



# Cutting fpga design time

A more sophisticated approach to reducing fpga debugging cycles.

By **Joerg Kalita**.

**F**or many fpga designs, basic knowledge of the place and routing capabilities of modern pld design tools, as well as a minimum of design preferences, is sufficient to produce reasonable results. But if a critical performance metric needs to be achieved, a more sophisticated approach is needed.

The tools embedded within today's pld software suites can help save time and money during the design implementation stage. The principles are outlined using Lattice Semiconductor's ispLEVER as an example.

Design information can be imported from a variety of sources, such as Vhdl, Verilog, schematic entry and EDIF. Pre-configured IP cores, including those from third parties, can also be imported. A Matlab/Simulink interface, along with synthesis and RTL simulation tools from Synplicity and Mentor Graphics (Synplify, Precision and Modelsim), are also provided.

ispLEVER imports data from various sources and constructs a common database. Different netlists can be bound together to handle mixed sources, such as Vhdl and Verilog, in one project – useful when the synthesis tool does not handle a mixed input.

After the database is constructed, the next step involves the Design Planner – the main control for the input of environmental conditions. Users can not only input the usual preferences, such as

frequency and I/O timing, but also relevant design information, such as pin placement, driver output, I/O standards and global attributes.

The Design Planner can also be used to check such things as critical paths using the Path Tracer. Paths can be displayed graphically and a Floor Planner is also available. In this way, designers can quickly get a feeling as to whether the pin out is optimal for the design and whether related logic blocks are well placed.

At this point, it is helpful to mark the components already in source code, as well as their associated logic blocks. This can be done using the 'HGROU' attribute. The associated logic block is then marked with a colour in the Floor Planner. Design information can be obtained from the static timing report, HTML Fitter or the Performance Analyst. If necessary, manual adjustments can be made.

The algorithms in ispLEVER are timing driven; they take the design engineer's preferences and attempt to obtain a reasonable fit. If preferences are not well defined, the resulting design will be less than optimal.

After logic synthesis, the process moves to timing driven place and route. This function is split between MAP Design (or, more commonly, mapper) and place and route (more commonly, the parser). The mapper takes the resulting netlist and allocates to it the fpga's logic elements. The parser then positions these logic elements and connects them.

At this point, no timing information has yet been generated. However, a theoretical maximum frequency can be determined. If this doesn't meet the desired criteria, the engineer can adjust the prefer-

ences to achieve the required performance.

Design techniques, such as pipelining, may be used. Although these can add register steps, they may allow the mapper and parser to achieve shorter interconnects and a higher frequency. Timing driven synthesis tools can also improve a design by trading timing, placement area and frequency.

The designer may want to specify how many iterations the parser can

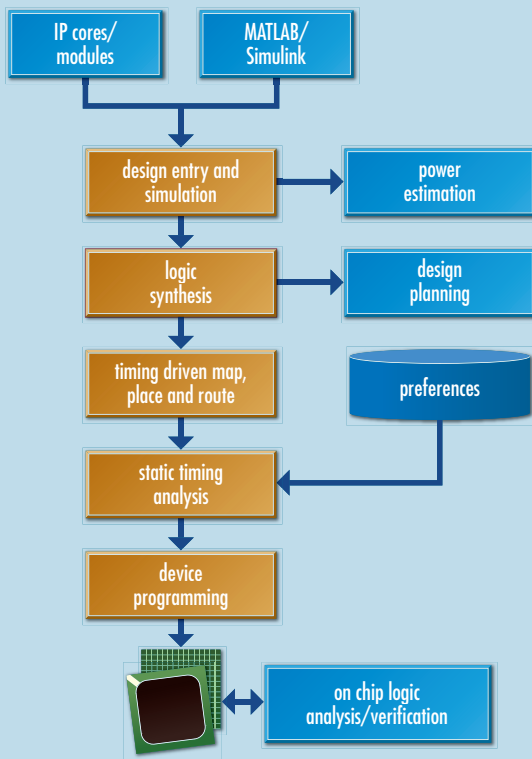
*"This cuts development time by helping debug designs in ways that more traditional hardware methods cannot."*

Joerg Kalita, **Lattice Semiconductor**

attempt. By adjusting the placement effort, combined with iterations in the place and route process, a significant amount of development time may be saved. When the design is complete and meets its overall goals, the process can be adjusted to tighten those constraints. This may result in the design fitting into



**Figure 1:** The typical design flow



a smaller device with the same pin out.

In many cases, this method leads to good 'pushbutton' results. But there are tools and design methods available that can speed the debugging cycle. These are divided into two groups:

- Tools that simplify debugging
- Partitioning and incremental design.

IspTRACY consists of three tools: Generator; Linker; and Logic Analyser. Small user defined IP modules can be used to feed live signal information from internal nodes (signals not accessible at device pins) back to the pc. This information can then be displayed and manipulated with the logic analyser interface, allowing functional simulation of the design at the hardware level.

This cuts development time by helping debug designs in ways that more traditional hardware methods cannot. When real time information is needed, for example to measure internal signals, the EPIC Design Editor can be used. This provides access to the physical implementation of the design. Details like route intercon-

nect, physical element programming and I/O buffer configuration can be examined or edited after place and route. Using a push button process, EPIC constructs a list with all internal nodes on one side and free pins on the other. The designer can then 'drag and drop' to connect any node with any free I/O.

### Incremental design

When relatively minor changes are required, it is desirable to make them without recompiling the design. This can be achieved using the EPIC Design Editor. Other methods, such as Guided Design Flow, Re-Entrant Routing and the Memory Initialisation Process, are accommodated in ispLEVER.


With Guided Design Flow, the designer presents a successful netlist (.ncd format) and if this matches the required design parameters, the object is used without being placed and routed again. Apart from saving time, this process uses proven timing parameters.

Where the timing performance of the overall design needs to be improved, Re-Entrant Routing can use a previously routed netlist. Alternatively, in the memory initialisation process, this netlist is modified only with respect to the value of the memory block. The mapper and parser are not involved.

With the increasing complexity of today's logic components, even low cost families, like the LatticeECP2M, can

reach 100k look up tables. For this reason, it can be useful to partition the design into modules, which then can be developed separately. Lattice calls this method semi-conductor block modular design.

The advantage is that engineers, who may be in different locations, can develop modules in isolation. There still needs to be a Design Architect, responsible for the total project, who budgets the resources for the separate groups and allocates the chip area location on the fpga. Within this control, design groups can develop, simulate and place and route, with the resulting logic placed only in the area allocated by the Design Architect. After all modules have been completed, the Design Architect assembles them and the whole function is tested.

A recent addition to ispLEVER is HDL Explorer, which integrates design creation, analysis, verification and documentation. The tool generates graphical representations of a design's hierarchical structure and connectivity, based on the source HDL. It is useful for IP integration, design maintenance and reengineering of complex fpga HDL designs. HDL Explorer also helps the designer visualise higher level abstractions of the design structure and reduces the time required for design management and documentation. 

### Author profile:

Joerg Kalita is a field applications engineer with Lattice Semiconductor.

**Figure 2:** The ispTracy logic analyser

