions (see figure ove)

Test Condition		R_1	R_2	C_L
A		200Ω	390Ω	50pF
B	Active High	∞	390Ω	50pF
	Active Low	200Ω	390Ω	50pF
C	Active High	∞	390Ω	5pF
	Active Low	200Ω	390Ω	5pF

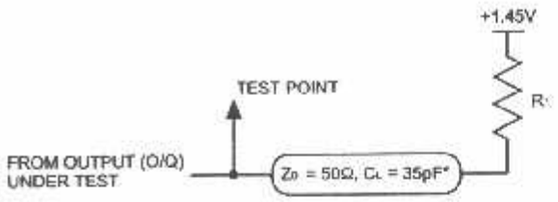


*C_L INCLUDES TEST FIXTURE AND PROBE CAPACITANCE

Switching Test Conditions (Continued)

GAL16V8D-3 Output Load Conditions (see figure at right)

Test Condition		R_1	C_L
A		50Ω	35pF
B	High Z to Active High at 1.9V	50Ω	35pF
	High Z to Active Low at 1.0V	50Ω	35pF
C	Active High to High Z at 1.9V	50Ω	35pF
	Active Low to High Z at 1.0V	50Ω	35pF



*C_L includes test fixture and probe capacitance.

Electronic Signature

n electronic signature is provided in every GAL16V8 device. It ontains 64 bits of reprogrammable memory that can contain user efined data. Some uses include user ID codes, revision numbers, inventory control. The signature data is always available to the ser independent of the state of the security cell.

OTE: The electronic signature is included in checksum calcula- ons. Changing the electronic signature will alter the checksum.

Security Cell

security cell is provided in the GAL16V8 devices to prevent un- thorized copying of the array patterns. Once programmed, this ll prevents further read access to the functional bits in the device. his cell can only be erased by re-programming the device, so the iginal configuration can never be examined once this cell is pro- ammed. The Electronic Signature is always available to the user, gardless of the state of this control cell.

Latch-Up Protection

AL16V8 devices are designed with an on-board charge pump negatively bias the substrate. The negative bias minimizes the tential of latch-up caused by negative input undershoots. Ad- ionally, outputs are designed with n-channel pull-ups instead of traditional p-channel pull-ups in order to eliminate latch-up due output overshoots.

Device Programming

AL devices are programmed using a Lattice Semiconductor- proved Logic Programmer, available from a number of manu- cturers. Complete programming of the device takes only a few conds. Erasing of the device is transparent to the user, and is ne automatically as part of the programming cycle.

Output Register Preload

When testing state machine designs, all possible states and state transitions must be verified in the design, not just those required in the normal machine operations. This is because, in system operation, certain events occur that may throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper treatment of these conditions, a way must be provided to break the feedback paths, and force any desired (i.e., illegal) state into the registers. Then the machine can be sequenced and the outputs tested for correct next state conditions.

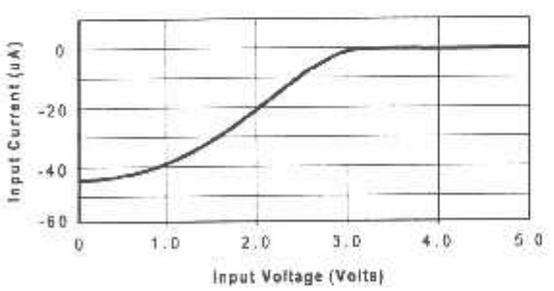
GAL16V8 devices include circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing text vectors perform output register preload automatically.

Input Buffers

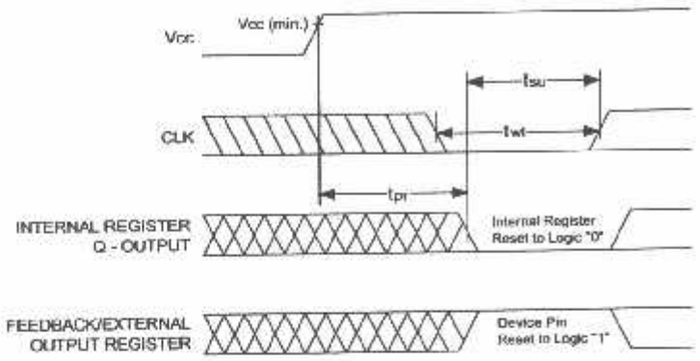
GAL16V8 devices are designed with TTL level compatible input buffers. These buffers have a characteristically high impedance, and present a much lighter load to the driving logic than bipolar TTL devices.

The GAL16V8 input and I/O pins have built-in active pull-ups. As a result, unused inputs and I/O's will float to a TTL "high" (logical "1"). Lattice Semiconductor recommends that all unused inputs and tri-stated I/O pins be connected to another active input, V_{CC}, or Ground. Doing this will tend to improve noise immunity and reduce I_{CC} for the device.

Typical Input Pull-up Characteristic

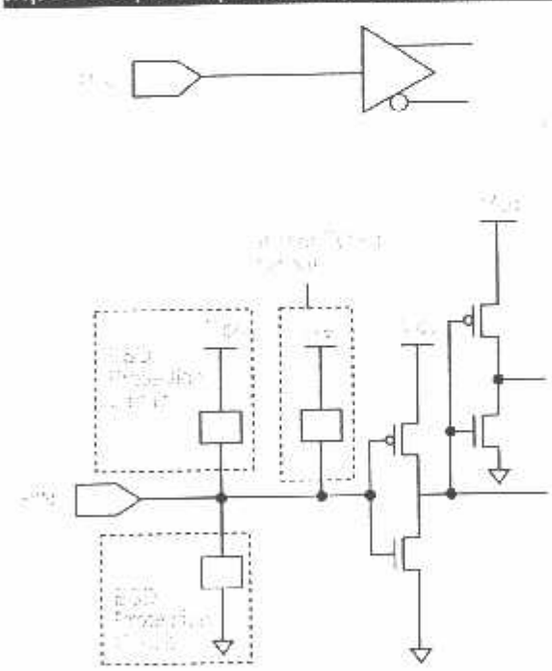


Power-Up Reset



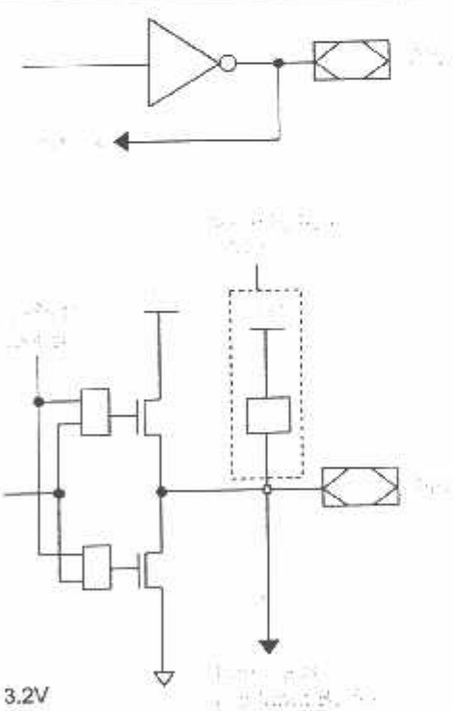
Circuitry within the GAL16V8 provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (tpr, 1µs MAX). As a result, the state on the registered output pins (if they are enabled) will always be high on power-up, regardless of the programmed polarity of the output pins. This feature can greatly simplify state machine design by providing a known state on power-up. Because of the asynchronous nature of system power-up, some conditions must be met to provide a valid power-up reset of the device. First, the Vcc rise must be monotonic. Second, the clock input must be at static TTL level as shown in the diagram during power up. The registers will reset within a maximum of tpr time. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met. The clock must also meet the minimum pulse width requirements.

Input/Output Equivalent Schematics



yp. Vref = 3.2V

Typical Input

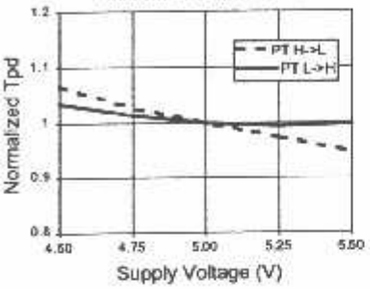


Typ. Vref = 3.2V

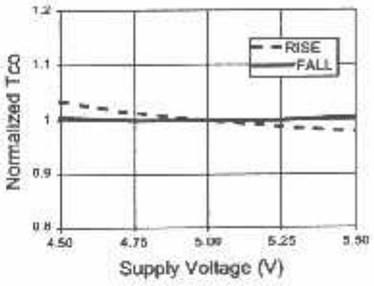
Typical Output

GAL16V8D-3/-5/-7 (IND PLCC): Typical AC and DC Characteristic Diagrams

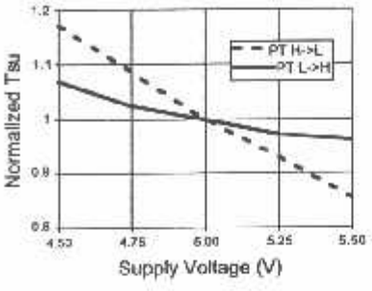
Normalized Tpd vs Vcc



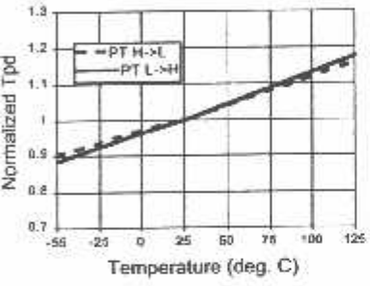
Normalized Tco vs Vcc



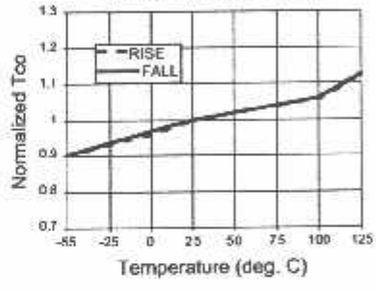
Normalized Tsu vs Vcc



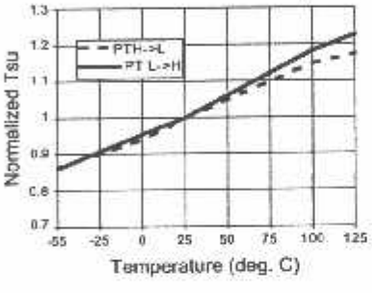
Normalized Tpd vs Temp



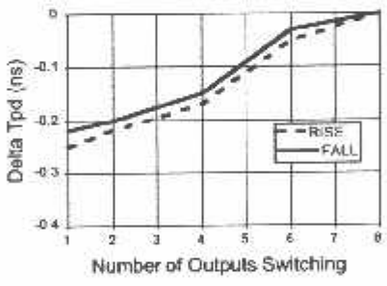
Normalized Tco vs Temp



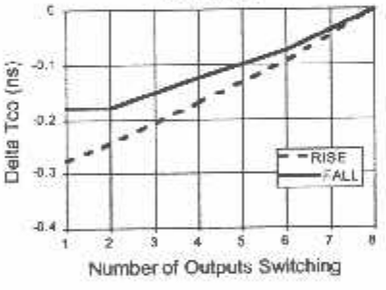
Normalized Tsu vs Temp



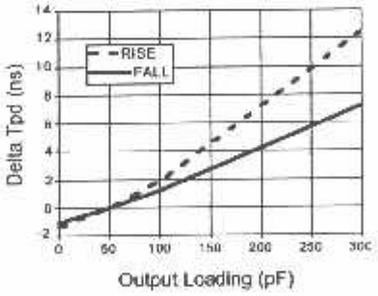
Delta Tpd vs # of Outputs Switching



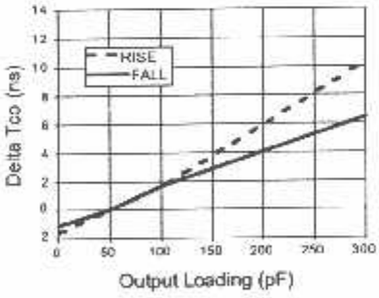
Delta Tco vs # of Outputs Switching



Delta Tpd vs Output Loading

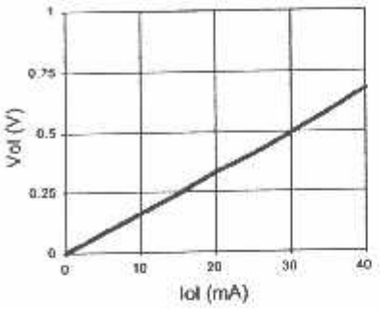


Delta Tco vs Output Loading

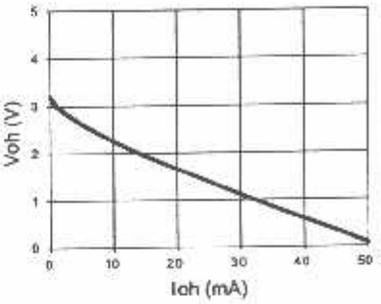


GAL16V8D-3/-5/-7 (IND PLCC): Typical AC and DC Characteristic Diagrams

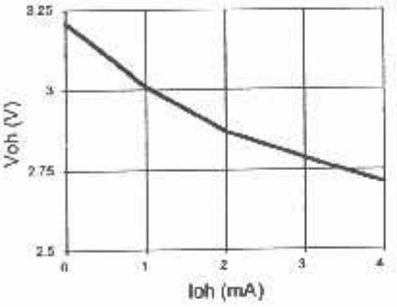
Vol vs Iol



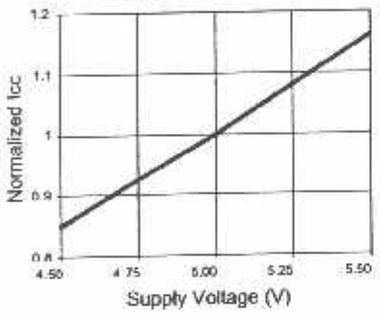
Voh vs Ioh



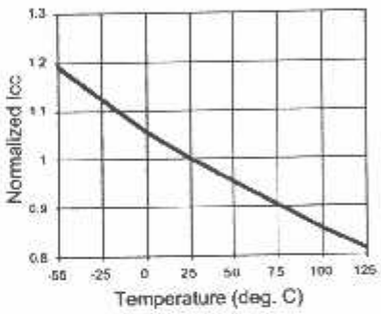
Voh vs Ioh



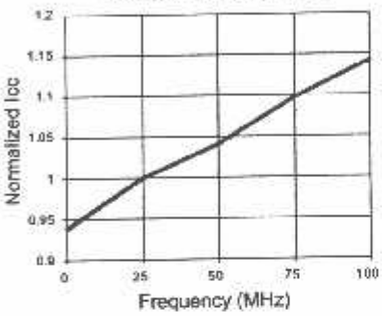
Normalized Icc vs Vcc



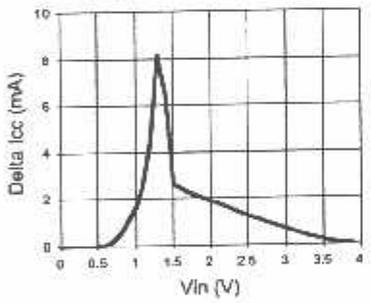
Normalized Icc vs Temp



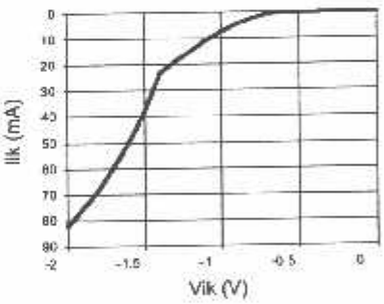
Normalized Icc vs Freq.



Delta Icc vs Vin (1 input)

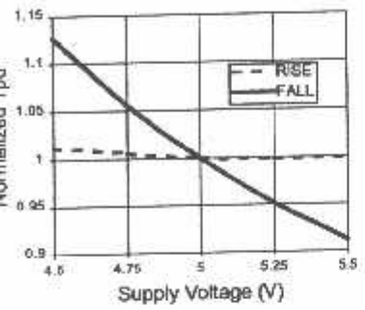


Input Clamp (Vik)

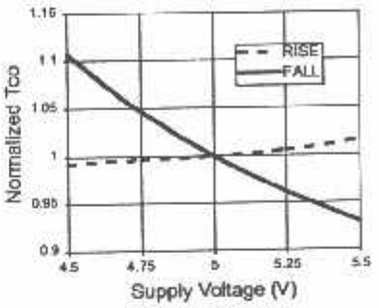


GAL16V8D-7 (Except IND PLCC)/-10L: Typical AC and DC Characteristic Diagrams

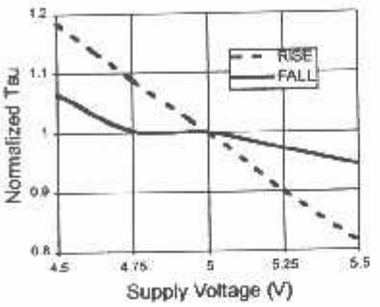
Normalized Tpd vs Vcc



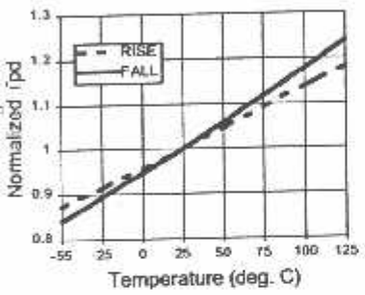
Normalized Tco vs Vcc



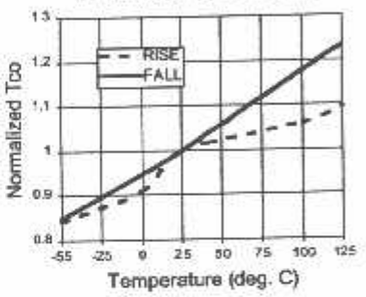
Normalized Tsu vs Vcc



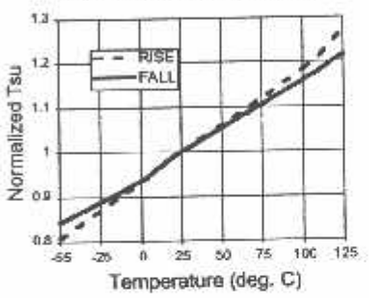
Normalized Tpd vs Temp



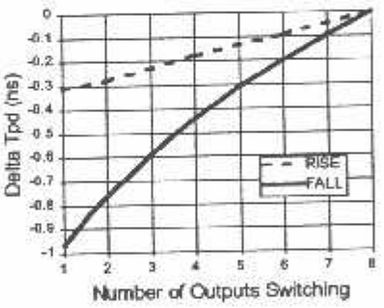
Normalized Tco vs Temp



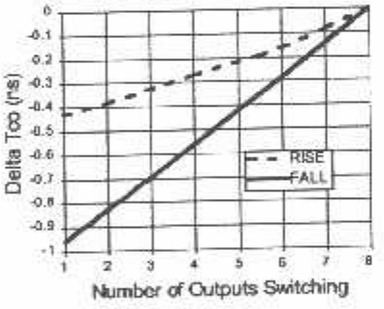
Normalized Tsu vs Temp



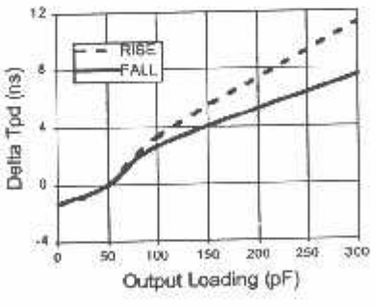
Delta Tpd vs # of Outputs Switching



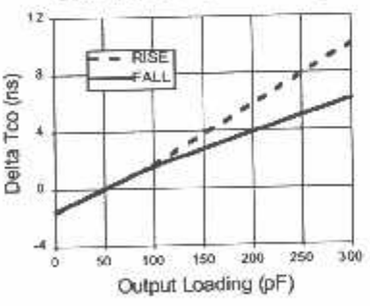
Delta Tco vs # of Outputs Switching



Delta Tpd vs Output Loading

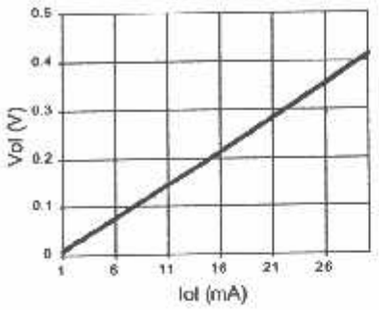


Delta Tco vs Output Loading

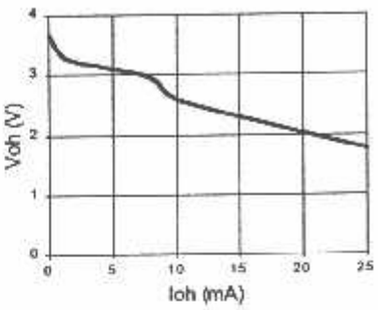


GAL16V8D-7 (Except IND PLCC)/-10L: Typical AC and DC Characteristic Diagrams

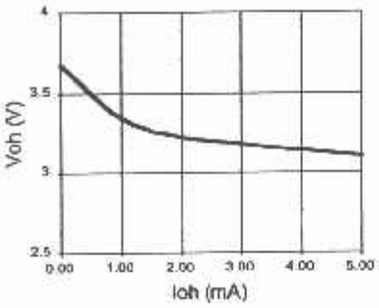
Vol vs Iol



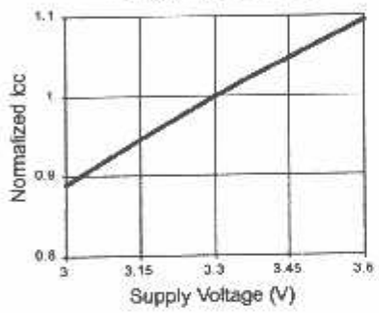
Voh vs Ioh



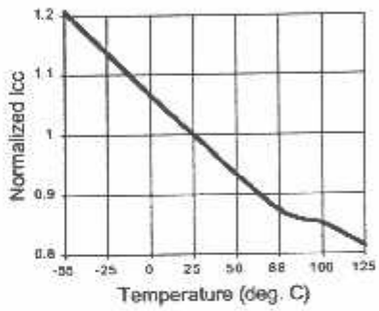
Voh vs Ioh



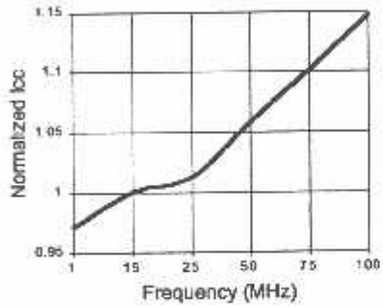
Normalized Icc vs Vcc



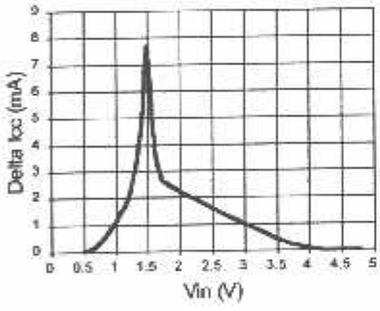
Normalized Icc vs Temp



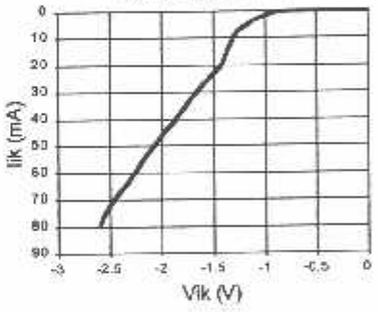
Normalized Icc vs Freq



Delta Icc vs Vin (1 Input)

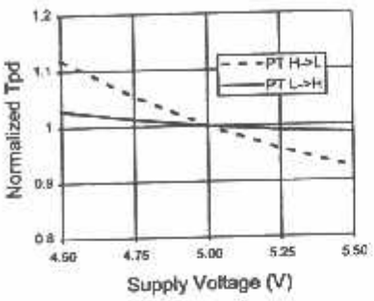


Input Clamp (Vik)

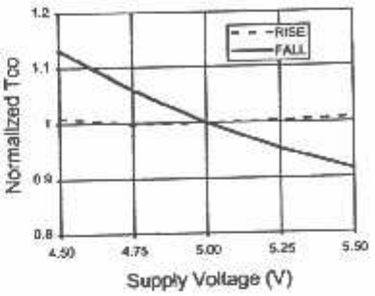


GAL16V8D-10Q (and Slower): Typical AC and DC Characteristic Diagrams

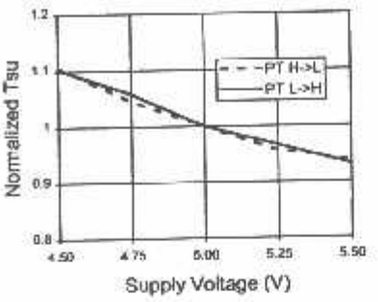
Normalized Tpd vs Vcc



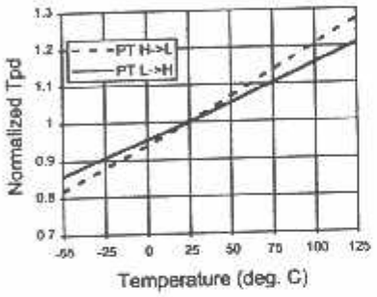
Normalized Tco vs Vcc



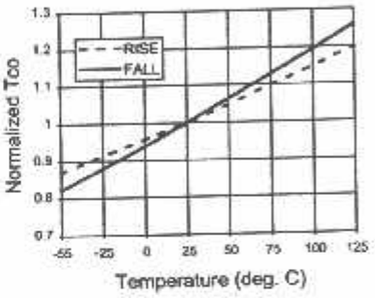
Normalized Tsu vs Vcc



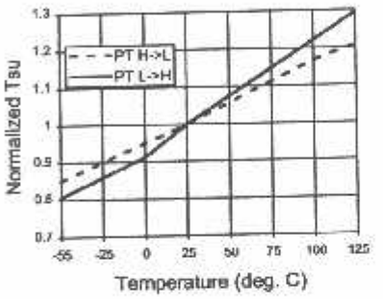
Normalized Tpd vs Temp



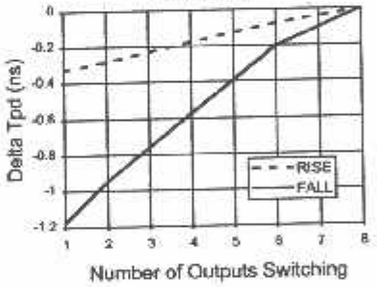
Normalized Tco vs Temp



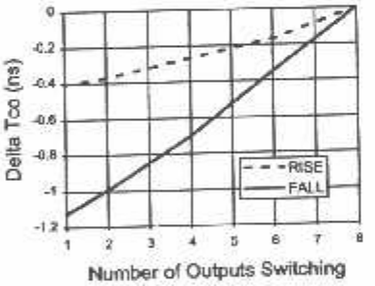
Normalized Tsu vs Temp



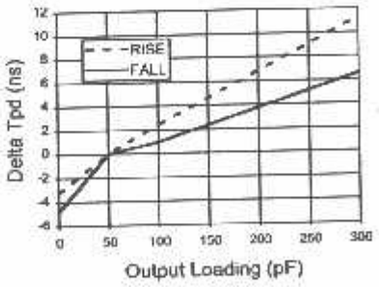
Delta Tpd vs # of Outputs Switching



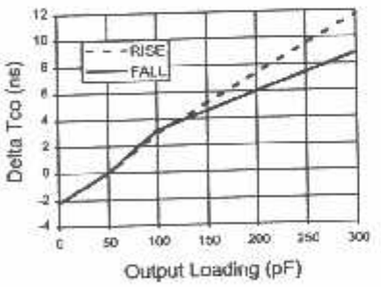
Delta Tco vs # of Outputs Switching



Delta Tpd vs Output Loading

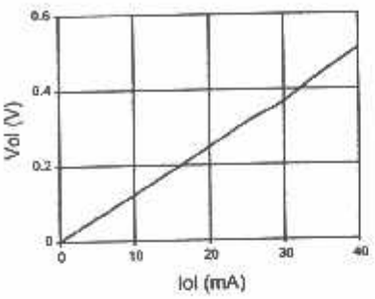


Delta Tco vs Output Loading

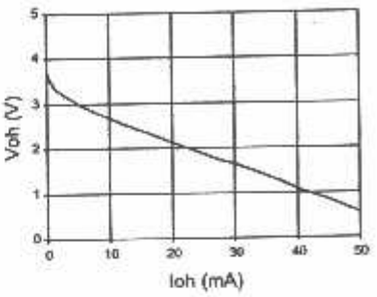


GAL16V8D-10Q (and Slower): Typical AC and DC Characteristic Diagrams

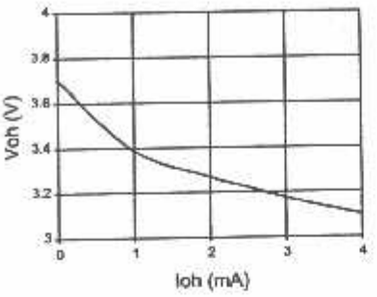
Vol vs Iol



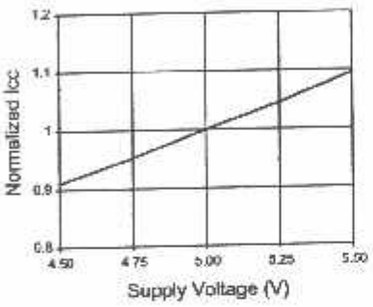
Voh vs Ioh



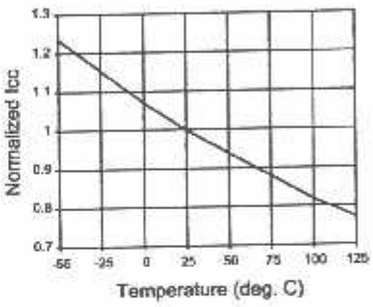
Voh vs Ioh



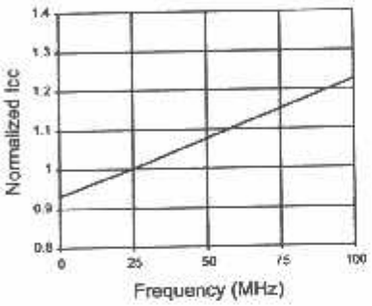
Normalized Icc vs Vcc



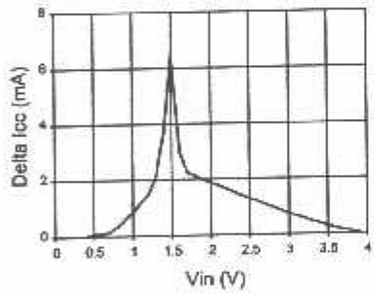
Normalized Icc vs Temp



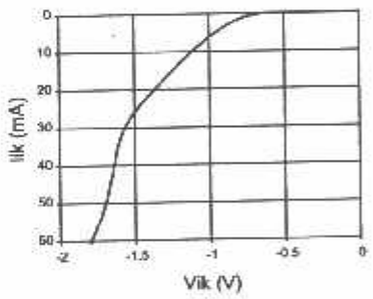
Normalized Icc vs Freq.



Delta Icc vs Vin (1 input)



Input Clamp (Vik)



GAL22V10

High Performance E²CMOS PLD
Generic Array Logic™

Features

- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - 4 ns Maximum Propagation Delay
  - Fmax = 250 MHz
  - 3.5 ns Maximum from Clock Input to Data Output
  - UltraMOS® Advanced CMOS Technology
- ACTIVE PULL-UPS ON ALL PINS
- COMPATIBLE WITH STANDARD 22V10 DEVICES
  - Fully Function/Fuse-Map/Parametric Compatible with Bipolar and UVCMOS 22V10 Devices
- 50% to 75% REDUCTION IN POWER VERSUS BIPOLAR
  - 90mA Typical Icc on Low Power Device
  - 45mA Typical Icc on Quarter Power Device
- E² CELL TECHNOLOGY
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- TEN OUTPUT LOGIC MACROCELLS
  - Maximum Flexibility for Complex Logic Designs
- PRELOAD AND POWER-ON RESET OF REGISTERS
  - 100% Functional Testability
- APPLICATIONS INCLUDE:
  - DMA Control
  - State Machine Control
  - High Speed Graphics Processing
  - Standard Logic Speed Upgrade
- ELECTRONIC SIGNATURE FOR IDENTIFICATION
- LEAD-FREE PACKAGE OPTIONS

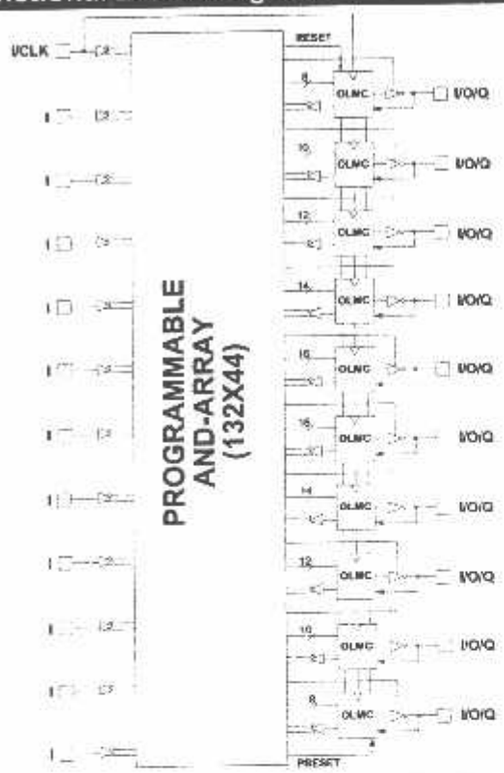
Description

The GAL22V10, at 4ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E^2) floating gate technology to provide the highest performance available of any 22V10 device on the market. CMOS circuitry allows the GAL22V10 to consume much less power when compared to bipolar 22V10 devices. E^2 technology offers high speed (<100ms) erase times, providing the ability to reprogram or reconfigure the device quickly and efficiently.

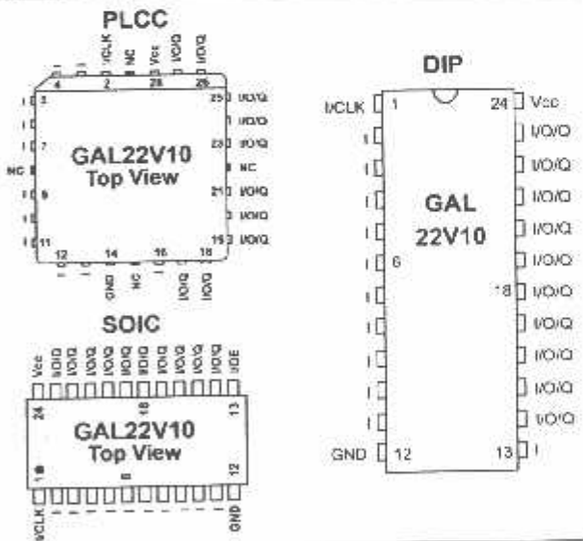
The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. The GAL22V10 is fully function/fuse map/parametric compatible with standard bipolar and CMOS 22V10 devices.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Functional Block Diagram



Pin Configuration



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A
Tel. (503) 268-8000; 1-800-LATTICE; FAX (503) 268-8556; http://www.latticesemi.com

August 2004

GAL22V10 Ordering Information

Conventional Packaging
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
4	2.5	3.5	140	GAL22V10D-4LJ	28-Lead PLCC
5	3	4	140	GAL22V10D-5LJ	28-Lead PLCC
7.5	4.5	4.5	140	GAL22V10D-7LP	24 Pin Plastic D P
	4.5	4.5	140	GAL22V10D-7LJ	28-Lead PLCC
10	7	7	55	GAL22V10D-10QP	24-Pin Plastic DIP
			55	GAL22V10D-10QJ	28-Lead PLCC
			130	GAL22V10D-10LP	24-Pin Plastic DIP
			130	GAL22V10D-10LJ	28-Lead PLCC
			130	GAL22V10D-10LS	24-Pin SOIC
15	10	8	55	GAL22V10D-15QP	24-Pin Plastic DIP
			55	GAL22V10D-15QJ	28-Lead PLCC
			130	GAL22V10D-15LP	24 Pin Plastic DIP
			130	GAL22V10D-15LJ	28-Lead PLCC
			130	GAL22V10D-15LS	24-Pin SOIC
25	15	15	55	GAL22V10D-25QP	24-Pin Plastic DIP
			55	GAL22V10D-25QJ	28-Lead PLCC
			90	GAL22V10D-25LP	24-Pin Plastic DIP
			90	GAL22V10D-25LJ	28-Lead PLCC
			90	GAL22V10D-25LS	24-Pin SOIC

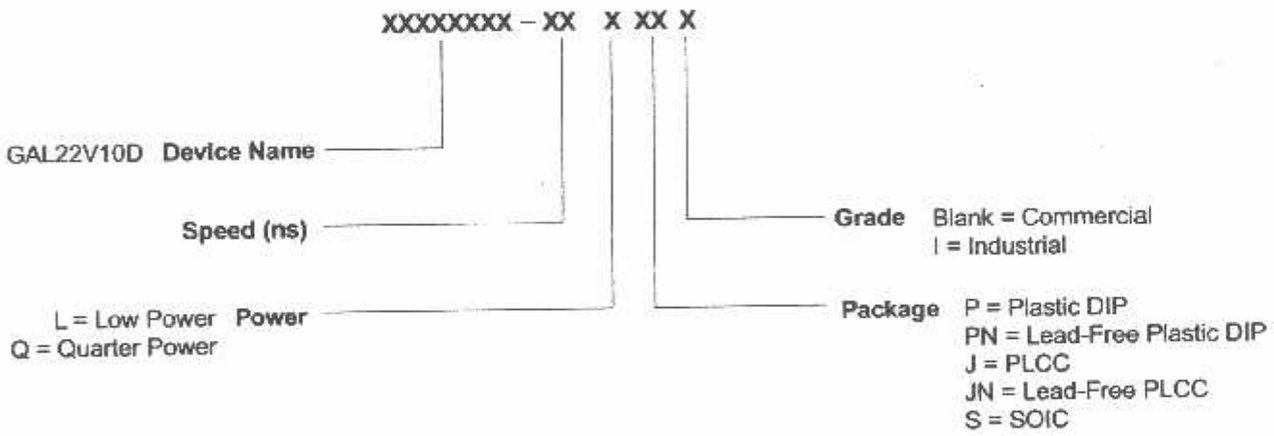
Industrial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
7.5	5	4.5	160	GAL22V10D-7LPI	24-Pin Plastic DIP
	4.5	4.5	160	GAL22V10D-7LJI	28-Lead PLCC
10	7	7	150	GAL22V10D-10LPI	24 Pin Plastic D P
			150	GAL22V10D-10LJI	28-Lead PLCC
15	10	8	150	GAL22V10D-15LPI	24-Pin Plastic DIP
			150	GAL22V10D-15LJI	28-Lead PLCC
20	14	10	150	GAL22V10D-20LPI	24 Pin Plastic D P
			150	GAL22V10D-20LJI	28-Lead PLCC
25	15	15	150	GAL22V10D-25LPI	24-Pin Plastic DIP
			150	GAL22V10D-25LJI	28-Lead PLCC

Lead-Free Packaging
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
4	2.5	3.5	140	GAL22V10D-4LJN	Lead-Free 28-Lead PLCC
5	3	4	140	GAL22V10D-5LJN	Lead-Free 28-Lead PLCC
7.5	4.5	4.5	140	GAL22V10D-7LPN	Lead-Free 24-Pin Plastic DIP
	4.5	4.5	140	GAL22V10D-7LJN	Lead-Free 28-Lead PLCC
10	7	7	55	GAL22V10D-10QPN	Lead-Free 24-Pin Plastic DIP
			55	GAL22V10D-10QJN	Lead-Free 28-Lead PLCC
			130	GAL22V10D-10LPN	Lead-Free 24-Pin Plastic DIP
			130	GAL22V10D-10LJN	Lead-Free 28-Lead PLCC
15	10	8	55	GAL22V10D-15QPN	Lead-Free 24-Pin Plastic DIP
			55	GAL22V10D-15QJN	Lead-Free 28-Lead PLCC
			130	GAL22V10D-15LPN	Lead-Free 24-Pin Plastic DIP
			130	GAL22V10D-15LJN	Lead-Free 28-Lead PLCC
25	15	15	55	GAL22V10D-25QPN	Lead-Free 24-Pin Plastic DIP
			55	GAL22V10D-25QJN	Lead-Free 28-Lead PLCC
			90	GAL22V10D-25LPN	Lead-Free 24-Pin Plastic Dip
			90	GAL22V10D-25LJN	Lead-Free 28-Lead PLCC

Part Number Description



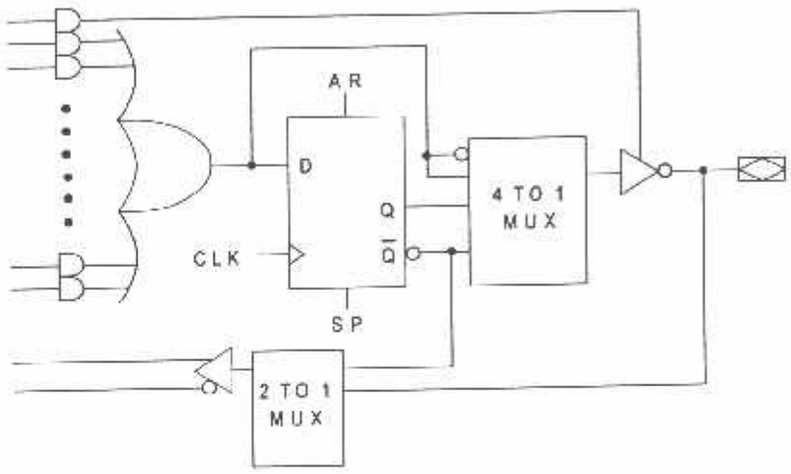
Output Logic Macrocell (OLMC)

The GAL22V10 has a variable number of product terms per OLMC. Of the ten available OLMCs, two OLMCs have access to eight product terms (pins 14 and 23, DIP pinout), two have ten product terms (pins 15 and 22), two have twelve product terms (pins 16 and 21), two have fourteen product terms (pins 17 and 20), and two OLMCs have sixteen product terms (pins 18 and 19). In addition to the product terms available for logic, each OLMC has an additional product-term dedicated to output enable control.

The output polarity of each OLMC can be individually programmed to be true or inverting, in either combinatorial or registered mode. This allows each output to be individually configured as either active high or active low.

The GAL22V10 has a product term for Asynchronous Reset (AR) and a product term for Synchronous Preset (SP). These two product terms are common to all registered OLMCs. The Asynchronous Reset sets all registers to zero any time this dedicated product term is asserted. The Synchronous Preset sets all registers to a logic one on the rising edge of the next clock pulse after this product term is asserted.

NOTE: The AR and SP product terms will force the Q output of the flip-flop into the same state regardless of the polarity of the output. Therefore, a reset operation, which sets the register output to a zero, may result in either a high or low at the output pin, depending on the pin polarity chosen.



GAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

Output Logic Macrocell Configurations

Each of the Macrocells of the GAL22V10 has two primary functional modes: registered, and combinatorial I/O. The modes and the output polarity are set by two bits (S0 and S1), which are normally controlled by the logic compiler. Each of these two primary modes, and the bit settings required to enable them, are described below and on the following page.

REGISTERED

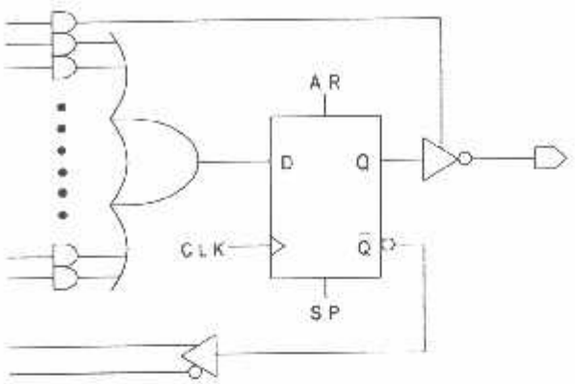
In registered mode the output pin associated with an individual OLMC is driven by the Q output of that OLMC's D-type flip-flop. Logic polarity of the output signal at the pin may be selected by specifying that the output buffer drive either true (active high) or inverted (active low). Output tri-state control is available as an individual product-term for each OLMC, and can therefore be defined by a logic equation. The D flip-flop's /Q output is fed back into the AND array, with both the true and complement of the feedback available as inputs to the AND array.

NOTE: In registered mode, the feedback is from the /Q output of the register, and not from the pin; therefore, a pin defined as registered is an output only, and cannot be used for dynamic I/O, as can the combinatorial pins.

COMBINATORIAL I/O

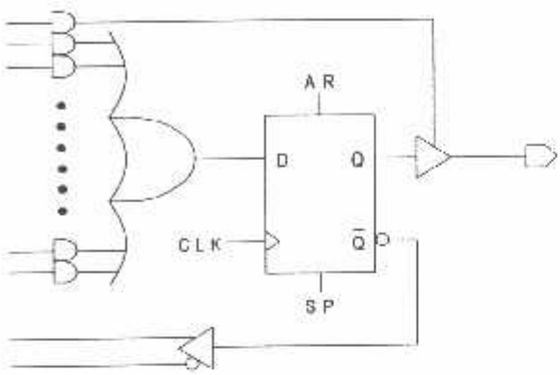
In combinatorial mode the pin associated with an individual OLMC is driven by the output of the sum term gate. Logic polarity of the output signal at the pin may be selected by specifying that the output buffer drive either true (active high) or inverted (active low). Output tri-state control is available as an individual product-term for each output, and may be individually set by the compiler as either "on" (dedicated output), "off" (dedicated input), or "product-term driven" (dynamic I/O). Feedback into the AND array is from the pin side of the output enable buffer. Both polarities (true and inverted) of the pin are fed back into the AND array.

Registered Mode



ACTIVE LOW

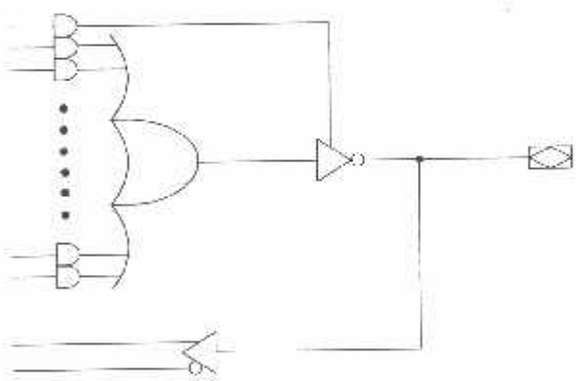
$S_0 = 0$
$S_1 = 0$



ACTIVE HIGH

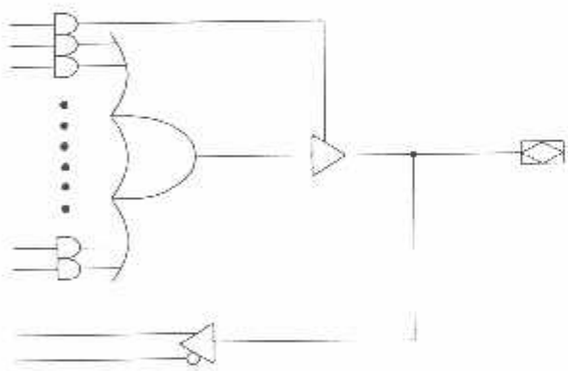
$S_0 = 1$
$S_1 = 0$

Combinatorial Mode



ACTIVE LOW

$S_0 = 0$
$S_1 = 1$

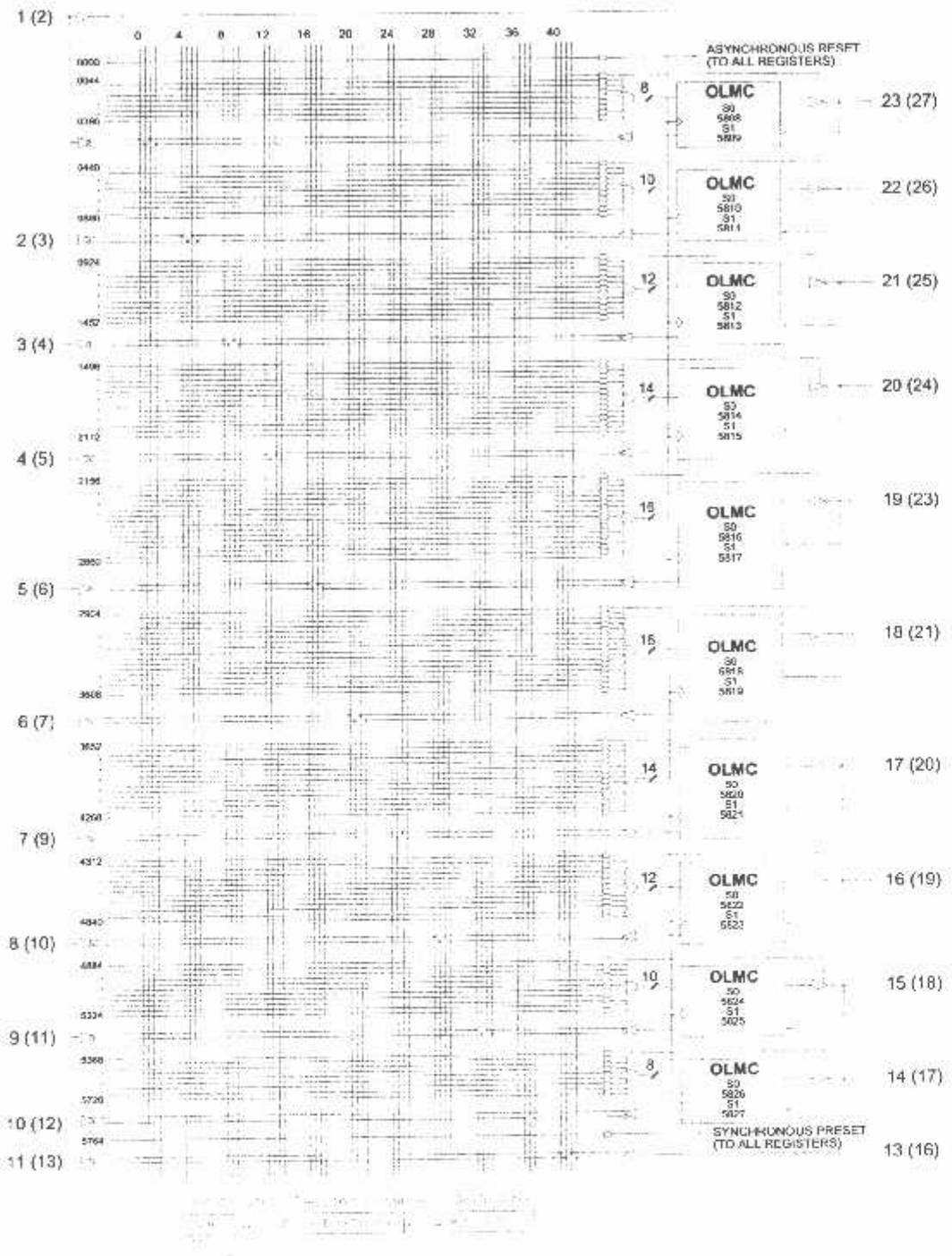


ACTIVE HIGH

$S_0 = 1$
$S_1 = 1$

GAL22V10 Logic Diagram / JEDEC Fuse Map

DIP (PLCC) Package Pinouts



Absolute Maximum Ratings[1]

Supply voltage V_{CC} ..... -0.5 to +7V
Input voltage applied ..... -2.5 to V_{CC} +1.0V
Off-state output voltage applied ..... -2.5 to V_{CC} +1.0V
Storage Temperature ..... -65 to 150°C
Ambient Temperature with
Power Applied ..... -55 to 125°C

1. Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied (while programming, follow the programming specifications).

Recommended Operating Conditions

Commercial Devices:
Ambient Temperature (T_A) ..... 0 to +75°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.75 to +5.25V

Industrial Devices:
Ambient Temperature (T_A) ..... -40 to 85°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.50 to +5.50V

DC Electrical Characteristics

Over Recommended Operating Conditions (Unless Otherwise Specified)

SYMBOL	PARAMETER	CONDITION		MIN.	TYP.[3]	MAX.	UNITS
V_{IL}	Input Low Voltage			Vss – 0.5	—	0.8	V
V_{IH}	Input High Voltage			2.0	—	Vcc+1	V
I_{IL}[1]	Input or I/O Low Leakage Current	$0V \le V_{IN} \le V_{IL}$ (MAX.)		—	—	–100	µA
I_{IH}	Input or I/O High Leakage Current	$3.5V \le V_{IN} \le V_{CC}$		—	—	10	µA
V_{OL}	Output Low Voltage	I_{OL} = MAX. $V_{IN} = V_{IL}$ or V_{IH}		—	—	0.4	V
V_{OH}	Output High Voltage	I_{OH} = MAX. $V_{IN} = V_{IL}$ or V_{IH}		2.4	—	–	V
I_{OL}	Low Level Output Current			—	—	16	mA
I_{OH}	High Level Output Current			—	—	–3.2	mA
I_{OS}[2]	Output Short Circuit Current	V_{CC} = 5V V_{OUT} = 0.5V T_A = 25°C		–30	—	–130	mA
COMMERCIAL							
I_{CC}	Operating Power Supply Current	V_{IL} = 0.5V V_{IH} = 3.0V f_{toggle} = 15MHz Outputs Open	L-4/-5/-7	—	90	140	mA
			L-10	—	90	130	mA
			L-15/-25	—	75	90	mA
			Q-10/-15/-25	—	45	55	mA
INDUSTRIAL							
I_{CC}	Operating Power Supply Current	V_{IL} = 0.5V V_{IH} = 3.0V f_{toggle} = 15MHz Outputs Open	L-7/-10	—	90	160	mA
			L-15/-20/-25	—	75	130	mA

1) The leakage current is due to the internal pull-up on all pins. See **Input Buffer** section for more information.
2) One output at a time for a maximum duration of one second. Vout = 0.5V was selected to avoid test problems caused by tester ground degradation. Characterized but not 100% tested.
3) Typical values are at Vcc = 5V and T_A = 25 °C

AC Switching Characteristics

Over Recommended Operating Conditions

PARAM	TEST COND.[1]	DESCRIPTION	COM -4		COM -5		COM/IND -7		UNITS
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{pd}	A	Input or I/O to Combinatorial Output	1	4	1	5	1	7.5	ns
t_{co}	A	Clock to Output Delay	1	3.5	1	4	1	4.5	ns
t_{cf}[2]	—	Clock to Feedback Delay	—	2.5	—	3	—	3	ns
t_{su}	—	Setup Time, Input or Fdbk before Clk↑	2.5	—	3	—	4.5	—	ns
t_h	—	Hold Time, Input or Fdbk after Clk↑	0	—	0	—	0	—	ns
f_{max}[3]	A	Maximum Clock Frequency with External Feedback, 1/(tsu + tco)	167	—	142.8	—	111	—	MHz
	A	Maximum Clock Frequency with Internal Feedback, 1/(tsu + tcf)	200	—	166	—	133	—	MHz
	A	Maximum Clock Frequency with No Feedback	250	—	200	—	166	—	MHz
t_{wh}	—	Clock Pulse Duration, High	2	—	2.5	—	3	—	ns
t_{wl}	—	Clock Pulse Duration, Low	2	—	2.5	—	3	—	ns
t_{en}	B	Input or I/O to Output Enabled	1	5	1	6	1	7.5	ns
t_{dis}	C	Input or I/O to Output Disabled	1	5	1	5.5	1	7.5	ns
t_{ar}	A	Input or I/O to Asynch. Reset of Reg.	1	4.5	1	5.5	1	9	ns
t_{arw}	—	Asynch. Reset Pulse Duration	4.5	—	4.5	—	7	—	ns
t_{arr}	—	Asynch. Reset to Clk↑ Recovery Time	3	—	4	—	5	—	ns
t_{spr}	—	Synch. Preset to Clk↑ Recovery Time	3	—	4	—	5	—	ns

) Refer to **Switching Test Conditions** section.
) Calculated from fmax with internal feedback. Refer to **fmax Description** section.
) Refer to **fmax Description** section. Characterized initially and after any design or process changes that may affect these parameters.

Capacitance (T_A = 25°C, f = 1.0 MHz)

SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	8	pF	V_{CC} = 5.0V, V_I = 2.0V
C_{IO}	I/O Capacitance	8	pF	V_{CC} = 5.0V, V_{IO} = 2.0V

Characterized but not 100% tested.

AC Switching Characteristics

Over Recommended Operating Conditions

PARAM.	TEST COND.[1]	DESCRIPTION	COM / IND -10		COM / IND -15		IND -20		COM / IND -25		UNITS
			MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{pd}	A	Input or I/O to Comb. Output	1	10	3	15	3	20	3	25	ns
t_{co}	A	Clock to Output Delay	1	7	2	8	2	10	2	15	ns
t_{cf}[2]	—	Clock to Feedback Delay	—	2.5	—	2.5	—	8	—	13	ns
t_{su}	—	Setup Time, Input or Fdbk before Clk↑	6	—	10	—	12	—	15	—	ns
t_h	—	Hold Time, Input or Fdbk after Clk↑	0	—	0	—	0	—	0	—	ns
f_{max}[3]	A	Maximum Clock Frequency with External Feedback, 1/(tsu + tco)	83.3	—	55.5	—	41.6	—	33.3	—	MHz
	A	Maximum Clock Frequency with Internal Feedback, 1/(tsu + tcf)	110	—	80	—	45.4	—	35.7	—	MHz
	A	Maximum Clock Frequency with No Feedback	125	—	83.3	—	50	—	38.5	—	MHz
t_{wh}	—	Clock Pulse Duration, High	4	—	6	—	10	—	13	—	ns
t_{wl}	—	Clock Pulse Duration, Low	4	—	6	—	10	—	13	—	ns
t_{en}	B	Input or I/O to Output Enabled	1	10	3	15	3	20	3	25	ns
t_{dis}	C	Input or I/O to Output Disabled	1	9	3	15	3	20	3	25	ns
t_{ar}	A	Input or I/O to Asynch. Reset of Reg.	1	13	3	20	3	25	3	25	ns
t_{arw}	—	Asynch. Reset Pulse Duration	8	—	15	—	20	—	25	—	ns
t_{arr}	—	Asynch. Reset to Clk↑ Recovery Time	8	—	10	—	20	—	25	—	ns
t_{spr}	—	Synch. Preset to Clk↑ Recovery Time	8	—	10	—	14	—	15	—	ns

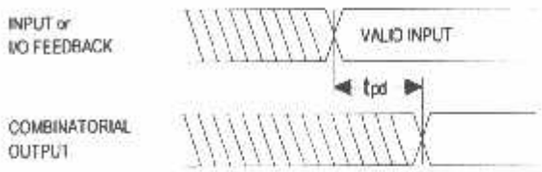
1) Refer to **Switching Test Conditions** section.
2) Calculated from fmax with internal feedback. Refer to **fmax Description** section.
3) Refer to **fmax Description** section.

Capacitance (T_A = 25°C, f = 1.0 MHz)

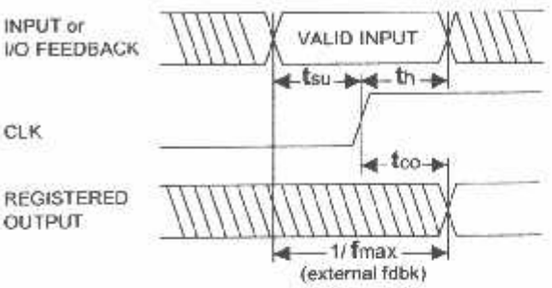
SYMBOL	PARAMETER	MAXIMUM*	UNITS	TEST CONDITIONS
C_I	Input Capacitance	8	pF	V_{CC} = 5.0V, V_I = 2.0V
$C_{I/O}$	I/O Capacitance	8	pF	V_{CC} = 5.0V, $V_{I/O}$ = 2.0V

*Characterized but not 100% tested.

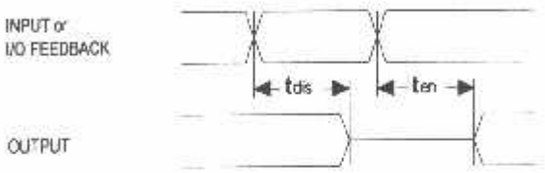
Switching Waveforms



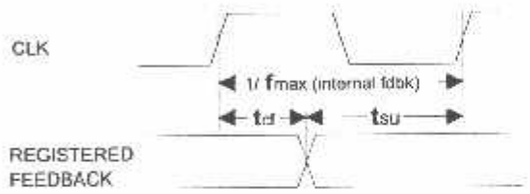
Combinatorial Output



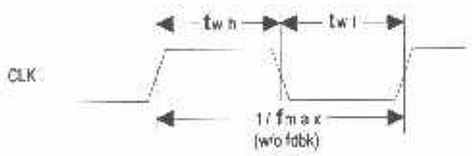
Registered Output



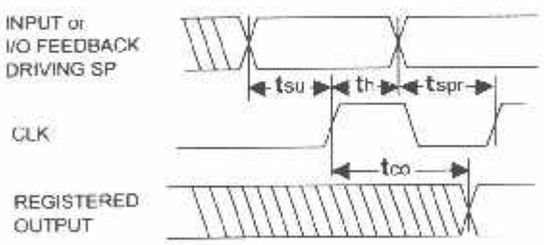
Input or I/O to Output Enable/Disable



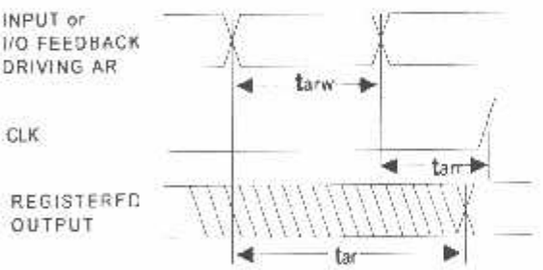
fmax with Feedback



Clock Width

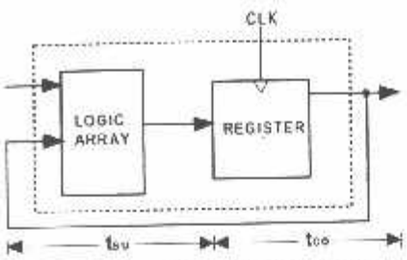


Synchronous Preset



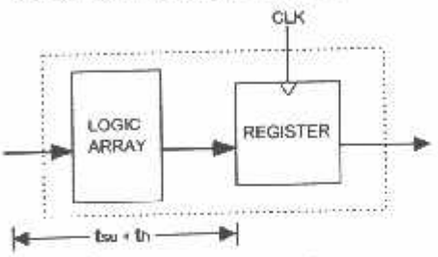
Asynchronous Reset

fmax Descriptions



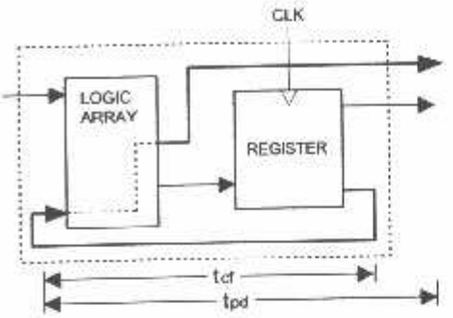
fmax with External Feedback 1/(tsu+tco)

Note: fmax with external feedback is calculated from measured tsu and tco.



fmax with No Feedback

Note: fmax with no feedback may be less than 1/(twh + twl). This is to allow for a clock duty cycle of other than 50%.



fmax with Internal Feedback 1/(tsu+tcf)

Note: tcf is a calculated value, derived by subtracting tsu from the period of fmax w/internal feedback (tcf = 1/fmax - tsu). The value of tcf is used primarily when calculating the delay from clocking a register to a combinatorial output (through registered feedback), as shown above. For example, the timing from clock to a combinatorial output is equal to tcf + tpd.

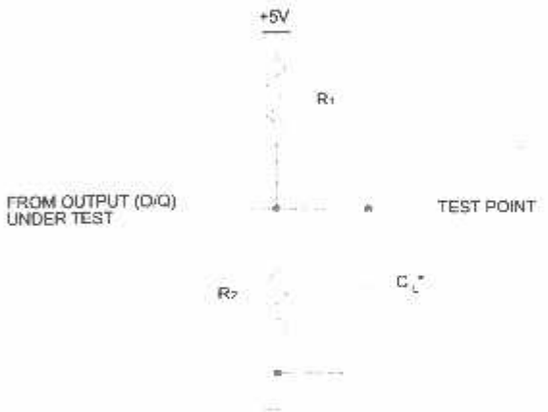
Switching Test Conditions

Input Pulse Levels		GND to 3.0V
Input Rise and Fall Times	D-4/-5/-7	1.5ns 10% – 90%
	D-10/-15/-20/-25	2.0ns 10% – 90%
Input Timing Reference Levels		1.5V
Output Timing Reference Levels		1.5V
Output Load		See Figure

3-state levels are measured 0.5V from steady-state active evel.

Output Load Conditions (except D-4) (see figure below)

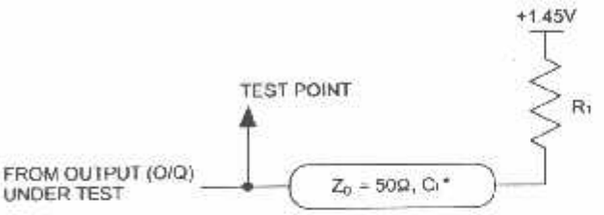
Test Condition		R_1	R_2	C_L
A		300Ω	390Ω	50pF
B	Active High	∞	390Ω	50pF
	Active Low	300Ω	390Ω	50pF
C	Active High	∞	390Ω	5pF
	Active Low	300Ω	390Ω	5pF



*C_L INCLUDES TEST FIXTURE AND PROBE CAPACITANCE

GAL22V10D-4 Output Load Conditions (see figure below)

Test Condition		R_1	C_L
A		50Ω	50pF
B	Z to Active High at 1.9V	50Ω	50pF
	Z to Active Low at 1.0V	50Ω	50pF
C	Active High to Z at 1.9V	50Ω	50pF
	Active Low to Z at 1.0V	50Ω	50pF



Electronic Signature

An electronic signature (ES) is provided in every GAL22V10 device. It contains 64 bits of reprogrammable memory that can contain user-defined data. Some uses include user ID codes, revision numbers, or inventory control. The signature data is always available to the user independent of the state of the security cell.

The electronic signature is an additional feature not present in other manufacturers' 22V10 devices. To use the extra feature of the user-programmable electronic signature it is necessary to choose a Lattice Semiconductor 22V10 device type when compiling a set of logic equations. In addition, many device programmers have two separate selections for the device, typically a GAL22V10 and a GAL22V10-UES (UES = User Electronic Signature) or GAL22V10-ES. This allows users to maintain compatibility with existing 22V10 designs, while still having the option to use the GAL device's extra feature.

The JEDEC map for the GAL22V10 contains the 64 extra fuses for the electronic signature, for a total of 5892 fuses. However, the GAL22V10 device can still be programmed with a standard 22V10 JEDEC map (5828 fuses) with any qualified device programmer.

Security Cell

A security cell is provided in every GAL22V10 device to prevent unauthorized copying of the array patterns. Once programmed, this cell prevents further read access to the functional bits in the device. This cell can only be erased by re-programming the device, so the original configuration can never be examined once this cell is programmed. The Electronic Signature is always available to the user, regardless of the state of this control cell.

Latch-Up Protection

GAL22V10 devices are designed with an on-board charge pump to negatively bias the substrate. The negative bias is of sufficient magnitude to prevent input undershoots from causing the circuitry to latch. Additionally, outputs are designed with n-channel pullups instead of the traditional p-channel pullups to eliminate any possibility of SCR induced latching.

Device Programming

GAL devices are programmed using a Lattice Semiconductor-approved Logic Programmer, available from a number of manufacturers (see the the GAL Development Tools section). Complete programming of the device takes only a few seconds. Erasing of the device is transparent to the user, and is done automatically as part of the programming cycle.

Output Register Preload

When testing state machine designs, all possible states and state transitions must be verified in the design, not just those required in the normal machine operations. This is because certain events may occur during system operation that throw the logic into an illegal state (power-up, line voltage glitches, brown-outs, etc.). To test a design for proper treatment of these conditions, a way must be provided to break the feedback paths, and force any desired (i.e., illegal) state into the registers. Then the machine can be sequenced and the outputs tested for correct next state conditions.

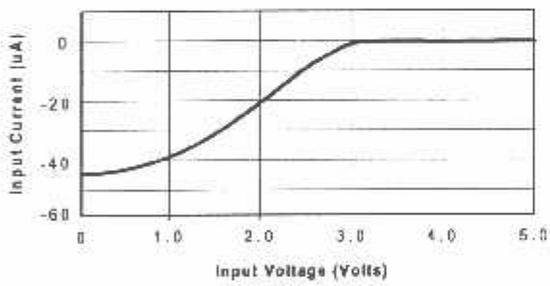
The GAL22V10 device includes circuitry that allows each registered output to be synchronously set either high or low. Thus, any present state condition can be forced for test sequencing. If necessary, approved GAL programmers capable of executing test vectors perform output register preload automatically.

Input Buffers

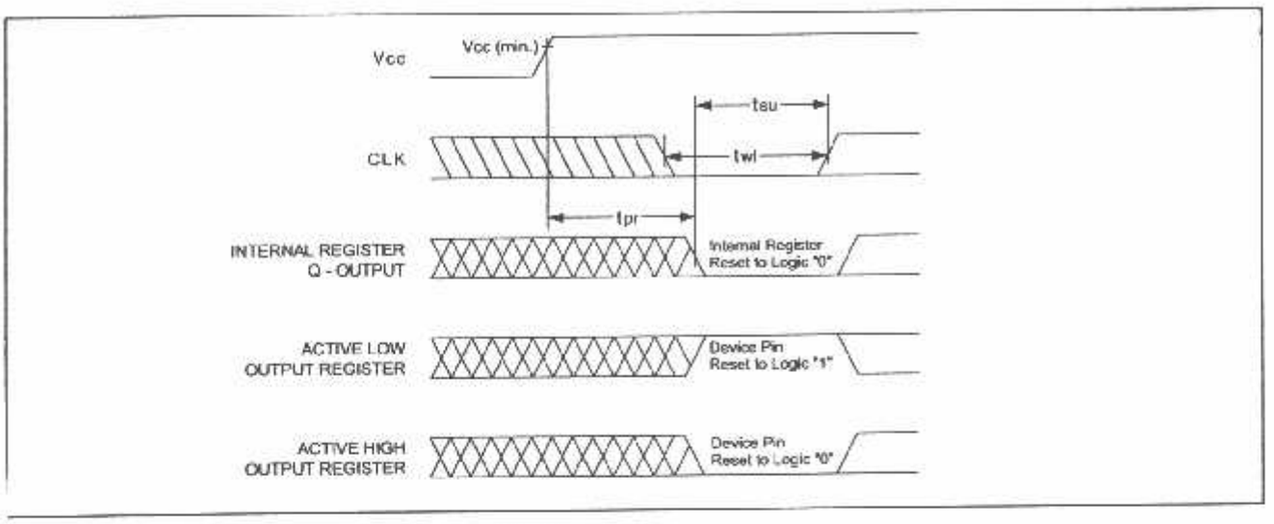
GAL22V10 devices are designed with TTL level compatible input buffers. These buffers have a characteristically high impedance, and present a much lighter load to the driving logic than bipolar TTL devices.

The input and I/O pins also have built-in active pull-ups. As a result, floating inputs will float to a TTL high (logic 1). However, Lattice Semiconductor recommends that all unused inputs and tri-stated I/O pins be connected to an adjacent active input, Vcc, or ground. Doing so will tend to improve noise immunity and reduce Icc for the device. (See equivalent input and I/O schematics on the following page.)

Typical Input Current

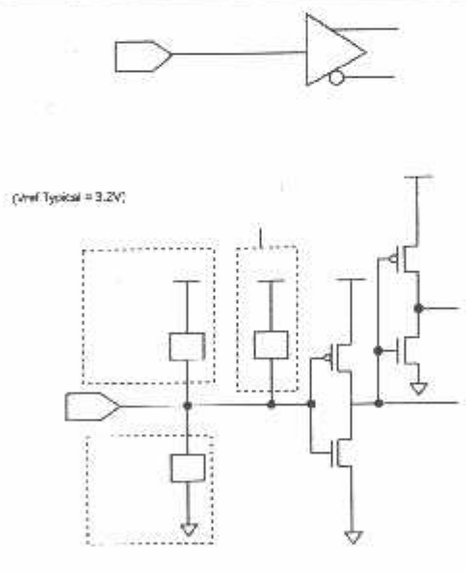


Power-Up Reset

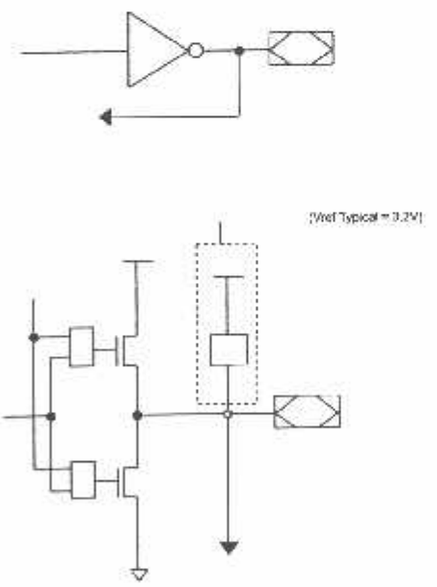


Circuitry within the GAL22V10 provides a reset signal to all registers during power-up. All internal registers will have their Q outputs set low after a specified time (tpr, 1µs MAX). As a result, the state on the registered output pins (if they are enabled) will be either high or low on power-up, depending on the programmed polarity of the output pins. This feature can greatly simplify state machine design by providing a known state on power-up. The timing diagram for power-up is shown below. Because of the asynchronous nature of system power-up, some conditions must be met to guarantee a valid power-up reset of the GAL22V10. First, the Vcc rise must be monotonic. Second, the clock input must be at static TTL level as shown in the diagram during power up. The registers will reset within a maximum of tpr time. As in normal system operation, avoid clocking the device until all input and feedback path setup times have been met. The clock must also meet the minimum pulse width requirements.

Input/Output Equivalent Schematics

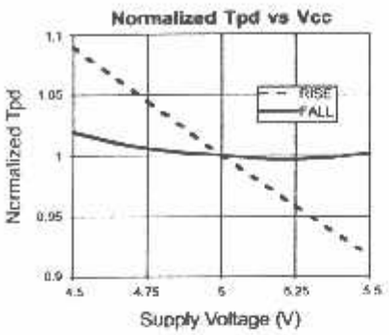


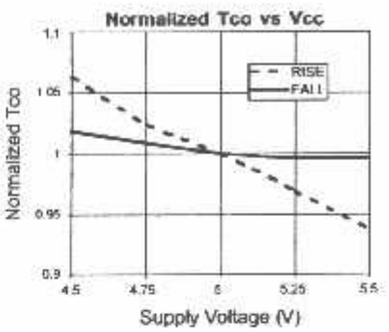
Typical Input

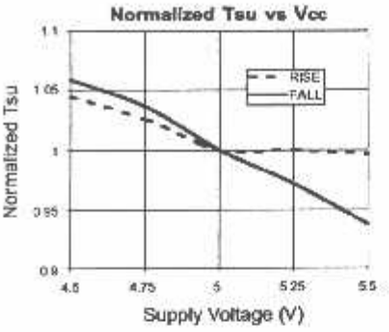


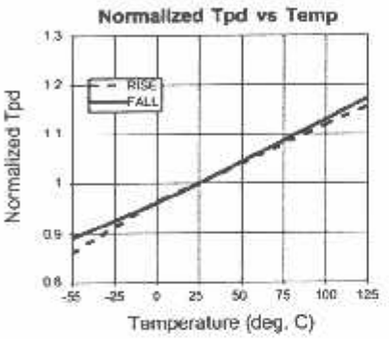
Typical Output

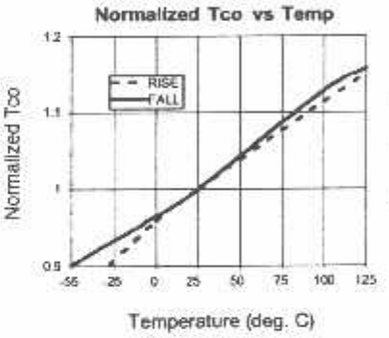
GAL22V10D-4/-5/-7/-10L (PLCC): Typical AC and DC Characteristic Diagrams

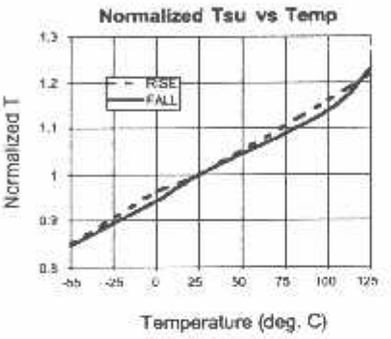


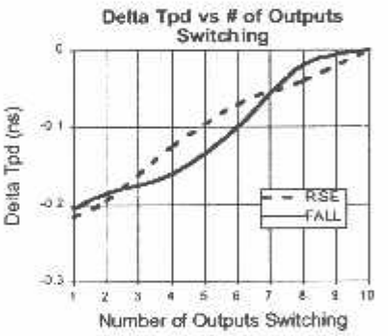


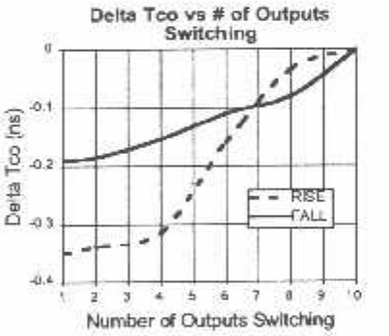


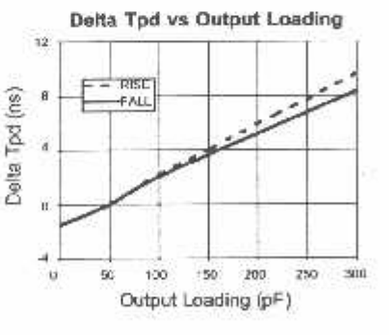


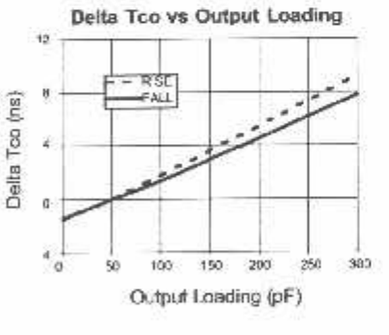






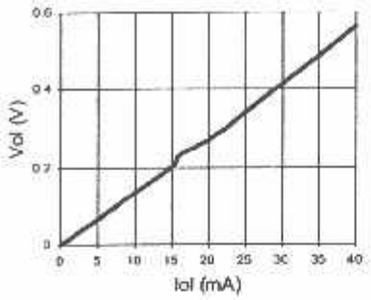




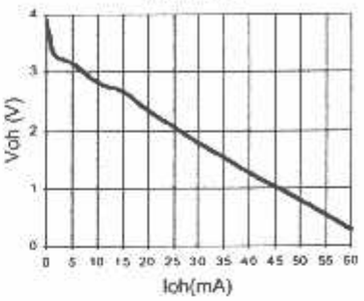


GAL22V10D-4/-5/-7/-10L (PLCC): Typical AC and DC Characteristic Diagrams

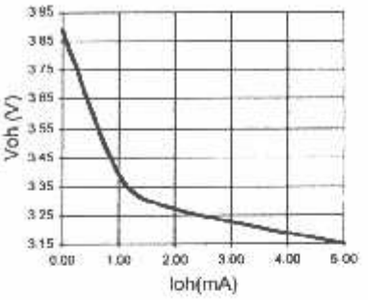
Vol vs Iol



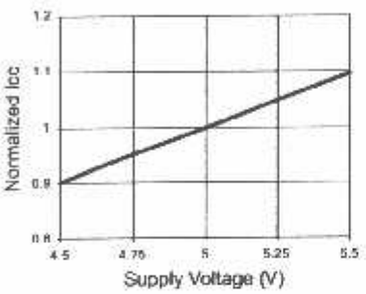
Voh vs Ioh



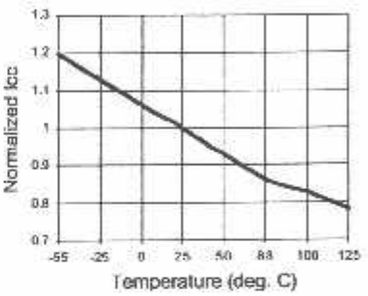
Voh vs Ioh



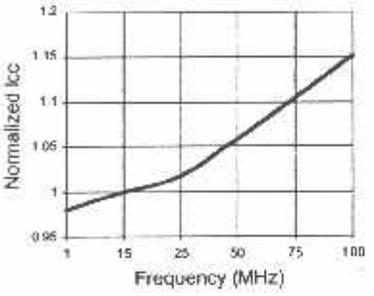
Normalized Icc vs Vcc



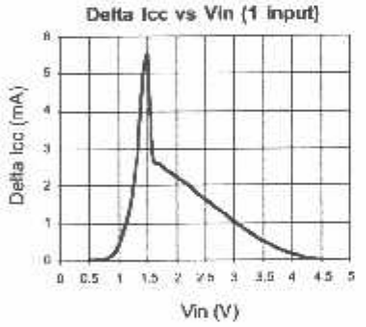
Normalized Icc vs Temp



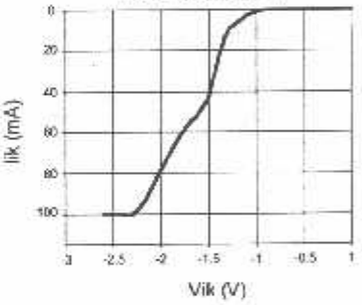
Normalized Icc vs Freq



Delta Icc vs Vin (1 input)

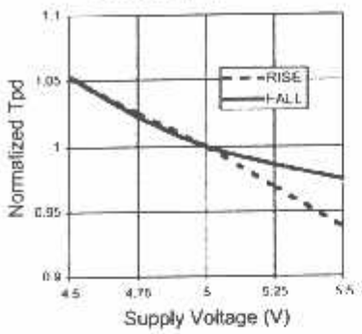


Input Clamp (Vik)

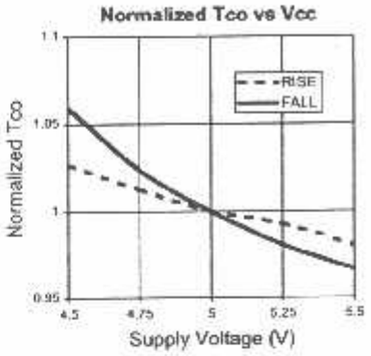


GAL22V10D-7/10L (PDIP): Typical AC and DC Characteristic Diagrams

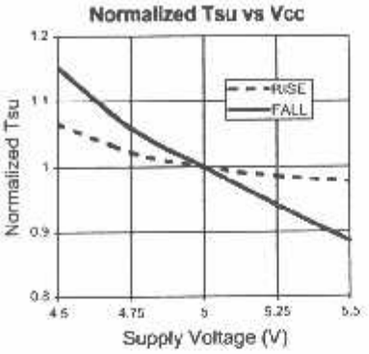
Normalized Tpd vs Vcc



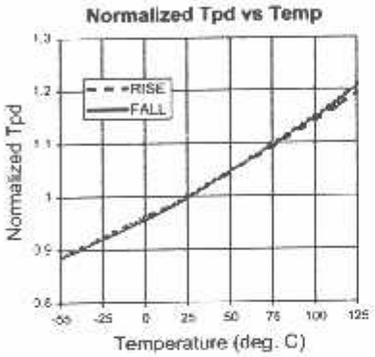
Normalized Tco vs Vcc



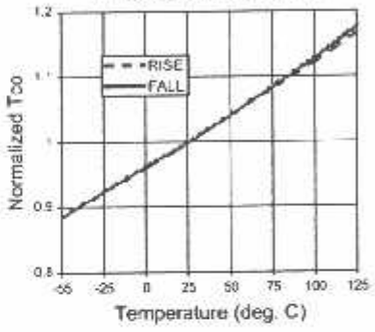
Normalized Tsu vs Vcc



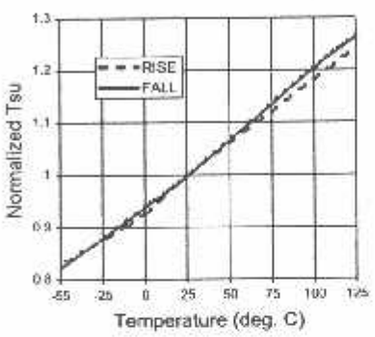
Normalized Tpd vs Temp



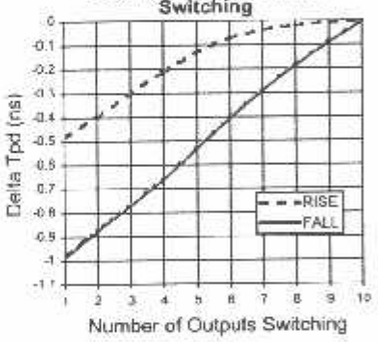
Normalized Tco vs Temp



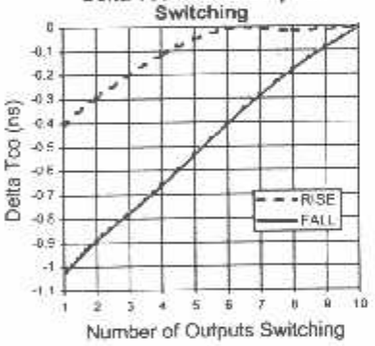
Normalized Tsu vs Temp



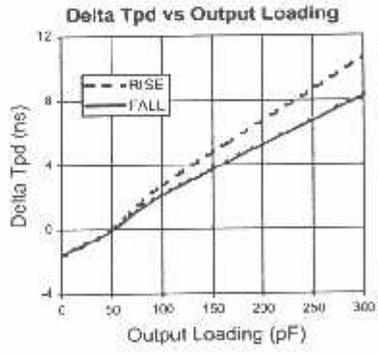
Delta Tpd vs # of Outputs Switching



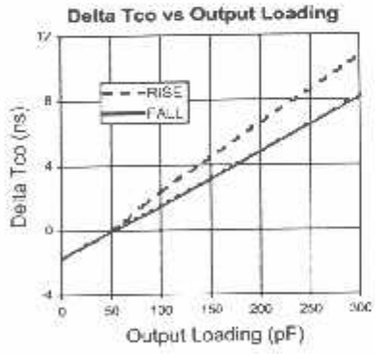
Delta Tco vs # of Outputs Switching



Delta Tpd vs Output Loading

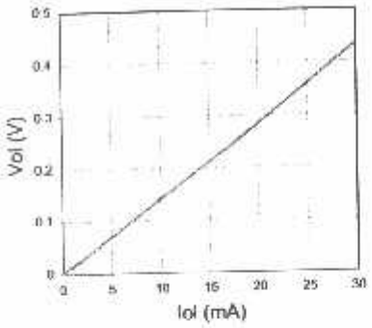


Delta Tco vs Output Loading

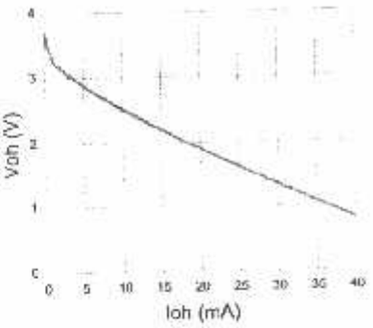


GAL22V10D-7/10L (PDIP): Typical AC and DC Characteristic Diagrams

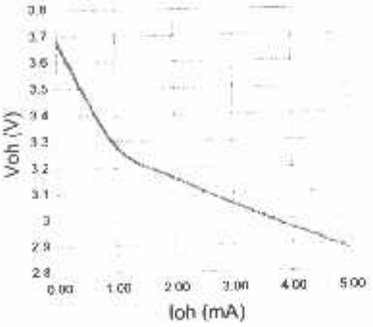
Vol vs Iol



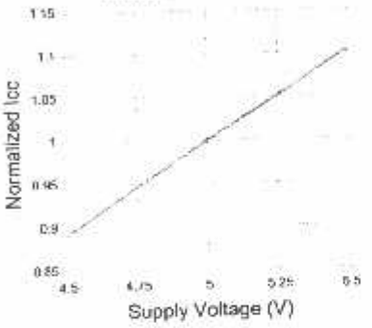
Voh vs Ioh



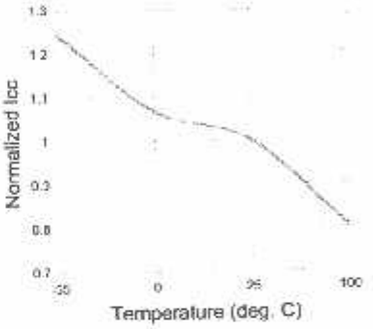
Voh vs Ioh



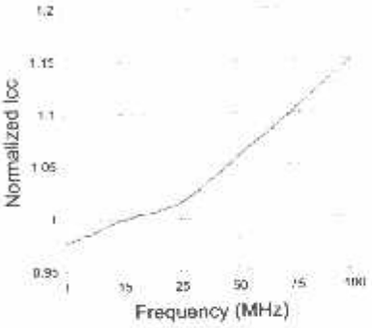
Normalized Icc vs Vcc



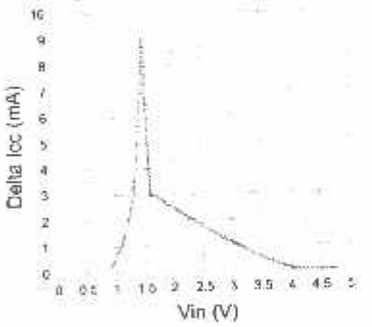
Normalized Icc vs Temp



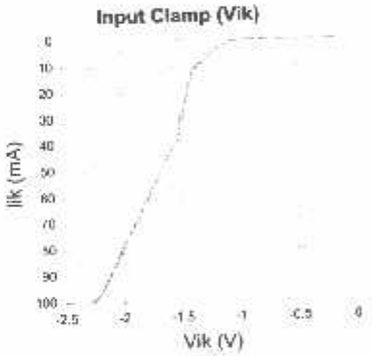
Normalized Icc vs Freq



Delta Isb vs Vin (1 input)

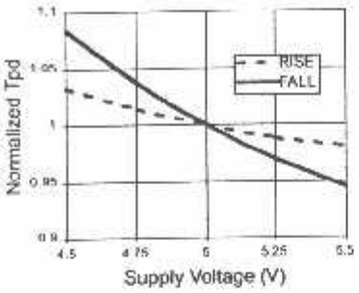


Input Clamp (Vik)

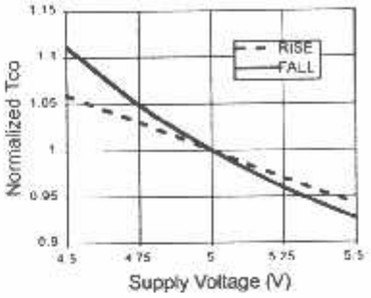


GAL22V10D-10Q and Slower (L & Q): Typical AC and DC Characteristic Diagrams

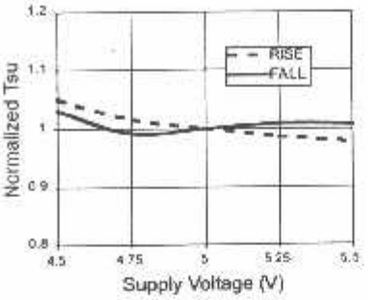
Normalized Tpd vs Vcc



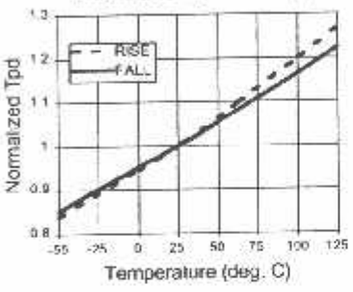
Normalized Tco vs Vcc



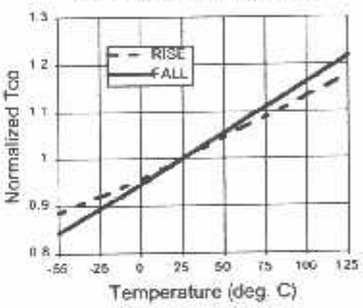
Normalized Tsu vs Vcc



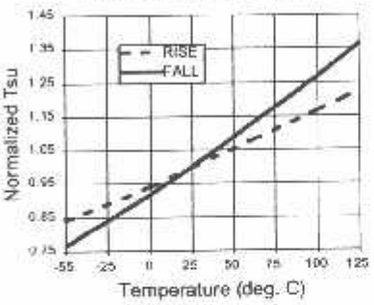
Normalized Tpd vs Temp



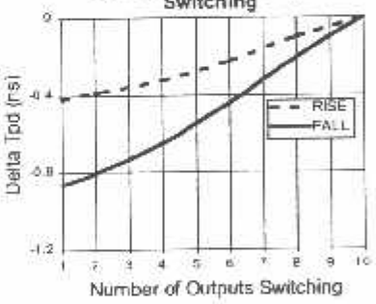
Normalized Tco vs Temp



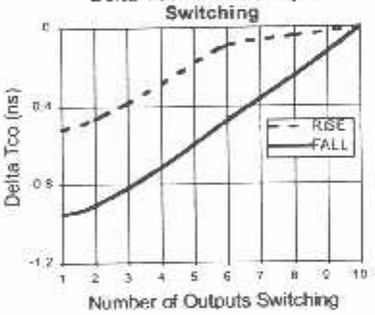
Normalized Tsu vs Temp



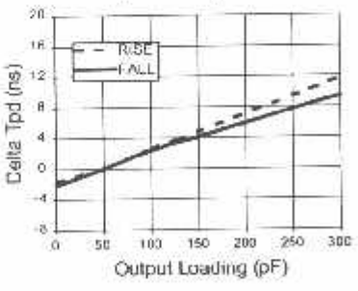
Delta Tpd vs # of Outputs Switching



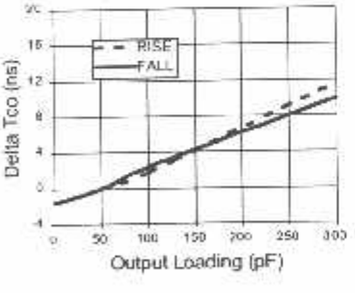
Delta Tco vs # of Outputs Switching



Delta Tpd vs Output Loading

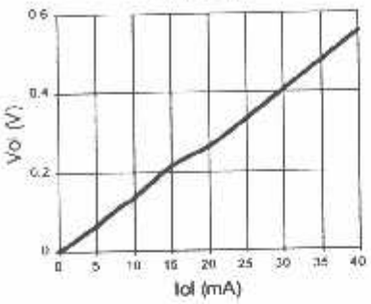


Delta Tco vs Output Loading

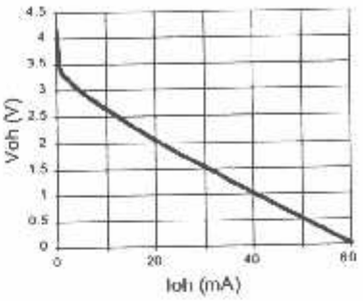


GAL22V10D-10Q and Slower (L & Q): Typical AC and DC Characteristic Diagrams

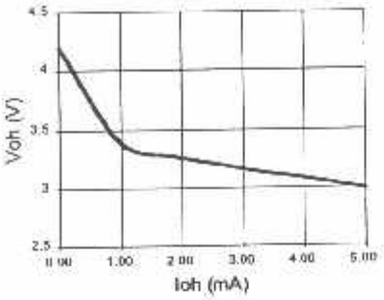
Vol vs Iol



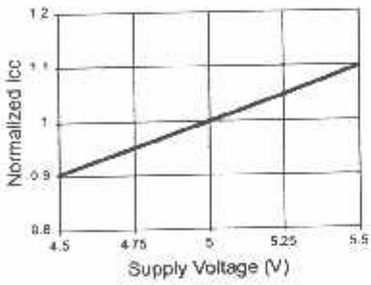
Voh vs Ioh



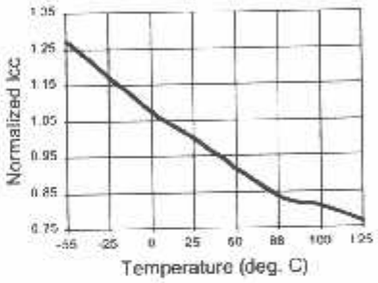
Voh vs Ioh



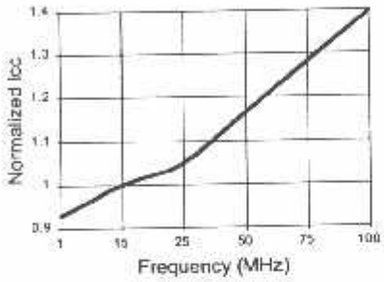
Normalized Icc vs Vcc



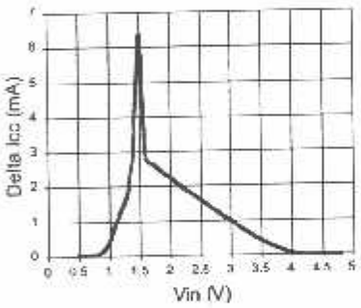
Normalized Icc vs Temp



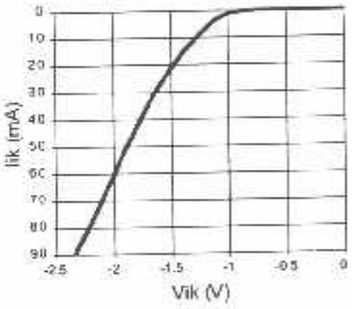
Normalized Icc vs Freq



Delta Icc vs Vin (1 Input)



Input Clamp (Vik)



INSTITUT TEKNOLOGI NASIONAL
JL. BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Muhammad Tohiron
Nim : 0017175
Masa Bimbingan : 10-Sep-2005 s/d 13-Mar-2006
Judul Skripsi : Perencanaan dan pembuatan Downloader IC GAL 16V8 dan 22V10 berbasiskan VHDL

NO	Tanggal	Uraian	Paraf Pembimbing
1.	2-03-2006	SUMBER GAMBAR	
2.	2-03-2006	BAB I	
3.	2-03-2006	BAB II	
4.	6-03-2006	BAB III	
5.	7-03-2006	KESIMPULAN	
6.	10-03-2006	BAB IV	
7.	13-03-2006	BAB I, II, III, IV, V	
8.	15-03-2006	ACC KOMPRE	
9.			
10.			

Malang, 15-03-2006
Dosen Pembmbing

Joseph Dedy Irawan, ST, MT

Form. S-4a

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

FORMULIR PERBAIKAN SKRIPSI

Nama Mahasiswa : Muhammad Tohiron
NIM : 00.17.175
Masa Bimbingan : 10 September 2005 s/d 13 Maret 2006
Judul Skripsi : Perencanaan dan Pembuatan Downloader IC GAL 16V8 dan 22V10 Berbasiskan VHDL

Materi Perbaikan	Paraf Penguji
Pengujian Alat • Mulai langkah awal di protel • Hasil pada alat peraga (LED) Kesimpulan • Dimuat juga spesifikasi alat • Feature yang ada Timing Diagram Pemrograman Pada IC GAL	22/5 '06

Disetujui :

Penguji I

Ir. M. Luqman, MS

Penguji II

Ir. Eko Nurcahyo

Mengetahui,
Dosen Pembimbing

Joseph Dedy Irawan, ST MT
NIP 132315178

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

LEMBAR BIMBINGAN SKRIPSI

Nama Mahasiswa	: Muhammad Tohiron
NIM	: 00.17.175
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika
Judul Skripsi	: Perencanaan dan Pembuatan Downloader IC GAL 16V8 dan 22V10 Berbasiskan VHDL
Mulai Bimbingan Skripsi	: 10 September 2005
Selesai Bimbingan Skripsi	: 13 Maret 2006
Pembimbing	: Joseph Dedy Irawan, ST MT
Dengan Nilai	: 82

Mengetahui
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP Y.1039500274

Disetujui
Dosen Pembimbing

Joseph Dedy Irawan, ST MT
NIP 132315178