Certify Technology Backgrounder

1. *Hardware Solution for RTL Verification*

Process technology is capable of delivering devices with large transistor counts in relatively small die-sizes enabling ASIC designers to integrate more functionality on silicon. The ability to integrate different functions from the system onto one ASIC has created the System-on-Chip (SoC) "revolution". Though SoC designs are offering unprecedented integration resulting in cost savings and higher performance, the resulting complexity of these designs makes functional verification a monumental task. It is not easy to evaluate the quality the verification test suite and ensure that it is testing all the functionality of the ASIC with suitable coverage for all possible corner cases.

The second problem faced by design and validation teams is that of at-speed or near-speed verification. Software-based simulation technology is not being able to keep pace with the demands of verifying designs in their real-world environments. Verifying these complex SoC designs against events (OS boot, bit-error rates, protocol compliance) and subjective criteria (video and communications applications) has created a need for at-speed or near-speed verification. For example, a change in the image or sound quality based on an algorithmic change in the design cannot be verified effectively by using software simulation technology. At-speed or near-speed verification is enabled through the use of hardware-based prototyping platforms.

SoC designs are designed for systems that need application software, OS and firmware support before they can be productized. Building FPGA-based prototypes allows system vendors to get early starts on OS ports, driver development, application software development and porting as well as system/vendor inter-operability testing. Time-to-market is not just dependent on the hardware functionality but on software availability.

"Software design and validation must begin at the earliest phase of an SoC design project, and EDA and software tools must work very closely together", according to Michael Kaskowitz, vice-president and general manager of the Embedded Software Division of Mentor Graphics Corp. He adds that "The lack of access to internal signals and buses call for a new designing paradigm that gives software tools access to hardware simulation models".

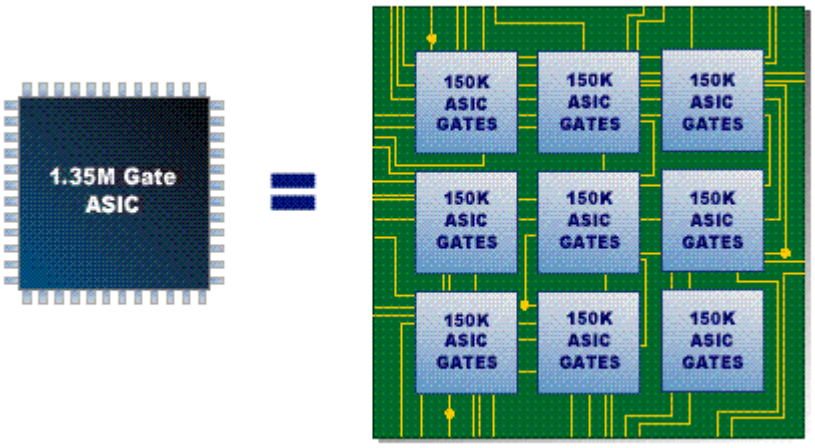
As using IP becomes a part of SoC design practice, integrating IP into designs successfully becomes a key to meeting spec and schedule requirements. Use of IP is also

fraught with problems such as validation and ease of its integration into the design. Hardware prototyping enables IP vendors and their customers alike, to evaluate, integrate and validate IP in SoC designs.

1.1 New FPGA Capabilties

FPGA vendors are now shipping devices based on state-of-the-art semiconductor fabrication process and packaging technology. Xilinx is shipping its Virtex FPGAs with gate counts range from 50,000 to 1,000,000 system gates running at clock speeds up to 200 MHz. Altera is shipping The APEX device family ranges from 100,000 to nearly 2 million system gates and is shipping on a 0.22-µm, six-layer-metal process. FPGAs today support 2.5V/3.3V operation, support a variety of I/O signaling standards and have PLL circuitry for minimizing clock skew inside these large devices. A variety of packages offering up to 500 user I/O in a variety of packages are available today. Looking at these features and their re-programmability, it is easy to understand why FPGAs are ideal for prototyping large SoC designs to possibly evaluate third party IP and verify SoC functionality.

On an average, the functionality provided by 1 ASIC gate maps to 4-6 FPGA gates, depending on the architecture of the FPGA device. Taking into account the high gate-counts of SoC designs and the "FPGA-ASIC gate equivalency" issue, we can see how most SoC designs will map into multiple FPGAs. ASIC designs of 200K-gate and greater will have to be partitioned across multiple FPGAs.



Designers now struggle with the problem of breaking up their ASIC into multiple FPGAs for prototyping. This is a tedious task often performed by the most experienced designers. Today, partitioning and synthesis are two separate steps. The user must partition logic blocks into FPGA size blocks, then synthesize each block into FPGA specific technology libraries. If the partitions do not fit into the desired FPGAs, the

partitioning must be redone and each FPGA must be re-synthesized, which is extremely time-consuming.

1.2 Tool and Methodology Limitations

Some of the problems a designer faces while prototyping designs are

- Implementing prototype using original source HDL
- Partitioning ASIC design without analysis and reporting tools
- Long synthesis runtimes
- Many synthesis-partition-P&R iterations
- Getting optimal performance from the prototype

The fundamental problem here is that of mapping a design targeted to an ASIC process to an FPGA technology. This places a stringent set of requirements on any solution that attempts to optimize FPGAs for ASIC prototyping. The solution has to be able to support high capacity, offer strong language support, deliver fast run times, provide ease-of-use and deliver best-in-class synthesis results. Additionally, this technology needs to solve inter-FPGA and intra-FPGA timing issues while keeping the design HDL intact.

2. Current approaches

2.1 Synthesis followed by Partitioning at the gate level:

One approach to prototyping a design is to attack the partitioning problem at the gate level netlist. This flow entails synthesizing the RTL to the target technology and then using a gate level netlist partitioner. The partitioned netlist is then passed on to the FPGA vendor place-and-route tools. Following this step, the prototype board is implemented by placing all devices on the board, routing the board and programming the FPGA devices.

Often, designers have to go through multiple synthesis-partition-P&R iterations before they get usable results. Should the partition fail to converge, the designer has to go back to the synthesis step. Multiple synthesis-partition-P&R iterations take significant amounts of time and compute resources for large designs. Multiple synthesis runs are usually unavoidable though, because design partitioning is usually done without accurate, implementation specific feedback.

Large ASIC designs are usually split into "synthesis-sized" blocks; the size of each of these blocks depends on the synthesis tool's capacity to handle designs efficiently and is typically 50K-60K gates. Therefore, time taken to synthesize the design is the product of the time taken to synthesize the design and the number of runs needed to get optimal results from synthesis.

$$\text{Time}_{\textit{one synthesis iteration}} = \frac{\text{Number of Gates in Design}}{\text{Number of Gates per Block}} \times \text{Number of Runs}$$

Once synthesis has completed, the gate-level netlist is partitioned and the netlist of each FPGA is sent to the Vendor Place and Route tools. If, at this time, the partition doesn't meet timing or area constraints, the designer has to go over the synthesis-partitioning-P&R loop again.

$$\text{Time}_{\textit{synthesis}} = (\text{Number of Iterations}) \times (\text{Time}_{\textit{one synthesis iteration}})$$

This is clearly a very time-consuming process, if you consider the fact that synthesis is just one part of the prototyping flow. We have not accounted for partitioning and place-and-route times.

A netlist partitioner works on the post-synthesized netlist and partitioning choices cannot impact the synthesis process. Thus, gate-level netlist partitioners don't always deliver the best solution when partitioning timing-critical logic across multiple FPGAs. This usually results in a lower performance prototype.

2.2 Manual RTL partitioning followed by synthesis:

Another approach to partitioning designs is to partition the design manually at the RTL level and then synthesize each partition separately to the target FPGA architecture. The synthesized netlist for each FPGA is then passed to the vendor place-and-route tools and the target prototype platform is implemented.

Partitioning at the RTL level allows user visibility into contextual information of the design. This makes it more intuitive to partition the design hierarchy at the RTL level. However, partitioning the RTL involves the tedious process of grouping and ungrouping of logic in a synthesis tool with no feedback on the partitioning implications of each choice, thus leaving open the possibility of multiple iterations of the partition-synthesis-P&R loop. Each FPGA is synthesized separately, precluding any opportunity for cross-partition optimizations. Also, the granularity of partitioning is driven by the hierarchy of the design because partitioning tends to occur at module boundaries. This flow may not be conducive to the single-source approach to verification of designs. In most cases, designers want the verified HDL to be the same as the original HDL source. For example, gaining visibility into the design under test may often require bringing a net at any level of hierarchy to the pins of the FPGA. Doing this may require changes to the original source. Changes to the original source are usually viewed as compromises to the quality of verification.

2.3 Crafting a Better Solution

Design verification currently accounts for well over 60% of the design cycle. This is only going to increase as SoC designs integrate more complex functionality and rely on hardware-based verification strategies. What the design and verification community needs is technology that shortens prototype development time while delivering optimal prototype performance.

In order to provide a productive path from ASIC HDL to FPGA-based prototypes, we need a technology that allows designers to work at the HDL level, guides them through the partitioning process by helping them make the right partitioning decisions. In addition to this, the new technology needs to provide extremely high quality-of-results for prototype performance. Another requirement from this technology would be extremely fast run times so that a designer could use what-if analyses for area, I/O and connectivity to drive the partitioning process.

It is our experience that allowing the design or verification engineer to interactively partition the design results in an optimal partition, both in terms of area utilization and timing. Automatic partitioners typically work on post-synthesis netlists and deliver low-utilization of FPGA resource. This means that the design is to be partitioned over a larger number of FPGAs. This usually results in a more expensive, slower prototype.

In summary, the best approach is a solution that enables users to partition efficiently, provides the best utilization of FPGA resources and has best-in-class synthesis technology to optimize the partitioned logic to the target FPGA architecture. Another requirement is to create a user model that integrates ease-of-use with the power of the underlying synthesis technology.

3. Partition Aware System Synthesis Technology

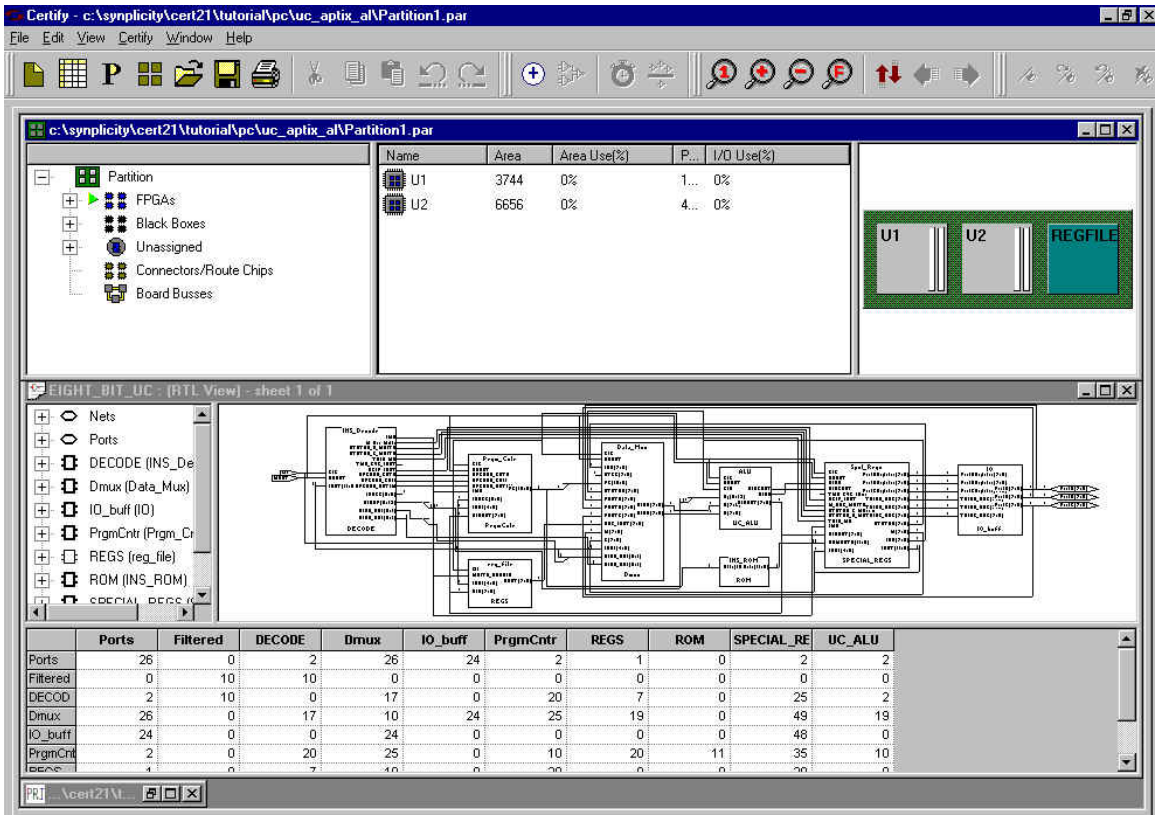
Partition Aware System Synthesis (PASS) is an extension of Synplicity's FPGA synthesis technology. PASS technology allows users to partition a design from within the synthesis process. The traditional approaches to multi-FPGA prototypes have 2 separate steps of synthesis and partitioning, often with non-optimal results. Each of these steps is usually done in isolation of the other step in the process and hence each has no way of passing control to the other. PASS technology combines the 2 processes into one, allowing for the best productivity and quality of results.

PASS® technology revolutionizes the user model for multi-FPGA partitioning. It is the ability to create prototypes at optimal performance that completes the value delivery of Certify, an implementation of the Partition Aware System Synthesis technology. Certify uses the following technologies to deliver the first in a new class of tools that merge physical and logical design into one seamless process:

3.1 BEST technology:

At the core of Synplicity's FPGA synthesis technology is its Behavior Extraction Synthesis Technology (BEST) algorithms. While traditional synthesis tools try to optimize gate-level representations of designs after compiling HDL, Synplicity's BEST algorithms represent designs in a higher level of abstraction than user HDL. This results in a smaller representation of the design and allows the synthesis engine to optimize the entire design on a global basis. This also obviates the need to break up a design into smaller hierarchical blocks as required by traditional synthesis tools. Having a smaller, more abstract version of the HDL means that the synthesis engine can synthesize large designs very quickly. This is a critical requirement for Partition Aware System Synthesis since fast synthesis technology is a key for real-time feedback to guide the partitioning process.

3.2 The User Model: Interactive Partitioning at HDL level:

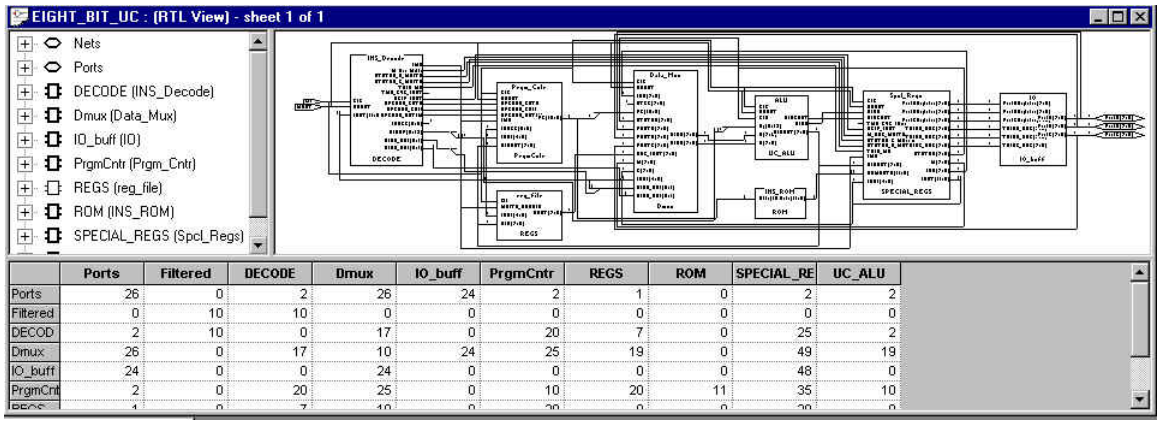


3.2.1 Intuitive UI

The Certify UI framework encapsulates 2 views that are central to the partitioning process. The logical view displays the HDL schematics while physical view displays the board onto which the design is being partitioned. The UI also delivers relevant, real-time feedback during the partitioning process.

3.2.1.1 RTL View

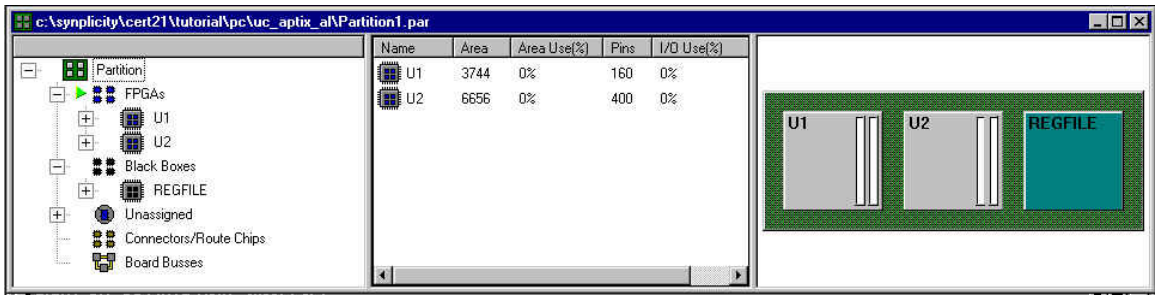
Synplicity's HDL Analyst option makes an ideal interface for analyzing HDL. The Synplify software uses HDL Analyst to allow designers to view and analyze their HDL code in schematic form. It also allows cross-probing between HDL and the schematic views. HDL Analyst allows users to graphically view and analyze critical paths in the design after it has been mapped to the target technology. It is also the ideal user interface for partitioning logic from HDL to the physical board and the corresponding FPGAs and system components. The RTL View is also annotated with the Area Estimates in target technology resources. Once the design is mapped to the multi-FPGA system, the designer can invoke the Technology View which shows the technology specific implementation of the design and cross-probe to the RTL to analyze timing paths. During system debug, the user can generate the RTL View for each FPGA and cross-probe back to the user HDL to investigate a potential problem. This feature, when used with source level partitioning delivers an integrated debug environment to isolate a potential bug and verify a fix for it.



	Ports	Filtered	DECODE	Dmux	IO_buff	PrgmCntr	REGS	ROM	SPECIAL_RE	UC_ALU
Ports	26	0	2	26	24	2	1	0	2	2
Filtered	0	10	10	0	0	0	0	0	0	0
DECOD	2	10	0	17	0	20	7	0	25	2
Dmux	26	0	17	10	24	25	19	0	49	19
IO_buff	24	0	0	24	0	0	0	0	48	0
PrgmCnt	2	0	20	25	0	10	20	11	35	10

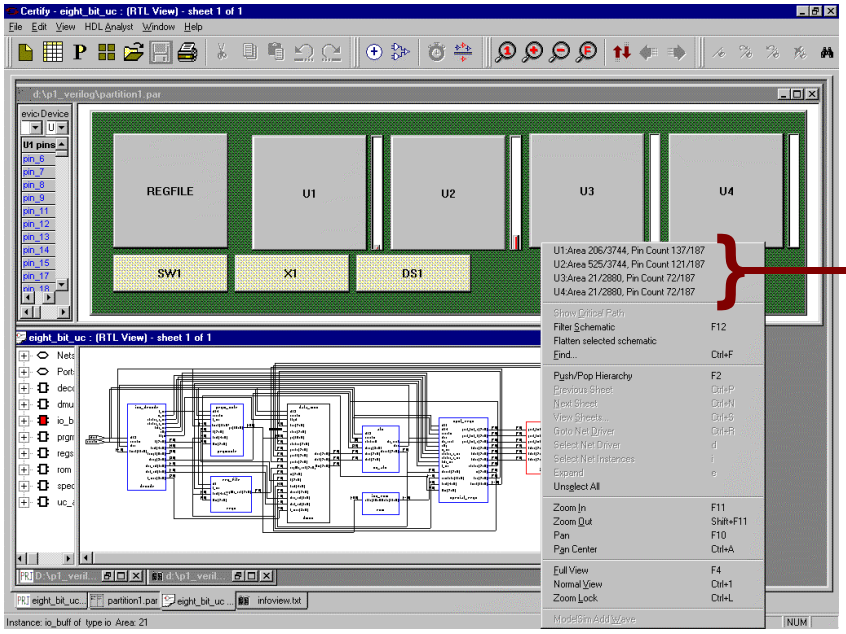
3.2.1.2 Partition View

The partition view represents the board on which the design is to be prototyped. It shows the components on the board, e.g. FPGAs, system-level black boxes, connectors and the connectivity between all these devices on the board. The partition view also displays I/O and area utilization graphs. These are updated in real time as logic is partitioned into a FPGA. Integrated with the partition view is a graphical browser-based view that provides partition status information. This view provides information on what logic has been partitioned and to which FPGAs. A user can undo logic assignments, get information on the FPGAs (Vendor, device, size, package, and I/O capacity) and make pin assignments from this view.



3.2.2 Guided Interactive Partitioning

The ability of PASS technology to allow designers to partition at the HDL level while providing them implementation technology specific information and analysis tools is key to Guided Interactive partitioning. The motivation behind this user model is to be able to converge on a legal partition without facing the traditional multiple synthesis-partitioning-P&R iterations of the traditional approaches. The designer is “guided” through the partitioning process by making informed area/IO/performance trades-off based on synthesis derived area and I/O estimates. The ability to do what-if analyses in real-time enables designers to evaluate multiple partitioning choices before making any logic assignment.



Select one or multiple RTL blocks and instantly see area and I/O impact for each available FPGA

The Certify tool's ability to report logical and physical connectivity allows it to guide the partitioning process over a variety of emulation platforms e.g. re-configurable platforms, pre-defined boards or Certify-created boards.

3.2.3 Partitioning flexibility

Partition Aware System Synthesis allows a powerful way of managing the design HDL. Certify software allows designers to replicate logic for I/O conservation and timing optimization, probe the design from the HDL level without having to modify the design HDL. The Certify tool allows designers to keep single source code model in place and enable them to test the actual design in hardware, not a version of the HDL created to overcome partitioning and performance problems.

One of the key benefits of the PASS enabled partitioning process is that the designer partitions the design on a post-compiled representation of the HDL. This allows the designer to take advantage of the BEST technology. This in turn allows designers to partition their design at any granularity. For example, if your HDL has an addition operator, the compilation process will extract an adder and display it in the RTL View; the user can then partition this adder to an FPGA. This overcomes the traditional shortcoming of partitioning at the RTL level where a designer is constrained to partition along hierarchical boundaries.

PASS technology uses constant propagation techniques to optimize I/O utilization in modules with feed-through connections. This is an example of how PASS is applied specifically to a partitioning problem. For example, if a control register in a module is set to a specific value and its inputs are driven from another module, I/O connectivity is optimized by generating the constants within the module itself.

Certify also gives the designer control in partitioning the design by offering such partitioning aids as logic replication, decomposition of large primitives, bit-slicing of primitives with bussed inputs, GUI-driven what-if analysis and connectivity reports.

3.2.4 System-level Black Box Support

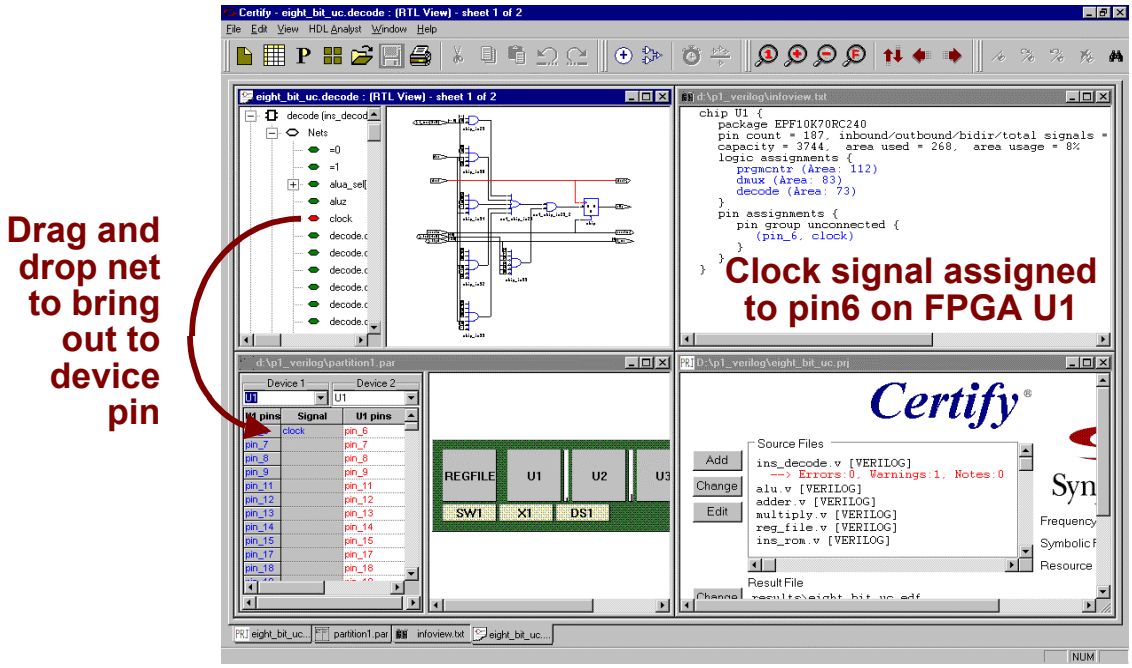
Bonded-out cores are often used for implementing IP cores and analog functionality on the prototyping board. Designs that have multi-ported memory structures are also often implemented using standard off-the-shelf components. Using Certify the designer can partition relevant logic into these system–level black boxes. The synthesis engine understands that it does not need to try and implement this functionality in the FPGA.

3.2.5 Source-level Partitioning

Multi-million gate designs map into many FPGAs. Synthesizing these FPGAs may take some CPU resources and time. Certify partitions the source code for each FPGA into a separate project file that can be used for synthesis. If, for example, the designer needs to verify a bug that only affects logic contained in one particular FPGA, he or she can synthesize, place-and-route, and reprogram only the one FPGA in question. This flow can be used to verify if a potential bug fix actually fixes the problem and does not create any new ones.

1.2.6 Automatic Probe Point Creation

Because the partitioning operation within Certify is to create a testable functional prototype, Certify assists the designer in incorporating their test strategy into the prototype. Specifically, Certify features the ability to select any signal in the design (regardless of where in the design hierarchy it exists), and bring that signal out to any FPGA pin for observation.



3.3 Timing Driven Optimization:

In the final mapping stage, when the tool maps the partitioned design to the target FPGA architecture, Partition Aware System Synthesis technology uses Synplicity's Direct Synthesis Technology . In this step the partitioned design is mapped directly to the target device. Synplicity has separate mappers for each FPGA vendor, these mappers perform technology mapping by instantiating architecture-specific primitives and make the requisite packing trades-off to deliver at the best QoR for the design.

In addition, Synplicity's Partition Aware System Synthesis mappers are timing-driven. While some FPGA synthesis tools try to optimize the levels of logic as a means to meeting timing, Synplicity's mappers optimize timing paths intelligently, based on timing constraints. Often, optimizing a circuit based on logic level reduction alone will result in sub-par timing performance.

Synplicity's FPGA synthesis mappers optimize designs across hierarchy boundaries by automatically budgeting timing constraints. This is an extremely powerful feature of our synthesis technology. Propagating slack between hierarchy blocks results in the most optimal implementation of a path for a given constraint.

Partition Aware System Synthesis mappers actually extend this functionality by optimizing system performance. PASS® mappers analyze timing at the SoC design level. As they map each partition to FPGA they propagate slack between partitions and are able to optimize inter-FPGA timing paths. Certify mappers understand the fact that a timing path may cross partition boundaries. They automatically infer I/O cells in the FPGA netlist. They will also infer I/O cell delays for each vendor technology and use them in the timing calculations.

The ASIC designer can use Certify without having to do explicit timing budgeting and iterating between synthesis runs to converge on the best constraints for optimal implementation. This provides a major benefit to the designer, both in terms of productivity and performance gains. This feature, in concert with Certify's high capacity allows ASIC designs can be imported into Certify without re-defining hierarchy or setting new timing constraints for time budgeting reasons.

Timing constraints can be set by using the Synthesis Constraints Optimization Environment (SCOPE), a graphical spreadsheet-based tool. SCOPE is a powerful tool that shields the user from syntax details of constraint files. It is also integrated with HDL Analyst. For example, to set a constraint on a register in FPGA1, create the RTL view for FPGA1, and then drag-and-drop the register to a cell in the "Register" tab of SCOPE. Also, SCOPE can used to set all constraints in a design, i.e., for synthesis and Place and Route.

3.3.1 Black box timing model

Synplicity's synthesis technology allows the integration of IP in an intelligent way. A designer can specify timing constraints at the inputs and outputs of the IP block. While

this IP may be instantiated in the HDL as a black box, Certify will use the black box timing constraints to optimize the logic around it and deliver a better logic implementation of the design as a whole. This is especially useful if the instantiated IP block is on a timing-critical path.

3.3.2 Module generators

Synplicity's synthesis technology extracts high level arithmetic constructs during HDL compilation. It also has the ability to generate modules for arithmetic operators and memories (RAM and ROM) in a design and map them to the target technology. FPGA vendors often provide a library of modules for their own technology, however Synplicity's module generators integrate the associated control logic and are optimized both for area and timing. Taking advantage of features like RAM inference allows full portability of the designer's HDL code to many vendor technologies.

3.4 Simulation support

Another powerful feature of Certify is its integration of Model Technology's ModelSim simulator; it extends the cross-probing between HDL and technology views to simulation. This integration between the HDL Analyst and ModelSim allows designers to debug their partitioned designs using simulation technology. For example, once design or verification engineers have run a test, they can select nets in HDL Analyst and have ModelSim display the simulation values for these nets. This cross-probing works from both the RTL and Technology view and provides a consistent debugging environment integrating schematics-to-source-to-simulator views through partitioning, synthesis, and simulation.

4. Certify2.1 Features

Certify2.1 release offers further evolution of the partitioning and synthesis technologies. Some of the specific enhancements are:

4.1 APEX support

Certify2.1 supports the EPF20K series of devices from Altera. Synplicity's Direct Synthesis Technology continues to deliver QoR gains and extremely fast run time advantages. Run time advantages directly translate to productivity gains on these large devices; Synplicity uses an innovative "atom-mapping" technique to take advantage of the APEX device architecture to deliver compelling performance.

4.2 Automatic Pin assignment

Certify1.1 supports manual pin assignment via a Pin Grid Array. A designer can assign nets to pins by simple drag-and-drop mechanism. While this is an effective way to create probe points for a design, it is a time consuming process when it comes to assigning all nets that traverse multiple FPGAs. Automatic Pin assignment support is especially critical for pre-defined platforms where all the inter FPGA connections would have to be assigned manually. Also, the designer would have to keep track of all the logic assignments made to each FPGA such that he/she may assign nets to pins.
Certify2.1 makes this process much simpler by supporting pin assignments in the following modes:
Automatic: Certify will assign all cross-partition nets to pins on the package
Bundle: Certify will assign a set of nets to a set of pins
Explicit: Designer manually assigns pins

Certify 2.1 supports any combination of explicit, automatic or vendor P&R based pin assignments. Automatic pin assignment will respect explicit pin assignments should they exist in the partition. The partition file will log the source of each pin assignment.

4.3 Certify Pin Multiplier

As SoC designs to integrate system-level functionality, they have to support the wide system bus interfaces. Wide datapaths help optimize the throughput vs. latency trade-off in these designs. This trend in high-end networking and compute applications means that I/O management often dominates the partitioning process. The inability to partition successfully because of connectivity issues leads to low area utilization for FPGAs. This, in turn, leads to larger, slower prototype implementations.

Synplicity's Certify Pin Multiplier technology helps resolve the I/O utilization issue by multiplexing multiple logical nets over a physical pin. The CPM technology allows manual and automatic multiplexing of logical nets. Certify will analyze the partitioned design and automatically suggest CPM candidates. Once a group of logical nets have been selected as CPM candidates. Once a group of logical nets have been selected as CPM candidates, they are assigned to a physical pin. Certify will instantiate the multiplexing and de-multiplexing logic, and then use timing information during the

mapping phase to assign signals to a time-slot on the shared physical pin. Thus 2 or 4 signals can share one physical pin and transmit data at separate times during a cycle. This is another example of PASS enabling the designer to select candidates for CPM and then have the tool derive timing relationships between signals and implement the logic accordingly. A GUI component will allow the user to browse and select nets for CPM.

4.4 Additional Partitioning Aids

Incorporated into Certify 2.1 are additional partitioning aids, used to help partition elements of the design that require some manipulation prior to being partitioned, or that enhance the overall performance of the ASIC prototype. These partitioning aids include:

1. Large MUX breakup
2. Bit-slicing of large primitives
3. Partitioning at process-level hierarchy
4. RTL view per FPGA partition
5. Logic replication
6. Connectivity matrix within HDL Analyst
7. Zippering of inputs and outputs

