

# Syndicated<sup>®</sup>

News, tips, and notes for improving the Quality of Results in FPGA and ASIC design

## In this issue...

MultiPoint™ Synthesis  
Technology – the Key to  
Designing Large, IP-Intensive  
SoCs and PSoCs . . . . . 1

Life in the Fast Lane –  
Benefits of Lightning-Fast  
Timing Analysis . . . . . 1

Synplicity Spotlight:  
Sunil Ashtaputre . . . . . 2

Tips and Hints . . . . . 5

Synplify 7.1 Software  
Update . . . . . 6

Synplicity® Success Stories:  
API NetWorks . . . . . 8

Synplicity Training Courses  
Now Available . . . . . 8

Upcoming Events:  
Noteworthy events,  
seminars, shows, and more . . 8



Synplicity®

Simply Better Results

## MultiPoint™ Synthesis Technology – the Key to Designing Large, IP-Intensive SoCs and PSoCs

by John Gallagher, Director of Product Marketing

**Circuit** complexity today continues to soar despite the fact that IC design tool capabilities remain fully one to two generations behind. Arguably, a prime contributor to the design productivity gap today is conventional synthesis technology. Some of the problems found in using current synthesis methodologies include:

- The maximum synthesizable block size is about 1/100th of the overall design size and is becoming a smaller percentage of the chip with each silicon generation.
- The work-around solutions to manage small block sizes require inordinate amounts of manual scripting, drastically reducing design quality and productivity by the engineer.
- The more partitioned blocks there are in a design, the worse delay QoR becomes because

the synthesis tool cannot “see” as many optimizations.

- Breaking designs according to arbitrary gate count limits is hard for a logic designer, who naturally breaks a design into functionality and timing.
- While top-down synthesis delivers the best possible delay QoR, it is limited in memory and runtime to handling smaller pieces of the entire design.
- IP, datapath, analog/mixed signal, and, in the future, embedded FPGA are all examples of elements of a design that should have special handling in synthesis, regardless of block size.
- 20M gate complexities (or larger) cannot be synthesized well with today’s “divide and conquer,” bottom-up approach.

“MultiPoint” continued on page 3

## Life in the Fast Lane – Benefits of Lightning-Fast Timing Analysis

by Khaled Zakharia, Product Marketing Manager

graphics are included in this article.

### Where the Timing Engine Fits In

Figure 1 (page 4) shows the typical synthesis flow. It is important to note that the timing engine is present in two distinct places of the diagram:

1. In conjunction with the mapper as it attempts to meet the specified timing constraints.
2. In the final step as it creates a timing report and identifies the critical paths.

“Life in the Fast Lane” continued on page 4

# Synplicity Spotlight

## Sunil Ashtaputre, Director of ASIC Product Development

**Back** to the future – for Sunil Ashtaputre, it's a fitting storyline. A product development engineer by trade, Sunil was trying something a bit different when he joined Synplicity, Inc. back in 1998. "I joined Synplicity because I wanted to move into marketing. Talking to customers and vendors about their needs – I really enjoy that," he said.

But, while Sunil was happily immersed in the product marketing responsibilities of the highly anticipated, but not-yet-developed, Synplify ASIC® solution, his best-laid plans were about to change. A sudden and unexpected vacancy at the director-level of product development for the Synplify ASIC product left the newly formed development team without a leader.

### A Take-Charge Engineering Leader

"I just couldn't stand the idea of a six-to-nine month search that could delay kicking off the Synplify ASIC product. So, I went to Bob Erickson (Synplicity's Senior VP of Engineering) and said, 'I'm going to move and start working on the product development side.' That's pretty much how and why I moved back to the engineering side," he said.

Crediting his time in marketing, Sunil embraced his new role of Director of ASIC Product Development with, what he calls, "a heightened awareness" of validating internal engineering priorities with feedback from customers, vendors, and application engineers.

### Priorities, Priorities, Priorities

From a development perspective, Sunil quickly admits the Synplify ASIC product is "new and unique." "We used some powerful, proven Synplicity technology – the front end, the user interface, and a lot of infrastructure pieces – but there was much that was completely developed from scratch and optimized very specifically for the Synplify ASIC tool. It's a bit of a hybrid," he noted.

Like most engineering managers, Sunil's primary responsibility in developing this product was that of constantly setting and re-setting priorities. Unlike his colleagues involved with the Synplify Pro® product, Sunil lives in a very reactive environment. "We do have a plan for the Synplify ASIC product, but we will change and update it very frequently based on new input we are getting from early customers," he explained.

Reacting to ever-changing issues and juggling a multitude of priorities, Sunil relies on a single development strategy to keep his team centered and the product successful. "We must remain focused on the core optimization and on those things that ensure the engine is continuously improving, so the Quality of Results will keep improving," he said. "The product has to be steered like that."

### Never a Dull Moment

Now that the Synplify ASIC tool is in its 2.3 release and has recently added major productivity gains through Synplicity's MultiPoint™ incremental synthesis technology, is the hard part over for Sunil? "It's definitely not over," he quickly answered. "The growing size and complexity of designs and the methodology shift in terms of the hand-offs with ASIC vendors – these are two major obstacles where mainstream ASIC designers need better solutions. And that's where we come in," he explained. "It will be a long time before we are completely done with everything we want to do for the Synplify ASIC solution. Perhaps never. But everywhere you look in this organization you will see Ken McElvain and Bob Erickson encouraging innovation and innovative thinking among our developers. Anything and everything is possible here," Sunil concluded.

Sunil Ashtaputre holds a BTech in electrical engineering from IIT-Bombay and an MS in computer science from North Carolina State. Prior to joining Synplicity, he was the Director of Design Methodology & Library Development Tools for Silicon Architects Division at Synopsys. He also served as a Manager at VLSI Technology, and started his career in place and route software development at Seattle Silicon. ●

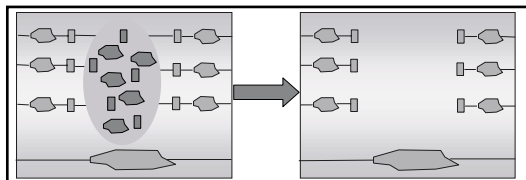


**Sunil Ashtaputre,**  
Director of ASIC Product Development,  
Synplicity, Inc.

*"It will be a long time before we are completely done with everything we want to do for the Synplify ASIC solution. Perhaps never."*

To address these shortcomings of traditional synthesis approaches as applied to high-complexity ICs, Synplicity has developed a new MultiPoint synthesis technology for multi-million gate system-on-a-chip (SoC) and programmable system-on-a-chip (PSoC) devices. By providing a high-productivity design methodology that is scalable to tens of millions of gates, the MultiPoint technology addresses design challenges that are emerging as application specific integrated circuits (ASICs) and programmable logic devices (PLDs) become more complex.

With MultiPoint synthesis, Synplicity combines new technology advances with the best of existing design methodologies to provide for the best timing and area results with low runtime. The technology uniquely and



**Figure 1: Interface Logic Models (ILM) are a key aspect of MultiPoint technology. ILMs are partial netlists that are typically 70-80% smaller than the design netlist.**

automatically creates Interface Logic Models (ILMs) based upon user-defined "compile points" (Table 1) or instructions to the synthesis tool for modeling and synthesizing a particular portion of the design. By thus reducing the design data for these modules, MultiPoint technology reduces memory requirements needed to synthesize large designs (Figure 1). This means a designer can hierarchically synthesize their design based on functionality and timing, not an arbitrary gate count limit. MultiPoint technology optimizes across design partitions using the same information needed for a top-down

synthesis flow, enabling the highest design performance. Additionally, ILMs can be written out for any netlist or synthesized design, even from third-party tools. The use of ILMs and compile points in an overall synthesis flow is illustrated in Figure 2.

MultiPoint technology incorporates a unique, difference-based incremental synthesis approach (Table 2), eliminating the need for re-synthesis, common with time-stamp-based incremental flows, by only re-synthesizing design entities that will have a different gate-level netlist due to code or constraint changes. This minimizes the impact of changes by using constraints and RTL to determine what must change. Design stability is ensured by way of locked compile points that isolate changes to the module(s) of interest.

With its ability to automatically model the IP and use the timing information for synthesis, the MultiPoint flow also eases intellectual property (IP) integration into a design. For example, MultiPoint technology optimizes logic both inside an instantiated IP block and in its adjacent modules without impacting port assignments for the IP core. For designs with replicated logic or IP blocks, MultiPoint synthesis allows

Features	Benefits
Use of ILMs	<ul style="list-style-type: none"> <li>Increased capacity</li> <li>Memory efficient</li> <li>Faster synthesis with same QoR as top-down flow</li> </ul>
Difference-based incremental synthesis	<ul style="list-style-type: none"> <li>Only modules that truly change are resynthesized</li> </ul>
Time-budgeting	<ul style="list-style-type: none"> <li>Eliminates manual constraints creation</li> <li>Much faster time-budgeting step</li> </ul>
Soft/Hard compile points	<ul style="list-style-type: none"> <li>Allows better QoR through boundary optimization</li> </ul>

**Table 2: Incremental synthesis that is based upon comparison of RTL, constraints, and properties minimizes design changes and runtime.**

you to control how each unique instance is treated in terms of boundary optimizations without the runtime penalty of re-synthesizing each instance.

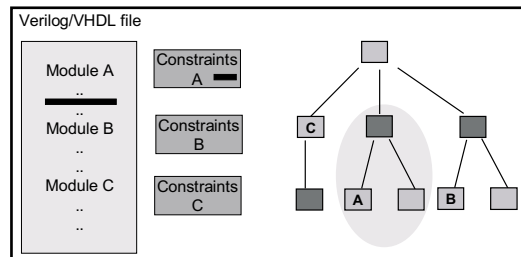
For ASIC design, MultiPoint technology delivers a methodology for implementing highly complex designs. With ASIC designs exceeding two million gate densities that includes some form of IP or replicated logic becoming more common, existing top-down or bottom-up methodologies are inadequate. The traditional bottom-up design flow, or synthesizing lower modules of a design before synthesizing the upper modules, can require many cumbersome scripts and time budgeting, and can inhibit boundary optimization and lead to sub-optimal design performance. Similarly, synthesizing the design hierarchy all at once in a top-down flow is ideal for delivering the best design performance, but is limited by the memory capacity of the compute, as well as long runtimes for synthesizing the entire design. With its unique incremental approach and use of compile points and ILMs, MultiPoint technology produces the Quality of Results (CoR) and ease-of-use of top-down flows, as well as the stability, fast runtimes, and capacity of a bottom-up flow.

Likewise, programmable device complexity abounds. Emerging PSoC devices include capabilities such as complex I/Os and embedded processors, and offer up to ten million PLD-gate capacities. With this increase in complexity come design challenges that are now equivalent to those for ASIC devices. Programmable logic vendors are responding to customer needs with new incremental place and route capabilities. The MultiPoint technology's new and complementary synthesis capability creates an effective and complete incremental flow for high-complexity FPGA designs. The MultiPoint flow can significantly improve runtimes for both synthesis and place and route.

With the new MultiPoint technology, Synplicity continues its long tradition of providing innovative and easy-to-use synthesis solutions. It represents the optimal compromise between top-down and bottom-up design approaches. The MultiPoint technology delivers an automated, high-productivity incremental synthesis solution for large designs in terms of runtime, quality of timing and area results, and integration of intellectual property blocks. ●

- Modules synthesized separately
- Firm & hard IP blocks
- Reused designs
- Datapath and custom blocks
- Embedded FPGA blocks
- Modules that are instantiated several times
- Modules having independent clock domains
- Timing critical modules whose layout is frozen
- Modules that are frozen after verification

**Table 1: Examples of compile points**



**Figure 2: Through the use of compile points and ILMs, the MultiPoint technology creates a very efficient synthesis flow. Integrity of sub-blocks is retained through the ILMs and runtimes, and computational requirements are dramatically reduced.**

### Timing Engine in Conjunction with the Mapper

While most ASIC designers regard a timing analyzer as being a back-end tool that is run a few times only after a design is fully synthesized, many fail to realize that it is also embedded in the heart of the synthesis flow. It is in here that a lightning fast timing engine, coupled with an efficient mapper, is critical in achieving very fast synthesis run times. To understand this better, consider the diagram in Figure 2.

First, this figure shows that the synthesis flow is comprised of two phases: compiling and mapping/timing.

Historically, the mapping/timing phase has been the bottleneck. To improve the efficiency, developers have come up with a timing engine that operates in incremental mode. Such a mode requires updating only the timing information that was affected by optimizations in the mapper. Since this is the most common case, the result is a much faster timing engine. Improving the performance of the timing engine yields the following benefits:

1. Permits the mapper to try out more permutations of possible design structures and optimizations within the same out of time, thereby yielding a more thorough analysis of the design.
2. Provides the ASIC designers with faster feedback. This allows them to experiment with different design structures and fix more timing violations within the same amount of time.

Both these benefits imply better QoR and a shorter time to market. In addition, these benefits are exhibited not only in the initial synthesis run, but also in further iterations and synthesis optimizations.

### Timing Engine as a Report Generator

The speed of the timing engine is still a factor here. A slow timing engine may, for example, hinder the user from asking for a large number of paths. Instead of asking for the 30 worst violations, a user may be tempted to request the 10 worst ones, fix those, and then see what the next 10 worst ones are. That process would require two extra cycles that otherwise would have been unnecessary.

Nonetheless, the speed of the timing engine is now overshadowed by its ability to provide the designer with useful information quickly. The critical factor now becomes how well the timing analyzer is integrated into the synthesis flow and how quickly the user interface can translate the results of the timing report into meaningful information for the designers. Rather than burden them with writing short scripts that search for keywords inside the timing reports, having a sophisticated, user-friendly GUI that crossprobes the critical paths back to the original HDL code and/or gate level netlist would be key in helping the designer to quickly and efficiently: identify the critical paths, visualize the logic blocks they traverse, and, grasp any redesigning or repartitioning tasks that need to be undertaken to fix the violations.

### Synplify ASIC® Software

The Synplify ASIC solution has an embedded timing engine identical to the one used in Synplify's FPGA synthesis products. It is designed and optimized for incremental mode. This is in line with one of the most fundamental rules in computer architecture: make the common case fast. Thus, with this design, Synplify ASIC software has eliminated a huge bottleneck in the synthesis flow, and created one of the fastest and most accurate timing engines around. This translates directly into a much faster synthesis engine and explains why the Synplify ASIC solution can boast extremely fast runtimes.

In addition, the GUI features of the HDL Analyst® environment are tightly integrated into the Synplify ASIC synthesis tool. This equips the designer with two extremely useful views: the RTL view and the technology view. These views allow the designer to crossprobe any timing violations or critical paths back to the RTL code or gate-level netlist respectively, thereby serving as quick, intuitive tools that replace an archaic and inefficient form of debugging.

In attempting to quantify the increase in productivity, it can be shown that by considering the application of the synthesis engine to various aspects of the development cycle – such as, design and exploration; spec changes; and, the synthesis-timing optimization loop – it is conceivable that the Synplify ASIC solution can shave off a month on a design that typically required an eight month cycle from concept to tape-out. This is equivalent to a 12.5% faster time to market. Given that most of the profits are made right after the product hits the market and before the rest of the competitors have jumped in, the net gains from choosing the right synthesis engine can be huge.

### Conclusion

The Synplify ASIC solution is designed with a lightning fast timing engine that is tightly integrated into the synthesis flow enabling the ASIC designer to not only experiment and iterate quickly on his design, but also to crossprobe critical paths back to the source code and glean a more thorough understanding of the violations faster. By simply selecting the right synthesis tool, the ASIC designer can shave off almost a month in development time, thereby obtaining a critical edge over competitors through a faster time to market. ●

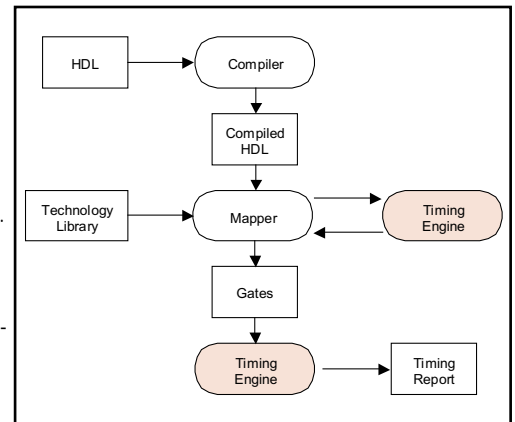


Figure 1: Timing Engine within the Synthesis Flow

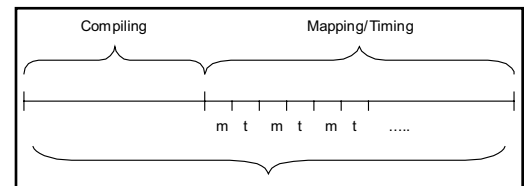


Figure 2: Breakdown of the synthesis flow into its major constituents

# Tips and Hints

## Amplify® Physical Optimizer™ Software

**Q:** What is the recommended flow for the Amplify Physical Optimizer now that it features both an automated and an interactive flow?

**A:** The simple answer is to start with the automated flow first and see if the performance goals are met. If the user is not able to achieve the performance goals using the Automated TOPS™ flow, then it will be necessary to apply physical constraints and use the Interactive TOPS flow.

The automated flow, or Automated TOPS flow, is a push-button physical synthesis flow. This flow, as the name implies, does not require the user to interact with or to constrain the physical synthesis implementation (other than to provide the appropriate timing constraints). While this flow is more simple to use than the Interactive TOPS flow, the resultant improvement in design performance is not as large as the improvements that may be achieved using the Interactive TOPS flow.

The Interactive TOPS flow requires the user to constrain critical portions of the design to physical regions. This flow gives the user control over the physical synthesis implementation and the quality of the results will depend greatly upon the quality of the physical constraints applied. This flow can be an iterative process where the user begins by applying a few constraints and gradually adds more constraints in successive synthesis / P&R passes, or the user can apply numerous constraints right at the start. This flow provides the user the most control and flexibility.

Which flow is best for you? Ultimately it will depend upon your design, design goals, and experience with the tools. If in doubt, start out with the very easy-to-use Automated TOPS flow and progress to the Interactive flow if necessary.

## Synplify® and Synplify Pro® Software

**Q:** How can I create multiple instances of a module in Verilog similar to VHDL generate capability?

**A:** The Synplify 7.1 product release supports the key features of Verilog IEEE Std. 1364-2001, also known as Verilog 2001. This support includes generate statements that can be used to create multiple instances of a module.

Verilog IEEE Std. 1364-1995, also known as Verilog 1995, support of instantiating and replicating a module was conservative and tedious. In order to address this issue, Verilog 2001 has added a new construct called generate. This new construct now has the capability of creating multiple instances of the object within a module and selectively controlling which instances or statements needed to be generated.

Verilog statements that can be placed in this generate block are:

- Module and primitive instances
- Initial and always procedural blocks
- Continuous assignments
- Net/Variable declaration
- Task/function definitions

## Types of Generate Statements

There are two types of **generate** statements: generate loop and conditional generates. Both these types of generate statements are supported by Synplify Pro 7.1 software.

### Generate loop

The primary use of generate loops is to create multiple instances within a for loop. The for loop in a generate statement is similar to the for loop in Verilog 1995 except for the following conditions:

- The loop index variable must be a genvar
- The assignments in the loop control must assign to the same genvar
- The contents of the loop must be within a named begin -end block

The genvar variable is a special integer variable used in generate loops. It can be assigned only 0 or positive numbers. A genvar can only be assigned a value as part of a generate loop statement. A genvar variable can be defined outside of the generate block or within a generate block. If declared outside, the variable can be used by any number of generate blocks. Take an example of creating a 32-bit adder using four 8-bit adders. It can be written as:

```
generate
genvar i ;
for ( i= 0; i <= 3 , i = i+1 )
begin : u
adder8 add ( sum [ 8*i+7 : 8 *i ] , co[i+1], a[8*i+7 :i*8], b[8*i+7 :i *8] , cin [i] ) ;

end
endgenerate
```

*"Tips and Hints" continued on page 6*

*“Tips and Hints” continued from page 5*

This will result in 4 instances u[0], u[1], u[2], u[3] of adder8. Please note that one of the conditions of the generate loop is to have the statements within a named begin -end block. This is to ensure a unique name can be created for the instances with the block name as the prefix and the loop variable count in square brackets

### Conditional Generates

Conditional generates can be created in two ways: using the if -else statements or the case statements . Both these statements can be used within the generate block and follows the Verilog 1995 definitions. Here is an example of using a generate statement to control which kind of adder is used depending on the parameter called adder\_width :

```
// if-else generate
generate
  if (adder_width < 8)
    ripple_carry #(adder_width) u1 (a, b, sum);
  else
    carry_look_ahead #(adder_width) u1 (a, b, sum);
endgenerate
```

The following example uses the case statement in order to specify which adder is used according to the parameter called adder\_width:

```
// case generate
parameter adder_width =1;
generate
  case (adder_width)
    1: adder1 x1 (c0, sum, a, b, ci);
    2: adder2 x1 (c0, sum, a, b, ci);
    default: adder # adder_width (c0, sum, a, b, ci);
  endcase
endgenerate
```

---

## Synplify® 7.1 Software Update

*by Steve Pereira, Product Marketing Manager*

The Synplify 7.1 software released last month introduced many new features and Quality of Results improvements for our users. In this article, you shall discover three of the major new capabilities of this release. For a complete list of detailed features, see the release notes in the support section of our web site.

### Altera Stratix Devices Target DSP Designers

Stratix devices are the latest SoPC (system on a programmable chip) devices from Altera. Stratix integrates DSP blocks, clock management circuitry, advanced memory, and high-speed interfaces into one device. Synplify worked closely with Altera to support this advanced architecture and, specifically, its DSP blocks. The Stratix DSP blocks are high-performance embedded DSP units optimized for applications such as:

- Rake receivers
- Voice over Internet protocol (VoIP) gateways
- Image processing applications
- Orthogonal frequency division multiplexing (OFDM) transceivers
- Multimedia entertainment systems

DSP blocks help eliminate performance bottlenecks in these types of applications, providing more reliable performance and also resulting in resource savings. Stratix devices use DSP blocks to achieve the high data throughput (up to 250MHz) necessary for computationally demanding applications. This allows for up to 2.0 giga multiply accumulate operations per second (GMACs) per DSP block.

### Software Support for Stratix Devices

Dedicated MAC DSP blocks may be configured as multipliers, multiplier adders, or multiplier accumulators. The Synplify Pro® synthesis software is ideal for DSP-type applications, as it is able to extract and perform true timing-driven synthesis on mathematical functions. The Quartus II software then maps these mega functions directly into the MAC DSP blocks. With our integrated support of Quartus (e.g., cross probing, etc.), DSP clock mapping allows design engineers to code at a higher level, resulting in great performance.

The MAC DSP blocks specify VQM netlist files by means of Altera's mega functions – Altmult\_add, altmult\_accum, and lpm\_mult. Structural VHDL and Verilog simulation netlists also return these functions. The Quartus II software then maps these mega functions directly into the MAC DSP blocks. Each MAC block

*“Synplify Software” continued on page 7*

consists of four 18-bit multipliers feeding into two-level adders. Below is a table describing how the Synplify solution handles the different configurations of the DSP blocks.

Configuration	The software Implements...
<b>Multiply Only Mode</b>	
A standalone signed or unsigned multiplier	The LPM_MULT Megafunction
A standalone signed or unsigned multiplier with registered inputs	Pipelined LPM_MULT Megafunction
A standalone signed or unsigned multiplier with registered inputs and outputs	Pipelined LPM_MULT Megafunction
<b>Multiply Add Mode</b> 2, 3, or 4 multipliers (all signed or all unsigned) feeding an adder	The ALTMULT_ADD Megafunction. The multiplier inputs and outputs can be registered. The output of the final adder can also be registered. The registers can have an asynchronous reset and/or enable. A maximum of four clocks (with or without enables) and four asynchronous resets can be used in an altmult_add to control the registers. The multiplier input sizes must be between 4 and 18.
<b>Multiply Accumulate Mode</b> 1 multiplier driving an accumulator $x = x + a*b$ or $x = x - a*b$	The ALTMULT_ACCUM Megafunction. The multiplier can be signed or unsigned. In this release, the multiplier input and outputs cannot be registered. The size of inputs a and b must be between 4 and 18, and the size of accumulator x must be between 8 and 52. The accumulator register can have an asynchronous reset and/or enable.

### Verilog 2001

A second important feature added to Synplicity's FPGA products is support for the new Verilog 2001 standard. Verilog 2001 contains features making the code easier to implement, easier to read, and also to enhance the functionality of the language. In addition to the Verilog 95 standard, you can now enable or disable Verilog 2001 features via the Synplify tool's user interface. This may be performed either on an entire project or on a file-by-file basis.


We focused first on the high value features such as Signed Arithmetic, Generate, ANSI Style port listing, and combinatorial sensitivity. Below is a table with the supported Verilog 2001 features and its benefit. Also included is a comparison to the older Verilog 95 format where applicable.

Feature	Verilog 95	Verilog 2001
Generate		Generate for making multiple instances of Modules and Procedures.
Comma-separated sensitivity list Verilog 95:	always@(sel or a)	always@(sel, a)
Wildcards in sensitivity list	always@(sel or a)	always@ *
Automatic width extension beyond 32 bits		
Combinatorial logic sensitivity		
Combined data and port types	output o1; reg o1;	output reg o1;
NSI-style port lists	module v95add(a,b,o) input [7:0] a; input [7:0] b; output wire [8:0] o; assign o = a + b; endmodule	module v2kadd( input [7:0] a, input [7:0] b, output wire [8:0] o ); assign o = a + b; endmodule
Enhanced conditional compilation		

### Advanced Optimizations for Xilinx Virtex2 Architecture

Quality of Results is always a top priority for the Synplicity development team and the 7.1 release is no exception. Several productivity and performance improving features for Xilinx include: DLL and DCM recognition, pipelined multipliers, and wider ROM support. With this release, you can pipeline Virtex-II multipliers (MULT18X18). When pipelining is enabled, the synthesis software creates a new pipelined multiplier, MULT18X18S, with the additional pipeline stages added. The pipelining optimization is reported in the log file along with the number of stages created. This dramatically speeds up functions like FIR filters.

Prior to the Synplify 7.1 product release, up to 16x1 ROMs were used to implement ROMs extracted by the Synplify tool's compiler. Now that Mux6 is available for VirtexE and MuxF7 and MuxF8 for Virtex-II, the synthesis software decomposes bigger ROMs using 64x1, 128x1, and 256x1 ROMs. These changes result in better utilization and higher performance.

Another feature that makes using DLLs easier is full recognition of DLL/DCM primitives in Virtex technologies. You no longer are required to set constraints on individual DLL or DCM outputs. You can specify clock constraints on DLL/DCM inputs and the DLL/DCM outputs – frequency and phase shift are calculated automatically. 

# Upcoming Events

## DAC 2002

June 10 - 14, 2002

Ernest N. Morial Convention Center

New Orleans, LA

[www.dac.com](http://www.dac.com)

Booth No. 1843

Demo Suite No. 5105

## Multi-million Gate Chip Synthesized and Verified in Hours

June 27, 2002

Four Points Sheraton

Sunnyvale, CA

[www.synplicity.com/chipbeforelunch.html](http://www.synplicity.com/chipbeforelunch.html)

## X-Fest 2002 – Programmable World Workshops

For locations and dates near you, go to

[www.insight-electronics.com/x-fest2002](http://www.insight-electronics.com/x-fest2002)

This list of Upcoming Events is subject to change.

Please go to the Synplicity web site at

[www.synplicity.com](http://www.synplicity.com) to view the most current schedule  
of events the company will be attending.

*Syndicated* is a quarterly publication of Synplicity, Inc. It's free to all subscribers and is also conveniently located online at [www.synplicity.com/syndicated](http://www.synplicity.com/syndicated). To be added to our mailing list, or to submit comments please e-mail:

[syndicated@synplicity.com](mailto:syndicated@synplicity.com)  
or call (US) +1 408 215-6000



**Synplicity**

Simply Better Results

Synplicity, Inc.  
935 Stewart Drive  
Sunnyvale, CA 94085 USA  
Phone: (US) +1 408 215-6000  
Fax: (US) +1 408 990-0290  
[www.synplicity.com](http://www.synplicity.com)

## Synplicity® Success Story: API NetWorks

API NetWorks is a leading developer of high-speed, I/O interconnect components for the performance and bandwidth-intensive communications, networking, and storage markets. Their bridge chip and switch solutions are used by industry-leading vendors to leverage the power of HyperTransport technology in their next-generation product designs, creating break-through control plane performance.

API, a dedicated user of technology from Synplicity for five years, migrated from the Synplify® tool to the Amplify® Physical Optimizer™ for synthesizing the FPGA version of a key new product – the Starfish AP4041, the industry's first HyperTransport switch technology.

Since volume shipments of the product would be accomplished in its ASIC form, the design was targeted for an ASIC from day one and was, therefore, not efficient for fitting into an FPGA. This presented challenges for FPGA implementation, which were compounded by the design's complexity – five internal clocks – and its sheer size of approximately six million gates.

"The Synplify solution always served our needs well," said Ron Pettyjohn, Engineer for API NetWorks. "For the Starfish AP4041 project, however, I knew we'd need the head start on physical design that only the Amplify product provides. Place-and-route runtimes on a design of this size are long and the Amplify product not only gave us the performance we needed, but significantly reduced the iterations through place-and-route allowing us to save weeks of time."

API NetWorks was able to meet all its engineering and schedule goals by using the Amplify solution to provide the Xilinx place-and-route tool with a head start on physical layout. By producing the product first as an FPGA, API NetWorks captured a great deal of customer interest and accelerated sales of the eventual ASIC implementation.

For more information on the API NetWorks story, visit [www.synplicity.com](http://www.synplicity.com). ☉

## Synplicity® Training Courses Available!

**Whether** you are a new or experienced user, Synplicity offers a variety of focused product training courses to help ensure your productivity in your daily job. With the flexibility of attending formal training classes, purchasing your own copy of the training material, or registering for the on-line, self-paced course, you choose an option that best fits your schedule and learning style.

### New On-Line, Self-Paced Courses

This month we're introducing our new on-line Synplify Pro® for Xilinx course. This course takes the designer through an introduction to the Synplify Pro tool, as well as Xilinx-specific and advanced features. All Xilinx FPGA designers will find all the tips they need. Take advantage of the special introductory price today! A similar course will be offered for Altera users later this year.

Any user who is evaluating or owns the Amplify® Physical Optimizer™ software can register for the free on-line, self-paced course for the Altera or Xilinx technology.

### Up-to-Date Classroom Training Classes

All courses are taught by Synplicity Corporate

Applications Engineers who are experts in each of our products. All training classes are given at Synplicity's headquarters and feature formal instruction, labs, and plenty of time to ask questions. Classes can also be arranged at your facility. The Synplify Pro and Amplify software training classes have been updated for the 7.1 and 3.1 releases (respectively). Classes are held monthly and filling fast, so be sure to register soon!

### Training Materials

Designers who wish to take a course at their own pace can now purchase the training material used in the instructor-led classes. The training material includes a course manual and lab instructions, with lab files available electronically. Orders are usually shipped within two weeks. For a list of courses and to register, go to

[www.synplicity.com/training](http://www.synplicity.com/training).

To view the current class schedule and register, or to get more information, just go to [www.synplicity.com/training](http://www.synplicity.com/training). Please e-mail all questions to [training@synplicity.com](mailto:training@synplicity.com). ☉