Compiling Multiple PLDs into ispLSI® and pLSI® Devices

Introduction

As high-density Programmable Logic Devices (PLDs) become more complex, they can combine larger designs previously implemented with low-density PLDs and SSI/MSI glue logic. The use of ispLSI and pLSI devices from Lattice Semiconductor can reduce manufacturing costs by: shrinking board size, simplifying test procedures, speeding development, and reducing the type and number of parts required to be kept in inventory. Designers familiar with PLDs and SSI/MSI devices can convert to Lattice ispLSI and pLSI devices with little effort. This application note addresses a procedure to convert a circuit designed with PLDs, MSI and SSI devices into the Lattice ispLSI and pLSI device format.

The basic steps required to convert the design are:

- Define the I/Os
- Convert the low-density PLD equations
- Combine the PLD source files
- Add any SSI and MSI functions
- Partition the logic into Generic Logic Blocks (GLBs)
- Import the file into the ispLSI and pLSI design environment
- Place and route using the pLSI and ispLSI Development System (pDS®)

Define the I/Os

The first task in the conversion process is to define the I/O pins of the Lattice ispLSI and pLSI device based on the circuit developed using lower-density devices. One must determine if the design is I/O-limited or gate-limited. If the design is I/O-limited, the circuit must be partitioned into a higher pin-count device, or two (or more) lower pin-count devices. A gate-limited design will mandate the design be partitioned into a higher-density ispLSI and pLSI device. This implies that there will be unused I/O pins. This can allow additional functionality to be designed into the device, providing it does not become gate-limited again.

A straightforward approach to estimate gate count is to use SSI, MSI and PLD equivalents. By adding up the total number of these circuit blocks required for a circuit, one can determine if the design will fit into a Lattice ispLSI and pLSI device. For example, the 1000 and 2000/LV family GLB (Generic Logic Block) of the Lattice ispLSI and pLSI family has 18 inputs and four outputs. Numerous functions implemented in 16V8, 20V8 and 22V10 devices can be fit easily into one GLB. However, in cases where five or more outputs are desired, partitioning into two GLBs will be necessary. Expanding this analogy, approximately one MSI device and two SSI devices can fit into a single GLB.

When converting a circuit implemented with MSI, SSI and PLDs, partitioning can be achieved by recognizing which nodes are best suited for interconnection within the ispLSI and pLSI device. The partitioning of logic will vary for different MSI, SSI or PLD devices. By determining which of these devices will be implemented completely within the Lattice ispLSI and pLSI device, it will become readily apparent which of the nodes should be kept within the ispLSI and pLSI device or allocated as an I/O pin. Signals which connect to a device not implemented within the Lattice ispLSI and pLSI device will be required to be an I/O. As a shot gun approach, one can simply draw a box around the circuit, count the I/O and gate requirement, and select the ispLSI and pLSI device meeting the requisite gate and I/O count. This task requires good engineering judgement and knowledge of device architecture to effectively use the ispLSI and pLSI device architecture.

Nodes which have a broad fanout should be considered for I/O unless all destination devices are implemented within the Lattice ispLSI and pLSI device. Naturally, nodes going off-board must be implemented as I/O pins on the Lattice ispLSI and pLSI device.

Clocking is another factor to consider when partitioning a circuit. In the Lattice ispLSI and pLSI 1000 device family, if the circuit requires more than the four global clocks available in the ispLSI and pLSI device, the circuit should be partitioned so that circuits with common clocks are in the same ispLSI and pLSI device. The global clock inputs are available on pins Y0, Y1, Y2 and Y3. Y0, Y1 and Y2 can be directly connected to any GLB, while Y2 and Y3 can be directly connected to any I/O cell. For the 3000 family, there are five clocks available. The pins Y0, Y1, and Y2 are GLB clocks and Y3 and Y4 are available for I/O cells. Alternatively, each GLB can generate its own Product Term (PT) clock from the output of a single product term within the GLB. This will allow up to 32 separate PT clocks within the ispLSI and pLSI 1032 device.

Compiling Multiple PLDs into ispLSI and pLSI Devices

PLD File Conversion

Once the circuit to be placed into the Lattice ispLSI and pLSI device has been defined, the process of converting the design into the ispLSI and pLSI format begins. Typically a design will be implemented with PLDs and a small number of MSI and SSI devices. Most of the PLD devices will have an associated source equation file. This file can be used as the basis for the design equations to be imported into the Lattice pDS Software.

Adding MSI and SSI Functions

By creating Boolean equations which emulate an SSI or MSI function, and subsequently importing that file into the Lattice pDS Software, SSI and MSI functions can be easily integrated into the design. Another method of implementing these functions is to look through the Lattice pDS Software Macro Library (or the ispLSI and pLSI Software Manual), to find the closest equivalent circuit to the function desired. This macro can then be edited if necessary, to provide the exact function required. The net result of either of these processes is to derive functionally correct equations which best utilize the ispLSI and pLSI device architecture.

Conversion of 3-Stated to Multiplexed Signals

Internal 3-state functions implemented in an ASIC or high density PLD can create problems such as undefined outputs. A better implementation of internal 3-state functions is to implement them with a ONE of N multiplexer function. The inputs to the multiplexer are the signals that are 3-stated together. The select lines of the multiplexer are individual 3-state enable signals. This technique is commonly used in the design of ASICs. Figure 1 illustrates an implementation of a 3-state function. The block diagram in Figure 2 shows a multiplexer emulating a 3-state function. The 3-state equations of Listing 1 would be rewritten for a ONE of N multiplexer as shown in Listing 2.

The AND function of the output enables (OE_A, OE_B) does not increase the number of product terms required to implement the various bus signal functions. This will always be true for product term oriented architectures

Figure 1. Implementation of 3-State Function

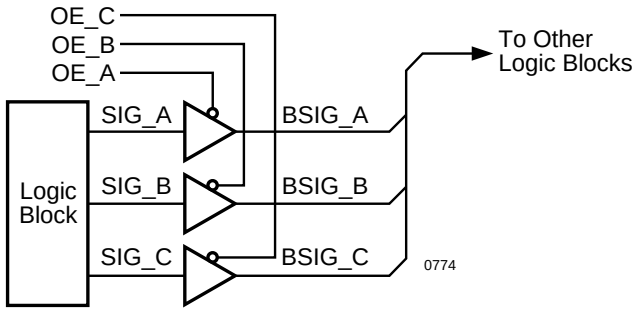
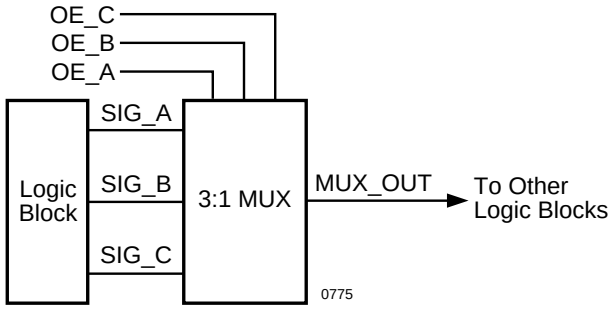


Figure 2. Block Diagram of Multilevel Emulating a 3-State Function



such as the Lattice ispLSI and pLSI devices. There are ten ONE of N multiplexer macros currently available in the ispLSI and pLSI Macro Library. By using these macros, the conversion may be readily accomplished by simply changing the default signal names within the Lattice macro.

Inversion Placement

Proper placement of active low internal signals may provide a significant savings in the utilization of the Lattice ispLSI and pLSI device resources as described in the following example.

Consider the equations shown in Listing 3. The original was entered in a "Product of Sums" form which becomes the "Sum of Products" form shown in Listing 4. Traditional PLDs require that logic be in Sum of Products form to be implemented in the architecture. This equation requires nine product terms to implement.

A better way to implement this function is to Demorganize (invert) the equation, as shown in Listing 5. By doing this, the implementation becomes two product terms versus nine for the non-inverted form. There are inversions available in each I/O cell and each input to the GLBs to re-invert the signal to get the original function.

Therefore, when manipulating equations to fit the Lattice ispLSI and pLSI architecture, consider placing inversions for active low outputs at the signal destination or at the I/O cell. The Lattice ispLSI and pLSI family can accommodate any active low signal with this technique as all inputs to the logic block have both true and complementary inputs. In

Listing 1. Original 3-State Equations

```
BSIG_A = SIG_A
BSIG_A.OE = OE_A

BSIG_B = SIG_B
BSIG_B.OE = OE_B

BSIG_C = SIG_C
BSIG_C.OE = OE_C
```

Listing 2. Multiplexer Equations

```
MUX_OUT = !OE_B & !OE_A & SIG_A #       // SELECT SIG_A
          !OE_B &  OE_A & SIG_B #       // SELECT SIG_B
           OE_B & !OE_A & SIG_C;        // SELECT SIG_C
*Note that OE_C is not needed in this implementation.
```

Listing 3. Original Function Required

```
OUT = (!IN1 # !IN2 # !IN3) & (!IN4 # !IN5 # !IN6);
```

Listing 4. Showing Sum of Products Form of Listing 14

```
out = (!in3 & !in6
     # !in2 & !in6
     # !in1 & !in6
     # !in3 & !in5
     # !in2 & !in5
     # !in1 & !in5
     # !in3 & !in4
     # !in2 & !in4
     # !in1 & !in4);
```

Listing 5. Showing Reduction of Product Terms with ! Use

```
out = !((!in1 # !in2 # !in3) & (!in4 # !in5 # !in60));
out  = (in1 & in2 & in3) # (in4 & in5 & in6);
```

other words a signal 'A' routed to a GLB, will have both 'A' and '!A' available within the GLB AND array. The outputs of the Lattice ispLSI and pLSI devices can also be selected as active high or active low.

Defining a Preset/Reset Mechanism

A frequently neglected but necessary requirement is a reset mechanism. All state machine designs should have a known power-up state. If a reset line is routed to all state machine registers for reset, significant routing resources will be unnecessarily used. The reset mechanism should take advantage of the hardware reset resources available in the Lattice ispLSI and pLSI device. Individual reset signals should be removed from the design equations and the hardware reset should be used. The Lattice ispLSI and pLSI devices have two reset mechanisms: a global reset for all registers and an asynchronous reset for each GLB or I/O cell.

Many high-density device architectures provide only reset and no preset mechanism. Consider complementing the output requiring preset and using the hardware reset. If that is not possible, make the preset synchronous by adding a preset term into the design equations.

Circuit Partitioning

The *.DOC files produced by third-party compilers are in an industry standard format. These files contain the reduced equations which are derived from the source file, JEDEC maps, high-level state machine language, truth table or standard Boolean equations. The individual PLD and SSI/MSI *.DOC files should be combined into a single source file for partitioning into the ispLSI and pLSI a device.

By grouping the equations into groups of no more than four outputs, the PLD equations can be partitioned to fit into the GLBs of the ispLSI and pLSI device since there are four outputs per GLB. Headers and trailers must be placed around the four equations to indicate to the Lattice pDS Software into which GLB the equations should be loaded. The syntax is shown in Table 1.

In the ispLSI and pLSI 1000 and 2000 device families, each GLB has 18 inputs, 20 product terms and four registered or combinatorial outputs. Additionally, there is product term combining among the four outputs and an optional Exclusive OR gate which is fed by a single product term and an AND/OR term. The software will automatically place a given set of four equations into a GLB. The ispLSI and pLSI 3000 and 6000 device families have 24 inputs in each Twin GLB™ (see Figure 3), a programmable AND array and two OR/exclusive-OR arrays, and either outputs which can be configured to be either combinatorial or registered.

If the PLD equations do not fit into a GLB, the Lattice pDS Software will give a message as to why. If there are too many inputs, the equation can be moved into another GLB and a new equation brought into the current GLB which does not exceed the limit of 18 inputs.

As previously stated, every GLB is allowed one clock. This clock may come from either one of the four global clocks or a clock generated from a product term (.PTCLK). Ensure all registered outputs in a GLB have a single clock.

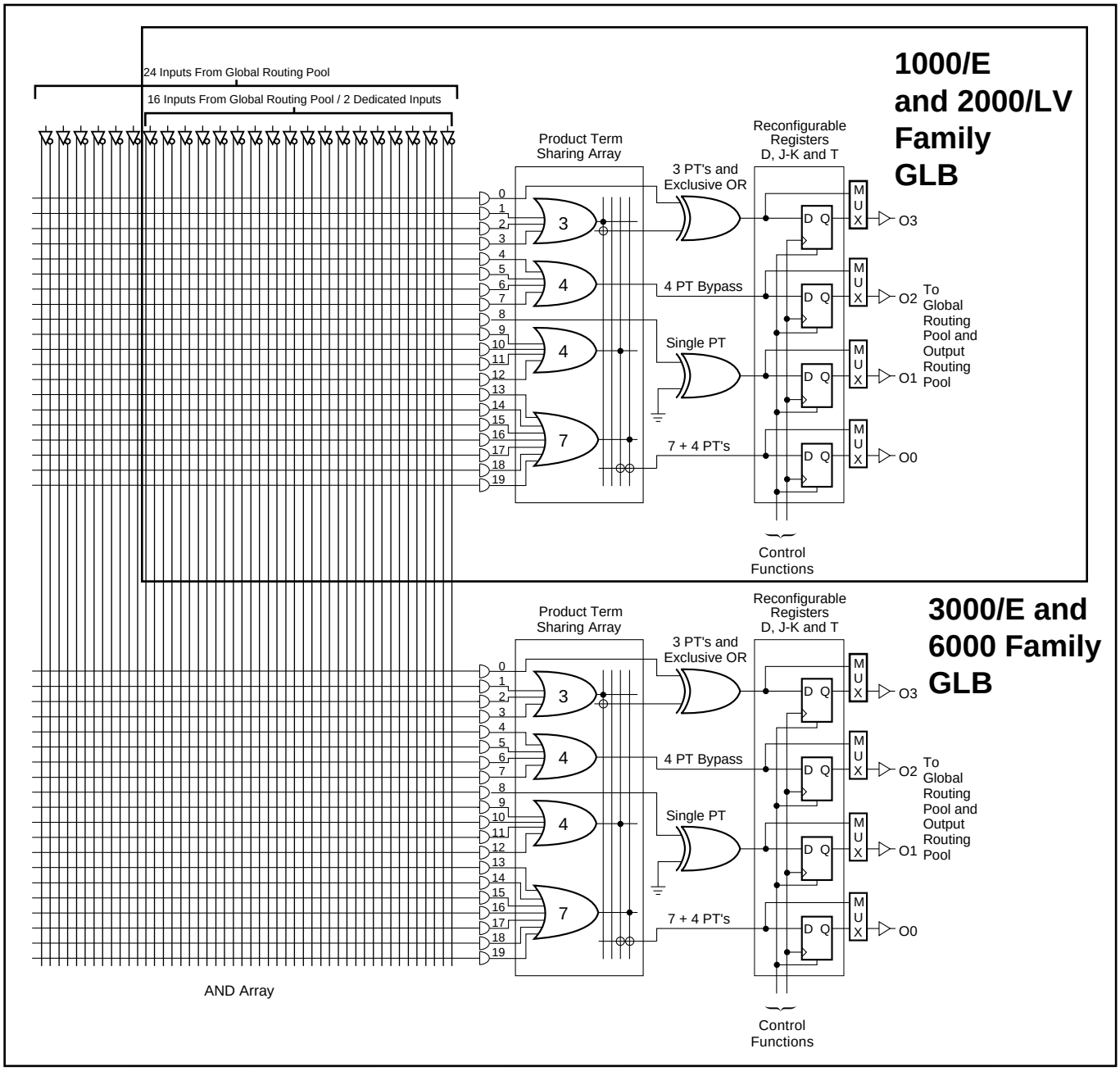
If an equation contains product terms which cannot be allocated into one GLB, consider exchanging a complex equation for one of less complexity in another GLB. If this trading of equations is not possible, simply move the equation into an empty GLB. In general, try to keep equations with common inputs in the same GLB. If a function requires a high number of product terms (product term combining), try to make use of the product term groups.

Moving a registered equation from one GLB to the next will not degrade performance as the interconnect delays between all GLBs are constant. Combinatorial equations

Table 1. Header and Trailer Syntax

Header	PLD Equations	Trailer
SYM GLB A0 <GLB NAME> 1;	Signal1.clk = ...;	END;
SIGTYPE Signal1 REG OUT;	Signal1 = ...;	END;
SIGTYPE Signal2 OUT;	Signal2 = ...;	—
SIGTYPE Signal3 CRITICAL OUT;	Signal3 = ...;	—
SIGTYPE Signal4 REG OUT;	Signal4 = ...;	—
EQUATIONS	—	—

Figure 3. GLB Diagram Showing Product Term Sharing Combinations



0845a

may have an extra GLB and unit interconnect delay added to the propagation delay if the implementation requires more than 18 inputs and 20 product terms. If an equation will not partition into a single GLB, the equation must be split into two equations and then cascaded. For Registered equations consider pipelining the intermediate equation(s) to keep the performance at the same level.

The previous steps are all that are required to place PLD type designs into the GLBs of the ispLSI and pLSI devices. Note that no syntax changes of the AND/OR portions of the equations were required.

Definition of I/O Cells

The final step in the conversion process is to define the I/O cells. The basic I/O cell definition for an input and output pin is shown in Listings 6 and 7 respectively. Because the device is routed according to signal names, all I/O cells will automatically be connected to the proper internal nodes. Other variations are shown in Table 2. For more details refer to the pDS Development Manual under Macro Library.

Import and Verify the Design

Now that the design has been partitioned into the GLBs, the device ASCII design source file needs to be imported into the Lattice pDS Software to be verified, placed and routed. By using the FILE and IMPORT LDF commands, the ASCII file containing the design will be imported into the Lattice pDS Software. The pDS Software will check the syntax of each GLB and I/O cell and translate the ASCII file to a binary LIF (Lattice Internal Format) file.

All syntax errors must be eliminated to successfully import a file. After a successful import into the Lattice pDS Software, any SSI or MSI devices can be placed into GLBs. This can be done by using the available macros from the Lattice Macro Library, by using the designer's custom macros, or using Boolean equations. After this is accomplished, select the DESIGN VERIFY command which performs a global design rule and connectivity check. After a successful global design verify, the design is ready for automatic place and route.

Listing 6. Basic Input I/O Cell Definition

```
SYM IOC IOXX 1;              // IOXX = IO CELL NUMBER
XPIN IO/I X_SIG;             // IO = IO PIN; I = DEDICATED INPUT CELL
IB11 (SIG, X_SIG);           // IB = INPUT SIGNAL
END;
```

Listing 7. Basic Output I/O Cell Definition

```
SYM IOC IOXX 1;              // IOXX = IO CELL NUMBER
XPIN IO/I X_SIG;             // IO = IO PIN; I = DEDICATED INPUT CELL
OB11 (SIG, X_SIG);           // OB = OUTPUT SIGNAL
END;
```

Table 2. I/O Cell Signal Type Description

I/O Cell Type	Signal Description
IBXX	Input Pin
IDXX	Input Register
ILXX	Input D Latch
OBXX	Tri-State Output Pin
OTXX	Tri-State Output Pin
BIXX	Bidirectional Pin
BIIDXX	Bidirectional Pin with Registered Input
BIILXX	Bidirectional Pin with Latched Input

Place and route is invoked with the DESIGN ROUTE commands. After place and route, the fuse map can be generated and the design downloaded to a programmer. In the case of an ispLSI device, it can be programmed via the ispDOWNLOAD™ Cable connected to a parallel port of an IBM-compatible PC.

Summary

Converting a design from low-density PLDs to Lattice ispLSI and pLSI high-density PLDs is quick and easy, as long as a few guidelines are followed:

1) Decide if the design is I/O- or gate-limited.

2) Choose the appropriate ispLSI or pLSI device.

3) Use as much of the original Boolean equations from the low-density source file as is practical.

4) Convert 3-state outputs to a ONE of N multiplexer scheme.

5) For reset functions, use the global reset for the entire device or the asynchronous reset for specific GLBs.

6) Use no more than 18 inputs or four outputs per GLB when partitioning the logic for 1000 or 2000 family devices and no more than 24 inputs and eight outputs for 3000 and 6000 family devices.

7) Use no more than one clock per GLB.

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