Versuchsbeschreibung Nr. 7

Verwendete Geräte:
1 PC
1 GAL-Programmiergerät
1 Software zur Programmierung
1 Netzgerät
1 Funktionsgenerator

Literatur:
1. Dieter Bitterle: GALs
2. Joachim Blank: Logikbausteine Grundlagen...

Es steht eine Beschreibung über den in diesem Versuch verwendeten programmierbaren Baustein, die Software und den Testkasten im Anhang zur Verfügung.

Inhalt: - Kennenlernen programmierbarer Logik

Aufgaben:

1.
- Entwerfen Sie einen 2-Bitvergleicher
- Programmieren Sie die Schaltung.
- Bauen Sie die Schaltung auf.
  Zum Nachweis der Funktion stehen Ihnen Schalter zur Eingabe und LEDs zur Ausgabe zur Verfügung.

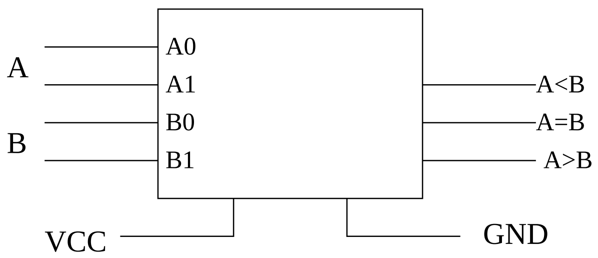


Abb.13: 2-Bitvergleicher

2. - Entwerfen Sie einen synchronen BCD-Abwärts-Zähler von 99 bis 00, mit synchronem Reset.
Gehen Sie dabei nach folgender Entwurfsstrategie vor:
- Entwurf des Zustandsdiagrammes,
- Entwurf der Übergangstabelle,
- Entwurf der logischen Gleichungen.

- Programmieren Sie die Schaltung.
- Bauen Sie die Schaltung auf und weisen Sie deren Funktion logisch nach.
- Beachten Sie, daß
  - I/O 0 nicht verwendet werden kann und
  - die I/O-Belegungen den Pins richtig zugeordnet sind.

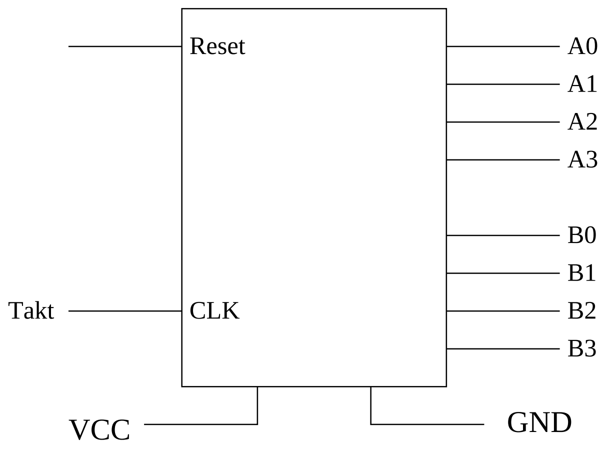


Abb.14: BCD-Abwärtszähler

Anhang zu Versuchsbeschreibung Nr. 7

Einführung

In diesem Versuch soll ein Baustein programmiert werden, bei dem es sich um den ispLSI2032 von Lattice handelt. ispLSI bedeutet In-System-Programmable-Large-Scale-Integration. Er gehört zur Familie der PLD's (Programmable Logic Device). Auf den folgenden Seiten ist der Baustein kurz beschrieben.

The Lattice pLSI and ispLSI Families

The Lattice programmable Large Scale Integration (pLSI) and in-system programmable Large Scale Integration (ispLSI) families are the logical choice for your next design project. They're the first programmable logic devices to combine the performance and ease of use of PLDs with the density and flexibility of FPGAs. And at 135 MHz system speed, and up to 14000 PLD gates, they're the world's fastest and highest density programmable logic devices!

There are three pLSI and ispLSI families to fit your specific application needs. Lattice's premier pLSI and ispLSI 1000 family implements high integration functions such as controllers, LANs and encoders at high speeds. The high performance pLSI and ispLSI 2000 family with its large number of I/Os handles timers, counters as well as timing critical interfaces to high speed RISC/CISC microprocessors. The highest density pLSI and ispLSI 3000 family integrates complete system logic, DSP functions, and entire encryption or compression logic into a single package, while delivering superior performance.

The ispLSI 1000, 2000 and 3000 families pioneer non-volatile, in-system programmability, a technology that allows real-time programming, less expensive manufacturing and end-user system reconfiguration.

All the development tools you need are available from Lattice - tools ranging from Lattice's own entry level software to higher level, third-party design environments. With these tools, you'll be completing your circuit designs in hours instead of weeks or months.

pLSI and ispLSI 1000: The Premier High Density Family

- ❑ 110 MHz system performance
- ❑ 10 ns pin-to-pin delay (maximum)
- ❑ 2000-8000 PLD gates
- ❑ 44-pin to 128-pin packages

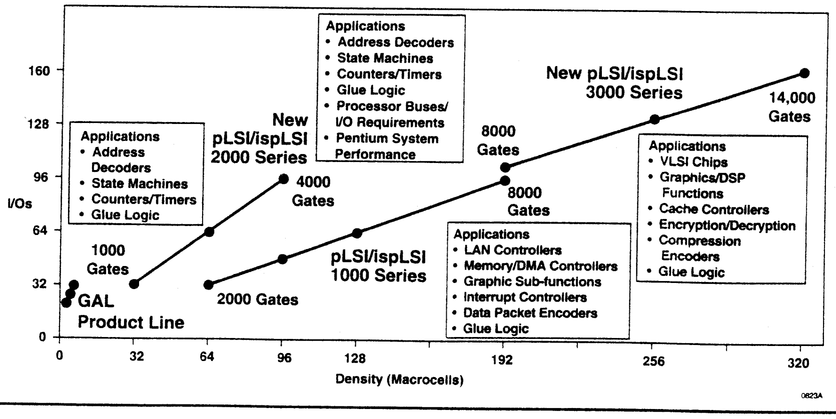
pLSI and ispLSI 2000: Unparalleled System Performance

- ❑ 135 MHz system performance (world's fastest!)
- ❑ 7.5 ns pin-to-pin delay (maximum)
- ❑ 1000-4000 PLD gates
- ❑ 44-pin to 128-pin packages
- ❑ High I/O to Logic Ratio

pLSI and ispLSI 3000: Density with Performance

- ❑ 110 MHz system performance
- ❑ 10 ns pin-to-pin delay (maximum)
- ❑ 8000-14000 PLD gates (world's largest!)
- ❑ 128-pin to 208-pin packages
- ❑ Boundary scan for enhanced testability (IEEE 1149.1)

Lattice's pLSI and ispLSI Families



Family Overview

From registers to counters, from multiplexers to complex state machines, these families of high-density programmable logic will address your high-performance system logic needs.

With PLD gate densities ranging from 1,000 to 14,000, the pLSI and ispLSI devices provide the range of programmable logic solutions you need to meet design requirements today and tomorrow.

Each device contains multiple Generic Logic Blocks (GLBs), architectured to maximize system flexibility and performance. And a generous supply of registers and I/O cells provides the optimum balance of internal logic and external connections. A global interconnect scheme ties everything together, enabling high logic utilization .

Table 1. pLSI and ispLSI Family Attributes

pLSI and ispLSI 1000

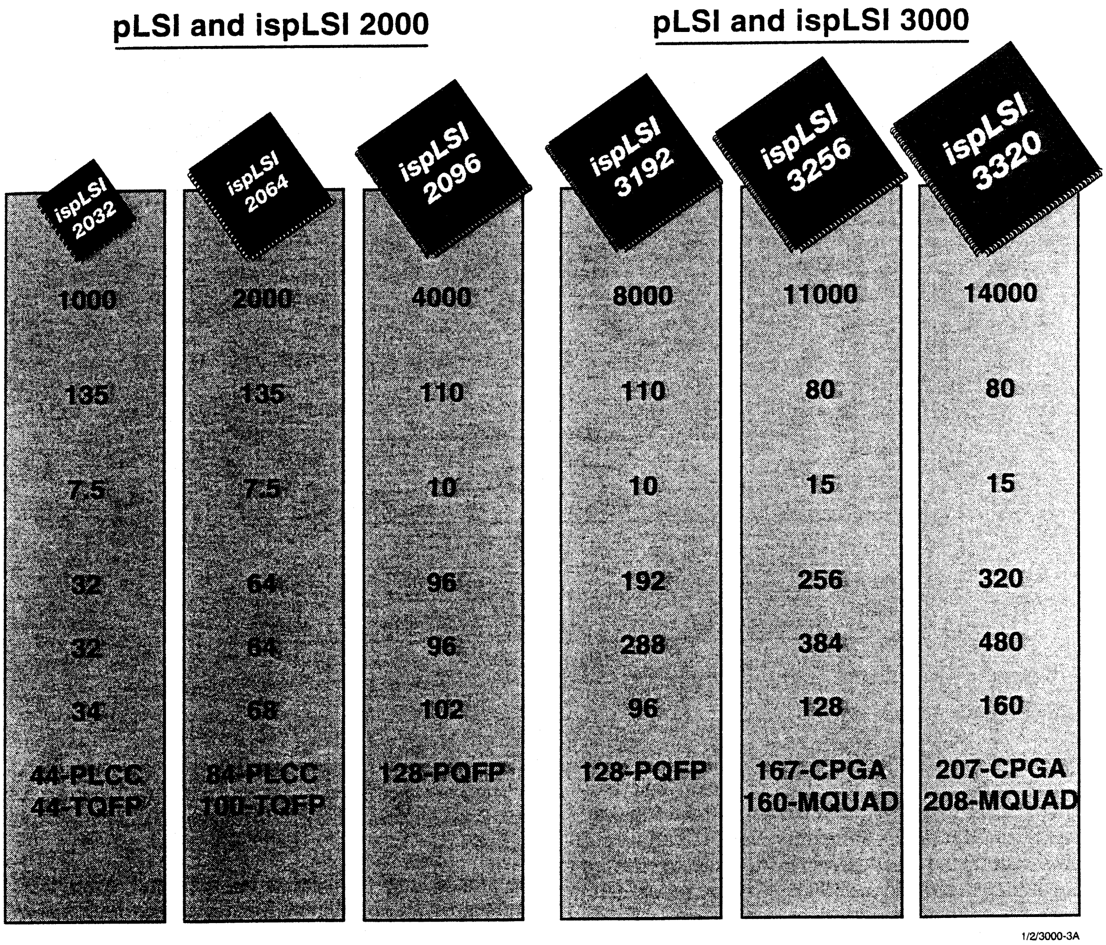
	ispLSI 1016	ispLSI 1024	ispLSI 1032	ispLSI 1048/1048C
Density (PLD Gates)	2000	4000	6000	8000
Speed: Fmax (MHz)	110	90	90	80
Speed: Tpd (ns)	10	12	12	15
Macrocells	64	96	128	192
Registers	96	144	192	288
Inputs & I/Os	36	54	72	106/110
Pins/Package	44-PLCC 44-TQFP 44-JLCC	68-PLCC 68-JLCC	84-PLCC 100-TQFP 84-CPGA	120-PQFP 128-PQFP

1/2/3000-2A

pLSI and ispLSI Architecture

The pLSI and ispLSI architecture was constructed with real system design requirements in mind. Figure 1 shows the representation of the pLSI 3256 architecture. This architecture provides the designer with the following advantages.

- ❑ High Speed
- ❑ Predictable performance
- ❑ Low power
- ❑ Flexible architecture
- ❑ Easy to use
- ❑ Design portability across all the families
- ❑ Non-volatile in-system programmable (ispLSI)
- ❑ Advanced Global Clock Network
- ❑ Boundary Scan (3000 Family)



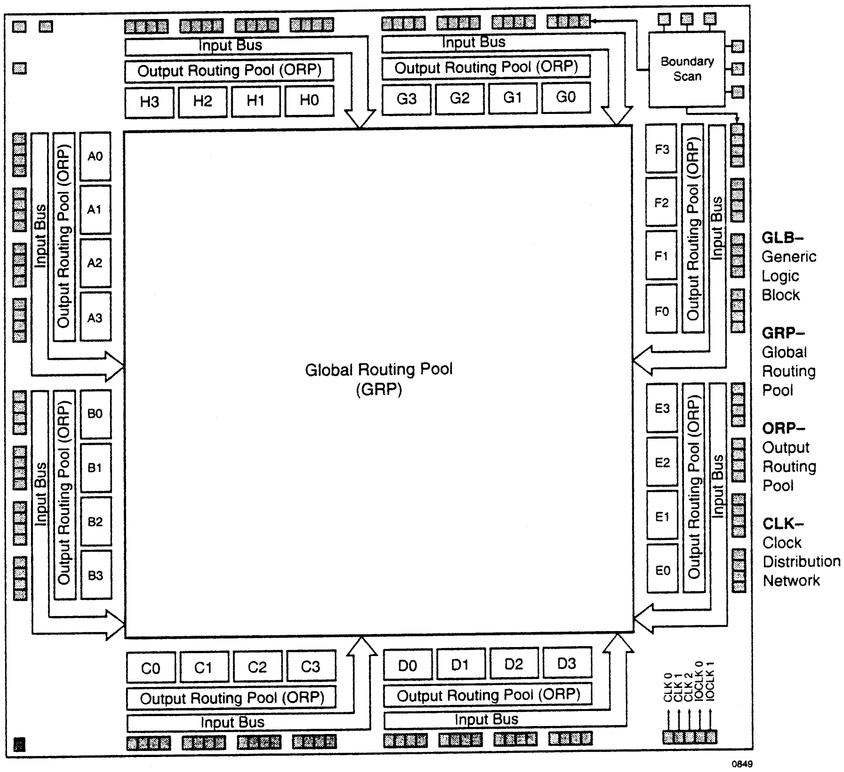
The Global Routing Pool

Central to the pLSI and ispLSI architecture is the Global Routing Pool (GRP), which connects all of the internal logic and makes it available to the designer. The GRP provides complete interconnectivity with fixed and predictable delays. This unique interconnect scheme consistently provides high performance and allows effortless implementation of complex designs.

The Output Routing Pool (ORP)

The Output Routing Pool (ORP) is a unique pLSI and ispLSI architectural feature which provides flexible connections between the GLB outputs and the output pins. This flexibility allows for "last minute" logic design changes to be implemented without changing the external pin-out.

Figure 1. pLSI 3256 Functional Block Diagram



Generic Logic Block (GLB)

The key element in the pLSI and ispLSI architecture is the Generic Logic Block (GLB). This powerful logic block provides a high input-to-output ratio for best logic efficiency. The GLB (figure 2) used in the pLSI and ispLSI 1000 and 2000 families feature 18 inputs which drive an array of 20 Product Terms (PTs). These product terms feed four outputs which effectively handle both wide and narrow gating functions. The pLSI and ispLSI 3000 family utilizes a Twin GLB which delivers wider logic functionality. The Twin GLB accepts 24 inputs and feeds two arrays of 20 Product Terms that ultimately drive two sets of four outputs.

The architectural flexibility of the pLSI and ispLSI GLB, combined with its optimum input-to-output ratio, allows the GLB to implement virtually all 4-bit and 8-bit MSI functions.

An additional element of architectural flexibility is the Product Term Sharing Array (PTSA). The PTSA allows the 20 PTs from the AND array to be shared with any and all of the four GLB outputs. This ability to share PTs between all of the four GLB outputs provides a highly efficient means to implement complex state machines by eliminating duplicate product term groups.

Each of the four outputs from the PTSA feeds into a flexible Output Logic Macrocell (OLMC), consisting of a D-type flip-flop with an Exclusive-OR gate on the input.

The OLMC allows each GLB output to be configured as either combinatorial or registered. Combinatorial mode is available as AND-OR or Exclusive-OR. Registered mode is available as D, T or J-K.

The power of the GLB is further enhanced by a flexible clock distribution network. This network provides a choice of clock signals to each GLB: global synchronous clock signals or internally generated asynchronous product term clock signals.

Figure 2. pLSI and ispLSI 1000 and 2000 Family GLB

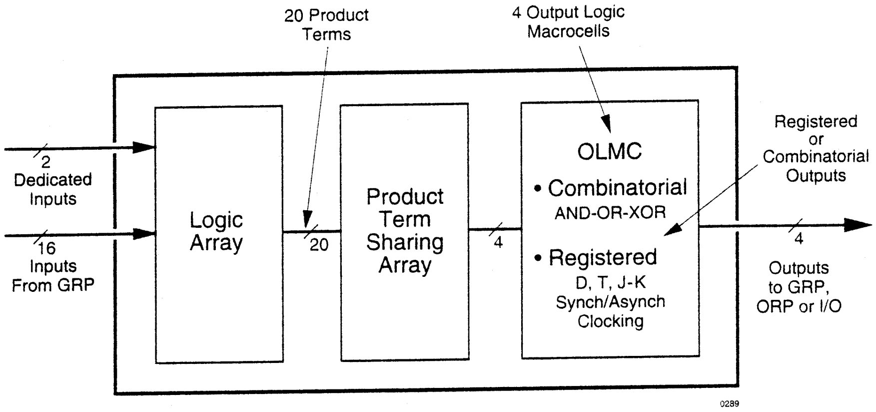


Figure 3. pLSI and ispLSI 3000 Family "Twin GLB"

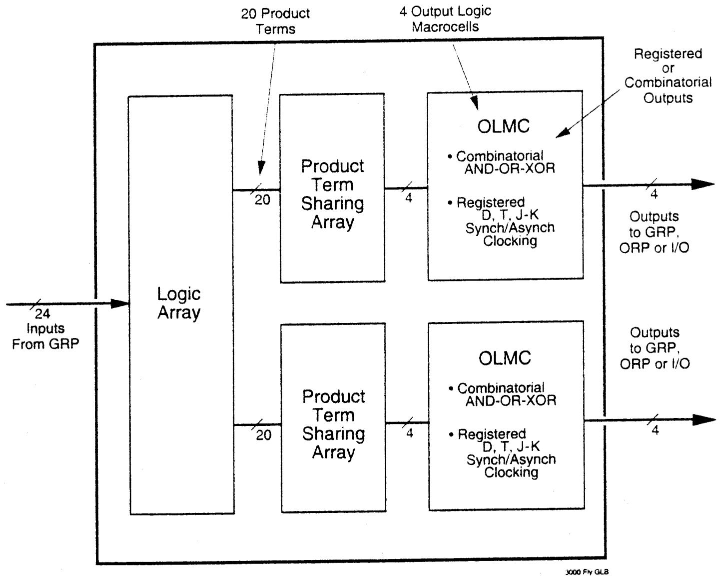
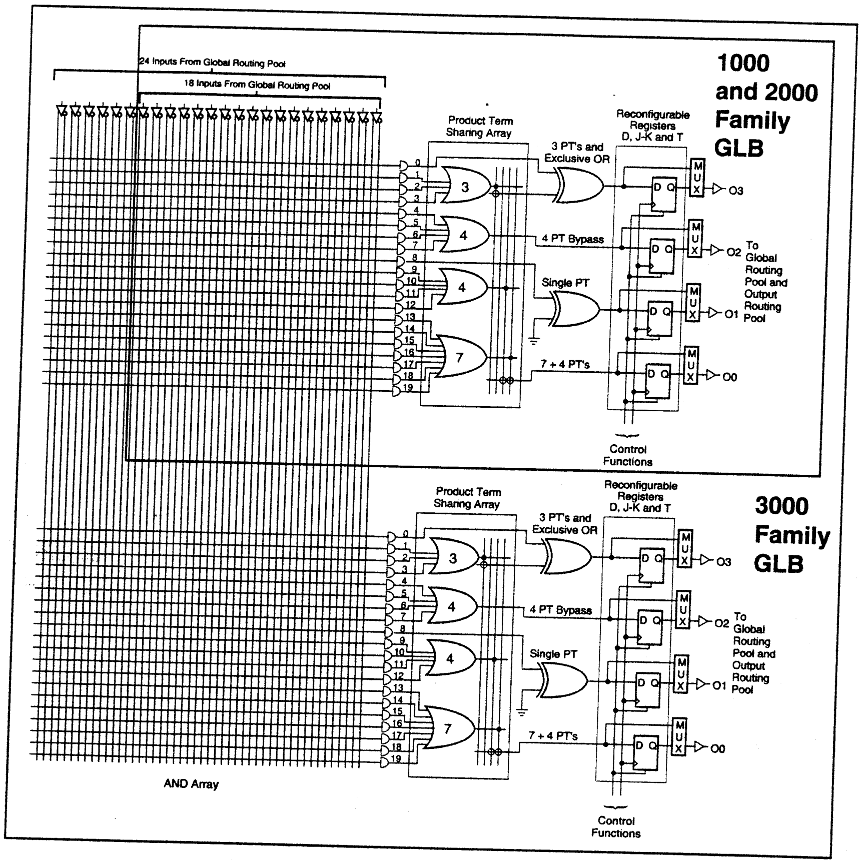


Figure 4. GLB: Multi-Mode Configuration



Standard Configuration

- GLB Outputs Comprised of 4, 4, 5 and 7 Product Terms
- The PTSA can Combine up to 20 Product Terms per GLB Output to Meet the Needs of Both Wide and Narrow Logic Functions

High-Speed Bypass Configuration

- For Speed-Critical Timing Paths
- Bypasses the PTSA and the Internal Exclusive-OR gate of the OLMC
- Provides Four Product Terms per Output
- Supports Design of Fast Address Decoders

Exclusive-XOR Configuration

- Utilizes Powerful Exclusive-XOR Architecture
- Great for Counters, Comparators and ALU Functions

Single PT Configuration

- Small Logic Functions at Fast Speed

Multi-Mode Configuration

- Individual Outputs are Independently Configurable
- PTSA Gives Flexibility in the Number and Selection of Product Terms per Output

Security Cell

A security cell is provided in the pLSI and ispLSI devices 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 reprogramming the device, so the original configuration can never be examined once this cell is programmed.

Device Programming

pLSI and ispLSI devices can be programmed using a Lattice-approved device programmer, available from a number of third party manufacturers. Complete programming of the device takes only a few seconds. Erasing of the device is automatic and is completely transparent to the user. In-system programming is also available with ispLSI devices which allows programming on the circuit board using Lattice programming algorithms and standard 5V system power.

Latch-up Protection

pLSI and ispLSI 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 internal circuitry to latch-up. Additionally, outputs are designed with n-channel pull-ups instead of the traditional p-channel pull-ups to eliminate any possibility of SCR induced latching.

In-system programmability

Lattice's ispLSI devices (in-system programmable) are the industry's only high-density programmable logic family offering non-volatile, in-system reconfigurability.

ispLSI devices are available in all three families: 1000, 2000 and 3000. The ispLSI devices are 100 percent functionally and parametrically compatible with their pLSI counterparts, with the added capability for 5-volt in-system programmability and reprogrammability.

Complex logic functions can be implemented in multiple ispLSI devices with complete on-board configurability. In-system programming of a multiple ispLSI chip solution is easily achieved through a proprietary in-system erase/program/verify technique.

In-system programmability can revolutionize the way you design, manufacture and service systems.

Prototype Board Designs

In-system programming allows you to program and modify your logic designs "in-system" without removing the device(s) from the board. This accelerates the system and board-level debug process and enables you to define the board layout earlier in the design process.

Fine Pitch Package Handling

When programming traditional PLDs, manual handling is required during both design/debugging and manufacturing stages. When using PQFPs or TQFPs, fragile leads as thin as 0.5 mm can easily bend in the programmer socket causing coplanarity damage. With ispLSI, you can solder these packages onto your printed circuit board and still program and reprogram the devices during debugging and manufacturing – without ever losing a single part due to bent leads.

Reconfigurable Systems

Your options become boundless when you have the ability to change the functionality of devices already soldered on a p.c. board. You can now implement multiple hardware configurations with the same circuit board design. A variety of protocols or system interfaces can be implemented on a generic board as the last step in the manufacturing flow.

Easier Field Updates

With software reconfigurable systems, field updates are as easy as loading a new configuration from a floppy or downloading it through a modem.

Enhanced Manufacturing Flow with ispLSI

Perhaps the most exciting benefit of the ispLSI family is its potential to streamline the manufacturing process by eliminating the separate programming and labeling steps usually associated with PLDs. Quality is enhanced when product handling steps are reduced, in this case, those associated with programming, labeling and re-inventorying multiple device types. Eliminating socketing further improves quality and reduces board cost. Figure 6 shows the enhanced manufacturing with the ispLSI device.

Figure 5. In-System Programmable Graphics Board

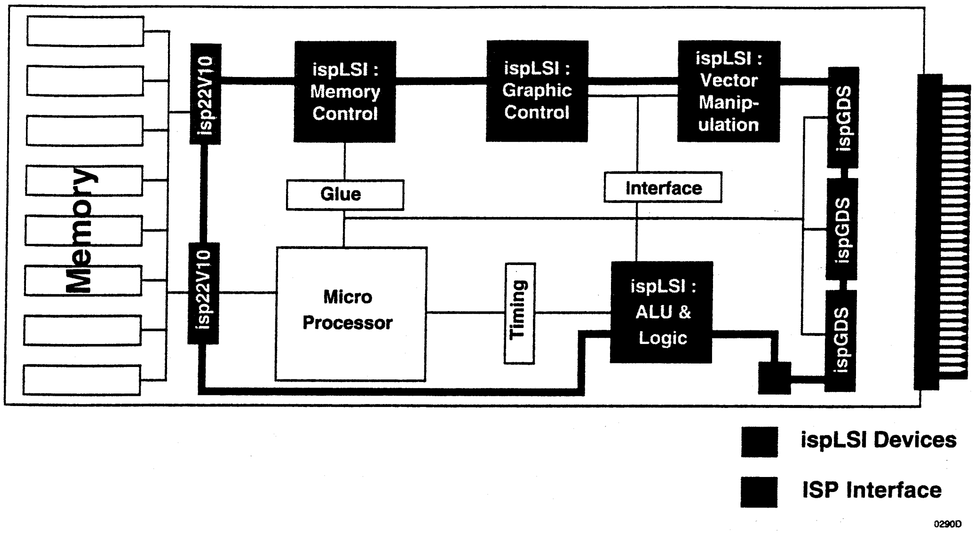
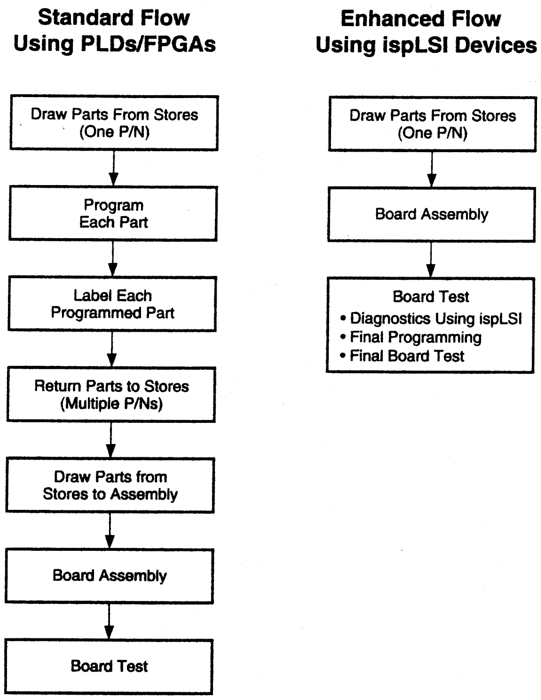
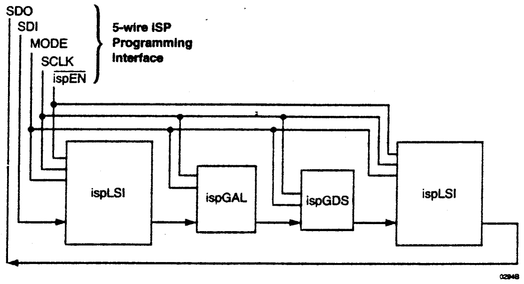


Figure 6. Manufacturing Flow Comparison



All necessary programming is achieved via five TTL-level logic interface signals (see figure 7). These five signals control the on-chip programming circuitry, which protects against inadvertent reprogramming via on-chip state machines. The ispLSI family can also be programmed using popular third-party logic programmers.

Figure 7. In-System Programming Interface (Multi-Chip Solution)



Boundary Scan

An emerging trend in board-level testing is boundary scan test, an attractive feature helping designers test system boards efficiently while lowering test and manufacturing costs. The pLSI and ispLSI 3000 family offers dedicated IEEE 1149.1 boundary scan support for all test functions required by the standard. By using pLSI and ispLSI devices you not only eliminate expensive "bed-of-nails" testers but also simplify testing of surface-mount boards, multi-layer boards and boards using fine-pitch packages. Boundary scan is ideal wherever tight board layout limits access to logic signals.

It only takes 4 pins to implement the boundary scan interface. The ispLSI 3000 devices share the four boundary scan signals with the in-system programming pins. This enhances the testability of system designs allowing logic to be reconfigured to improve controllability and observability.

Lattice Development Systems

The Lattice pLSI/ispLSI Development System (pDS) software is used to implement designs in pLSI and ispLSI devices. Design alternatives can be quickly implemented using Lattice's low cost pDS software or the pDS+ family of Fitters that interface with third-party development software packages. This section describes the pDS and pDS+ Development Systems. Programmer support is also discussed.

pLSI/ispLSI Development System (pDS)

Features

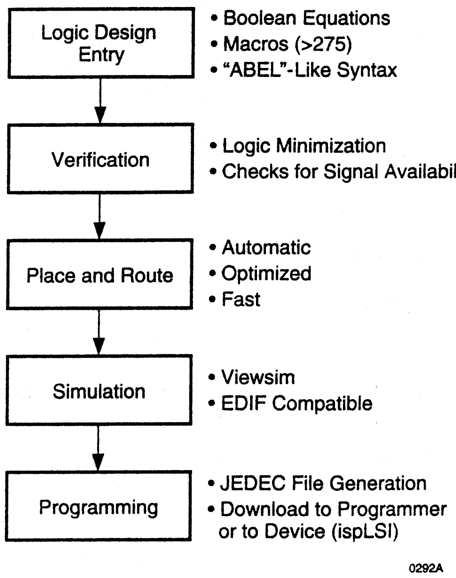
- ❑ High-Performance, Low-Cost Development Environment
- ❑ Supports pLSI and ispLSI Device Families
- ❑ Boolean Logic and Text File Design Entry
- ❑ Windows Based Graphical User Interface
- ❑ Over 275 Macros Available
- ❑ Automatic Place and Route
- ❑ Static Timing Table
- ❑ Logic Simulation with Viewlogic™ Viewsim™
- ❑ JEDEC File Download Direct to Programmer or ispLSI Device

General Description

All pLSI and ispLSI families are supported by Lattice's low-cost pDS software. It runs on IBM-compatible (386/486/Pentium) PCs with Microsoft® Windows.

The graphical user interface employs an easy-to-use mouse and pull-down menu driven approach. Combined with Boolean logic data entry using an ABEL™-like syntax, pDS makes design entry with pLSI and ispLSI quick and straightforward (see figure 8).

Figure 8. pDS Design Flow



The pDS software supports over 275 macros to assist the design process. These macros cover most TTL functions, from gate primitives to 16-bit counters. The software also supports user-definable macros which can be modifications of existing macros or custom creations.

The pDS software automatically verifies the design, performs logic minimization and checks for signal availability.

The Lattice Place and Route software assigns pins and critical speed paths while routing the design.

Quick compilation speeds the design, debug and rework process dramatically. Incremental design techniques are also supported.

Timing and functional simulation is available from Lattice, using Viewsim simulation software.

The Windows graphical user interface makes programming easy, using pull-down menus, intuitive point-and-click commands and self explanatory instructions. Without any up-front training, designs can be completed within hours instead of days or weeks.

pLSI/ispLSI Development System Plus (pDS+)

Features

- ❑ Supports pLSI and ispLSI Device Families
- ❑ Schematic Capture, State Machine, Design Entry HDL, and Boolean Equations
- ❑ Expanded Macro Library (>300)
- ❑ Automatic Logic Minimization and Partitioning
- ❑ Automatic Place and Route
- ❑ Logic and Timing Simulation
- ❑ EDIF Compatible
- ❑ JEDEC File Download Direct to Programmer or ispLSI Device

General Description

For higher level design entry environments, Lattice offers pDS+ development software packages, which expand on the core capabilities of pDS. Schematic capture, state machine, HDL and Boolean entry are supported, along with an expanded macro library.

The pDS+ software utilizes industry standard third-party design environments such as Viewlogic's Viewdraw™ and Data I/O's ABEL.

Running on IBM compatible (386/486/Pentium) PCs or workstation platforms, pDS+ software supports automatic logic minimization and partitioning as well as place and route, resulting in high logic utilization.

For logic and timing simulation, support is available from Lattice through Viewlogic Viewsim simulation tools.

Third Party Programming Support

The pLSI and ispLSI families are supported by popular third-party logic programmers including Data I/O, Logical Devices, BP-Microsystems, Stag, System General, SMS Micro Systems and Advin. Table 2 describes each vendor's specific programmer models that support the pLSI and ispLSI devices. No proprietary, expensive, high pin-count programmers are required.

High pin-count socket adapters are available from Emulation Technology, Procon Technology, EDI Corporation and Logical Systems Corporation.

Additionally, the ispLSI family can be programmed on the board (in-system), which eliminates the need for a stand-alone programmer. For specific details refer to the Lattice Programming Tools Guide available from your local Sales Representative.

Table 2. Programming Support

Programmer Vendor	Model
Advin Systems	Pilot-U84
	Pilot-U40
	Pilot-GL/GCE
BP Microsystems	PLD-1128
	CP-1128
Data I/O	2900
	3900
	Unisite 40/48
Logical Devices	Allpro 40
	Allpro 88
SMS Micro Systems	Sprint Expert
Stag	System 3000
	ZL30/A
System General	TURPRO-1

isp Engineering Kit

The ispLSI family may also be programmed with Lattice's isp Engineering Kit Model 100 for PCs and Model 200 for Sun workstations. The kit is designed for engineering purposes only and is not intended for production use. By connecting an 8 wire cable to the parallel printer port of a PC, JEDEC files can be easily downloaded into the ispLSI device. Additionally, this cable can be connected directly to the circuit board facilitating on-board in-system programming.

2000 Family Architectural Description

pLSI and ispLSI 2000 Family Introduction

The basic unit of logic of the pLSI and ispLSI 2000 family is essentially the same as that of the pLSI and ispLSI 1000 family. However, there are some specific architectural differences: Global clock structure, I/O Cell and OE structure, and ORP structure. A functional block diagram of the 2032 device is shown in figure 1. These architectural differences are described in detail below.

Global Clock Structure

The clock GLB distribution network of the 1000 family has been eliminated and replaced by three dedicated global GLB clock input signals CLK0, CLK1, and CLK2. These three clocks are used for clocking all the GLBs configured as registers in the device. They feed directly to the GLB clock input via a clock multiplexer. CLK0 is associated with system clock pin Y0, CLK1 corresponds to system clock pin Y1, and finally CLK3 corresponds to system clock pin Y2. This is illustrated in figure 2. The GLB global clocks do not have inversion capability, but all GLBs continue to have the capability of generating their own asynchronous clocks using the clock product term (PT12) with inversion capability. The GLB global clocks and the GLB product term clock feed to their corresponding clock multiplexer shown in figure 3.

Figure 1. ispLSI 2032 Functional Block Diagram

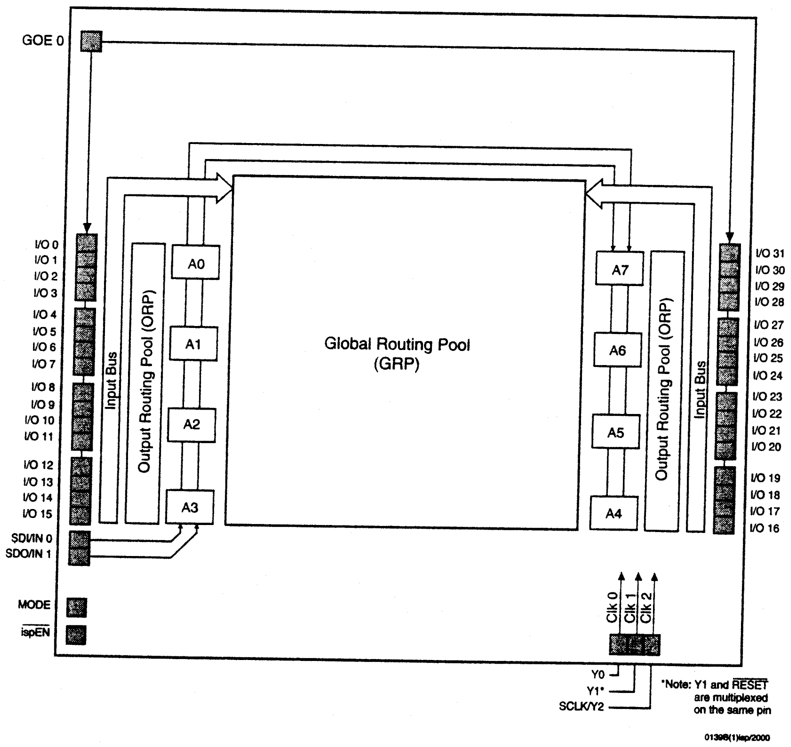


Figure 2. Global Clock Structure

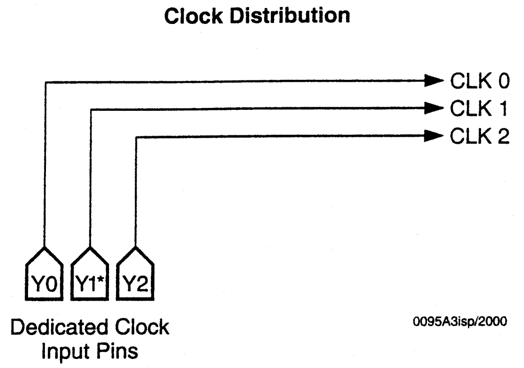
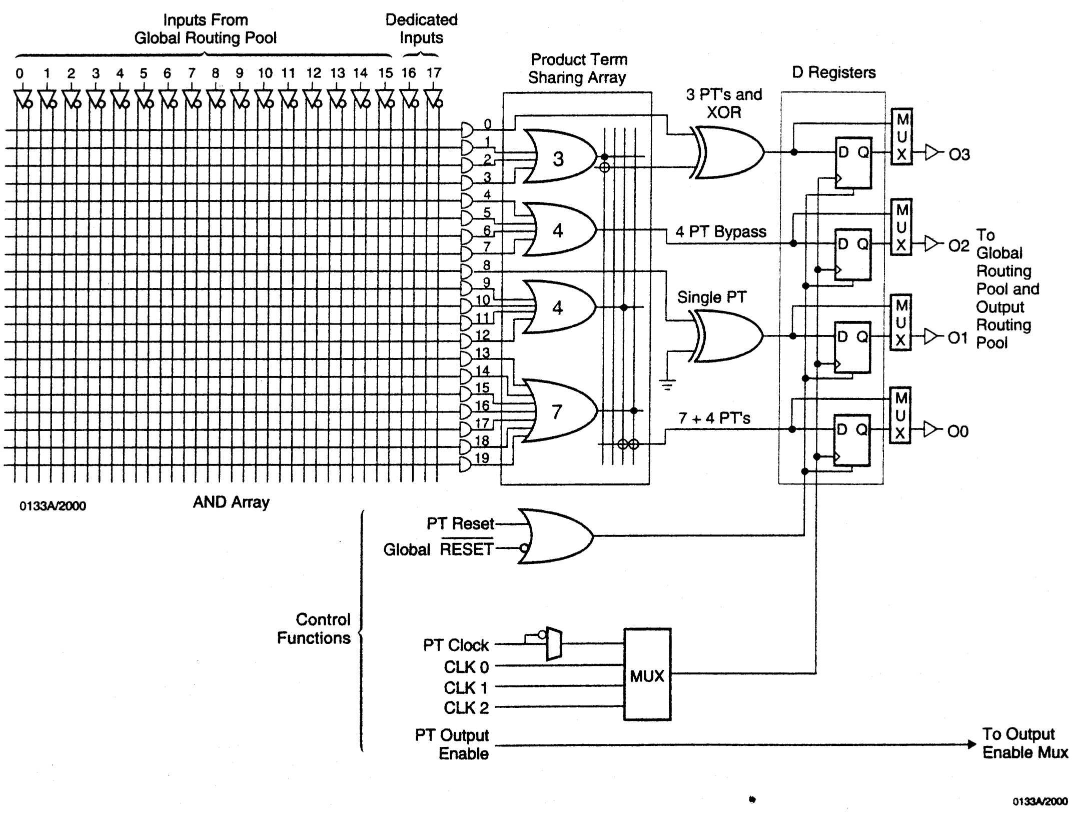


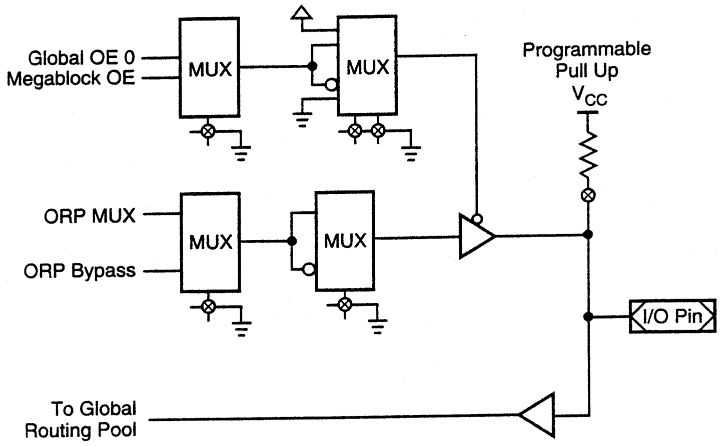
Figure 3. GLB with Clock Multiplexer Scheme



I/O Cell and OE Structure

The reconfigurable input register or latch has been removed to simplify the I/O cell architecture. Each I/O cell can be individually programmed to be a combinatorial input, combinatorial output, or a bi-directional I/O pin with 3-state control. With the simplified I/O cell architecture, the I/O clocks have also been removed. This is illustrated in figure 4. The product term output enable (PTOE) signal is still generated within each GLB using product term 19. The PTOE is generated in one of the eight GLBs. In addition to the PTOE, there is a global output enable (GOE) pin which can control any of the device's 3-state output buffers. The multiplexing between the GOE and PTOE is illustrated in figure 5. The 2032 device has one GOE, and the 2064 and 2096 devices each have 2 GOEs.

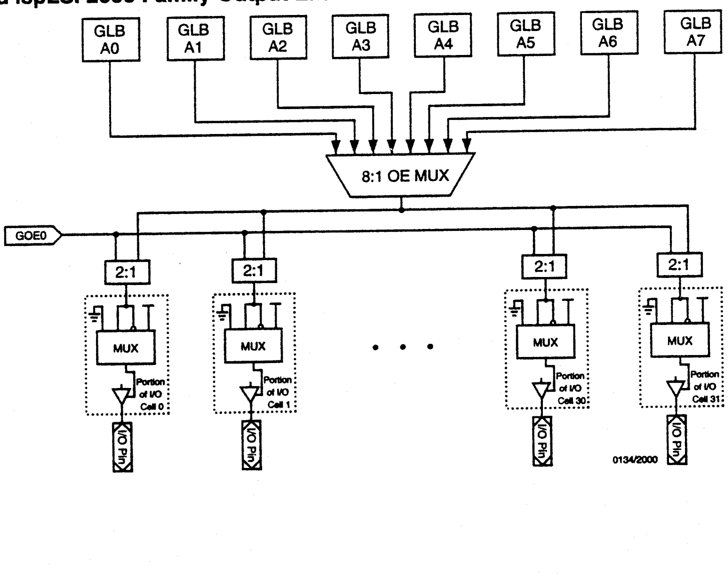
Figure 4. pLSI and ispLSI 2000 Family I/O Cell Architecture



Note:

⊗ Represents an E^2CMOS Cell.

Figure 5. pLSI and ispLSI 2000 Family Output Enable Controls



Output Routing Pool (ORP)

Each megablock now contains two ORPs to increase output routability. A set of four GLBs is associated with one of the two ORPs within the megablock. The 16 outputs of the four GLBs within a megablock will feed to any of the 16 associated I/O cells. In the 1000 family, the 32 GLB outputs feed only 16 associated I/O cells. In this device family, 32 GLB outputs of a megablock can feed 32 I/O cells. Output routability has doubled. This is illustrated in figure 6. Each GLB output has an ORP bypass capability so more designs can have critical output signals. This is shown in figure 7.

Figure 6. pLSI and ispLSI 2000 Family Output Routing Pool

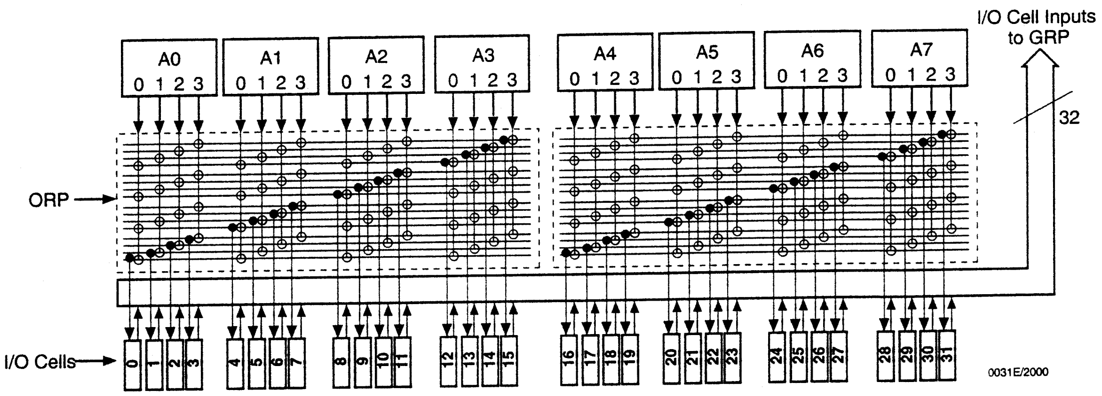
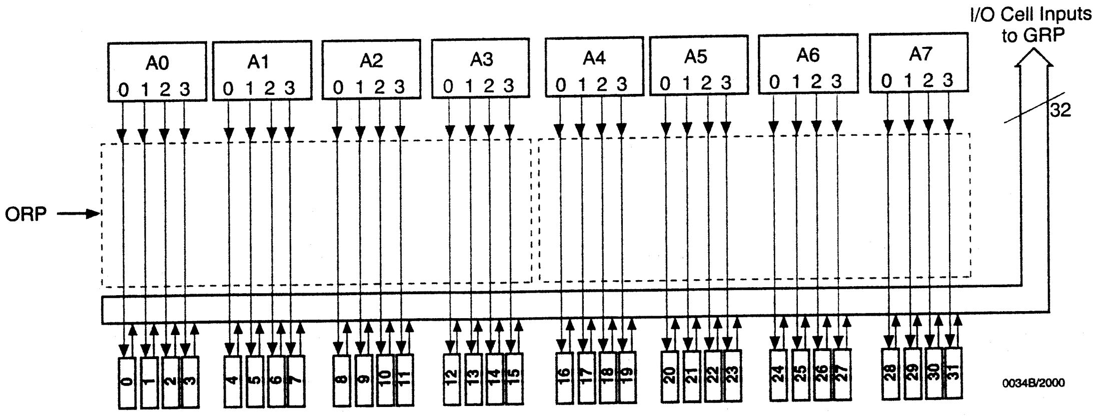


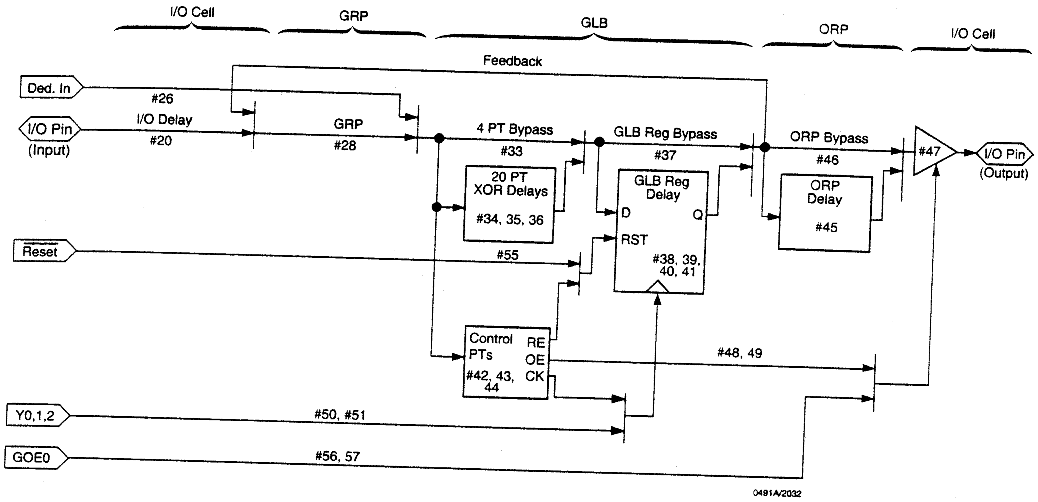
Figure 7. pLSI and ispLSI 2000 Family Output Routing Pool Showing Bypass



Timing Model

The task of determining the timing through the device is simple and straightforward. A device timing model is shown in figure 8. To determine the time that it takes for data to propagate through the device, simply determine the path the data is expected to follow, and add the various delays together (figure 8). Critical timing paths are shown in figure 8, using data sheet parameters. Note that the Internal timing parameters are given for reference only, and are not tested. (External timing parameters are tested and guaranteed on every device).

Figure 8. pLSI and ispLSI 2032 Timing Model



Derivations of tsu, th and tco from the Product Term Clock[1]

tsu = Logic + Reg su - Clock (min)
= (tiobp + tgrp + t20ptxor) + (tgsu) - (tiobp + tgrp + tptck(min))
= (#24 + #28 + #35) + (#38) - (#20 + #28 + #44)
2.2 ns = (1.0 + 1.3 + 4.7) + (0.8) - (1.0 + 1.3 + 3.3)

th = Clock (max) + Reg h - Logic
= (tiobp + tgrp + tptck(max)) + (tgh) - (tiobp + tgrp + t20ptxor)
= (#20 + #28 + #44) + (#39) - (#20 + #28 + #35)
1.6 ns = (1.0 + 1.3 + 3.3) + (3.0) - (1.0 + 1.3 + 4.7)

tco = Clock (max) + Reg co + Output
= (tiobp + tgrp + tptck(max)) + (tgco) + (torp + tob)
= (#20 + #28 + #44) + (#40) + (#45 + #47)
8.8 ns = (1.0 + 1.3 + 3.3) + (0.7) + (1.3 + 1.2)

1. Calculations are based upon timing specs for the pLSI and ispLSI 2032-135L

Lattice Semiconductor Corporation

ispLSI® and pLSI® 2032

High Density Programmable Logic

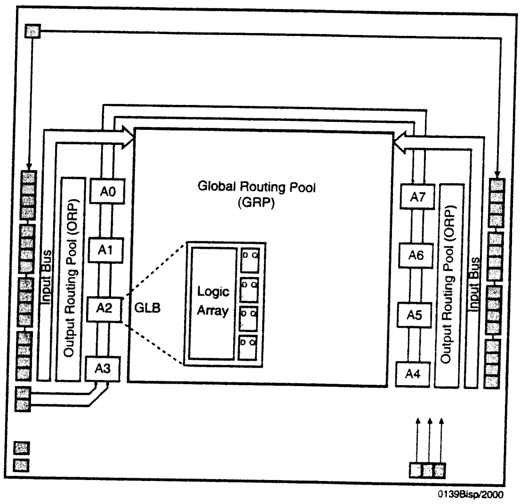
Features

- HIGH DENSITY PROGRAMMABLE LOGIC
  - 1000 PLD Gates
  - 32 I/O Pins, Two Dedicated Inputs
  - 32 Registers
  - High Speed Global Interconnect
  - Wide Input Gating for Fast Counters, State Machines, Address Decoders, etc.
  - Small Logic Block Size for Random Logic
- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - fmax = 154 MHz Maximum Operating Frequency
  - tpd = 5.5 ns Propagation Delay
  - TTL Compatible Inputs and Outputs
  - Electrically Erasable and Reprogrammable
  - Non-Volatile
  - 100% Tested at Time of Manufacture
  - Unused Product Term Shutdown Saves Power
- ispLSI OFFERS THE FOLLOWING ADDED FEATURES
  - In-System Programmable™ (ISP™) 5-Volt Only
  - Reduce Time-to-Market, Increase Manufacturing Yields, and Improve Product Quality
  - Reprogram Soldered Devices for Faster Prototyping
- OFFERS THE EASE OF USE AND FAST SYSTEM SPEED OF PLDs WITH THE DENSITY AND FLEXIBILITY OF FIELD PROGRAMMABLE GATE ARRAYS
  - Complete Programmable Device Can Combine Glue Logic and Structured Designs
  - Three Dedicated Clock Input Pins
  - Synchronous and Asynchronous Clocks
  - Programmable Output Slew Rate Control to Minimize Switching Noise
  - Flexible Pin Placement
  - Optimized Global Routing Pool Provides Global Interconnectivity
- pLSI/ispLSI DEVELOPMENT SYSTEM (pDS®)

  pDS+™ Software
  - Industry Standard, Third Party Design Environments
  - Schematic Capture, State Machine, HDL
  - Automatic Partitioning and Place and Route
  - Comprehensive Logic and Timing Simulation
  - PC and Workstation Platforms

  pDS Software
  - Easy to Use PC Windows™ Interface
  - Boolean Logic Compiler
  - Manual Partitioning
  - Automatic Place and Route
  - Static Timing Table

Functional Block Diagram



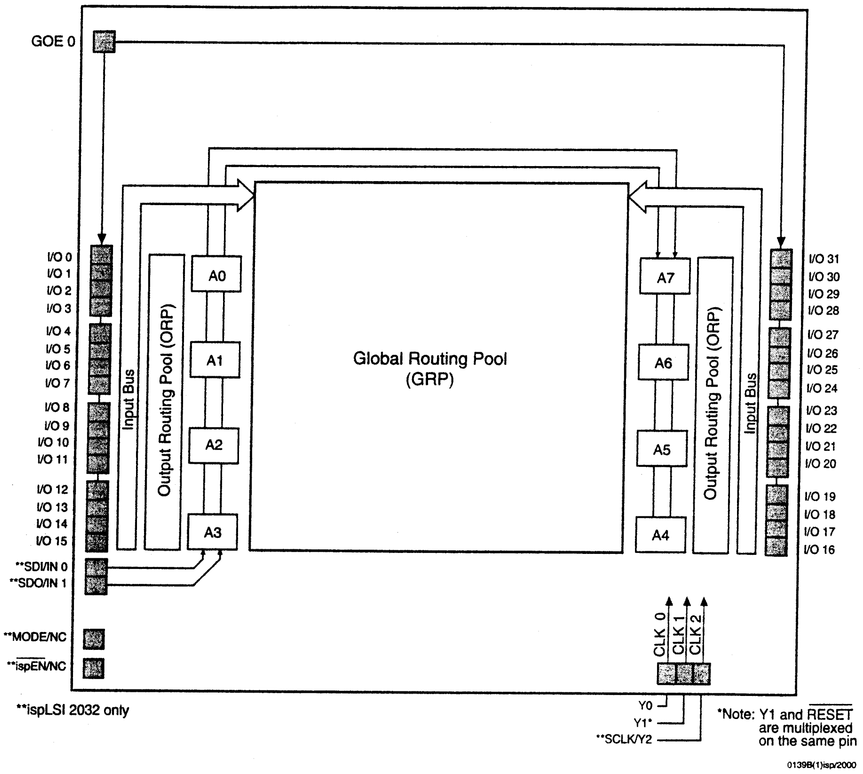
Description

The Lattice ispLSI and pLSI 2032 are High Density Programmable Logic Devices. The devices contain 32 Registers, 32 Universal I/O pins, two Dedicated Input Pins, three Dedicated Clock Input Pins, one dedicated Global OE input pin and a Global Routing Pool (GRP). The GRP provides complete interconnectivity between all of these elements. The ispLSI 2032 features 5-Volt in-system programmability and in-system diagnostic capabilities. The ispLSI 2032 offers non-volatile "on-the-fly" reprogrammability of the logic, as well as the interconnect to provide truly reconfigurable systems. It is architecturally and parametrically compatible to the pLSI 2032 device, but multiplexes four input pins to control in-system programming.

The basic unit of logic on the ispLSI and pLSI 2032 devices is the Generic Logic Block (GLB). The GLBs are labeled A0, A1 .. A7 (see figure 1). There are a total of eight GLBs in the ispLSI and pLSI 2032 devices. Each GLB is made up of four macrocells. Each GLB has 18 inputs, a programmable AND/OR/Exclusive OR array, and four outputs which can be configured to be either combinatorial or registered. Inputs to the GLB come from the GRP and dedicated inputs. All of the GLB outputs are brought back into the GRP so that they can be connected to the inputs of any GLB on the device.

Functional Block Diagram

Figure 1. ispLSI and pLSI 2032 Functional Block Diagram



The devices also have 32 I/O cells, each of which is directly connected to an I/O pin. Each I/O cell can be individually programmed to be a combinatorial input, output or bi-directional I/O pin with 3-state control. The signal levels are TTL compatible voltages and the output drivers can source 4 mA or sink 8 mA. Each output can be programmed independently for fast or slow output slew rate to minimize overall output switching noise.

Eight GLBs, 32 I/O cells, two dedicated inputs and two ORPs are connected together to make a Megablock (see figure 1). The outputs of the eight GLBs are connected to a set of 32 universal I/O cells by the ORP. Each ispLSI and pLSI 2032 device contains one Megablock.

The GRP has as its inputs, the outputs from all of the GLBs and all of the inputs from the bi-directional I/O cells. All of these signals are made available to the inputs of the GLBs. Delays through the GRP have been equalized to minimize timing skew.

Clocks in the ispLSI and pLSI 2032 devices are selected using the dedicated clock pins. Three dedicated clock pins (Y0, Y1, Y2) or an asynchronous clock can be selected on a GLB basis. The asynchronous or Product Term clock can be generated in any GLB for its own clock.

Absolute Maximum Ratings [1]

Supply Voltage V_{CC} .... -0.5 to +7.0V

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

Case Temp. with Power Applied .... -55 to 125°C

1. Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. 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).

DC Recommended Operating Condition

SYMBOL	PARAMETER	MIN.	MAX.	UNITS
T_A	Ambient Temperature	0	70	°C
V_{CC}	Supply Voltage	4.75	5.25	V
V_{IL}	Input Low Voltage	0	0.8	V
V_{IH}	Input High Voltage	2.0	V_{CC}+1	V

Table 2 - 0005/2032

Capacitance (T_A=25°C, f=1.0 MHz)

SYMBOL	PARAMETER	TYPICAL	UNITS	TEST CONDITIONS
C_1	Dedicated Input Capacitance	6	pf	V_{CC} = 5.0V, V_{IN} = 2.0V
C_2	I/O Capacitance	7	pf	V_{CC} = 5.0V, $V_{I/O}$ = 2.0V
C_3	Clock Capacitance	10	pf	V_{CC} = 5.0V, V_Y = 2.0V

Table 2 - 0006

Data Retention Specifications

PARAMETER	MINIMUM	MAXIMUM	UNITS
Data Retention	20	–	Years
ispLSI Erase/Reprogram Cycles	1000	–	Cycles
pLSI Erase/Reprogram Cycles	100	–	Cycles

Table 2 - 0008A-2032-isp

In-System Programmability

The ispLSI devices are the in-system programmable versions of the Lattice high density programmable Large Scale Integration (pLSI) devices. By integrating all the high voltage programming circuitry on-chip, programming can be accomplished by simply shifting data into the device. Once the function is programmed, the non-volatile E^2CMOS cells will not lose the pattern even when the power is turned off.

All necessary programming is done via five TTL level logic interface signals. These five signals are fed into the on-chip programming circuitry where a state machine controls the programming. The simple signals for the interface include isp Enable ($\overline{\text{ispEN}}$), Serial Data In (SDI), Serial Data Out (SDO), Serial Clock (SCLK) and Mode (MODE) control. Figure 4 illustrates the block diagram of one possible scheme of the programming interface for the ispLSI devices. For details on the operation of the internal state machine and programming of the device please refer to Lattice's In-System Programmability Manual.

The device identifier for the ispLSI 2032 is 0001 0101 (15 hex). This code is the unique device identifier which is generated when a read ID command is performed.

Figure 4. ISP Programming Interface

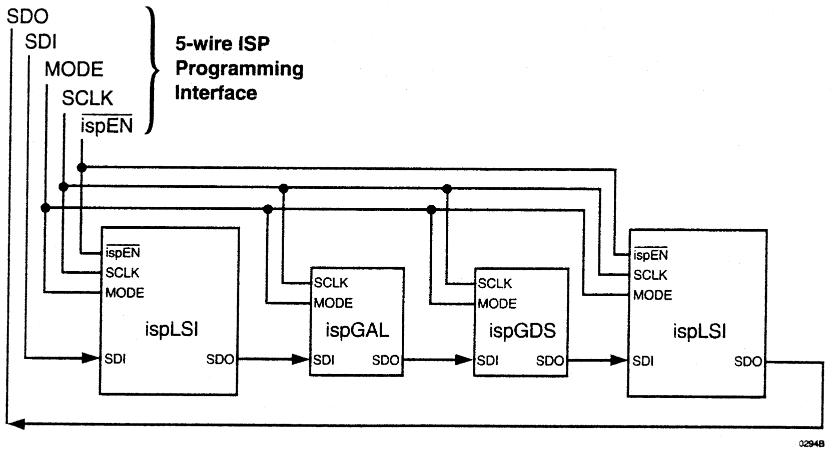
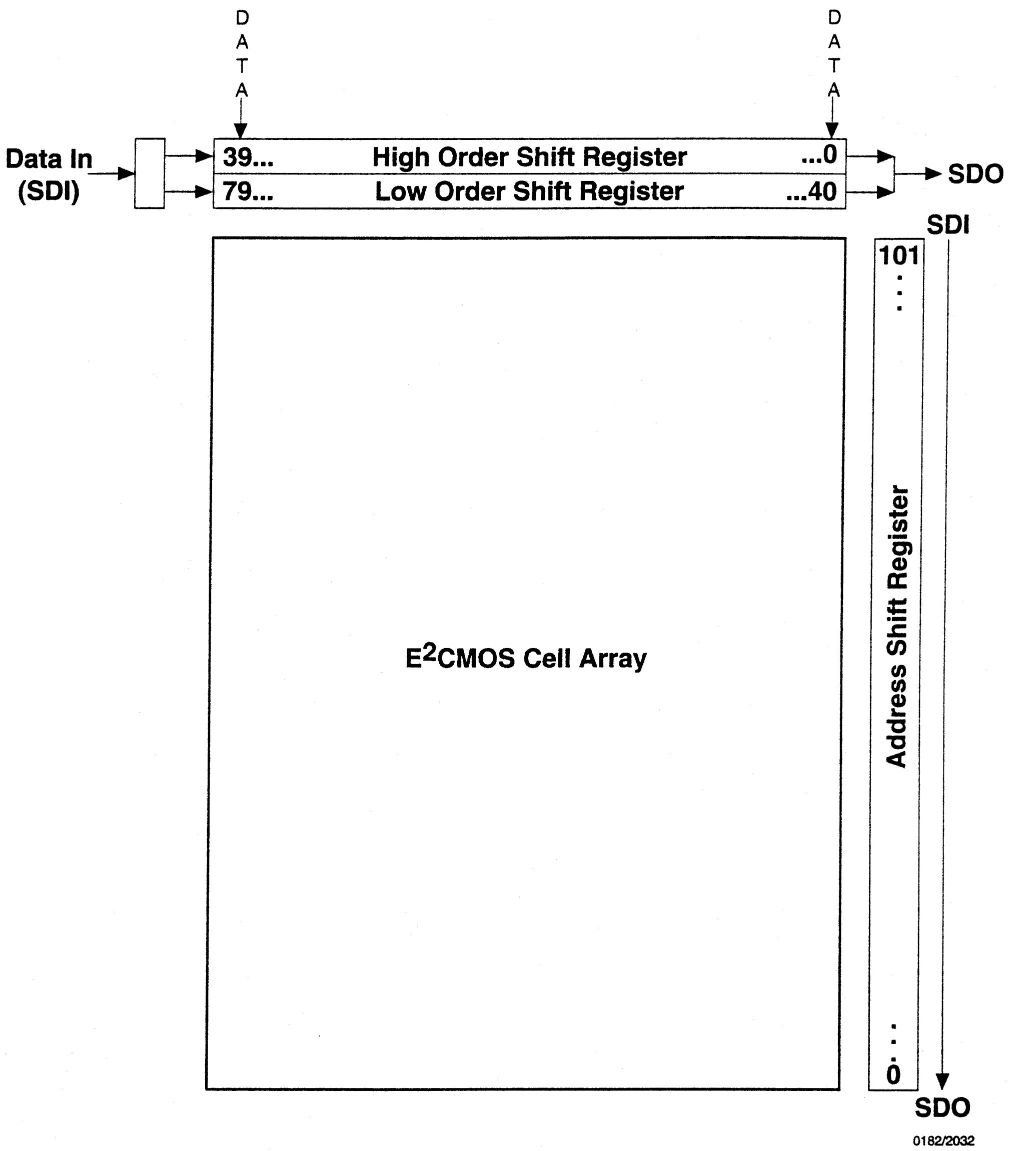


Figure 7 illustrates the address and data shift register bits for the ispLSI 2032. For a detailed explanation refer to the 1994 ISP Manual, Section 2.

Figure 7. ispLSI 2032 Shift Register Layout



Note: A logic "1" in the address shift register enables the row for programming or verification. A logic "0" disables it.

Pin Description

NAME	PLCC PIN NUMBERS	DESCRIPTION
I/O 0 - I/O 3 I/O 4 - I/O 7 I/O 8 - I/O 11 I/O 12 - I/O 15 I/O 16 - I/O 19 I/O 20 - I/O 23 I/O 24 - I/O 27 I/O 28 - I/O 31	15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 31, 32, 37, 38, 39, 40, 41, 42, 43, 44, 3, 4, 5, 6, 7, 8, 9, 10	Input/Output Pins - These are the general purpose I/O pins used by the logic array.
GOE 0	2	Global Output Enable input pin.
Y0	11	Dedicated Clock input. This clock input is connected to one of the clock inputs of all the GLBs on the device.
$\overline{\text{RESET}}$/Y1	35	This pin performs two functions: - Dedicated clock input. This clock input is brought into the Clock Distribution Network, and can optionally be routed to any GLB and/or I/O cell on the device. - Active Low (0) Reset pin which resets all of the GLB and I/O registers in the device.
$\overline{\text{ispEN}}$**/NC	13	Input - Dedicated in-system programming enable input pin. This pin is brought low to enable the programming mode. The MODE, SDI, SDO and SCLK controls become active.
SDI*/IN 0	14	Input - This pin performs two functions. When $\overline{\text{ispEN}}$ is logic low, it functions as an input pin to load programming data into the device. SDI/IN0 also is used as one of the two control pins for the isp state machine. When $\overline{\text{ispEN}}$ is high, it functions as a dedicated input pin.
MODE*/NC	36	Input - When in ISP Mode, controls operation of ISP state-machine.
SDO*/IN 1	24	Output/Input - This pin performs two functions. When $\overline{\text{ispEN}}$ is logic low, it functions as an output pin to read serial shift register data. When $\overline{\text{ispEN}}$ is high, it functions as a dedicated input pin.
SCLK*/Y2	33	Input - This pin performs two functions. When $\overline{\text{ispEN}}$ is logic low, it functions as a clock pin for the Serial Shift Register. When $\overline{\text{ispEN}}$ is high, it functions as a dedicated clock input.This clock input is brought into the Clock Distribution Network, and can optionally be routed to any GLB and/or I/O cell on the device.
GND	1, 23	Ground (GND)
VCC	12, 34	V_{CC}

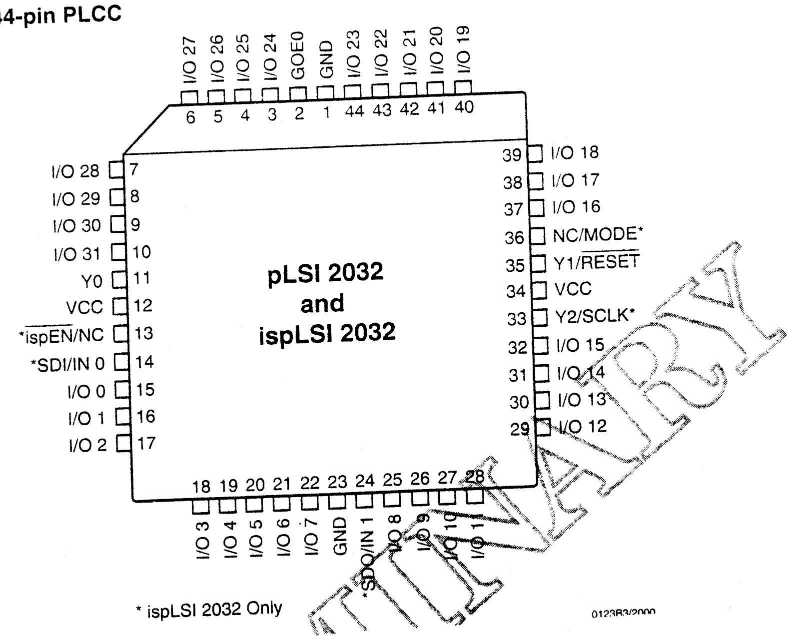
Table 2 - 0002A-08isp/2000

* ispLSI 2032 only

** $\overline{\text{ispEN}}$ for ispLSI 2032 only; NC for pLSI 2032 must be left floating or tied to V_{CC}, must not be grounded or tied to any other signal.

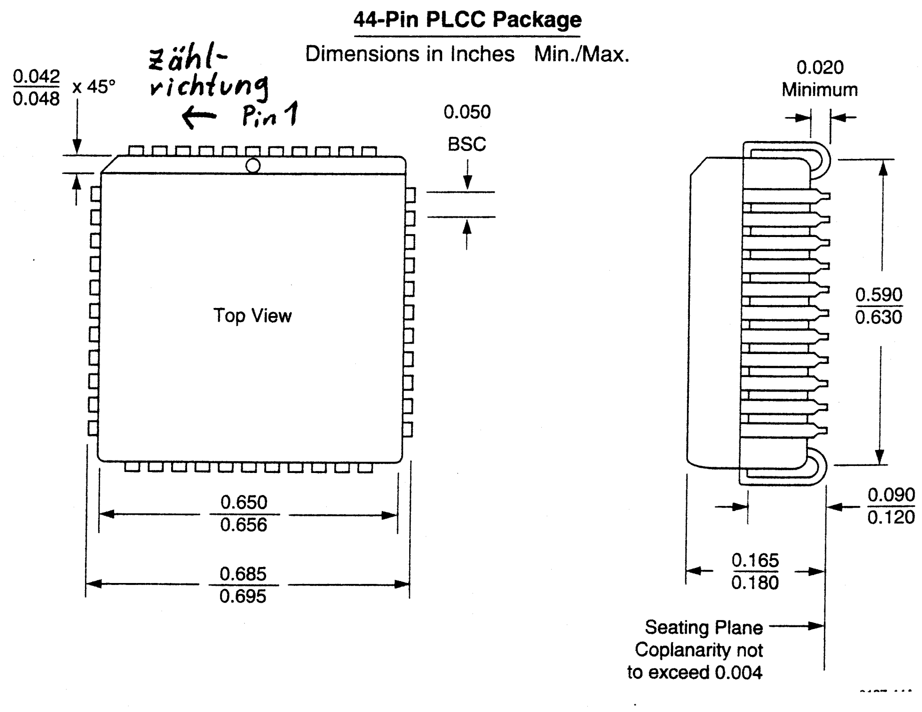
Pin Configuration

pLSI and ispLSI 2032 44-pin PLCC



* ispLSI 2032 Only

Package Diagram



Getting Started

If you have not previously installed the Lattice pDS software, see the *Installation Procedure* in Chapter 1.

1. To start Windows, type **WIN** at the DOS prompt (C:>).
2. Install the Lattice pDS software according to the installation instructions. A new program group called LATTICE is created. This program group should contain three icons. Select the icon which looks like the Lattice company logo (see Figure 3-1).
3. To start the pDS software, double-click on the Lattice Logo Icon.

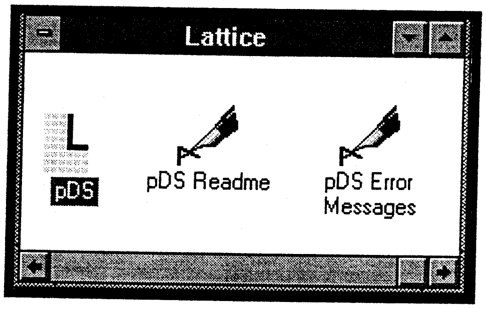


Figure 3-1. Lattice Program Group Window

Before proceeding any further, learn to perform some of the following Microsoft Windows tasks:

- Selecting a menu item using the mouse
- Using Open, Save, and Save As menu items
- Entering commands and text into Message windows and dialog boxes
- Moving around the screen with the scroll bars
- Editing text using the keyboard and mouse to perform the following:
  - Select the insertion point
  - Select text by highlighting it
  - Cut, paste, and copy text

If you are unfamiliar with any of these options, take some time to go through the *Microsoft Windows User's Guide.* If you have ever worked with the Apple Macintosh, you will find that many of the commands and operations are similar. For a complete description of the pDS Design Manager menus, see the *pLSI and ispLSI Development System Reference Manual.*

A Brief Tour of the Screen

Once you invoke the Lattice pDS software, two windows are displayed (see Figure 3-2).

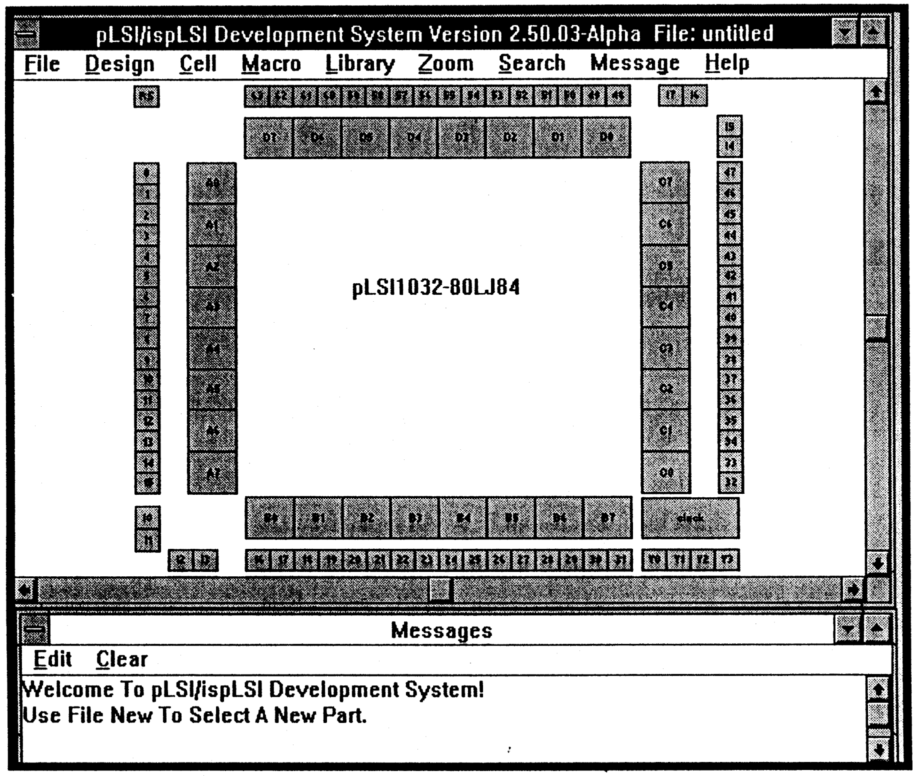


Figure 3-2. Design Manager Window

The larger of the two windows displays a graphic representation of the pLSI 1032 logic diagram. This window is called the Design Manager window. Enter the design by editing equations in the Design Manager window.

The smaller of the two windows is the Messages window, located at the bottom of the screen. The pDS software communicates with you by placing messages in the Messages window.

The part that is displayed in the block diagram shows the elements of the pLSI 1032 that can be modified by the user. These elements are the GLBs, the IOCs, the dedicated input (DI) pins, and the dedicated clock input pins (see Figure 3-2).

Enter the design into the development software by double-clicking on the box that you wish to edit and entering equations or macros (library elements already partitioned and optimized for high performance) into the Edit Cell window that appears (see Figure 3-3).

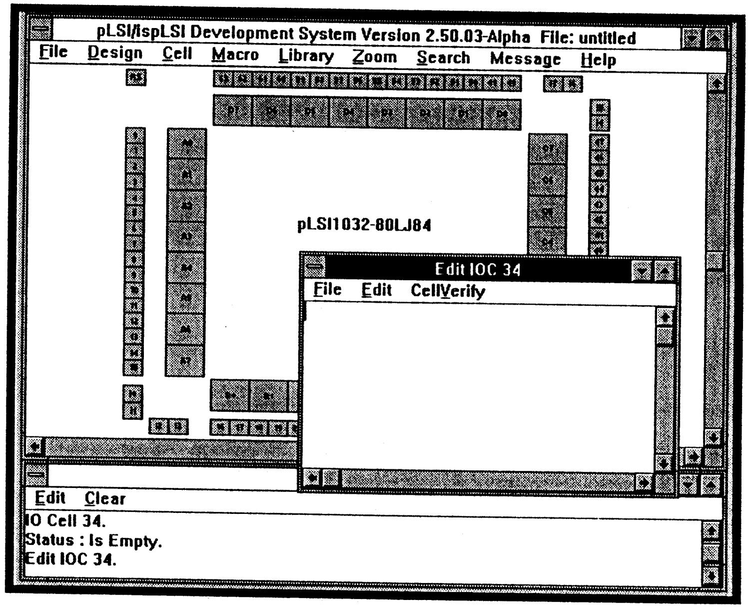


Figure 3-3. Open Edit Windows

The method of specifying the logic configuration into a cell depends on what type of cell it is. Logic configurations for GLBs are entered using a combination of Boolean equations or macros from the Lattice System Library. Configuration data for IOCs are entered using macros only. The complete set of macros which describes all possible combinations of input, output, and IOC logic configurations is contained in the *Lattice System Macro Library*. Configuration data for the dedicated input pins and the dedicated clock input pins are entered using a subset of the IOC macros. Because these pins are inputs only and do not have input registers, many of the standard IOC macros cannot be used.

The Design Flow

Before starting our sample design, it is valuable to understand the design flow. Observe the following steps to complete a design. Refer to Figure 3-4 for more information.

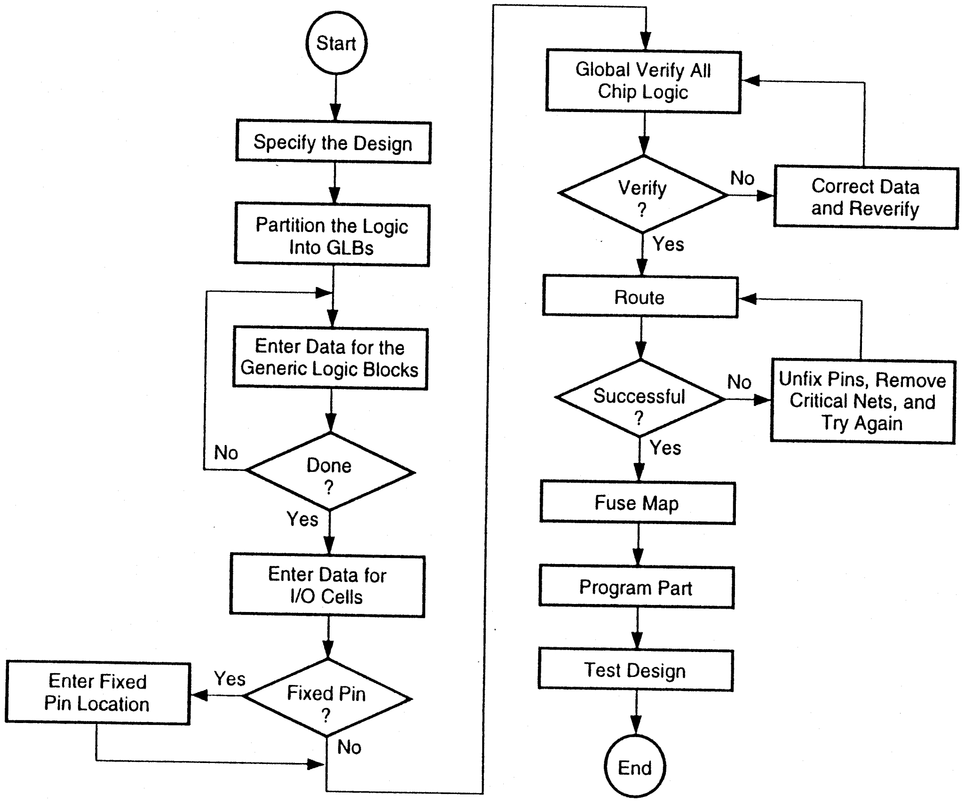


Figure 3-4. Design Process Flow

Specify the Design

A design is specified using one of the two approaches. With the first method, you use an existing design, consisting of 7400 Series TTL Logic elements, and fit the design into the pLSI part. With the second method, you design a circuit using Boolean equations or macros that are optimized for best performance and utilization of the pLSI architecture.

The first approach consists of simply selecting macros from the Lattice library that approximate the functions of the TTL or CMOS circuits and then connecting them to each other. Using this approach, a design can be completed quickly and has a high probability of working the first time because the circuit has been tested.

The second approach ensures better performance and higher utilization, but may require some circuit redesign. Many designs are a combination of the two approaches.

To select the correct pLSI device, partition the design into GLBs, and count the number of GLBs and IOCs used. Next, select the pLSI device that can hold the amount of logic required. Selection of the proper device is based on the amount of logic required and the number of IOCs needed.

The best utilization and routability are achieved by allowing the software to assign the I/O pin placement. It is a good idea to design the pLSI part first, and then lay out the printed circuit board or wire-wrap board after the device has been routed. Once the software intelligently assigns the pin placement the first time, the pins can be fixed, and changes can be made to the logic with few problems.

Partition the Logic

Partitioning consists of carving the logic into chunks that conveniently fit into the pLSI Generic Logic Blocks (GLBs). These general rules should be followed when partitioning logic:

- Consult the *Lattice System Macro Library* and decide if any of the logic can be implemented using the standard macros. Macros are already partitioned and are optimized for high utilization and high performance. Macros are also the fastest method to input logic into the design.
- Know the capabilities of the GLB. For this pLSI 1032 part, there are 16 GRP inputs, 2 dedicated inputs, and 4 outputs. The GLB has 20 product terms (PTs). The registers in the GLB share a common clock. The registers within the GLB also share a common reset product term.
- When an output is fixed to a specific I/O pin, the signal that is used to generate that output must be generated within the same megablock.
- There is only one Product Term Output Enable (PTOE) signal per megablock in our example part. Outputs which share a common PTOE signal should be placed in the same megablock. In the 2000 and 3000 family, use the global OE whenever possible.
- Signals that are related to each other, such as those used for counters, shift registers, etc., should be placed into the same megablock. This reduces routing congestion.

Compile the Design

Compile the Lattice pDS software design by using the following logical sequence:

1. **Entering the design.** Enter Boolean equations or macros into the various cells and blocks on the pLSI device using a built-in text editor. After each cell is entered, do a **CellVerify** to check for syntactical or logical errors within that cell.
2. **Verifying the design.** Verify globally after all the design is entered. Verification looks for such problems as inputs that are not connected to the GLBs or nets that have duplicate names. The design must completely pass a **Design ➠ Verify** before any of the following steps can happen.
3. **Routing the design.** The Router interconnects the GLB and IOC inputs and outputs. The option of fixing certain input and output signals to specific device pins is available.
4. **Generating the Fuse Map.** Using the verified and routed design, create the fuse map (a standard ASCII fuse file) file necessary to program the device. It is a modified format JEDEC file, and the file generated has a suffix of .JED.

Program the Part

Once the design has been compiled, program the part. This can either be done on the board if using in-system programming (ISP) or in a separate programmer. Using a separate programmer requires that the device be first programmed and then placed in the PC board or in the socket of the target system.

Test the Design

To test the design using the trial and error method, put it on the PC board and see if it works correctly. If corrections are needed, redefine the logic in the appropriate GLBs or IOCs, and recompile the design. Or, test the design using a low-cost functional tester. Because all Lattice devices are electrically erasable and reprogrammable devices, the same device can be used again.

The Sample Design

The sample design is a simple one. We are going to design a 4-bit binary counter using Boolean equations and place it into a pLSI 1032 device. We will then take the design through the compilation process, generate a fuse map file (JEDEC), and program a device.

The counter has the following specifications:

- 4-bit synchronous binary counter
- Active high carry in (CI) and carry out (CO) pins
- Active high count enable (CE) pin
- Synchronous reset pin

Figure 3-5 shows the schematic diagram and Figure 3-6 shows the logic symbol for this counter. Because the counter has five outputs (Q0, Q1, Q2, Q3, and CO), it occupies two GLBs.

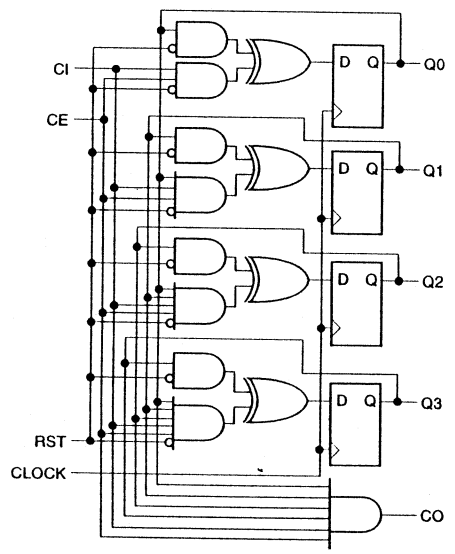


Figure 3-5. Counter Schematic Diagram

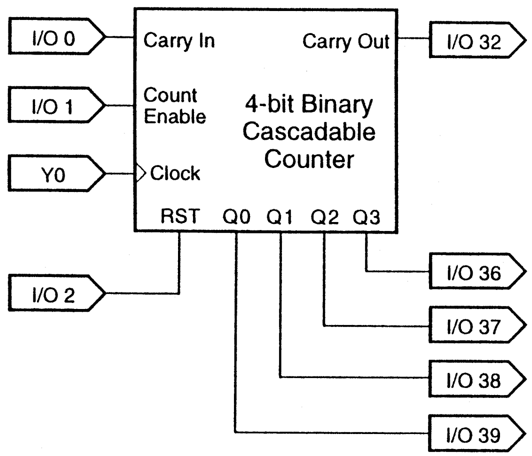


Figure 3-6. Sample Cascadable Counter Logic

In this design example, the clock and I/O pins are compatible with the Lattice ispLSI 1032 Demonstration board. This allows the design to be tested easily.

The input signals CI, CE, and RST are connected to three bits of the 8-bit DIP switch, and the five outputs are connected to five of the discrete LED outputs.

Define the Counter

To define the counter, first write the equations. The equations for the 4-bit binary counter are shown in Listing 1.

Listing 1. Counter Equations

```
Q0 = (Q0 & !I_RST) $$ (CI & CE & !I_RST);
Q1 = (Q1 & !I_RST) $$ (Q0 & CI & CE & !I_RST);
Q2 = (Q2 & !I_RST) $$ (Q0 & Q1 & CI & CE & !I_RST);
Q3 = (Q3 & !I_RST) $$ (Q0 & Q1 & Q2 & CI & CE &
     !I_RST);
CO = Q0 & Q1 & Q2 & Q3 & CI & CE;
```

There are two inputs to the Exclusive-OR gate in front of the D input to the register see (Figure 3-5). The data input is the one that receives its input from the feedback of the same register. It is left of the $$ (XOR) symbol in the above equations. The other input is connected to the control terms Cascade In and Count Enable and are the control input. When the control input to the XOR is a 0, the output of the XOR follows the data input (Hold). When the control Input is a 1, the output of the XOR is inverted from the input (Increment).

When either CI or CE is low and RST is low, the Q0 output from the counter remains in its current state (Hold). When CI and CE are both high and RST is low, the Q0 output toggles on each successive clock (Increment). When RST goes high, the inputs to the data side of the XOR gate and the control side go low. This causes the output of the counter to go low on the next clock edge (Reset).

Each successive stage operates similarly, except during transition (Increment), when the outputs of all previous stages are at logic level one. The Carry Out signal is only generated when all the stages have reached a 1 and both Cascade In and Counter Enable are a 1.

Enter the Equations

Once the equations have been defined, enter them into the GLBs. Follow these steps:

1. From within Windows, double-click on the Lattice Icon to start the pDS software (Figure 3-1).
2. When the Lattice software starts, it displays the block diagram of the pLSI 1032 part. Double-click on the GLB C1 box to open it for editing. The empty Edit Cell window opens.
3. Enter the equations shown in Listing 2 into the Edit Cell window for GLB C1. See Figure 3-7.

Listing 2. GLB Equations

```
SIGTYPE Q0 REG OUT;
SIGTYPE Q1 REG OUT;
SIGTYPE Q2 REG OUT;
SIGTYPE Q3 REG OUT;
EQUATIONS
 Q0.CLK = I_CLK;
 Q0 = (Q0 & !I_RST) $$ (CI & CE & !I_RST);
 Q1 = (Q1 & !I_RST) $$ (Q0 & CI & CE & !I_RST);
 Q2 = (Q2 & !I_RST) $$ (Q0 & Q1 & CI & CE & !I_RST);
 Q3 = (Q3 & !I_RST) $$ (Q0 & Q1 & Q2 & CI & CE &
      !I_RST);
END;
```

4. Verify the equations by clicking on the **CellVerify** menu option. If an error appears in the Messages window, find out what is wrong and correct it. Things to look for are typing errors, missing semicolons, or incorrect symbols. Reverify after making corrections.
5. Select **File ➟ Update** from Edit Cell menu. **Update** saves the edits you make and closes the Edit Cell window.

❖ **TIP:** You may also close the Edit GLB C1 window by double-clicking on the Control Menu box (the dash mark in the top left corner). Your edits are automatically saved.

6. Double-click on the GLB C2 box to open it for editing.

7. Enter the following equations into the Edit GLB C2 window:

```
SIGTYPE CO OUT;
EQUATIONS
   CO = Q0 & Q1 & Q2 & Q3 & CI & CE;
END;
```

8. Verify the equations by clicking on the **CellVerify** menu option. See Figure 3-7.

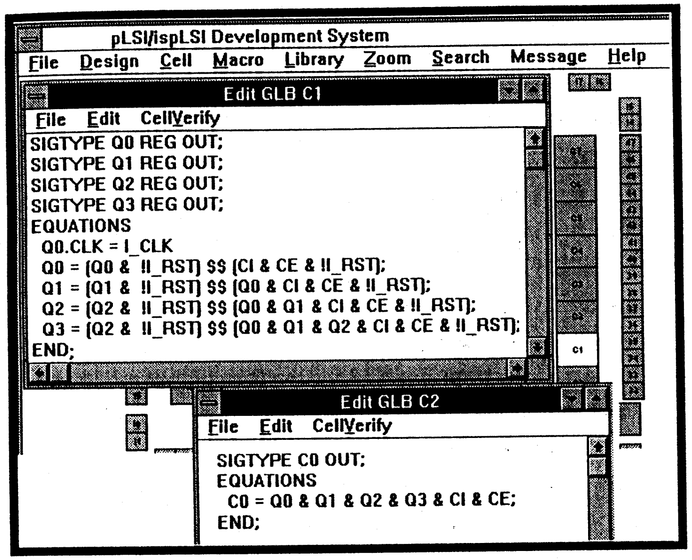


Figure 3-7. Cell Entry Windows with Counter Equations

9. Select **File** ➟ **Update** from Edit Cell menu. **Update** saves the edits you make and closes the Edit Cell window.

Connect the Inputs, Outputs, and the Clock

At this point, the logic for the counter is completely specified, but we still must connect the clock, the inputs, and the outputs.

10. Double-click on clock input Y0 to open it. It may be necessary to **Zoom** ➟ **In 2X** on the dedicated clock input area of the Logic Design Editor (the GUI) to determine which pin is Y0.

11. Enter the following equations into the Edit Cell window for dedicated clock input Y0:

```
XPIN CLK XI_CLK LOCK 20;
IB11 (I_CLK, XI_CLK);
```

Since this design example uses a pLSI 1032 device, the Y0 clock is pin 20, therefore the clock signal is locked to pin 20.

12. Verify the equations by clicking on the **CellVerify** menu option.

13. Select **File** ➟ **Update** from Edit Cell menu. **Update** saves the edits you make and closes the Edit Cell window.

14. Repeat Steps 10 through 13 for the Carry In input pin located at I/O 0 using these equations:

```
XPIN IO XCI LOCK 26;
IB11 (CI, XCI);
```

15. Repeat Steps 10 through 13 for the Count Enable input pin located at I/O 1 using these equations:

```
XPIN IO XCE LOCK 27;
IB11 (CE, XCE);
```

16. Repeat Steps 10 through 13 for the Reset input pin located at I/O 2 using these equations:

```
XPIN IO XI_RST LOCK 28;
IB11 (I_RST, XI_RST);
```

17. Repeat Steps 10 through 13 for the Q0 output pin located at I/O 39 using these equations:

```
XPIN IO XQ0 LOCK 75;
OB11 (XQ0, Q0);
```

18. Repeat Steps 10 through 13 for the Q1 output pin located at I/O 38 using these equations:

```
XPIN IO XQ1 LOCK 74;
OB11 (XQ1, Q1);
```

Note: With the Lattice pDS software, you can have only two Edit Cell windows open at the same time. This means that you can **Copy** the equations from IOC 39 and **Paste** them into IOC 38. The data in both cells are similar, and you can use the Edit Cell's **Edit** commands to make changes.

19. Repeat Steps 10 through 13 for the Q2 output pin located at I/O 37 using these equations:

```
XPIN IO XQ2 LOCK 73;
OB11 (XQ2, Q2);
```

20. Repeat Steps 10 through 16 for the Q3 output pin located at I/O 36 using these equations:

```
XPIN IO XQ3 LOCK 72;
OB11 (XQ3, Q3);
```

21. Repeat Steps 10 through 13 for the Carry Out output pin located at I/O 35 using these equations:

```
XPIN IO XCO LOCK 68;
OB11 (XCO, CO);
```

Now, the inputs, outputs, and clocks are connected, and the equations for the counter have been entered and verified. The design is complete and ready to be globally verified. Before proceeding, save your design.

Save the Work

22. From the Design Manager menu bar, select **File** ➟ **Save As**. The pDS software prompts you for the name of the file that you are saving. Type in the name COUNTER. The suffix .lif (Lattice Internal Format) is automatically appended. Select the drive and the directory. Click OK. Check the Messages window (Figure 3-9) for a successful verification message.

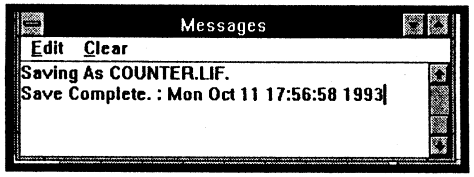


Figure 3-8. Successful Design Save

Verify and Route the Design

The next step in the development process is to globally verify the integrity of the design. Global verify first performs a **CellVerify** on GLBs or IOCs which have not already been verified, or which have changed since the last cell verification. Then it checks interconnections between the GLBs, looking for problems such as outputs which are not used or inputs that are not connected.

23. From the Design Manager menu bar, select **Design** ➟ **Verify**. This starts the global verify process. If **Verify** finds any problems, it lists them in the Messages window at the bottom of the screen. The verifier also creates a netlist file that the Router uses to route the design. Once the design passes **Verify**, it is ready to be routed (Figure 3-9).

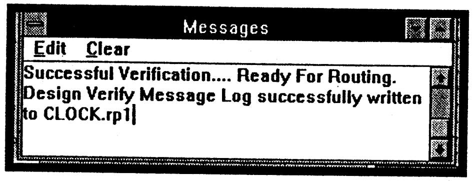


Figure 3-9. Successful Design Verification Message

24. From the Design Manager menu bar, select **Design** ➟ **Route**. This module places I/O pins that have not previously had their positions defined, and interconnects all the logic blocks and IOCs on the device. When **Route** is invoked, a list of all the I/O pins displays. If you have not previously defined which signals are connected to which pins, this is the time to do it.

Fast Route enables the Router to work very quickly, prioritizing speed over the best, most efficient design. **Strong** enables the Router to find the better design, prioritizing the best design layout over the amount of time it takes to finish the route. Routing is an entirely automatic process, and requires no intervention. As before, problems are listed in the Messages window.

Generate the Fuse map

25. The last step in the compilation process is to generate the fuse map. From the Design Manager menu bar, select **Design** ➠ **Fuse Map**. Like **Route**, **Fuse Map** is an entirely automatic process, and should require no intervention once you set the options. Selecting the pull up control for specific pins or individual pins is available. The output from the fuse map program is the .jed file used to program the part.

The design is now complete. Because it was given a name previously (`COUNTER.lif`), you can simply click on the **Save** command in the **File** menu to save the work. A backup file is automatically generated for you (called *design*.lbk) All that remains is to generate a JEDEC file, program the part, and test the design.

Review of the Syntax

This is a brief review of the syntax used in the example design. For complete information, see Chapter 3, "LDF Language Reference" in the *pLSI and ispLSI Development System Reference Manual.*

The operators that the Lattice software uses are similar to those used by the Data I/O ABEL program with the addition of $$. The operators and an example of how they are used are shown in Table 3-1. The Precedence of Evaluation is also shown where 1 is the highest precedence. The highest precedence will evaluated first.

Table 3-1. Precedence of Evaluation

Operator	Precedence	Description	Example
!	1	NOT	!A
$$	2	XOR (XOR Gate in GLB	A $$ B
&	3	AND	A & B
#	4	OR	A # B
$	5	XOR (Soft)	A $ B
!$	5	XNOR (Soft)	A !$ B

In addition to the equations, the following other lines need to be included in the GLB or IOC definition:

`SYM;` The symbol line consists of five parts:

- Keyword SYM, indicates what type of line this is to be.
- Symbol Name, either GLB or IOC.
- Cell location.
- Symbol Level, used by other software packages. For our purposes, always use 1.
- Symbol User Name, an assigned name that appears in the GLB or IOC in place of its location designation.

`SIGTYPE;` Used to define signal attributes within a GLB. OUT defines a combinatorial output. REG OUT defines a Registered Output.

`EQUATIONS;` Indicates the start of the EQUATION section for a GLB.

`MACRO;` Indicates the usage of a macro logic element from the *Lattice System Macro Library.*

`END;` Signifies the end of an EQUATION section, a GLB or IOC definition, a DECLARATION section, or a macro definition. There can be more than one END statement in a GLB.

Comments are indicated by preceding the comment with two forward slashes:

// This is a comment line example.

Programming the Device

The **Fuse Map** program generates a fuse (.jed) file that is programmed into a pLSI 1032 or an ispLSI 1032 device. Program the part using one of the following methods:

- Programmer for pLSI and ispLSI device
- In-system programming for ispLSI device
- A stand-alone device

For a stand-alone programmer, follow the programmer's manufacturing instructions.

An ISP down load cable is offered as an option with the pDS software. The cable connects to the parallel port on a PC and controls the programming process. If the target system is designed to use in-system programming (ISP), the part can be programmed right on the board.

All ISP devices can be programmed using the following methods. To program a pLSI/ispLSI 1032 with a counter.jed JEDEC file, follow these steps:

1. Select **Design ➠ Down Load** from the Design Manager menu. This invokes the Down Load window.

   The Down Load window prompts for the name of the JEDEC file.
2. Select `COUNTER.JED` from the file list and then click OK. It may already have `COUNTER.JED` as the default file name. If so, then just click OK.

Programming takes a few seconds. If an error is encountered, it is listed in the Messages window.

When programming is complete, the device is reset and returned to the operating mode. It can then be tested by applying the required input stimulus and looking at the outputs.

Language Elements and Conventions for Defining Logic

This section provides a description of the pDS software language elements. To effectively use the pDS software language elements, familiarize yourself with the basic syntax information contained in the *pLSI and ispLSI Development System Reference Manual*, Chapter 2, "LDF Language Reference."

- Naming and Notation Conventions
- Boolean Logic Operators
- GLB Dot Extensions
- Keyword Summary

Conventions

Conventions are the rules that govern design entry for names and notations.

Notation Conventions

Notation conventions define the style and format for syntax entries and are given in Table 4-1.

Table 4-1. Notation Conventions

Symbol	Definition
<X>	X is an identifier
[X]	X is optional
{X}*	One or more repetitions of X
X | Y	A choice, X or Y
UPPERCASEBOLD	Keyword or delimiting punctuation
lowercasebold	Keyword - for internal use only
italic	Comments

Naming Conventions

Signal and pin names cannot be longer than 31 characters, including the dot extension. Legal characters in a signal name include the following:

- Alphabetic characters
- Numerals
- Underscore "_"

Starting characters for a signal name must be either the underscore "_" or alphabetic. Names are **not** case sensitive but cannot start with a digit. These constraints are consistent with Electronic Design Interchange Format (EDIF), which eliminates conversion difficulties.

Boolean Logic Operators

Boolean logic is the means of syntax entry for all EQUATIONS sections. Design entry using Boolean logic is the most fundamental level of input. A list of Boolean operators is shown in Table 4-2. Operator precedence is also shown. You can use parentheses to override precedence. XOR ($) and XNOR (!$) will be expanded to sum-of-products. If operator precedence is the same, it will be processed from left to right.

Table 4-2. Boolean Logical Operators

Operator	Precedence*	Description	Example
!	1	Inversion (NOT)	!A
$$	2	XOR (XOR gate in GLB)	A $$ B
&	3	AND	A & B
#	4	OR	A # B
$	5	XOR	A $ B
!$	5	XNOR	A !$ B
=		Assignment**	A = B

* Precedence 1 is the highest and will be evaluated first.
** Must be left-most operator. Can be used only once in an equation.

Rules for using the ! operator

Rule 1 - Do not use ! on the left side of an equation or in a SIGTYPE statement. Define the complement of an equation (C # D) as an asynchronous output of a GLB.

INCORRECT ! USAGE:

```
SYM GLB A0 1;
   SIGTYPE A OUT;
   EQUATIONS
      !A = ( C # D ); // the ! is on the left side
   END;
END;
```

INCORRECT ! USAGE:

```
SYM GLB A0 1;
   SIGTYPE !A OUT; // the ! is in a SIGTYPE statement
   EQUATIONS
      A = ( C # D );
   END;
END;
```

CORRECT ! USAGE:

```
SYM GLB A0 1;
   SIGTYPE A OUT;
   EQUATIONS
      A = !( C # D ); // the ! is on the right side
   END;
END;
```

Rule 2 - Do not use ! as part of a name of an input name of an input macro (IB11, IB21, and so on) or in the XPIN statement.

INCORRECT ! USAGE:

```
SYM IOC IO0 1;
XPIN IO !XA; //! in an XPIN statement
IB11 (A,!XA); //! in input name
END;
```

CORRECT ! USAGE:

```
SYM IOC IO0 1;
XPIN IO XA;
IB11 (!A,XA);
END;
```

Rule 3 - Do not use ! as part of an input name of an output macro (OB11, OT11, and so on) or in the XPIN statement.

INCORRECT ! USAGE:

```
SYM IOC IO0 1;
XPIN IO XQ0;
OB11 (XQ0, !Q0)); //! in input name
END;
```

CORRECT ! USAGE: (use OB21 instead of ! and OB11.

```
SYM IOC IO0 1;
XPIN IO XQ0;
OB21 (XQ0, Q0));
END;
```

GLB Dot Extensions

The GLB dot extensions are defined in the Declaration and the Partitioned sections of the source file. Use dot extensions for GLBs to denote specific architectural connections, and define specific internal nodes and signals within the design. GLB dot extensions are defined in Table 4-3. Use the syntax *signal_name*.ext to enter a dot extension. The GLB has D flip-flops, a product term for IOC output enable (OE), a product term clock, a product term reset, and combinational feedback, all of which can be utilized using the dot extensions. Use dot extensions within GLBs and macros to refer to circuit elements such as register clocks and OEs. These extensions are also used to connect them to a selected signal.

The D flip-flop in a GLB has a clock that is driven by either a product term clock or a global clock. It also has a reset that is driven by product term. These features are supported with the dot extensions .clk, .ptclk, and .re.

Table 4-3. GLB Dot Extensions

Dot Ext.	Description	Left of =[1]	Rt of =[2]	Macro Usage[3]	Active Hi/Lo[4]
.d	Identifies the signal as the D input of a flip-flop	X	X	X	Either
.q	Identifies the signal as the Q output of a flip-flop		X	X	Either
.pin	Defines the signal as a combinatorial logic feedback into the Global Routing Pool		X	X	Either
.re	Identifies the signal as the active high product term reset signal for all the registers in the GLB	X		X	Hi only
.clk	Identifies the system clock to be used for all registers in the GLB	X		X	Either
.ptclk	Identifies the signal as the product term clock for all registers in the GLB	X		X[5]	Either
.oe	Identifies the signal to be used as the OE signal for the IOCs in the megablock within which it is located	X			Either

Table 4-3. GLB Dot Extensions

1. These dot extensions can be used on the left side of the equation.
2. These dot extensions can be used on the right side of the equation.
3. These dot extensions can be used to define the input/output signals for a macro being used in a GLB or an IOC.
4. These dot extensions are implied for a registered signal that appears on the right side of an equation.
5. These .ptclk or .re dot extensions can be used in a macro usage statement only for a parameter position that corresponds to the clock or reset input for the macro.

For more information about dot extensions, see the *pLSI and ispLSI Development System Reference Manual.*

Keyword Summary

LDE syntax keywords are reserved words you use to define different sections or specific commands in the source file. Table 4-4 summarizes the LDE syntax keywords and a brief definition. A detailed description for each table entry is given in the *pLSI and ispLSI Development System Reference Manual.*

Table 4-4. LDE and LDF Keyword Summary and Definitions

Keyword	Definition
AUTHOR	Identifies the design owner
COMMENT	Statement of explanation
CONSTANT (Global & Local)	Assigns a non-changing value to a named identifier
DECLARE	Indicates the beginning of a Declaration Section
DESCRIPTION	Optional, denotes start of comment section
DESIGN (Name)	Indicates name of design
END	Signifies the end of the .LDF design, equations sections, GLB section, macro section, declare section, or an IOC.
EQUATIONS	Starts equation section for GLB, IOC, or macro
LDF	Identifies design file version and type
LIBRARY	Names the libraries containing design macros
MACRO (Creation)	Defines and creates user macro library
MACRO (Usage)	Predefines a logic block in a macro definition, GLB, or IOC
MACROCOMMENT	Identifies comment section of macro

Table 4-4. LDE and LDF Keyword Summary and Definitions

Keyword	Definition
MACROFAMILY	Organize MACRO by family type
MACROGROUP	Organize MACRO by function type
MACROTYPE	Defines access permissions during macro creation
X	Denotes macro can be used only
RX	Denotes macro can be read or used only
RWX	Denotes macro can be read, copied, modified, etc.
OPTION (Programming)	Indicates programming options that affect processing the design file
PART	Identifies device type
PROJECTNAME	Defines project name of design file
REVISION	Indicates the design revision level (optional)
SET (Global & Local)	Enables a group of signals to function as one unit
SIGTYPE	Define GLB or macro signal definitions
ASYNC	Asynchronous signal directing the software not to duplicate a signal to enhance routing
CRITICAL/CRIT	Identifies pin or signal using critical path
FASTMIN	Fast minimization
NOMIN	No minimization
OE	Output Enable
OUT	Signal output of GLB
REG	Register

Table 4-4. LDE and LDF Keyword Summary and Definitions

Keyword	Definition
STRONGMIN	Strong minimization (exhaustive)
SYM	Defines a GLB or an IOC component
GLB	Keyword for a GLB cell
GND	Indicates a reserved constant for logic 0
IOC	Keyword for IO, DI, clock, and global OE cells
VCC	Indicates a reserved constant for logic 1
XPIN	Specifies IOC external pin name and attribute
LOCK	Specifies the pin and IOC that should be used
CRITICAL/CRIT	Specifies that the output routing pool should be bypassed
SLOWSLEW	Slowslew rate is selected for the Xpin
PULLUP	Pullup is selected for the Xpin

Design Entry Process

The Lattice Design Manager window is the graphic user interface (GUI). The remainder of this chapter discusses how to employ the Design Manager to enter your logic into the design.

To start the program, double-click on the pDS icon in the Microsoft Windows application (Figure 4-1).

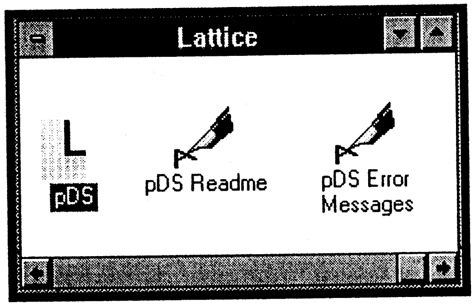


Figure 4-1. pDS Icon on the Windows Desktop

The Design Manager window shows the pDS program name, the software release number and date, and the design file name in the banner. The menu bar contains menu topics related to the functions you use to create the design. Access the various pull-down menus using the mouse or by pressing the Alt key and the desired underlined character. Within any dialog box, use the Tab key on the keyboard to move the cursor to different fields. Then use the arrow keys to scroll down a menu to make your selection. Press the underlined key to select.

Select the Design Manager submenus and options by clicking on them with the left mouse button or by entering the underlined character key. The dots following some of the menu items in the pull-down menus indicate there is additional information in another window associated with this menu item.

The Messages window welcomes you to the system and prompts you to select a new part (Figure 4-2).

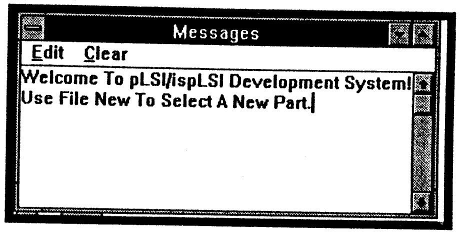


Figure 4-2. Messages Window

The **File** menu contains functions to create, open, save, import (from .ldf), export (to .ldf), print, setup design header information, generate reports, delete a design, and exit (Figure 4-3).

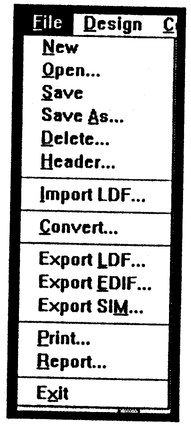


Figure 4-3. **File** Menu Window

Creating a Design

Use **File ➟ New** (Figure 4-4) to create a new, untitled design. The **File ➟ New** menu allows you to perform the following functions:

- Select a part version for your new design
- Change to a different speed grade
- Change to a different package
- Change to the ISP family of a part
- Select a grade of device (i.e., commercial, industrial, or military)

The following section quickly directs you on how to select a new part. The selection boxes are described in detail following the procedure.

To create a new design:

1. Select **File ➟ New**.

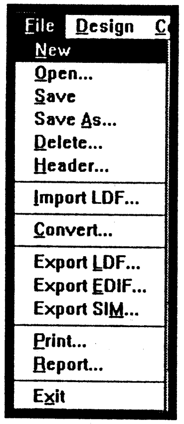


Figure 4-4. **File ➟ New** Menu Item Window

2. Check the boxes for ispLSI or pLSI devices and choose the 1000, 2000, or 3000 part family. The field optimizes once you qualify the selection. See Figure 4-5.
3. Select a part from the list. The part is highlighted and your selection changes the lower boxes to reflect that particular device's options (i.e., speed, grade, power, etc.).
4. Select a family of part types (pLSI or ispLSI) from the **Family** box.

❖ **TIP:** Use the down arrow for any box to make a selection and your choice is highlighted in the box head.

5. Select a **Part Version**.
6. Select a **Speed**.
7. Select the **Power**.
8. Select a **Package**.
9. Select a **Grade**.
10. Click OK.

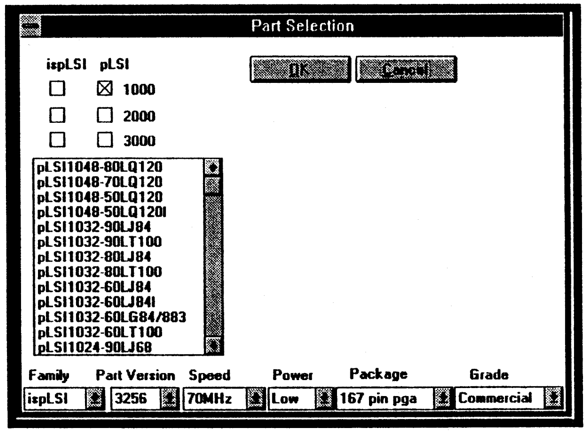


Figure 4-5. Part Selection Window

Now the new device's graphic representation window appears on the screen. You can begin entering your logic into the cells. If you select **Cancel**, the Part Selection window closes and the Design Manager window displays the original part.

To open an Edit Cell (logic entry) window, see *Editing a Cell* later in this chapter. To save a design, see *Saving a New Design* later in this chapter. The cells are all empty, so you cannot verify. To see logic strings, macros, and syntax, import a design file from the pDS software package kit. To import a design file, see *Using Import LDF* in Chapter 11.

Part Families

When the Part Selection window appears showing various fields from which to choose, the Lattice pDS software offers you the ability to narrow your parameters of part selection. Specifically, checking the device type boxes greatly qualifies the options and speeds up the process. The following options are available:

pLSI 1000 series	ispLSI 1000 series
pLSI 2000 series	ispLSI 2000 series
pLSI 3000 series	ispLSI 3000 series

Part Version

This box lists the types of parts available to you for the particular device and family series you chose. Presently, Lattice offers the following parts:

pLSI/ispLSI1016	pLSI/ispLSI1024	pLSI/ispLSI1032
pLSI/ispLSI1048	pLSI/ispLSI2032	pLSI/ispLSI3256

Speed

Currently, Lattice offers devices with speed grades of 135, 110, 90, 80, 70, 60, and 50 MHz. Use the down arrow to select the speed grade you need. It is highlighted in the box head.

Power

Lattice only offers low power devices.

Package

This list box gives the package type and pins available for the part you chose.

Grade

This list box offers commercial, industrial, and military device types.

Selecting Design Options

Immediately after you click OK in the select the Part Selection window, the Set Design Options window appears (Figure 4-6). This window shows the minimization options for the design set during verification.

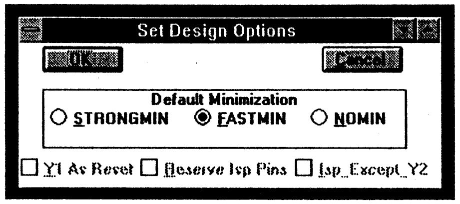


Figure 4-6. Set Design Options Window

The Set Design Options window can be invoked anytime throughout the design process by selecting **Design ➟ Options** (Figure 4-7).

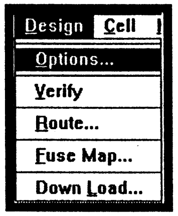


Figure 4-7. **Design ➟ Options** Menu Item Window

To set the Design Options:

1. Select the minimization default.
2. Select the ISP options from the check boxes, if available for that device.
3. Click OK.

The window in Figure 4-6 sets minimization attributes for the design as follows:

- **STRONGMIN** selects an exhaustive, Berkeley Espresso minimization
- **FASTMIN** provides a time-saving, single-pass minimization process (default)
- **NOMIN** bypasses the minimization process

The default is **FASTMIN**. Depending on which device you are using, the **Y1_AS_RESET** box, **Reserve Isp Pins** box, or the **Isp_EXCEPT_Y2** boxes may be enabled or be grayed (not available). These options involve certain device features and affect the routability if you select them. For more information on the verification options that affect routability, see Chapter 6, "Design Verification," and Chapter 7, "Design Routing."

✍ **Note:** Default minimization characteristics are only assumed by equations that have not been assigned a minimization attribute in the logic text. To assign minimization on an output-by-output basis, refer to Chapter 2, "Language Reference," in the *pLSI and ispLSI Development System Reference Manual.*

Setting Header Information

After the Design Options window closes, the Design Header Title Block window automatically appears (see Figure 4-8).

To set header information:

1. Select **File ➠ Header** (Figure 4-9).
2. Type in the information for the desired fields.
3. Click OK.

Filling in all of this information is optional. Click OK to accept the defaults.

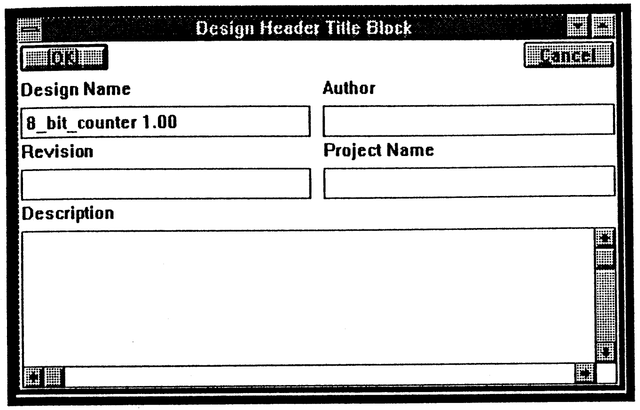


Figure 4-8. Design Header Title Block Window

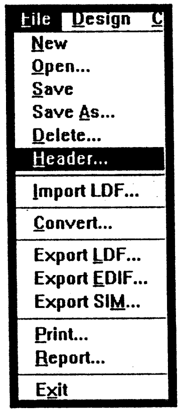


Figure 4-9. **File ➠ Header** Menu Item Window

Use the Design Header Title Block window to enter the following design information:

- **Design Name** – Name of your design
- **Revision** – Revision number or date
- **Author** – Designer's name
- **Project Name** – Name of project

- **Description** – User design notes window

If you currently are working on a design, and do not save it before starting a new one, you are prompted in the Messages window to name and save the current design first (Figure 4-10). It is good practice to periodically save your design. To save a design, see *Saving a New Design* next.

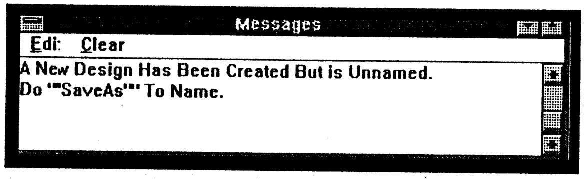


Figure 4-10. Save New Design Prompt Message Window

Saving a New Design

The **File ➡ Save As** function (Figure 4-11) prompts you to enter a file name before saving a new design. In case you forget to name your design and choose **Save** from the **File** menu, the File Save As window appears.

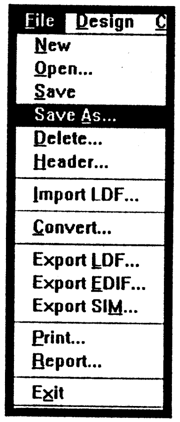


Figure 4-11. **File ➡ Save As** Menu Item Window

Or, use **File ➡ Save As** to save a copy of the open design under a different file name or to a different directory (Figure 4-12).

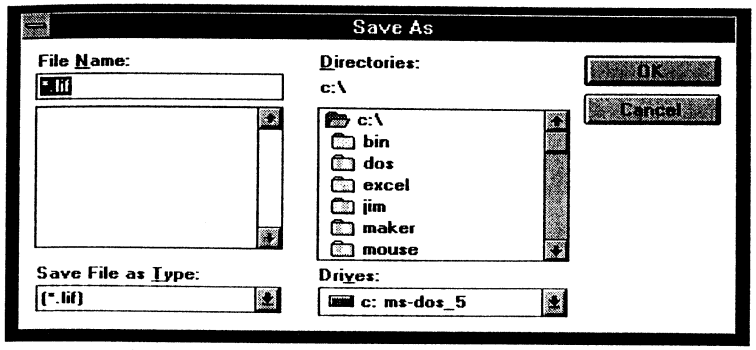


Figure 4-12. **File ➡ Save As** Window

To perform a Save As:

1. Select **File ➡ Save As**. See Figure 4-11.
2. Enter the name of the design in the File Name field.
3. Select the desired directory. Use the up and down arrows, if necessary.
4. Select a file type from the Save File As Type pull-down menu.
5. Select the drive from the pull-down menu.
6. Click OK.

If you save to a file that is already in the current directory, you are prompted to replace the existing file (Figure 4-13). Click OK to replace it or Cancel to return to the Save As File Name window. The new "**Save As**" design name and its path become the current working design and path.

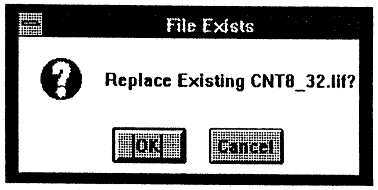


Figure 4-13. File Exists Window

Now that you have completed opening a design, naming and saving it to a directory on a specific drive, you can begin entering your design logic. Using the Design Manager menu, you can create and edit your design easily. After you are satisfied with the syntax, perform a verification, checking each cell specifically or globally. Global logic verification is explained in detail in Chapter 6, "Design Verify." Individual cell verification is discussed in the following section, *Logic Entry*.

Logic Entry

The **Cell** menu (Figure 4-14) provides functions to enter and edit Boolean equations or macros in a cell, as well as functions to manipulate information between cells. Any cell you select becomes highlighted in the Design Manager window to indicate it is activated. When using **Cell** menu functions, you are prompted to select the cell in which to perform the desired function.

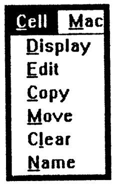


Figure 4-14. **Cell** Menu Window

Editing a Cell

To enter equations or macro calls for a GLB or an IOC:

1. Select **Cell** ➟ **Edit** (Figure 4-15).

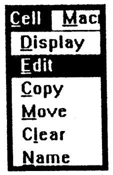


Figure 4-15. **Cell** ➟ **Edit** Menu Item Window

2. Select the desired GLB or IOC. Its Edit Cell window appears (Figure 4-16).

❖ **TIP:** Double-click on the GLB or IOC box from the GUI to invoke its Edit Cell window.

3. Enter your logic.
4. Click **CellVerify**.
5. Check the Messages window for successful verification notice. Follow the instructions in the window.

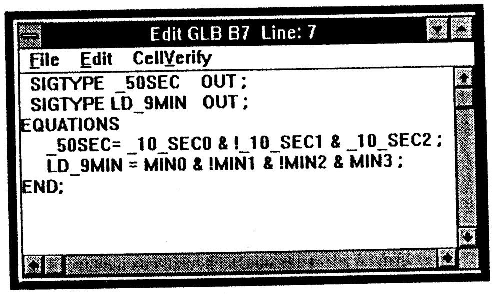


Figure 4-16. Cell Edit Window (GLB)

Notice that the cell's box icon in the GUI is outlined in green (on color monitors). The outline tells you at a glance that the cell contains logic. Empty cells remain outlined in black.

Using the Edit Cell Menus

The Edit window submenu options are **File**, **Edit**, and **CellVerify**.

Edit Cell's File Menu

File allows you to **Update** the cell contents or **Exit** (Figure 4-17). **Update** saves or refreshes the cell contents and closes the cell window. **Exit** issues a warning box that the cell contents changed since the last save (Figure 4-18). Select **Yes** to ignore your edits before closing the Edit Cell window, **No** to save the changes, or **Cancel** to close the warning box.

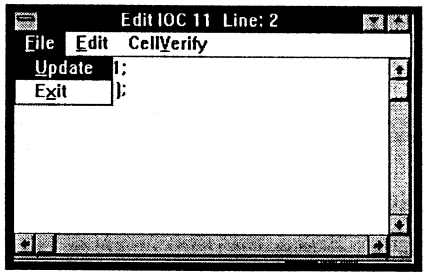


Figure 4-17. Edit Cell's **File** Menu

❖ **TIP:** Double-click on the Control Menu box (dash mark) of a cell to save edits *and* close it.

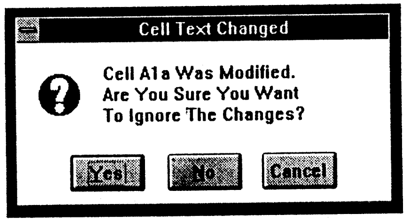


Figure 4-18. Cell Text Changed Warning Box

Edit Cell's Edit Menu

Edit allows you to **Undo**, **Cut**, **Copy**, **Paste**, and **Delete** information in the selected GLB (Figure 4-19).

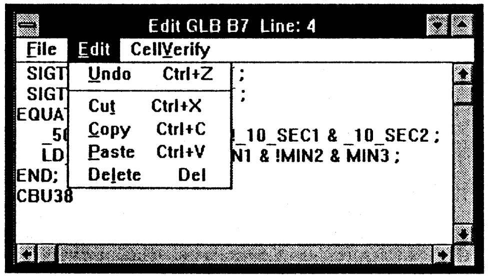


Figure 4-19. Edit Options, GLB Window

Edit Cell's CellVerify Menu

CellVerify performs a verify on the selected cell (Figure 4-20).

When you finish entering logic directly into a cell, select the **CellVerify** function from the Edit Cell window. **CellVerify** first checks for syntax errors, then for design rule accuracy. Any error messages are listed in the Messages window. To correct errors, edit the cell equations, then repeat the

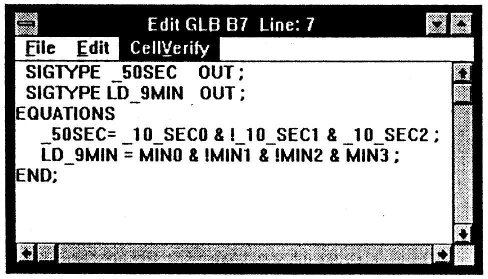


Figure 4-20. CellVerify Function Window

CellVerify. To resolve verification errors, use the **Help ➠ Error Messages** command from the Design Manager window. See the *pLSI and ispLSI Development System Reference Manual* for information on the **Help** menu.

CellVerify status is shown in the Messages window (Figure 4-21).

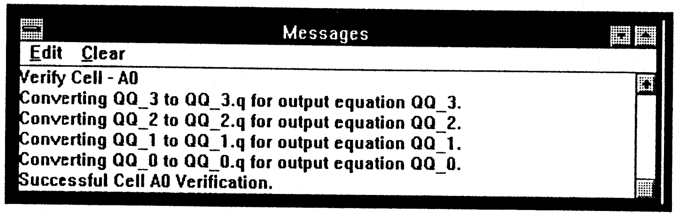


Figure 4-21. Cell Verify Status Window

For detailed information about GLBs or IOCs, refer to design architecture in the *Device Data Sheets.*

In the previous section *Editing a Cell*, we discussed how to open an Edit Cell window and enter your design syntax. After you create a design and you want to append it, there are two ways to modify your design. Use the Design Manager menu **Cell** commands for **Display**, **Edit**, **Copy**, **Move**, **Clear**, and **Name** functions, or open a specific Edit Cell window and use the **Edit** menu's **Undo**, **Cut**, **Copy**, **Paste**, and **Delete** functions.

Copying Cell Information

The **Cell ➠ Copy** command copies information from a cell and places it in a destination cell.

To copy the contents of one cell to another cell:

1. Select **Cell ➠ Copy** (Figure 4-22).

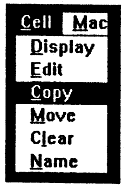


Figure 4-22. **Cell ➠ Copy** Menu Item Window

2. Select the source cell box from the Design Manager window. See the Messages window prompt (Figure 4-23).
3. Select the destination cell box from the GUI.

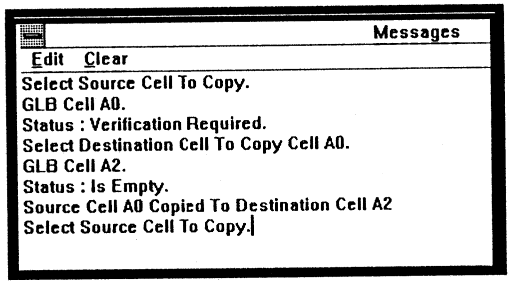


Figure 4-23. Copy Prompt to Select Cell Window

If the selected destination cell already contains information, you are warned that inserting new information will overwrite the cell's current contents (Figure 4-24, shown only if the destination cell edit window is not already open).

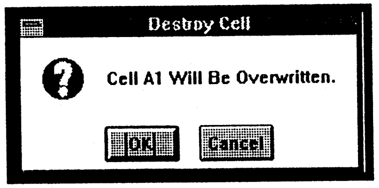


Figure 4-24. Copy Overwrite Prompt Window

4. Click OK to copy the information.

✍ **Note:** If the destination Edit Cell window is open, the warning box does not pop up and the copied information does not overwrite the destination cell's current contents, but it is **actually** copied.

✍ **Note:** If you highlight a block of information in the destination cell, it is replaced by whatever information you identify to copy and place, whether the information is from the same cell or from another cell. Be sure to identify the insertion point with the **cursor only**, unless you actually mean to remove a body of information from the destination cell.

❖ **TIP:** If you make a mistake during the **Copy** function, simply select **Undo** from the destination cell's **Edit** pull-down menu.

❖ **TIP:** Use **Cell** ➠ **Copy** to copy a cell's entire contents to another cell. Otherwise, open the Edit Cell window by double-clicking on its box icon and use the **Undo**, **Cut**, **Copy**, **Paste**, and **Delete** commands from the **Edit** menu to manipulate the information.

5. To terminate any **Cell** ➠ **Copy** function, click white space anywhere in the Design Manager window. The Messages window verifies the **Copy** command termination.

❖ **TIP:** You can use the Messages window as a scratch pad for editing, cutting, pasting, and copying.

Moving Cell Information

The **Cell** ➠ **Move** command removes information from a cell and places it in a destination cell. **Cell** ➠ **Move** functions are similar to **Cell** ➠ **Copy** functions, but **Cell** ➠ **Move** deletes the information from the source cell.

To move the contents of one cell to another:

1. Select **Cell** ➠ **Move** (Figure 4-25).

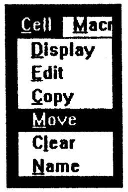


Figure 4-25. **Cell** ➠ **Move** Menu Item Window

2. Select the destination cell box from the GUI. The Messages window prompts you (Figure 4-26).

   If the selected destination cell already contains information, you are warned that inserting new information will overwrite the cell's current contents (shown only if the destination cell edit window is not already open, Figure 4-24.)

3. Click OK to move the information.

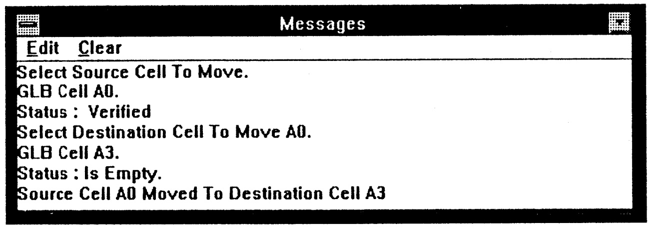


Figure 4-26. Move Function Prompts Window

✍ **Note:** If the destination Edit Cell window is open, the warning box does not pop up and the move occurs.

❖ **TIP:** Use the **Cell** ➠ **Move** function if you want to empty a cell by moving its entire contents to another cell. Otherwise, open the Edit Cell window by double-clicking on the box icon and use the **Undo**, **Cut**, **Copy**, **Paste**, and **Delete** commands from the **Edit** menu to manipulate the information.

✍ **Note:** You cannot copy or move IOC logic into a GLB cell or vice versa using the **Cell** pull-down menu. However, you can cut and paste selected text between *any* cells using the **Copy**, **Cut**, and **Paste** features from the **Edit** pull-down menu in the Edit Cell window.

4. To terminate any **Cell** ➠ **Move** function, click white space anywhere in the Design Manager window. The Messages window verifies the **Move** command termination.

❖ **TIP:** You can use the Messages window as a scratch pad for editing, cutting, pasting, and copying.

Clearing Cell Information

The **Cell** ➠ **Clear** command (Figure 4-27) deletes the contents of the selected cell so you are prompted to verify your choice before the command executes.

To clear a cell of syntax:

1. Select **Cell** ➠ **Clear** (Figure 4-27).

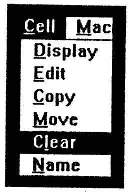


Figure 4-27. **Cell** ➠ **Clear** Menu Item Window

2. The Messages window prompts you to select the cell to clear.
3. Select the designated cell to clear. The **Clear/Delete** window appears asking for verification of the clear function (Figure 4-28).
4. Click Clear/Delete or Cancel.

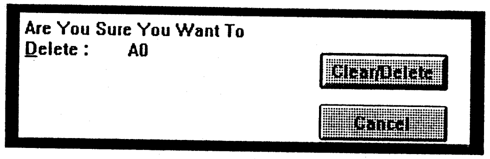


Figure 4-28. Clear/Delete Cell Window

5. To terminate any **Cell** ➠ **Clear** function, click white space anywhere in the Design Manager window. The Messages window verifies the **Clear** command termination.

Naming a Cell

The **Cell** ➠ **Name** command (Figure 4-29) enables you to reassign the name of the cell. This is helpful as you input your logic to personalize your design with recognizable tags on functional logic blocks. For instance, if you know you will use many IOC macros in a certain area, you can designate them incrementally in the GUI. You may use a maximum of 32 characters to rename a cell. Since the boxes are very small, only 3-4 characters will fit into a cell box, unless you use the **Zoom** ➠ **In 2X**. So the name is truncated to fit into the box. For information on how to perform zooms, see the *pLSI and ispLSI Development System Reference Manual.*

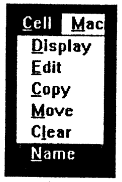


Figure 4-29. **Cell ➟ Name** Menu Item Window

To name a cell:

1. Select **Cell ➟ Name**. The Name Cell As window appears (Figure 4-30).

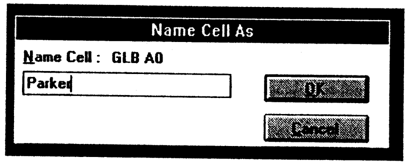


Figure 4-30. Name Cell As Window

2. Enter the desired cell name.
3. Click OK.

The name you specify is assigned to the cell in the GUI, as shown in (Figure 4-31). The Messages window shows the new name, also.

To terminate any **Cell ➟ Name** function, click white space anywhere in the Design Manager window. The Messages window verifies the **Name** command termination.

❖ **TIP:** You can use the Messages window as a scratch pad for editing, cutting, pasting, and copying.

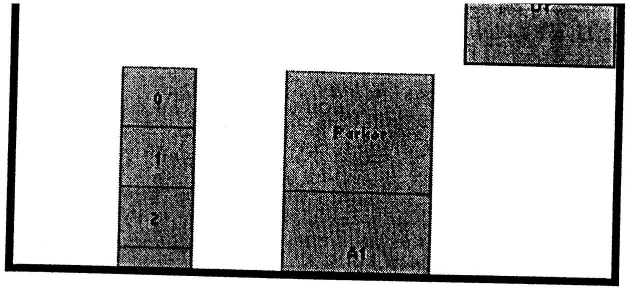


Figure 4-31. New Cell Name (Parker) in GLB Window

Displaying Cell Information

For quick access to cell information, use the **Cell ➟ Display** command. The syntax is displayed in the Messages window, but its Edit Cell window does not open to clutter your workspace. This is useful for comparing information between cells.

To see the contents of a cell:

1. Select **Cell ➟ Display** (Figure 4-32). Follow the instructions in the Messages window.
2. Select the desired cell to display from the GUI. The Messages window displays the contents (Figure 4-33).

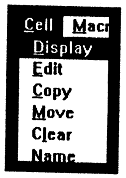


Figure 4-32. **Cell ➟ Display** Menu Item Window

❖ **TIP:** You can use the Messages window as a scratch pad for editing, cutting, pasting, and copying.

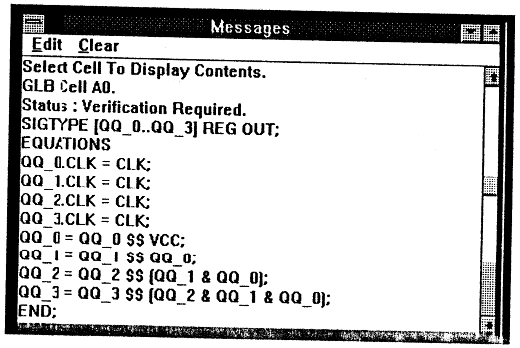


Figure 4-33.

Display Cell Information

Beschreibung der Bausteinprogrammierung

This chapter contains the ISP Daisy Chain Downloading procedure for the PC. The PC procedure supports both the Microsoft Windows and DOS environments. This chapter is organized into the following sections:

- Programming ISP Devices
- Windows Downloading
- DOS Downloading

There are two methods you can use to program ISP devices. One method sends a JEDEC file to a third-party programmer. The second method is to send a JEDEC file via an ispDOWNLOAD™ cable directly from your system to a device. This chapter contains information about the following:

- Options for programming a device using in-system programming (ISP) for ISP devices already installed on a printed circuit board
- Programming a device using industry-standard, third-party PLD programmers

For additional information about using a device programmer for the PC, see the *isp Engineering Kit Hardware Assembly Manual, Model 100.* For additional information about how to program multiple ISP devices in a daisy-chained configuration, as well as general ISP interface and unique programming features of each ISP device, see the *Lattice Handbook 1994.*

Programming ISP Devices

Using a PC

To program ISP devices in-system, you can either use the isp Engineering Kit Model 100 or implement the ISP algorithm for your own system and the ISP Daisy Chain Download software.

The isp Engineering Kit Model 100 contains the following:

- Programming module
- 25-pin parallel port adapter
- ispDOWNLOAD™ cable
- System cable
- Power supply converter (110VAC/9VDC @ 200 mA) – North America and Asia only
- *isp Engineering Kit Hardware Assembly Manual, Model 100*

A socket adapter board, which is purchased separately, is required if you want to program the ISP devices directly on the programming module. A unique socket adapter board is available for each package type. An ISP sample is included with each socket adapter board. Table 3-1 shows a partial list of the socket adapters now available from Lattice.

Table 3-1. Socket Adapters

Socket Adapter Part Number	Pins	Device Type	Package Type
pDS4102-J44	44	ispLSI 1016	JLCC
pDS4102-T44	44	ispLSI 1016	TQFP
pDS4102-J68	68	ispLSI 1024	JLCC
pDS4102-J84	84	ispLSI 1032	JLCC
pDS4102-T100	100	ispLSI 1032	TQFP
pDS4102-Q120	120	ispLSI 1048	PQFP
pDS4102-Q128	128	ispLSI 1048C	PQFP
pDS4102-J44	44	ispLSI 2032	PLCC
pDS4102-G167	167	ispLSI 3256	CPGA
pDS4102-M160	160	ispLSI 3256	MQFP

The part number for the isp Engineering Kit Model 100 is pDS4102-PM. To order the kit or the socket adapters, please contact a Lattice Sales Representative.

✍ ***NOTE*** The isp Engineering Kit Model 100 is now offered in a European edition featuring AC Power Supply Converter compatibility. Order part number pDS4102E-PM for use in Europe.

Windows Downloading

Lattice high-density ISP parts have an advanced feature that allows you to program and reprogram the parts in system without removing them. This feature eliminates using sockets for these devices, which avoids the common reliability problems associated with sockets. In-system programming can be done with an ispDOWNLOAD™ cable from the computer to the target device.

The software first scans to determine if there is a download cable connected to the device. If an ispDOWNLOAD™ cable is not attached, an alert message appears in the Messages window. When you invoke the download program, the ISP Daisy Chain Download window appears (Figure 3-1).

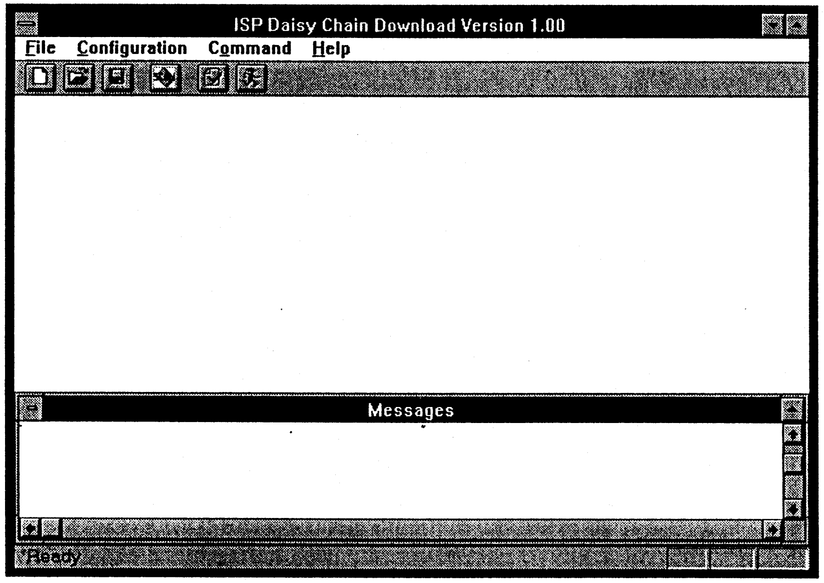


Figure 3-1. ISP Daisy Chain Download Window

The menu banner contains **File**, **Configuration**, **Command**, and **Help** pull-down options. Each menu option will be discussed briefly in the following sections.

The **File** menu contains the following options:

- **New** – Creates a new configuration setup
- **Open** – Opens a previously saved configuration setup
- **Save** – Saves a named configuration setup
- **Save As** – Names and saves a configuration setup
- **Exit** – Exits the ISP Daisy Chain Download program

The **Configuration** menu contains the following options:

- **Port Assignment** – Sets the parallel port(s) for the download configuration
- **Scan Board** – Scans the board and detects the configuration setup for you

The ISP Daisy Chain Download software can detect the first port assignment if you have the cables connected properly when you start the program. Use **Configuration ➠ Port Assignment** to *reset* the port.

The **Command** menu enables you to verify or download a configuration in the following two ways:

- **Command ➠ Check Configuration Setup**
- **Command ➠ Run Operation**

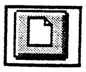
Once you set up the configuration properly using the **File** menu, verify your setup configurations with **Command ➠ Check Configuration Setup**. It is best to run the **Check Configuration Setup** operation before you download to check that your JEDEC device files are loading onto the correct devices, but that operation is optional. Execute the ISP Daisy Chain Download software using **Command ➠ Run Operation**. Your design files are programmed onto the devices to your specification. For more information about daisy-chaining your devices, see the *Lattice Handbook 1994*.

The **Help** menu contains the following options:

- **About ISP Daisy Chain Download** – Gives the version number of the ISP Daisy Chain Download software

Creating a New Configuration

To generate a new configuration:



1. Select **File** ➠ **New** from the ISP Daisy Chain Download menu (Figure 3-2). Or click the New icon in the ISP Daisy Chain Download tool bar. The New Configuration window appears (Figure 3-3). This window allows you to set the number of devices you want to program. The limit is 30. If you use the download cable (part number pDS4102-DL), we recommend that you configure no more than eight ISP devices in a daisy chain.

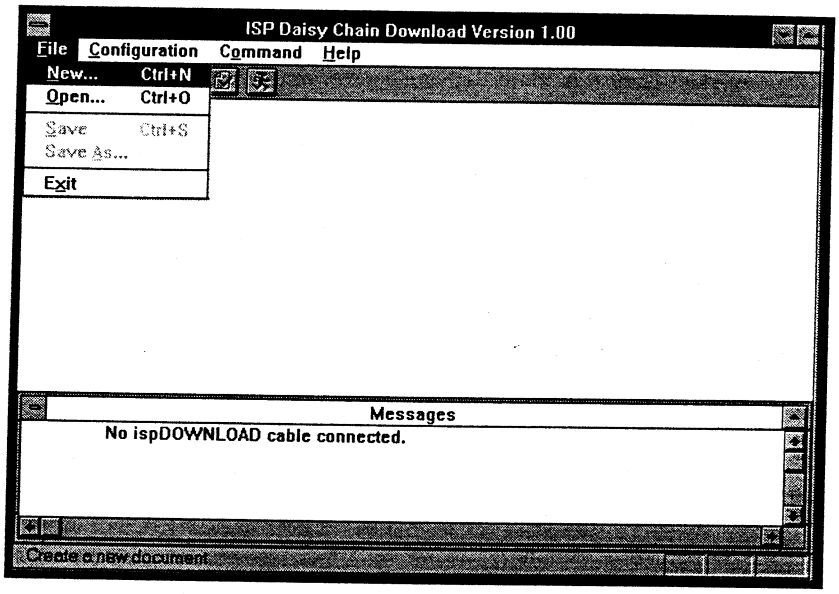


Figure 3-2. **File** ➠ **New** Menu Option

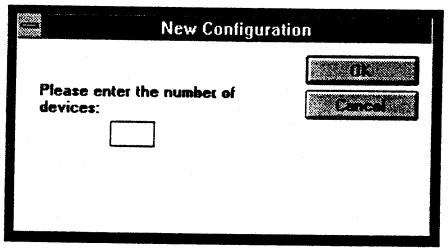


Figure 3-3. New Configuration Window

If you choose a number greater than 30, a dialog box appears (Figure 3-4) telling you to re-enter a valid number of devices.

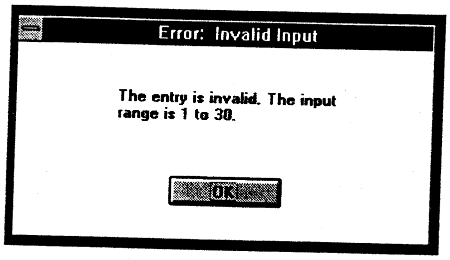


Figure 3-4. Invalid Input Dialog Box

2. Enter a number in the device field, click **OK** to execute, and Figure 3-5 appears. Select **Cancel** to close the window and return to the ISP Daisy Chain Download window. The scroll arrows and box appear to the right of the Status column if you choose a number greater than six. The default device is the ispLSI1016.

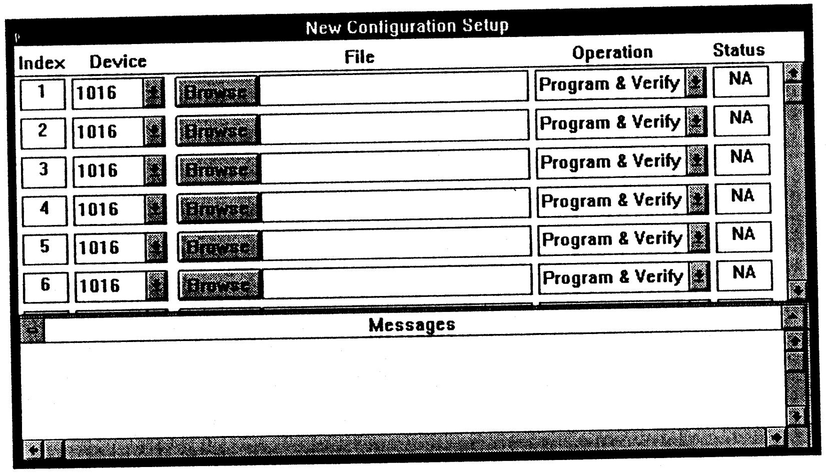


Figure 3-5. New Configuration Setup Window

The Configuration Setup window includes the following features and options:

- **Index Number** – Gives the order of the devices identified. As many as six are displayed per screen. This column is not editable.

- **Device** – Gives the device(s) identified. Use the pull-down arrow to select the device type when creating a configuration. The following are the available devices:

• ispLSI1016	ispLSI3256
• ispLSI1024	ispGAL22V10
• ispLSI1032	ispGDS14
• ispLSI1048	ispGDS18
• ispLSI1048C	ispGDS22
• ispLSI2032	

- **File** – Displays the file name (.jed). Choose **Browse** to locate a file or enter the file name in the field.

- **Operation** – Lists the function to perform for each file. The Operation options are the following:
  - Program & Verify
  - Verify
  - Checksum
  - Read & Save
  - Erase
  - No Operation

  The default is Program & Verify.

- **Status** – Shows the success factor after running a download procedure. The result possibilities include the following:
  - Pass
  - Fail
  - Hexadecimal Value
  - Done
  - NA (not applicable)

  The Status column displays NA until you execute the ISP Daisy Chain Download software for the configuration setup.

3. Select the device type in the **Device** field. Use the pull-down arrow to activate the pull-down menu (Figure 3-6).

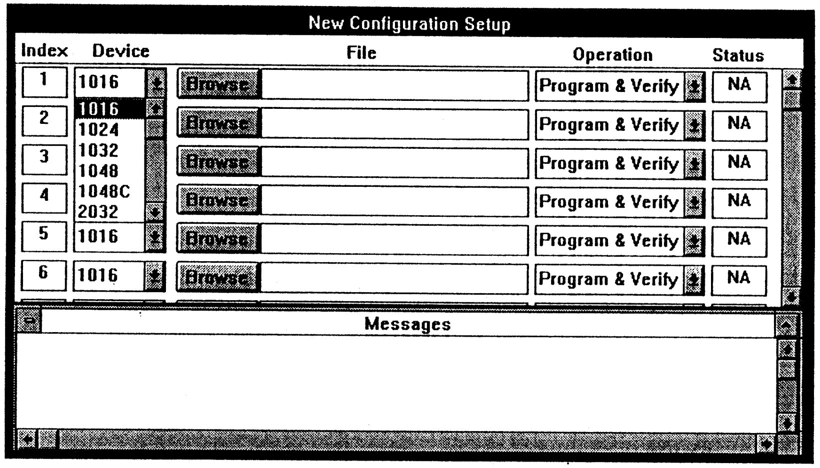


Figure 3-6. Device Pull-down Menu

4. Enter the file name or select **Browse** to find a file (Figure 3-7). Use the scroll down arrow keys to scroll through the lists. Make sure your choices are highlighted and choose **OK**. The Browse window closes and the .jed file name you chose appears in the Configuration Setup window for that device.

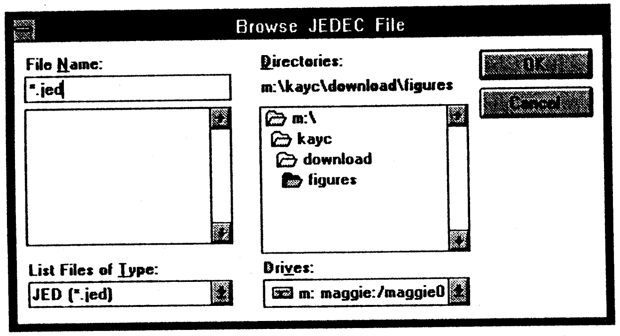


Figure 3-7. Browse JEDEC File Window

5. Select the **Operation** (Figure 3-8). Use the down arrow to activate the pull-down menu.

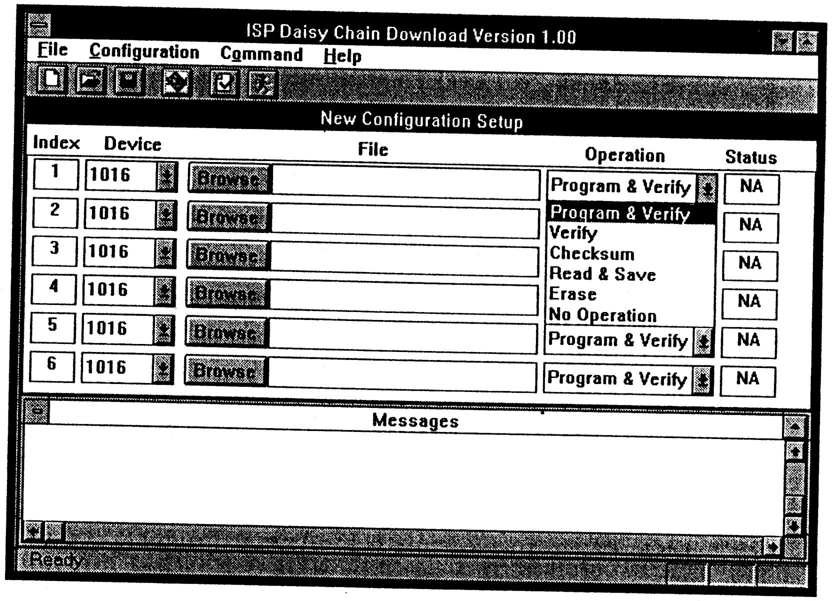


Figure 3-8. Operation Selection Options

The **Operation** options are defined next.

- **Program & Verify** – Performs a download of the design pattern, then verifies the chip program (device pattern) with the original .jed file to ensure the device was programmed correctly. The file name must be specified.
- **Verify** – Compares the chip program (device pattern) with the specified .jed file.

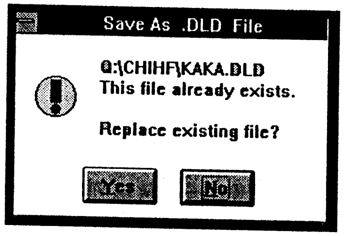


Figure 3-9. File Overwrite Dialog Box

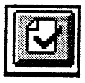
- **Checksum** – Computes the check sum value of the chip and sends the information to the Status field for the corresponding device.
- **Read & Save** – Reads the chip program and writes it into the specified JEDEC file. The file name must be specified. If the file name already exists, a dialog box appears (Figure 3-9) asking if you want to overwrite the existing file.
- **Erase** – Erases the chip program and the security fuse for the device you specify.
- **NOP** – Indicates no operation will be performed for that device.

Verifying a Configuration Setup

Once you enter all the specific information for each device, you should run a check configuration to identify any errors before downloading.

> *NOTE* The **Command ➠ Run Operation** command will also run a verification of your setup prior to downloading each file to your specifications.

To check a configuration setup:



1. Select **Command ➠ Check Configuration Setup** (Figure 3-10) from the ISP Daisy Chain Download menu. Or click the Check icon in the ISP Daisy Chain Download tool bar.

> ❖ *TIP* You can use **Command ➠Check Configuration Setup** to check your configuration setup and be certain you have all the information selected and the devices are ported properly. This process is particularly helpful if you do not want to download to an ISP device right then.

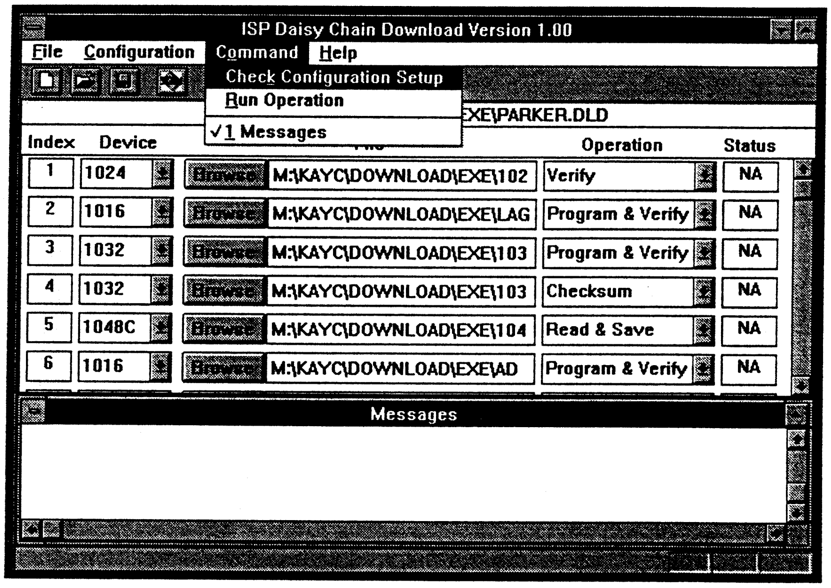


Figure 3-10. **Command ➠ Check Configuration Setup** Menu Option

2. Check the Messages window and make any necessary corrections.

The Messages window is a clipboard for the log file created every time you perform either a **Command ➠ Check Configuration Setup** or a **Command ➠ Run Operation**. Access the log file by typing `downld.log` from any text editor. The log file does not append itself; it only records the last function performed.

Downloading a File

To download a file:

1. Select **C**<u>**o**</u>**mmand ➟ **<u>**R**</u>**un Operation** (Figure 3-11) to program the device(s) with your design file(s). Or click the Run icon in the ISP Daisy Chain Download tool bar.

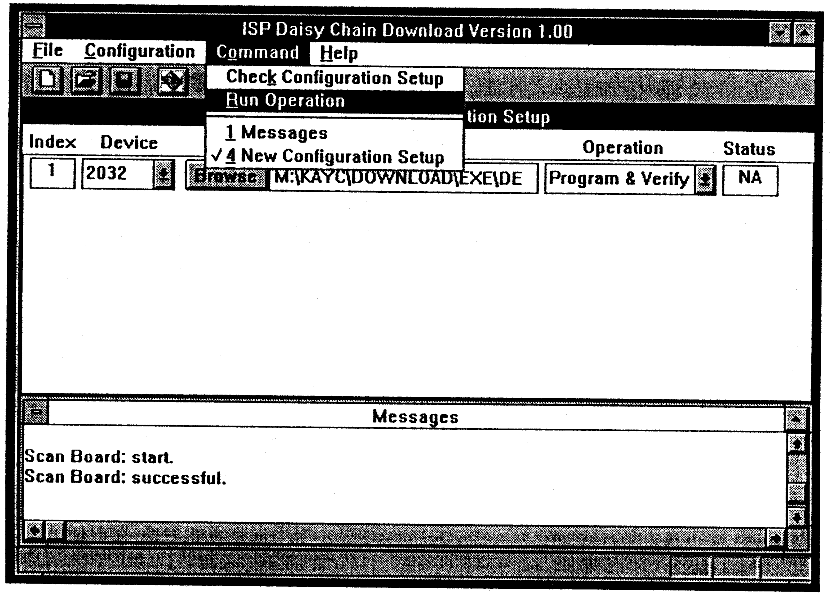
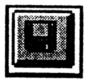


Figure 3-11. **C**<u>**o**</u>**mmand ➟ **<u>**R**</u>**un Operation** Menu Option

Saving a New Download Configuration

Once you set all the parameters for the configuration, save it.

To save a new download configuration:



1. Select <u>**F**</u>**ile ➟ Save **<u>**A**</u>**s** from the ISP Daisy Chain Download menu (Figure 3-12). Or click the Save icon in the ISP Daisy Chain Download tool bar. The standard Save As window appears (Figure 3-13).
2. Type in the name of your design file in the File Name field. It must have a .dld extension. If you enter more than eight characters, the file name is truncated. The full name of a file is not recognized against the truncated file name so the truncated file name overwrites itself each time you perform a save.
3. Choose the directory and file type.
4. Click **OK**. The *f_name*.dld appears in the main ISP window.

   If the file name already exists, a warning box to overwrite the file appears (Figure 3-14). Click **Yes** to overwrite the existing file. Click **No** to return to the Save As window and save the file with a new name.

   Click **Cancel** to close the Save As window. The configuration setup is not saved.

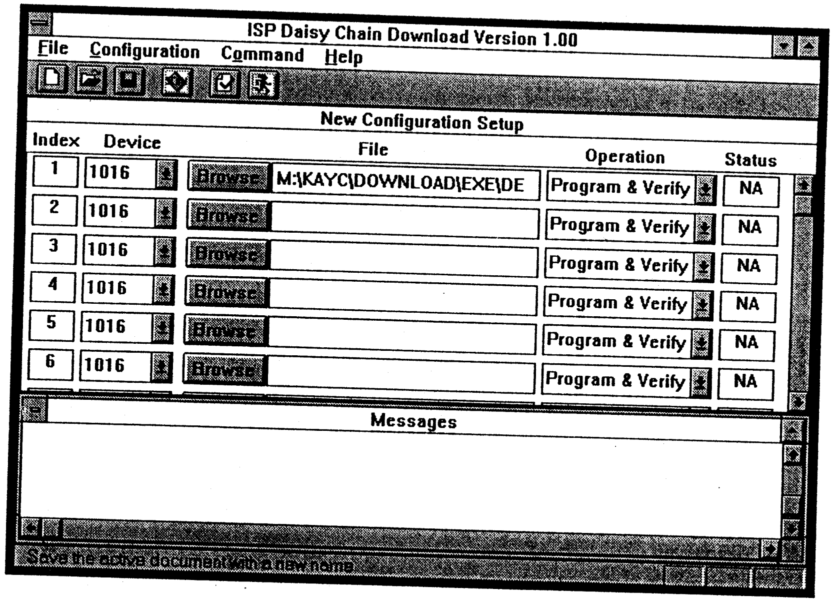


Figure 3-12. <u>**F**</u>**ile ➟ Save **<u>**A**</u>**s** Menu Option

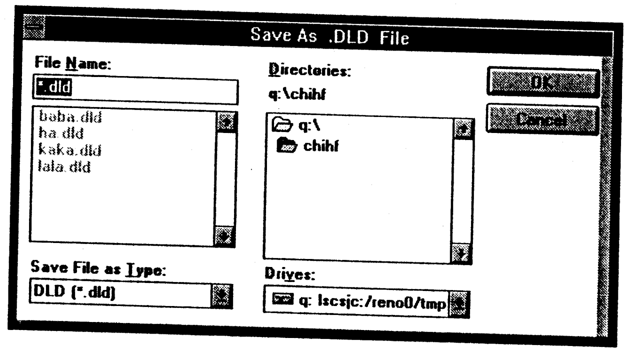


Figure 3-13. Save As Window

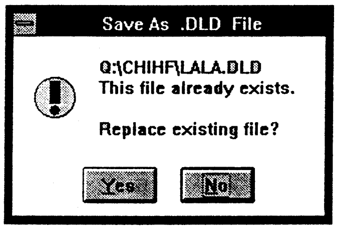
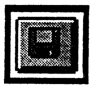


Figure 3-14. Overwrite Existing File Window

Saving a Configuration

Save your *f_name*.dld file while you work to ensure no data are lost.

To save a .dld file:



1. Select **File** ➠ **Save** from the *f_name*.dld ISP Daisy Chain Download menu (Figure 3-15). Or click the Save icon in the ISP Daisy Chain Download tool bar.

   If you have already named and saved the file before, the save takes place transparently and the icon dims.

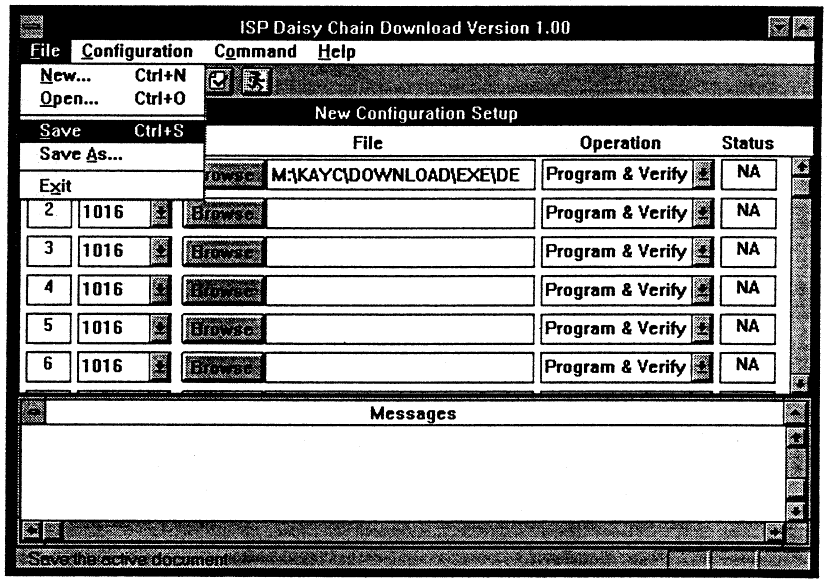


Figure 3-15. **File** ➠ **Save** Menu Option

If you are working with a .dld file created with a text editor using the DOS environment, see page 3-25.

Detecting a Configuration

If you want the software to identify all the devices available for programming, regardless of how you have them ported or mounted, use the **Configuration ➠ Scan Board** command. This command detects all the available ISP devices and lists their order and device type. Once the ISP Daisy Chain Download software identifies this information, complete the process by filling in the .jed file names and choosing the **Operation** for each device, exactly like the new configuration procedure beginning on page 3-7.

To identify the ISP devices automatically:

1. Select **Configuration ➠ Scan Board** from the ISP Daisy Chain Download menu (Figure 3-16). Or click the Scan icon in the ISP Daisy Chain Download tool bar. If you connected more than 30 devices to the board, an error message appears (Figure 3-17). The ISP Daisy Chain Download software can only detect and read as many as 30 devices per file. Otherwise, the Scan Board Configuration Setup window appears (Figure 3-18) with the **Index** and **Device** fields filled.

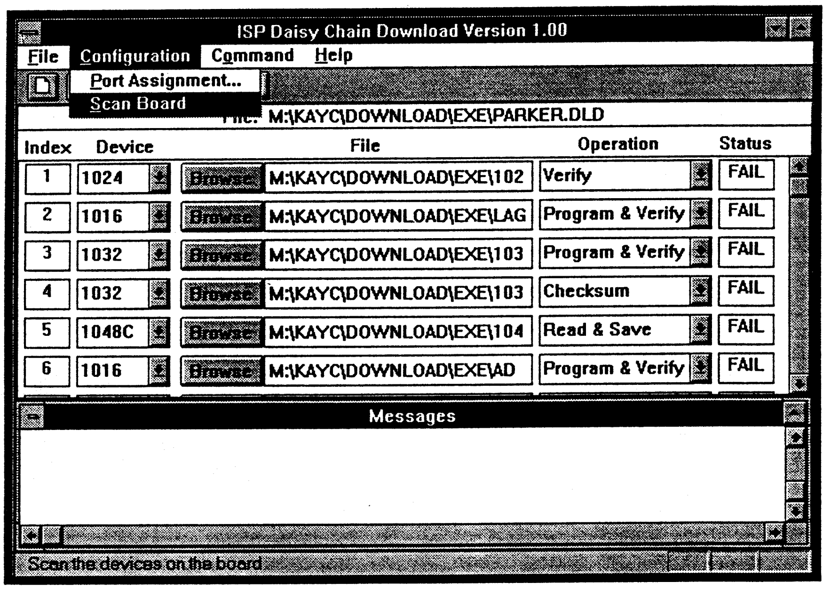


Figure 3-16. **Configuration ➠ Scan Board** Menu Option

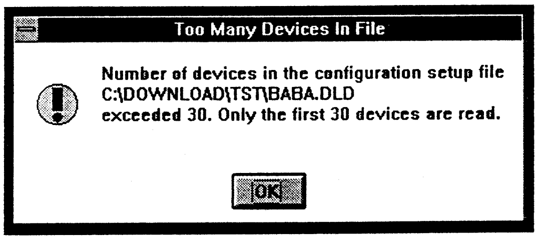


Figure 3-17. Number of Devices Detected Dialog Box

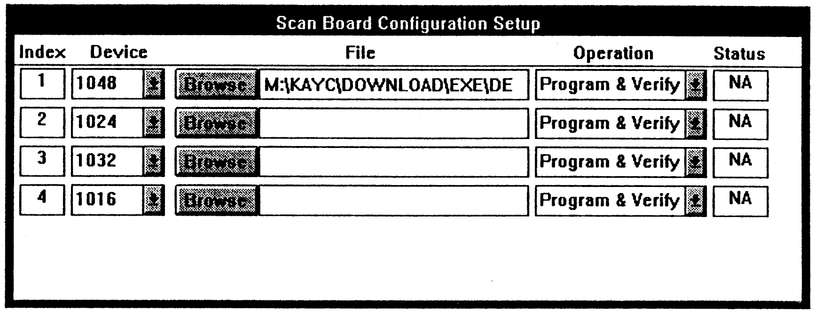


Figure 3-18. Scan Board Configuration Setup Window

2. Enter the file name or select **Browse** to find a file. Use the down arrow keys to scroll through the lists. Make sure your choices are highlighted and choose **OK**. The Browse window closes and the .jed file name you chose appears in the Scan Board Configuration Setup window for that device.
3. Select the **Operation**. Use the down arrow to activate the pull-down menu.

The **Operation** options are defined on page 3-10.

When you perform a **Read & Save** if the file name already exists, a dialog box appears (Figure 3-19) asking if you want to overwrite the existing file.

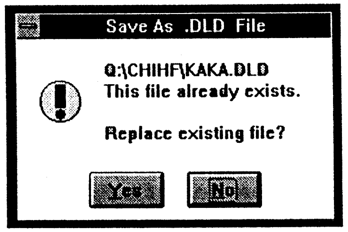
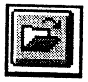


Figure 3-19. File Overwrite Dialog Box

Opening a Configuration



To open a configuration:

1. Select **File** ➠ **Open** (Figure 3-20). Or click the Open icon in the ISP Daisy Chain Download tool bar. The standard Open window appears (Figure 3-21).
2. Fill in the fields as required to access the file. Click **OK** to execute or **Cancel** to close the window.

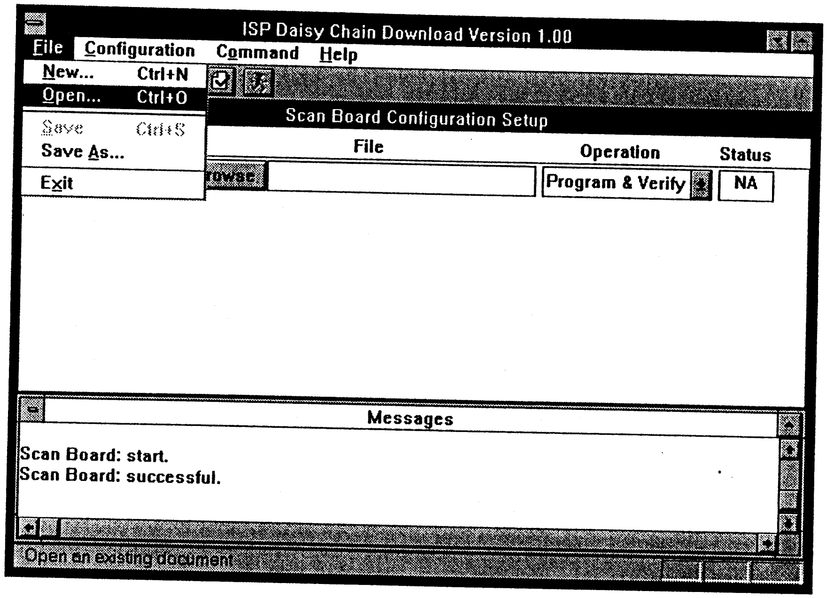


Figure 3-20. **File** ➠ **Open** Menu Option

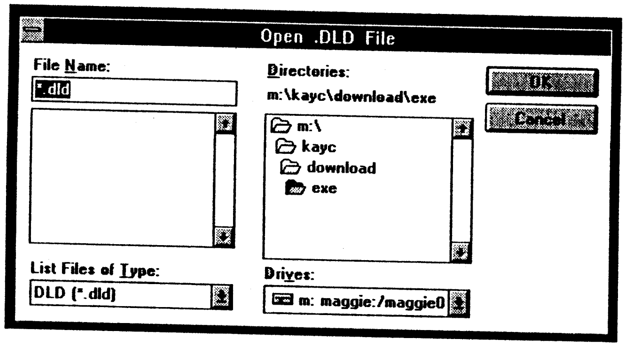


Figure 3-21. Open Window

Changing the Port

When you launch the ISP Daisy Chain Download software with the cables properly connected, it identifies the first port it detects. This menu allows you to change the port setup to accommodate your porting needs.

To change the port setup:

1. Select **Configuration ➠ Port Assignment** (Figure 3-22). The Port Assignment window appears (Figure 3-23).

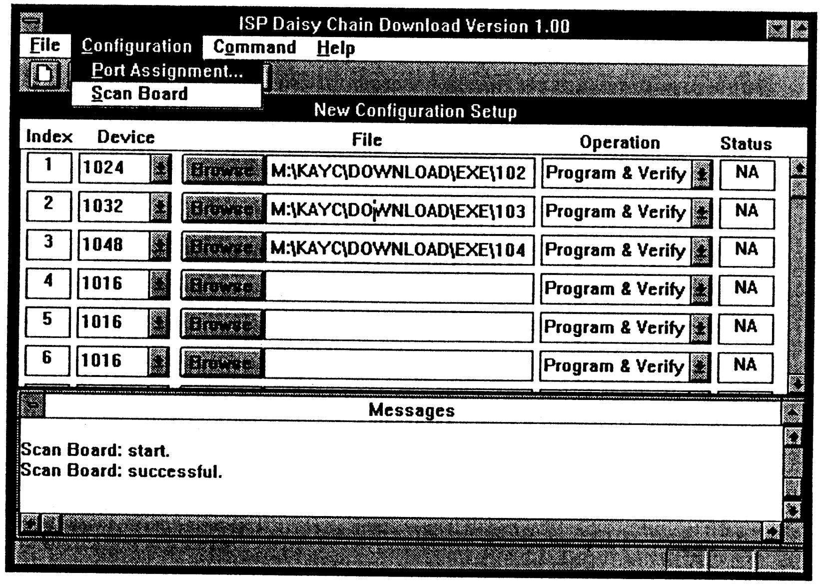


Figure 3-22. **Configuration ➠ Port Assignment** Menu Option

2. Use the pull-down arrow to select a port. Make sure your choice is highlighted. It appears in the top of the box.
3. Click **OK** to execute with your changes or **Cancel** to close the box with no changes.

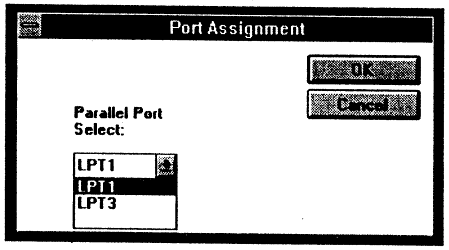


Figure 3-23. Port Assignment Window

If the cable is not connected or cannot be detected, the dialog box below appears (Figure 3-24).

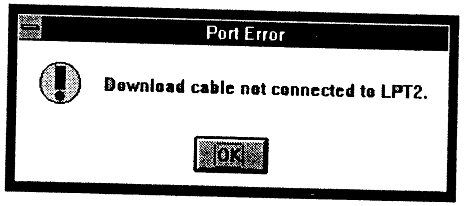


Figure 3-24. Cable Not Connected Dialog Box

If the cable is connected properly, but the power is not on, the dialog box in Figure 3-25 appears.

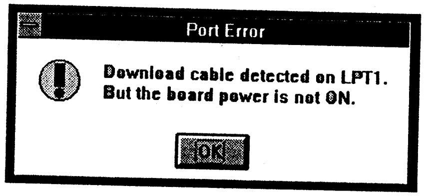


Figure 3-25. Cable Connected, Power Not On Dialog Box

Exiting the Program

To quit the ISP Daisy Chain Download software:

1. Select **File** ➠ **Exit** from the ISP Daisy Chain Download menu (Figure 3-26). The Save Configuration Setup windows appears (Figure 3-27), if you have not previously saved this file.
2. Click **Yes** to save the changes and exit the ISP Daisy Chain Download program. Click **No** to exit without saving the changes. The changes are not saved to the file that was open and the application closes. If you created a new configuration and did not name it, the standard Save As window appears (Figure 3-28). Fill in the fields and click **OK** to exit the ISP Daisy Chain Download software.

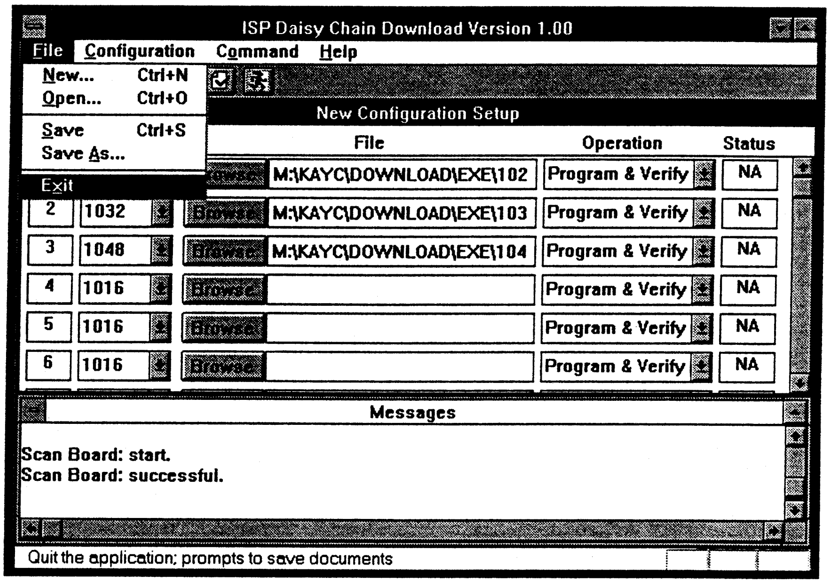


Figure 3-26. **File** ➠ **Exit** Menu Option

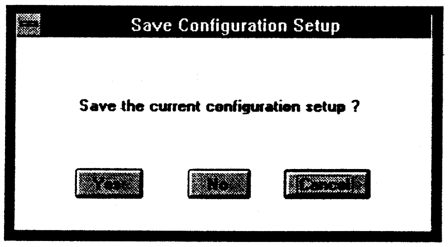


Figure 3-27. Save Configuration Setup Window

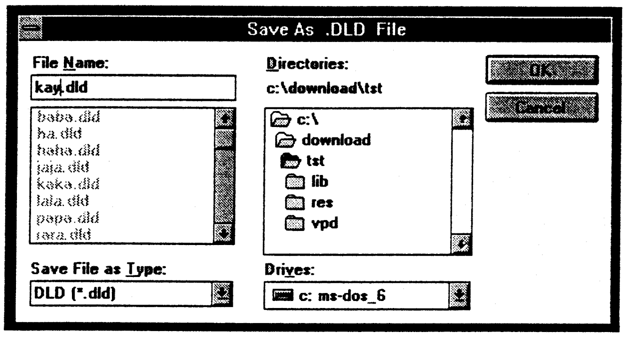


Figure 3-28. Save As Window

ispDOWNLOAD Cable Overview

This document describes how to assemble the Lattice ispDOWNLOAD™ Cable to perform in-system programming of Lattice in-system programmable (ISP™) devices. The Lattice ispDOWNLOAD Cable facilitates in-system programming of ISP devices from the parallel printer port of a PC using the Lattice pLSI/ispLSI Development System (pDS®), pLSI/ispLSI Development System Plus (pDS+™) PC-based design tools, ispGDS Download Software, or ispGAL Download Software.

ispDOWNLOAD Cable

The ispDOWNLOAD Cable consists of three primary components:

- 25-Pin Parallel Port Adapter
- 6' System ispDOWNLOAD Cable with RJ-45 Connectors
- 6' System ispDOWNLOAD Cable with RJ-45 to 8-pin Modular AMP Connector

NOTE Two cables are provided so you may use either an RJ-45.jack or an 8-pin AMP connector on your printed circuit board, depending on your application.

Connecting the ispDOWNLOAD Cable

Perform the following steps to connect the ispDOWNLOAD Cable to a printed circuit board. Refer to Figure 1 for assistance.

- ispDOWNLOAD Cable with RJ-45 Connectors

1. Connect the parallel port adapter directly behind the security key connected to a printer port on your PC, or to another available printer port.
2. Connect the appropriate ispDOWNLOAD Cable RJ-45 modular plug to the back of the parallel port adapter. Connect the other end of the cable to your printed circuit board that uses Lattice in-system programmable devices.
3. Ensure that the RJ-45 jack in your application follows the ISP signal pinout orientation shown in Figure 1.

- ispDOWNLOAD Cable with RJ-45 to 8-Pin Modular AMP Connector

1. Connect the parallel port adapter directly behind the security key connected to a printer port on your PC, or to another available printer port.
2. After connecting the ispDOWNLOAD cable RJ-45 modular plug to the back of the parallel port adapter, connect the AMP connector to your printed circuit board that uses Lattice in-system programmable devices. Your system needs to have a matching 0.100" center-to-center spacing 8-pin header, keyed to the cable's AMP connector. Refer to Figure 1 for the location of the ISP signal pinout orientation for the AMP connector.

Before using either cable, refer to the appropriate data book or datasheet for information on device pinouts and read the following descriptions about the cable signals:

- The VCC and GND pins are used to provide power from your system to a CMOS buffer in the parallel port adapter.
- During download, the ISP signals SDI, SCLK, and MODE are driven by the buffer in the parallel port adapter and $\overline{\text{ispEN}}$ is driven to a logic 0 by the pDS software.
- The SDO signal between your system and the computer's printer port is buffered by the parallel port adapter.
- Upon completion of the download process, the SDI, SCLK, and MODE signals from the cable are placed in a high impedance state and $\overline{\text{ispEN}}$ is driven to a logic 1 by the pDS software.
- You may need to place a .01 uf capacitor on your system board between the $\overline{\text{ispEN}}$ and GND signals to filter out any possible glitches generated from the computer's printer port during programming and verification.

NOTE $\overline{\text{ispEN}}$ is not used on all Lattice in-system programmable devices. For example, $\overline{\text{ispEN}}$ is not present on the ispGDS™ devices or on the ispGAL™ devices or on the ispGAL™ 22V10. See the appropriate device data sheet for details.

- The plugged signal line on the AMP connector is reserved for internal testing and should not be used.
- The "No Connect" line on the RJ-45 jack should not be used.

Programming the Device

After you have configured the hardware as shown in Figure 1, you are ready to download your pattern into the device. The pDS and pDS+ Development Software packages include software that controls the programming operation. For detailed instructions see the **ISP Daisy Chain Download Reference Manual**. ispGDS and ispGAL22V10 Download Software is available from the Lattice Literature Department and the "Device Programming" section is contained within the data sheet. Refer to Lattice Application Note number 2507 for more information.

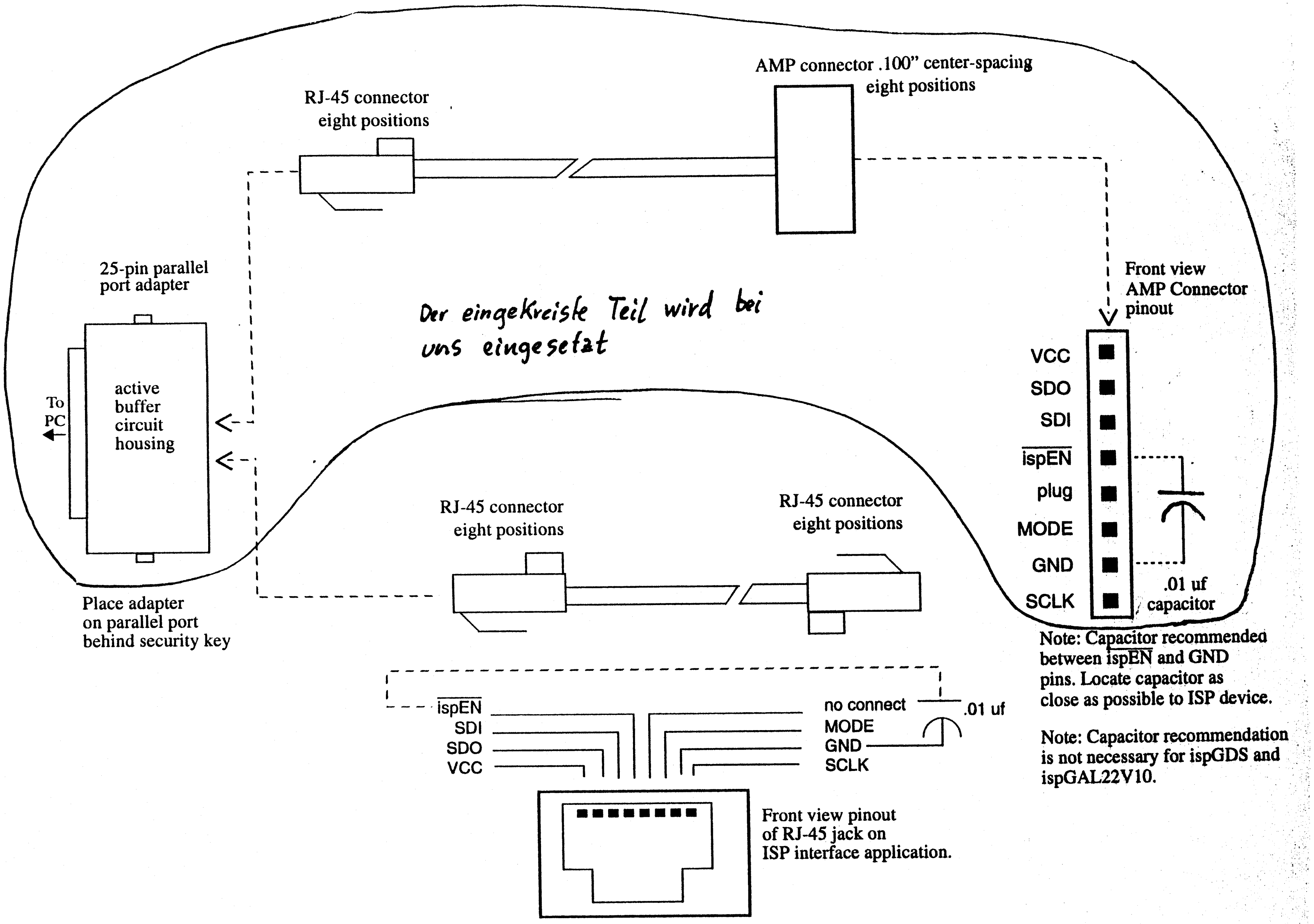


Figure 1. ispDOWNLOAD Cable Description

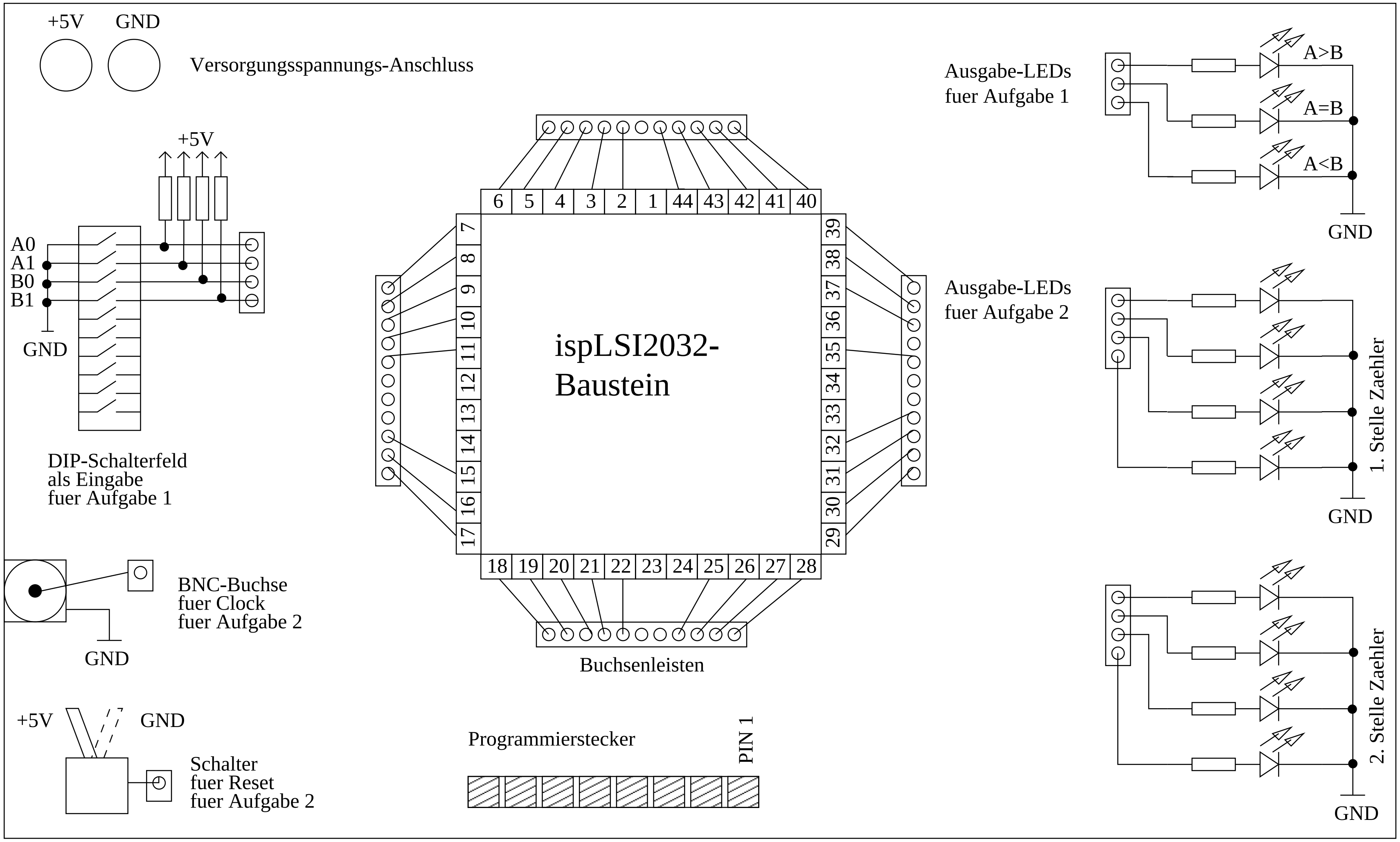
Beschreibung des Testkastens für die Programmierung

Für die Programmierung und den Test des ispLSI2023 steht ein Testkasten zur Verfügung, der auf der folgenden Seite abgebildet ist.

Der Testkasten ist so aufgebaut, daß man die Versorgungsspannung für die Schaltung an die Buchsen +5V und GND anschliessen muß.

Das Programmierkabel muß an den Programmierstecker, der sich im unteren Teil des Kastens befindet, angeschlossen werden. Beim Anschluss des Programmierkabels ist darauf zu achten, daß der Pin 1 des Programmierkabels auch an Pin 1 des Steckers angeschlossen wird !

Die I/O-Pins des Bausteines sind auf die Buchsenleisten geführt, die sich direkt beim Baustein befinden. Die Ein- bzw. Ausgabemöglichkeiten für die einzelnen Aufgabenteile sind am Rand des Kastens plaziert (rechts die Eingabeteile, links die Ausgabenteile). Auch die Ein- und Ausgabebauteile sind auf Buchsenleisten geführt, so daß die individuelle Verbindung zwischen den IO-Pins und den Ein- und Ausgabeteilen über einfache Steckleitungen aus dem Werkzeugkoffer erfolgen kann.


+5V
GND
Versorgungsspannungs-Anschluss
+5V
A0
A1
B0
B1
GND
DIP-Schalterfeld
als Eingabe
fuer Aufgabe 1
ispLSI2032-
Baustein
6 5 4 3 2 1 44 43 42 41 40
7 8 9 10 11 12 13 14 15 16 17
39 38 37 36 35 34 33 32 31 30 29
18 19 20 21 22 23 24 25 26 27 28
Buchsenleisten
Ausgabe-LEDs
fuer Aufgabe 1
A>B
A=B
A<B
GND
Ausgabe-LEDs
fuer Aufgabe 2
1. Stelle Zaehler
GND
2. Stelle Zaehler
GND
BNC-Buchse
fuer Clock
fuer Aufgabe 2
GND
+5V
GND
Schalter
fuer Reset
fuer Aufgabe 2
Programmierstecker
PIN 1