Lattice™
Semiconductor
Corporation

Optimizing an ispLSI Design

Introduction

Getting the most out of the Fitter effort is an important aspect of the design activity. Most designs will route to specifications with little or no extra input. These specifications may be utilization, performance, pin locking or combinations of these. To get the most out of your designs, there are techniques that can help the development tools in that effort. This application note discusses some of the techniques that may increase design optimization.

In general, optimizing a design for *speed* implies that more logic resources will be used. Similarly, optimizing a design for *area* implies that lower speed will result. Thus, a trade-off exists between area optimization and delay optimization. The time required to compile very large or dense designs can be significant. It is always best to increase the Fitter's efficiency to minimize the time required to compile your design.

Device routing, device resource utilization, and Fitter efficiency are controlled by Design Attributes and Fitter Control Options. Choosing optimal values for routability is a trial and error operation due to the complex nature of the compilation process and its dependency on the characteristics of the design. The following sections focus on how to choose Design Attributes and Fitter Control Options to achieve the best overall results.

If your design is not too large for the intended device, yet it does not pass through the Fitter due to routing or resource limitations, the design is probably overconstrained and some attributes or parameters must be relaxed to achieve routability. Listed below are the Design Attributes and Fitter Control Options in their order of effectiveness in improving routability and resource utilization. The most difficult ones for the Fitter are listed first. Use this list as a guide to determine which attributes have the most impact on the compilation process. But remember, these are only guidelines.

Optimizing for Area (Device Utilization)

The primary Fitter Control Options and Design Attributes that affect logic density, and hence chip utilization, are the following:

CRIT
EFFORT
LOCK
LXOR2
MAX_GLB_IN
MAX_GLB_OUT
PRESERVE
PROTECT
SCP/ECP, SAP/EAP, and SNP/ENP
STRATEGY

If your primary design consideration is placing the largest amount of logic into a device, the following guidelines will be helpful:

1. Remove all PRESERVE attributes to allow the Fitter to optimize your design better.
2. Use the USE_GLOBAL_RESET control option to allow the Fitter to use the global reset pin instead of an I/O pin.
3. Remove all CRIT attributes to allow the Fitter to use the Output Routing Pool.
4. Try different levels of EFFORT (levels 1 through 3).
5. Remove all pin specifications (LOCK) to allow the Fitter to choose pin locations.
6. Remove all LXOR2 attributes to allow the Fitter to decide on XOR usage when appropriate.
7. Increase MAX_GLB_IN to allow the Fitter to use Generic Logic Block (GLB) resources more extensively.
8. Increase MAX_GLB_OUT to allow the Fitter to use GLB resources more extensively.
9. Remove all PROTECT attributes to allow the Fitter to optimize your design better.
10. Remove all path restrictions (SCP/ECP, SAP/EAP or SNP/ENP) to allow the Fitter to use any possible mapping scheme.
11. Use hard macros in your design wherever possible.
12. Use STRATEGY AREA.

Optimizing an ispLSI Design

Although the above guidelines provide a good starting point for achieving a denser design in an ispLSI® or pLSI® device, the methods employed by the pDS+™ Fitter may not always produce optimum results and unexpected outcomes may occur. When unexpected results are encountered, try different Design Attributes and Fitter Control Options. Use caution when experimenting with extreme values for Fitter Control Options or with extensive use of one or more Design Attributes. This may lead to a denser implementation of the design, but unsuccessful routing.

Reset and Clock Lines

The use of dedicated global reset and clock pins can free up logic resources and unburden the routing resources. Use of global reset eliminates high fan-out. If the designer requires an active high reset, signal inversion will be required outside the device if the global reset pin is to be used. The alternate method to specify resets and clocks in a design is via Product Terms. This method uses logic resources and is unavoidable if the designer requires reset and clock signals which are created using combinatorial logic. Product Term Reset and Clock signals created within a GLB are available only to the GLB in which they are created. Therefore, for every four GLB registers using the Product Term Clock, the logic for the Product Term Reset/Clock is duplicated.

Global Clocks

Always try to use a global clock rather than a Product Term Clock (PTCLK). The GLB I/O global clock signals are routed through dedicated routing channels independent of the logic routing channels. In general, the use of global clocks will increase design routability.

Product Term Clock (PTCLK)

Typically, the PTCLK gets its source from a signal being routed through the GRP. If the PTCLK is used in more than four GLBs, it may help to make it a global clock instead. If there are more unique clocks in the design than there are global clocks available, the rule of thumb is to route the fastest on the global clock and the slower on PTCLKs. If the clocks are relatively slow (less that 20MHz) use global clocks for the clock signals that go the largest number of GLBs. This will increase the probability of a successful route on highly-utilized devices

Product Term Reset (PTRESET)

As with PTCLKs, PT RESET is routed in the GRP. ALL registers automatically have the global reset pin connected. In the case of the ispLSI 1016, 1016E, 2032 and 2032LV devices, if the ‘Y1 as reset’ option is not used, the reset function will still be asserted on power-up or after programming. PTRESET is often redundant to the global reset function. Most designs do not require PTRESET. Product Term control functions are generated by PT12 or PT19. Using PT controls will lower the number of PTs available for other logic functions. PT control can also increase GLB usage, if used indiscriminately.

ISP OFF

For input-intensive designs, it may help to specify ISP OFF. This allows the Fitter to use the dedicated inputs which are muxed with the ISP™ programming pins (dual function pins). Be aware that dedicated inputs only feed their associated megablock. This does not mean that the In-System Programming feature cannot be used. ISP ON can be specified and implemented via a property file or in the source file.

Pin Locking

Occasionally, pin assignments must be locked, restricting routing resources. Try removing lock attributes wherever possible for better routability. When locking I/Os, it is best to intermingle input and output signals. Never lock input-only signals to a megablock.

Hard XOR

Another Lattice feature which can free up logic resources are the two input XOR gates built into the GLB architecture. These are referred to as hard XORs or LXOR2s. These built-in logic functions can be used in place of combinatorial logic, reducing delay and resources used.

Logic Utilization

For best results, it is advisable to use up to 90% logic utilization, effectively leaving at least one GLB in every megablock for routing flexibility and future design revisions.

Software Tools

Lattice offers two software tools for the ispLSI device families: pLSI Development System (pDS®) and pLSI Development System Plus (pDS+). pDS+, also known as ‘the Fitter,’ is designed to interface with third-party design entry tools, such as Synario, Viewlogic, Synopsys, Mentor Graphics, etc. Each tool provides designers with a different level of freedom in the entry and partitioning phase. Optimization techniques are best explained in terms specific to the tool and the design entry technique used.

pDS

pDS provides for the manual partitioning of logic, giving the designer total control of the placement of logic equations within GLBs. Manual entry and partitioning can yield higher device utilization. However, a designer must be familiar with the architecture of the Lattice ispLSI device in order to take advantage of this interactive capability. In manual partitioning, there are certain techniques that may help pDS route the design logic more easily.

1. Try to group logic using common inputs in the same GLBs. This will lower the fan-out of the signals and the number of connections required in the Global Routing Pool (GRP). Fan-out in Lattice devices is counted by the number of GLBs to which a signal travels.

2. Always try to use a global clock rather than a Product Term Clock (PTCLK). Global clocks are routed only to the clock inputs of GLBs and the I/O registers on dedicated clock lines. This increases the routability of the design. The PTCLK typically gets its source from a signal being routed through the GRP. If the PTCLK is used in more than four GLBs, it may help to make it a global clock. If there are more unique clocks in the design than there are global clocks available, the rule of thumb is to route the fastest as global clocks and the slower as PTCLKs. If the clocks are relatively show (less than 20 MHz), use the global clocks for those clocks that go to the most GLBs. This will increase the probability of a successful route.

3. As with PTCLKs, the Product Reset (PTRESET) is also routed in the GRP. Global reset pins are automatically connected in ALL registers. In the case of the ispLSI 1016, 1016E, 2032 and 2032LV devices, if the “Y1 as reset” option is not used, the reset function will still be asserted on power-up and after programming. PTRESET is often a redundant function to the global reset. Most designs do not require PTRESET. Product Term control functions are generated by PT12 or PT19. The use of PT control functions therefore reduces the number of PTs available for other logic functions. Unnecessary use of PT control functions will increase the number of GLBs used and adversely affect device utilization.

4. If the design allows, try to use a GLB register instead of a GLB latch. A latch is implemented with the and/or array and combinatorial feedback. The register of flip flop is embedded in the GLB architecture, allowing the software to more easily place logic in front of the “D” input. Using the GLB register may achieve higher utilization.

5. When a product term control function is used, (PTCLK or PTRESET), try to group all logic under that control function in the same GLB.

6. Consider the specifications of the design. If fast paths are required, use the CRITical design attribute. The use of the CRIT attribute instructs the router to use the four product term bypass in the GLB or the Output Routing Pool (ORP) Bypass for the I/O. When a four product term bypass in the GLB is specified (using CRIT), it may increase utilization if other critical logic is grouped in the same GLB.

7. In the partitioning process, leave an output unused in each GLB, if possible. This will allow the router to duplicate at least one of the GLB output signals and thereby increase routing success. This duplication is done automatically if the software encounters a conflict in connecting a signal to all its destinations.

8. If the design permits, it is best to not lock all I/O cells in a single megablock with inputs only. It is better to intermingle inputs and outputs or bi-directional I/O functions when locking pins. This will also help reduce ground bounce and increase noise immunity.

9. Reserving “ISP” pins will instruct the router to reserve those pins. Because ISP pins can be used as dedicated inputs, reserving them cuts the number of available pins during routing.

pDS+ Routing and Utilization

Device routing, device resource utilization and Fitter efficiency are controlled by Design Attributes and Fitter Control Options. Choosing optimal values for routability is a trial and error operation due to the complex nature of the compilation process and its dependency on the characteristics of the design. The following tips enable designers to take advantage of the Lattice architecture for better routing and utilization.

Macros

Macros are complex logic functions such as arithmetic functions (e.g. adders), encoders, decoders, counters, logic gates, MUX/DMUX, registers and various types of I/Os. Using macros can increase both performance and utilization of device resources.

There are two types of macros – hard and soft. Soft macros can never be hard macros, but hard macros can be made soft. A soft macro is a predefined netlist of a particular logic function. When a soft macro is used, it will be routed according to available device resources. Hard macros are pre-mapped to the Lattice architecture to optimize for resource utilization and performance. For example, CBD28 (eight-bit down counter with asynchronous clear, enable, carry-in/out) is a hard macro which occupies 2.25 GLBs. This is a very efficient utilization of GLB resources and greatly enhances performance. However, the Fitter will not allow additional logic to be placed in the remaining (.75) portion of the unused GLB. Making this a soft macro removes this restriction, but causes the macro to be routed in available resources and could affect performance.

It should be pointed out that not all design entry tools allow for the use of hard macros.

VHDL/Verilog designers can take advantage of macros to improve utilization and timing efficiency. It has been observed that certain macros have successfully routed in two levels of logic (i.e. a MAG8, eight-bit magnitude comparator). In contrast, functions written in raw VHDL/Verilog-HDL code may route in as many as four levels of logic, illustrating the effectiveness of macros in making the design process easier and more efficient.

Sweeping Your Design

‘Sweep’ is a term which means to expose your design to all or multiple possible user-defined Fitter Control Options. The options varied in a ‘sweep’ are:

Effort (-e switch, values 1..4)

Max_GLB_In (-m switch, values 2..24)

Max_GLB_Out (-n switch, values 1..4)

Strategy (-s switch, values a(area), d(delay), n(no_optimize))

A ‘sweep’ file consists of many dpm commands, with each Fitter control option varied in value until all reasonable options have been attempted. ‘Sweep’ files are available for both the PC and Unix workstations. For example, a portion of a ‘sweep’ file (for PC DOS applications) might look like this:

```
dpm  -m 18 -n 4 -s d -e 4 %1 -p   %2
copy %1.rpt %1d.484
del %1.fxp
dpm  -m 18 -n 4 -s d -e 3 %1 -p   %2
copy %1.rpt %1d.384
del %1.fxp
dpm  -m 18 -n 4 -s d -e 2 %1 -p   %2
copy %1.rpt %1d.284
del %1.fxp
dpm  -m 18 -n 4 -s d -e 1 %1 -p   %2
copy %1.rpt %1d.184
del %1.fxp
dpm  -m 18 -n 4 -s a -e 4 %1 -p   %2
copy %1.rpt %1a.484
del %1.fxp
dpm  -m 18 -n 4 -s a -e 3 %1 -p   %2
copy %1.rpt %1a.384
del %1.fxp
```

To launch a ‘sweep,’ you must have a ‘sweep’ file in the project directory or in a directory that is in the path statement. The ‘sweep’ file is a simple batch file with the filename `sweep.bat`. The ‘sweep’ is started by issuing the following command:

```
sweep design_name part_name
```

(The default extension of dpm is .laf, which is not required. The part_name is the complete device name as it appears in release notes, the Lattice Semiconductor Data Book or the lscpart.lst file in the config directory of Lattice software).

The results from this ‘sweep’ batch file are written to a unique report file that can be reviewed by the designer to help determine the best control options for performance, utilization or both. The copy command, shown in the example above, writes the original Fitter .rpt file to files named by the following command:

```
copy %1.rpt %1a.384
```

The file name ‘%1a.384’ represents a compiler/Fitter run with the Fitter control options shown below:

Strategy = a (area)

Effort = 3

Max_GLB_In = 18

Max_GLB_Out = 4

The ‘%1’ represents the design_name entered in the ‘sweep’ command. Once the ‘sweep’ has completed, you can review the report files to identify successful routes and their respective performances.

Hard XOR Gates

Use Hard XOR in GLBs to decrease product terms and increase optimization. To ensure the more efficient XOR gate is used whenever possible, there are several things a designer can do. First, combinatorial inputs to the XOR gate should be described as temporary nodes. Then the temporary node should be substituted for each of the inputs. This will help the Fitter to reduce the number of Pts and increase performance. It should also be noted that the Fitter will not always use the hard XOR function. It is dependent on the logic and how the logic is presented to the Fitter.

Registers

If the design allows, try to use a GLB "D" register instead of a latch. A latch is implemented with combinatorial logic and does not use the "D" register. This will increase the number of product terms and the number of GLBs used. A "D" register is embedded in the device hardware and using it will improve device optimization.

Other Suggestions

In some counter designs, timing and routing can be improved by not using (count-1) or (count+1) declarations. Instead, declare each state explicitly as a function of combinatorial terms and "D" registers. If possible, use XOR gates as described earlier in this section. This method may require more effort for large counters, but the result may be a decrease in product terms, enhanced routing and a possible increase in performance.

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