Beginner's Guide to ispLSI® and pLSI® Using pDS® Software

Introduction

This Beginner's Guide is designed to familiarize you with the Lattice ispLSI and pLSI devices and the Lattice pLSI and ispLSI Development System (pDS). To accomplish this, the complete design of a simple four-bit counter is discussed from specification through programming and testing the part. The following assumptions are made. First, you have read and understood the ispLSI 1032E data sheet. Next, you have the documentation for Microsoft® Windows™ readily available. Everything else is included in the Beginner's Guide.

The Lattice pDS Software is designed to run under Microsoft Windows Version 3.1 or later or Windows 95 (see Figure 1).

Windows is an industry standard Graphical User Interface (GUI) with pull-down menus, text editing commands and screen control commands. Because the Lattice interface is the same as other Windows programs, it is very easy to learn. If you know how to run any Windows program, you can run the Lattice software.

It is necessary to have Windows for the Lattice pDS Software to run. Windows runs on most standard IBM PCs or clones. If your computer runs Windows 3.1, it will run the Lattice pDS Software. The recommended system configuration for running pDS Software is:

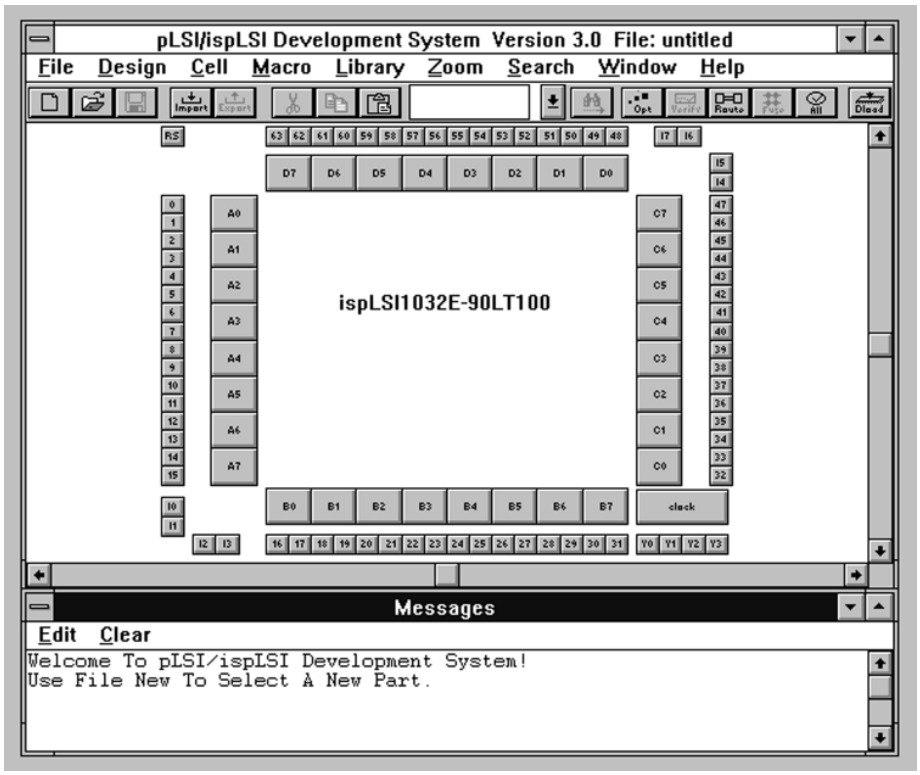
- 486 or Higher Processor
- 8 Megabytes of RAM
- 40 Megabytes Disk Space
- Floppy Disk Drive
- Microsoft Windows Compatible Mouse
- VGA or Super VGA Graphics

In addition, the pDS Software requires that either a spare parallel printer port be available to perform in-system programming, or a spare serial port be available to communicate with an RS-232 controlled programmer.

Getting Started

If you have not previously installed Lattice pDS Software, see the installation procedure included with your development system.

Figure 1. Lattice pDS Software Graphical User Interface



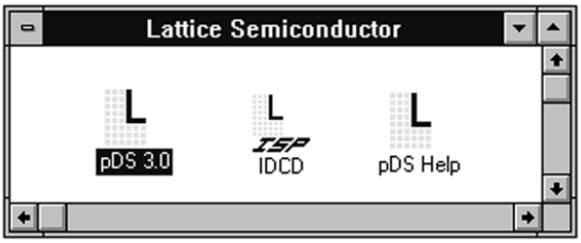
1. Install the Lattice pDS Software according to the installation instructions. A new program group called Lattice Semiconductor is created. This program group contains icons resembling the Lattice company logo (see Figure 2).
2. To start the pDS Software, double click on the Lattice pDS 3.0 logo icon.

Before proceeding, some Microsoft Windows tasks that you should be able to perform include:

- 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 using scroll bars
- Editing text using the keyboard and mouse to:
  - Select the insertion point
  - Select text by highlighting it
  - Cut, paste and copy text

If you are unfamiliar with any of these options, take time to go through the Windows Users Guide. If you have worked with an Apple™ Macintosh™, you will find that many of the commands and operations are similar.

Figure 2. Lattice Program Group Window



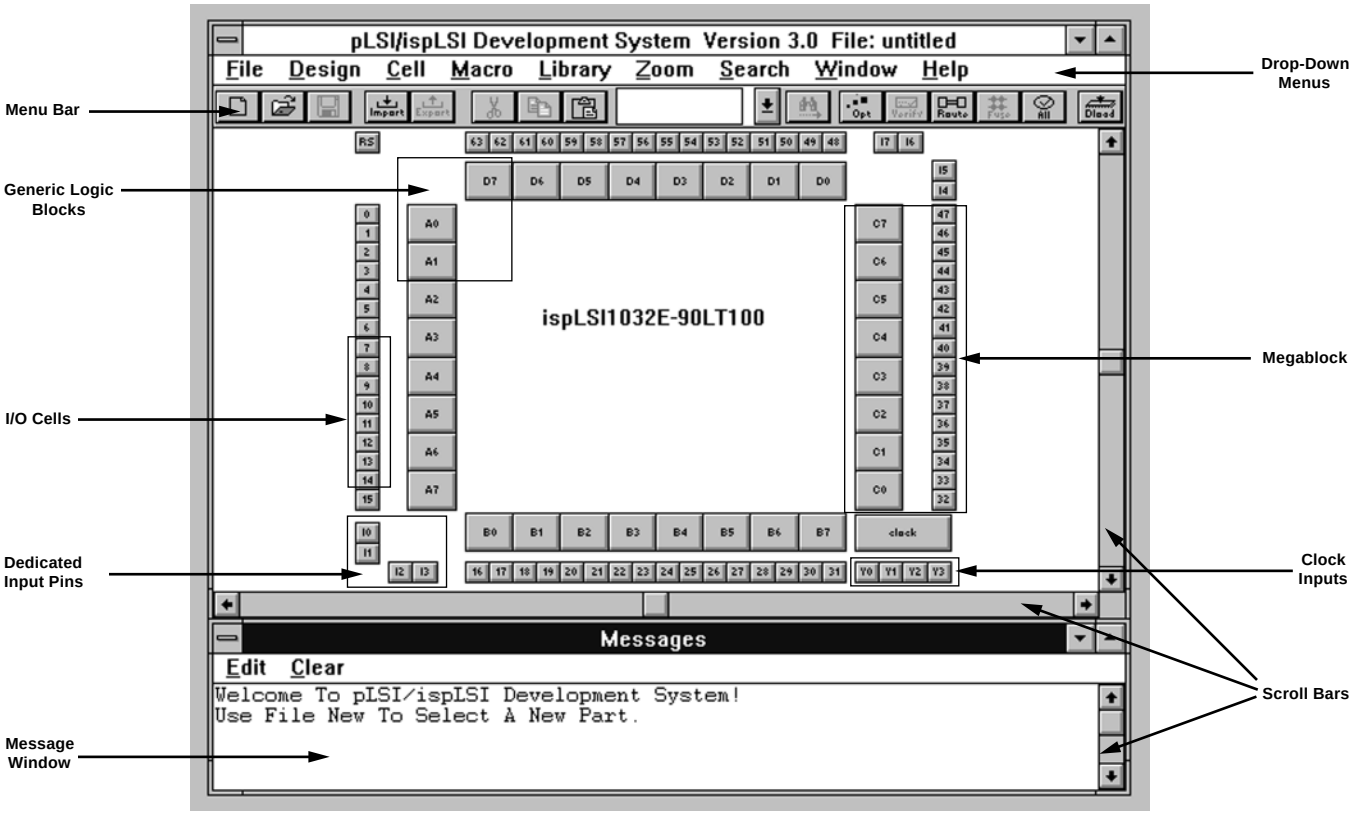
A Brief Tour of the Screen

Once you invoke the Lattice pDS Software, two windows are displayed (see Figures 8 and 9).

The larger of the two windows displays a graphical representation of the ispLSI 1032E logic diagram. This window is called the Data Entry Window. The design is entered by editing equations in the Data Entry Window.

The smaller of the two windows is the Message Window and it is located at the bottom of the screen. The pDS Software communicates with you by placing messages in the message window.

Figure 3. Design Entry Window



The part displayed in the block diagram shows the elements of the ispLSI 1032E that can be modified by the user. These elements are GLBs, I/O Cells, dedicated input pins and clock input pins, as shown in Figure 3.

The design is entered by clicking the element you wish to edit and entering equations or macros (library elements already partitioned and optimized for high performance) into the Edit Windows (see Figure 4).

The method of entering configuration data into a cell depends on the type of cell:

Configuration data for Generic Logic Blocks is entered using a combination of Boolean Equations or macros from the Lattice Standard Library or TTL Library.

Configuration data for I/O Cells is entered using macros only. There is a complete set of macros which describes all possible combinations of input, output, and I/O cell configurations.

Configuration data for the Dedicated Input Pins and the Clock Input Pins is entered using a subset of the I/O Cell macros. Because these pins are inputs only, and do not have input registers, only certain I/O Cell macros apply.

The Design Flow

Before starting our sample design, it is valuable to understand the Design Flow. The following steps are necessary to complete a design. Refer to Figure 5 for more information.

Specifying the Design

A design is specified using one of the two approaches. With the first method, you use an existing design, consisting of TTL-like logic elements and fit the design into the ispLSI part. With the second method, you design a circuit that is optimized for best performance and utilization of the ispLSI architecture.

The first approach consists of simply selecting macros from the Lattice library that match the functions of the TTL or CMOS circuits and then connecting them to each other. Using this approach, a design can be completed quickly with a high degree of initial success because the macro circuit elements have 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.

Figure 4. Open Edit Windows

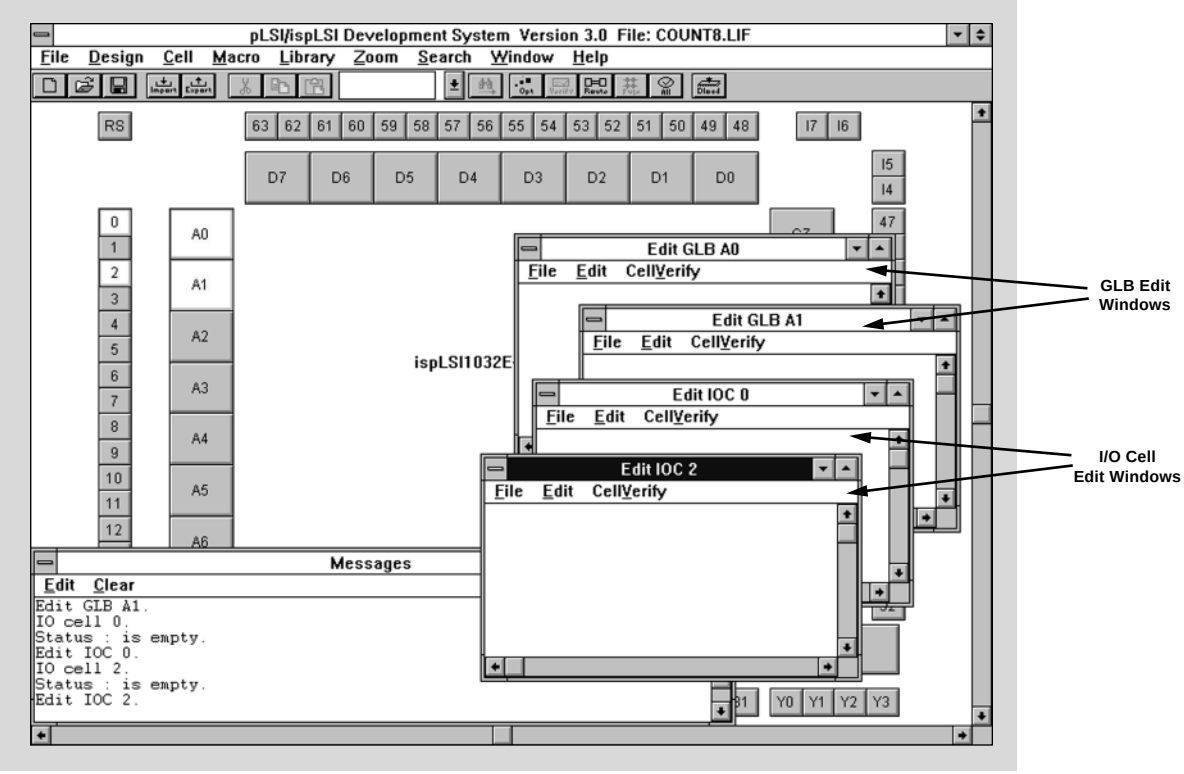
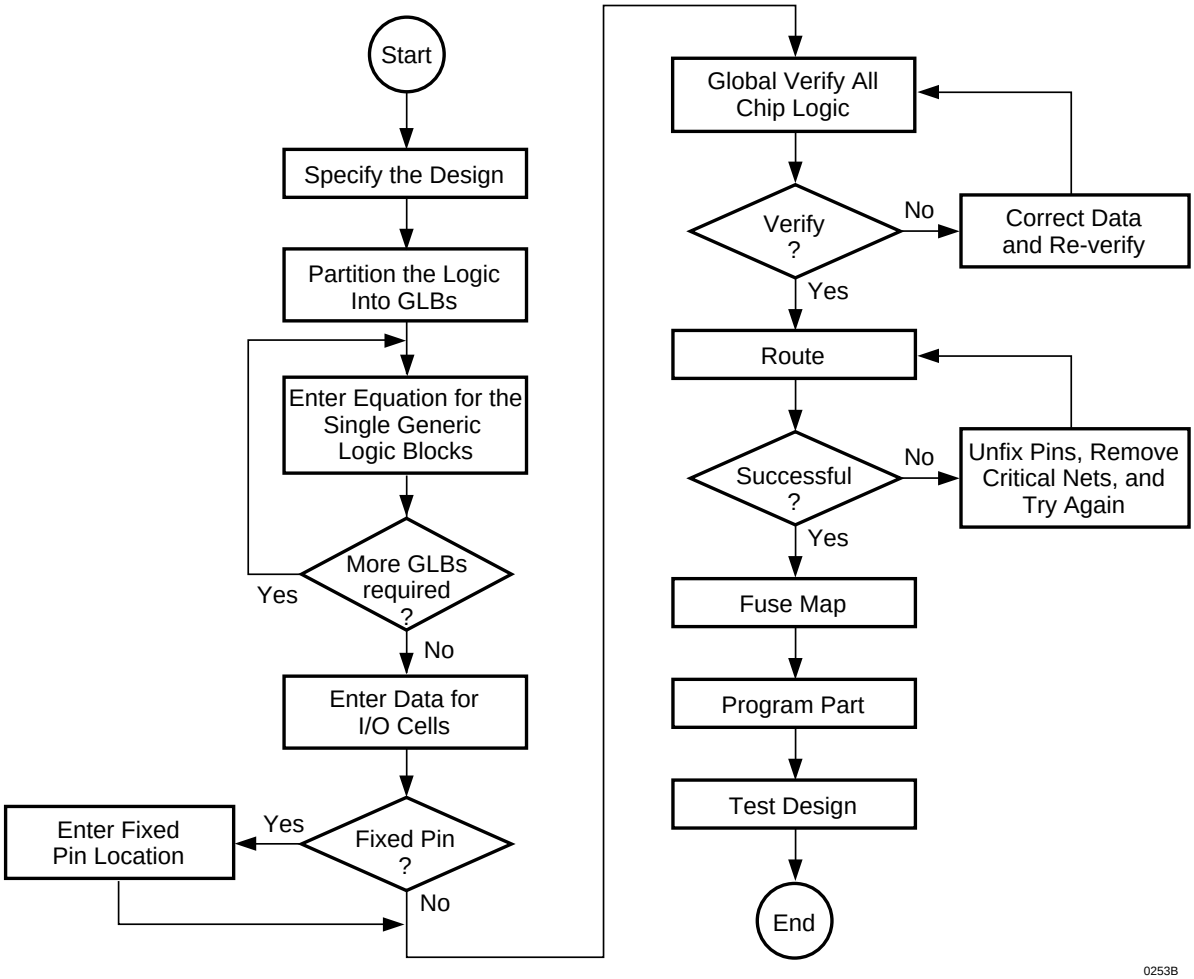


Figure 5. Design Process Flow



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

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

Partitioning the Design

Partitioning consists of dividing the logic into groups that conveniently fit into the ispLSI Generic Logic Blocks. These general rules should be followed when partitioning logic:

- Look at the 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 the logic design.
- Know the capabilities of the GLB. It has 18 inputs and four outputs. The GLB has 20 Product Terms (PTs) that are grouped together in groups of four, four, five, and seven PTs. The registers in the GLB share a common clock. The registers within the GLB also share a common Reset Product Term.

Figure 6. Counter Schematic Diagram

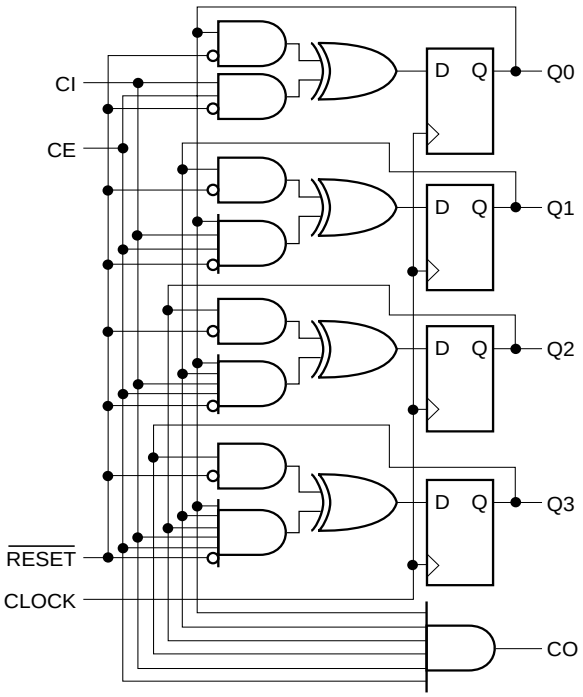
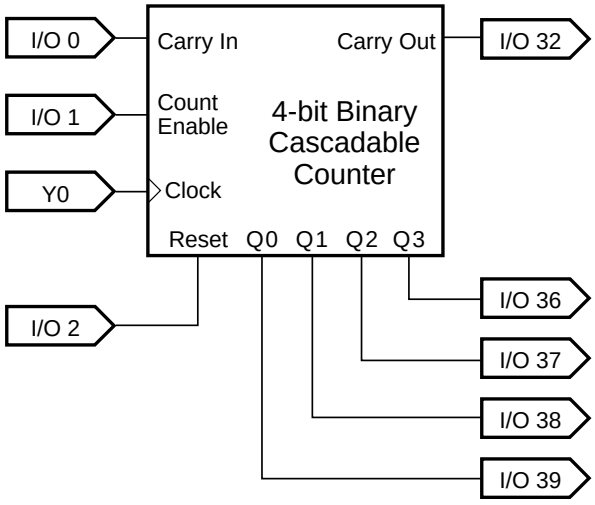


Figure 7. Sample Cascadable Counter Logic



- When an output has been 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 Output Enable signal per Megablock. Outputs which share a common Output Enable signal should be placed in the same Megablock (see Figure 3).
- Signals that are related to each other, such as those used for counters, shift registers, etc., should be placed into the same Megablock. This is done to reduce routing congestion.

Compiling the Design

Compiling the design is done using the Lattice pDS Software and consists of four steps:

1. **Entering the design**. Boolean equations or macros are entered into the various cells and blocks on the ispLSI device using a built in text editor. After each cell has been entered, a *Local Verify* is done to check for syntax or logic errors within that cell.
2. **Verifying the design**. This is done globally after all the design has been entered. This 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 Global Verify before any of the following steps can happen.
3. **Routing the design**. This is the next step after a successful Verify. The Router interconnects the Generic Logic Block and I/O Cell inputs and outputs. The option of fixing certain input and output signals to specific device pins is now available.
4. **Generating the Fusemap**. This takes the verified and routed design and creates the *JEDEC* file (a standard binary fuse file) necessary to program the part. This is a modified format JEDEC file with a suffix of .JED.

Programming the Part

Once the design has been compiled, the next step is to 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 part be removed from the target system socket and inserted into a programmer.

Testing the Design

The last step in the process is testing the design. The design is tested by putting it on the board and seeing if it works correctly. If corrections need to be made, the appropriate GLBs or I/O Cells are reprogrammed, and the design is recompiled. Because the ispLSI 1032E is an electrically erasable and reprogrammable part, the same part can be used again and 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 an ispLSI 1032E device. We will then take the design through the compilation process, generate a fuse file and program a part.

The counter has the following specifications:

- A 4-bit Synchronous Binary Counter.
- Active High Cascade In (CI) and Cascade Out (CO) Pins.
- An Active High Count Enable (CE) Pin.
- A Synchronous Reset Pin.

Figure 6 shows the schematic diagram and Figure 7 shows the logic symbol for this counter. Because the counter has 5 outputs (Q0, Q1, Q2, Q3, and Cascade Out) it occupies two GLBs.

Listing 1. Counter Equations

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

Listing 2. GLB Equations

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

In this design example, the Clock and I/O pins are assigned to be compatible with the Lattice ispLSI 1032E for a specific printed circuit board. This allows the design to be tested easily.

The input signals Cascade In, Count Enable and Reset are connected to three bits of the 8-bit DIP switch, and the five outputs are connected to five of the discrete LED outputs.

Defining the Counter

In defining the counter, the first step is to write the equations. The equations for the 4-bit binary counter are expressed in Listing 1.

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

When either Cascade In or Count Enable is low and RST is low, the Q0 output from the counter remains in its current state (Hold). When Cascade In and Count Enable 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.)

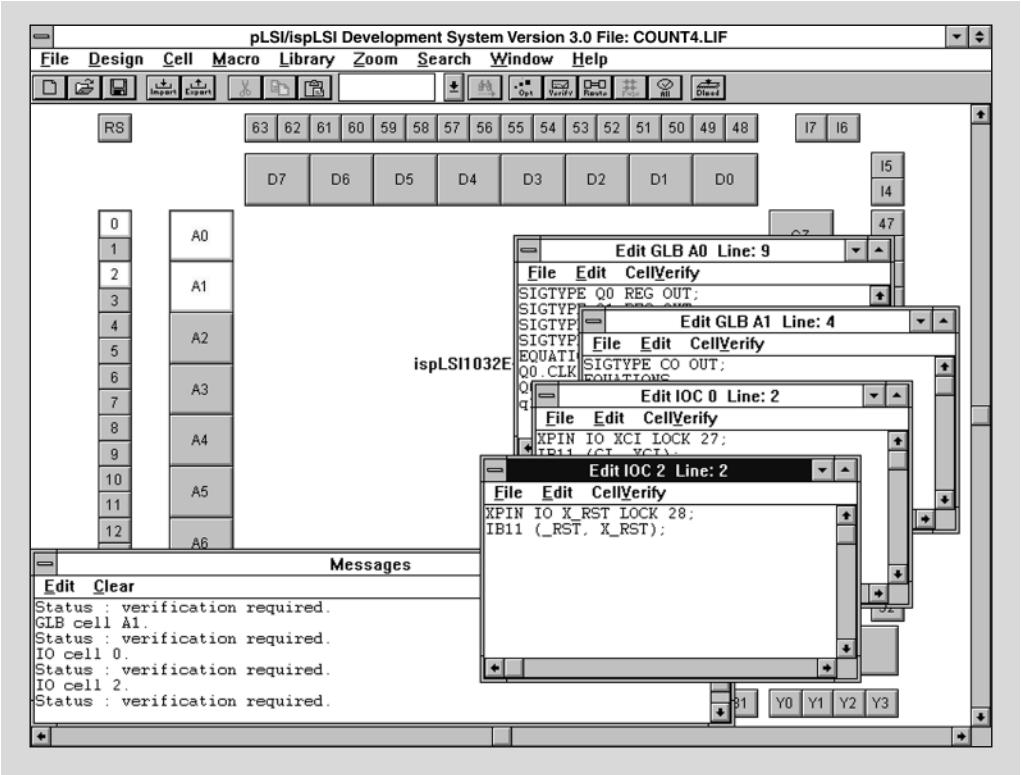
Each successive stage operates similarly, except during transition, (Increment), when the outputs of all previous stages are at Logic Level One. The Carry Out signal is only generated when all the stages have reached a one and both Cascade In and Counter Enable are a one.

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

1. From within Windows, start the Lattice pDS Software by double clicking on the Lattice Icon.

2. When the Lattice software starts, it displays the block diagram of the ispLSI 1032E part. Open GLB C1 for editing by double clicking on it. The edit window displays.

Figure 8. Examples of Cell Entry Windows with Equations



3. Enter the equations shown in Listing 2 into the edit window for GLB C1.

4. Verify the equations by clicking on the Cell Verify menu option. If errors appear in the Message window, find out what is wrong, and correct it. Things to look for are typing errors, missing semicolons, or incorrect symbols. Re-verify after making corrections.

5. Close the Edit window for GLB C1 by selecting the Done option from the Cell Edit Menu.

6. Open GLB C2 for editing by double clicking on it.

7. Enter the following equations into the edit window for GLB C2:

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

8. Verify the equations by clicking on the Cell Verify menu option.

9. Close the Cell Edit window by clicking on the Done option in the menu bar.

At this point, the logic for the counter is completely specified, but we still must connect the Clock and the Inputs and Outputs.

10. Open Clock Input Y0 by double clicking on it. It may be necessary to Zoom in on the Clock area of the Logic Diagram to determine which pin is Y0.

11. Enter the following equations into the edit window for Clock Input Y0:

```
XPIN CLK X_CLK LOCK 20;
IB11 (_CLK, X_CLK );
```

12. Verify the equations by clicking on the Cell Verify menu option.

13. Once the cell verifies correctly, close the Cell Edit window by clicking on the Done option in the menu bar.

14. Repeat Steps 10 through 13 for the Cascade 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 X_RST LOCK 28;
IB11 (_RST, X_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 multiple Edit Windows open at the same time. This means that you can Copy the equations from I/O Cell 39 and Paste them into I/O Cell 38. The data in both cells is similar, and you can use the Windows editing 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 13 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 71;
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 work.

22. From the Menu Bar, select the File Option, and choose Save As. The pDS Software prompts you for the file name. Type in the name COUNTER. The suffix .LIF (Lattice Internal File) is automatically appended.

Review of the Syntax

This is a brief review of the syntax used in the example design. For complete information see the Language Reference section of the Software Manual included with the Lattice pDS Software.

The operators that the Lattice pDS Software uses are similar to those used by the Data I/O ABEL program. The operators and an example of how they are used are shown in the table below. The Precedence of Evaluation is also shown where 1 is the highest precedence (see table below).

Table: 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, there are several other lines that need to be included in the GLB or I/O Cell definition. They are:

SYM; The symbol line consists of five parts:

- The Keyword SYM that indicates what type of line this is to be.
- The Symbol Name. This is either GLB or IOC.
- The Cell location.
- The Symbol Level used by other software packages. For our purposes, always use a 1.
- The Symbol User Name. This is 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; Indicate the start of the Equation Section for a GLB or I/O Cell.

MACRO; Indicates the usage of a Macro Logic Element from the Macro Library.

END; Signifies the end of an Equation Section, a GLB or I/O Cell 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 an example comment line.
```

Programming the Device

The Fuse map program generated a fuse (.JED) file which needs to be permanently programmed into an ispLSI or pLSI 1032E device. Programming the part is done using one of three methods:

- In-system program for ispLSI device
- Motherboard Programmer for ispLSI device
- RS-232 Link Programmer for ispLSI or pLSI device

An ispDOWNLOAD™ Cable is provided with 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, the part can be programmed right on the board or using the isp Engineering Kit.

For designers who need to integrate ISP into their on-board programming control using a microprocessor, Lattice provides ispCODE™ (C source code) to allow for customization of the ISP user interface.

For a programmer controlled by a serial RS-232 link, the Lattice pDS Software can call up the Windows Terminal Program. By using your PC to emulate a terminal, you can give the programmer the commands necessary to set it up to receive the .JED file. The Download command in the Windows Terminal program transfers the file to the programmer. Because the .JED file is an ASCII format, a text download is used.

For programming the ispLSI 1032E part, follow these commands.

1. From the Design Menu select the Program Option. This invokes the In-system Programming module.
2. The ISP module prompts for the name of the JEDEC file. Click on COUNTER.JED in the file list and then click OK. It may already have COUNTER.JED as the default file name. If so, click OK.
3. Programming takes a few seconds. If any errors are encountered, they are listed in the message box.

When programming is complete, the part is reset and sent back into operating mode. It can then be tested by applying the required inputs and reviewing the outputs.

Advanced Design Concepts

Working with Macros

The Lattice pDS Software comes with a library of over 200 macro logic elements. These logic blocks are similar to 7400 TTL logic. Some example macros are listed in Table 1.

For complete information on the Macro or TTL Libraries, refer to the *Macro Reference Manual* and *TTL Reference* that come with the Lattice pDS Software. In addition to using macros from the Lattice libraries, you can create custom macros from scratch or modify macros from the Lattice library to satisfy design requirements.

We are going to take a macro from the library that is identical to the counter just created, and cascade it with the counter. The macro element used to do this is named CBU24. The schematic diagram is shown in Figure 9.

1. Read in the previous design using the File Open command. The name of the file is COUNTER.LIF.
2. Choose the Library menu option and highlight Select to invoke the library window. Click on the System Lib button and then click OK.
3. Invoke the Macro Window by clicking on MACRO in the Menu Bar.
4. Select the macro *CBU24* from the list of macros.
5. Click on GLB C3 to select it.
6. Click on the *PLACE* command in the Macro Menu. This places the first half of the four-bit counter macro in GLB C3. The signal names placed in the GLB are the default signal names, and need to be changed to correspond to the signal names so that the router is able to connect them.
7. The original text in the cell was:

   ```
   CBU24_2(CAO,[Q0..Q3],CAI,EN);
   ```

   Change that to read:

   ```
   CBU24_2(CAO,[Q4..Q7],CO,CE);
   ```

   The default signal names are changed to match those already used in as shown in Table 2.
8. Perform a Cell Verify to ensure that no errors were introduced.
9. Click on *DONE* to close that GLB.

Table 1. Macro Logic Element Examples

Macro Name	7400 Part Equivalent	Description	Number of GLBs Used
AND2	7408	2 Input AND Gate	1/4
XOR2	7486	2 Input Exclusive OR Gate	1/4
FJK21	74112	J-K Flip-Flop with Asynchronous Clear	1/4
CBU34	74161	4-Bit Preloadable Binary Counter with Reset	1 1/4
BIN27	74247	BCD to 7 Segment Decoder	2
SRR38	74166	8-Bit Parallel In-Serial Out Shift Register	2
ADDF4	74283	4-Bit Full Adder with Look Ahead Carry	4 3/4

Table 8- 0002

Figure 9. Custom Binary Counter Cascaded with a Standard Macro Counter

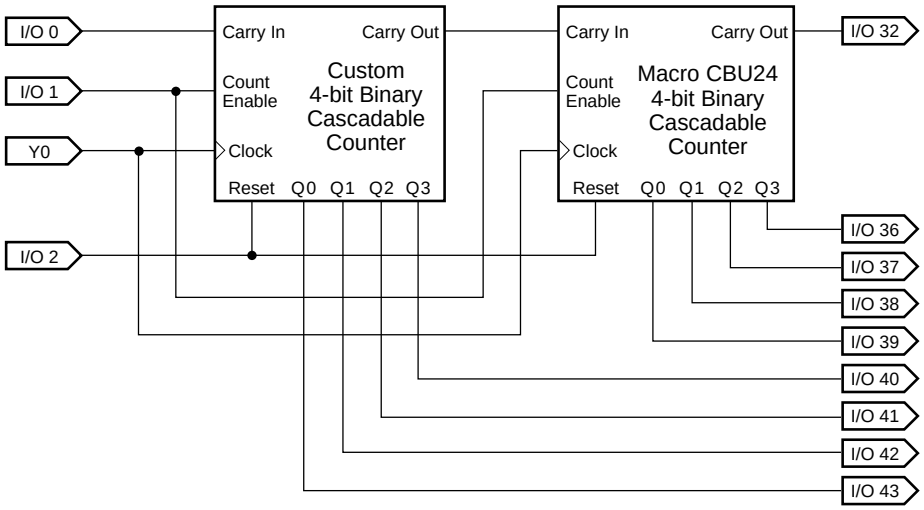


Table 2. Default Signal Names

Default	Signal	Is	Notes
CAO	Cascade Out	CAO	This is a new signal. We can use the default name.
Q0..Q3	Counter Outputs	Q4..Q7	We used Q0 through Q3 in the first counter. We need to assign new names so the router will not get confused.
CAI	Cascade In	CO	CO is the name that we assigned to the Cascade Out pin on the counter that we designed.
EN	Enable	CE	We called our Enable pin CE (Count Enable). This comes from a pin external to the device.

10. The macro occupies two GLBs, so the second half of the macro now needs to be placed. Click on *GLB C4* to place the second half.

11. As before, the signal names that were placed in the GLB were the default names. They also need to be edited. The Lattice software placed the following code into the cell:

    `CBU24_1([Q0..Q3],CAI,CLK,EN,CD);`
    Change it to read:

    `CBU24_1([Q4..Q7],CO,_CLK,CE,RST);`
    As before, we have changed the default signal names to match those that we are already using. See Table 3.

12. As before, perform a *CELL VERIFY*, and click on *DONE* when finished.

    The counter has now been placed, and the inputs connected, but the outputs are still floating. Connect them to the I/O cells as you did with the previous counter.

13. Select the macro called *OB11* from the macro list.

14. Click on IO Cell *#40* to select it.

15. Click on PLACE in the Macro Window. This configures I/O Cell #40 as an output buffer, but it used the default signal names. The text placed in the cell was:

    `OB11 (XO0,A0);`

Table 3. Renaming Default Signal Names

Default	Signal	Is	Notes
Q0..Q3	Counter Outputs	Q4..Q7	We used Q0 through Q3 in the first counter. We need to assign new names so the router will not get confused.
CAI	Cascade In	CO	CO is the name that we assigned to the Cascade Out pin on the counter that we designed.
CLK	Clock	_CLK	We named the signal that we brought in on pin Y0 _CLK
EN	Enable	CE	We called our Enable pin CE (Count Enable). This comes. in from pin 27.
CD	Clear Direct	RST	Our reset signal was brought in on pin 28 and called RST.

You should change it to read:

```
OB11 (XO4,Q4);
```

Q4 is the name of the first output of the counter. XO0 was changed to XO4 so there would not be duplicate I/O cell names when we place the next cells.

15. Click on IO Cell *#41* to select it.

16. Click on *PLACE* in the Macro Window. Change the default signal names to match those used in your design:

    ```
    OB11 (XO0,A0);
    ```

    Becomes:

    ```
    OB11 (XO5,Q5);
    ```

17. Use the same technique to connect I/O cell #42 to counter output Q6.

18. Use the same technique to connect I/O cell #43 to counter output Q7.

All the outputs are now connected, and the design is complete. As in the first design, you now need to do a *Global Verify* on the design, *Route* the nets and generate the *Fuse map*. You can see from this exercise how simple it is to complete a design using macros.

The use of macros is not limited to those in the Lattice Macro Libraries. Sometimes the standard macro is close to, but not exactly what you need. You can copy any of the standard Lattice soft macros into a personal library and modify them to meet specific needs. You can also create macros using Boolean equations and save them in your personal library for future use.

Summary

This application note describes how to design using pDS Software from definition to completion. In this Beginner's Guide, we:

- Looked at Lattice pDS Software and its various elements.
- Explained the design flow from beginning to end.
- Looked at the syntax needed for entering a design.
- Defined a small counter and partitioned it into GLBs.
- Entered the counter design into the development system.
- Took the design through the compilation process. (Verify, Route, and Fusemap).
- Programmed a part.
- Tested the design.
- Changed the design and introduced the use of macros.
- Recompiled the design and tested it.

If you have followed this procedure, you are ready to complete a design of your own.

LATTICE SEMICONDUCTOR CORPORATION
5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.
Tel.: (503) 681-0118
FAX: (503) 681-3037
http://www.latticesemi.com

November 1996