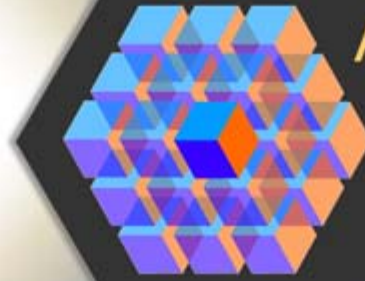


sponsored by

Gabe on EDA • EDAMarket • EDA Confidential



# Assembling the Future

A Newsletter About the Design  
and Production of Electronics

ISSUE 007 • AUGUST 2011

**SUBSCRIBE**

**PDF and Archives**

**for Printing**

**twitter**

## In this issue:

- **High Level Synthesis and Thoughts About Globalization**  
by Gabe Moretti
- **The Maturation of High-level Design**  
by John Sanguinetti
- **Taking the Load Off RTL Design and Verification**  
by Shawn McCloud
- **Power Optimization for High-level Synthesis**  
by Kiran Vittal
- **United States Innovation Within Global Innovation**  
by Gabe Moretti

---

## High Level Synthesis and Thoughts About Globalization

**Gabe Moretti**

One subject of this newsletter, High Level Synthesis (HLS), represents one of the areas of research and development most important to the ability of the electronic industry to progress toward true system level design unencumbered by manual translation of such design into an implementable representation. I moderated an HLS panel at this year's DAC that proved that HLS tools are widely used and trusted by designers.

Complexity requires designers to work at the highest possible level of abstraction in order to have the possibility to envision the entire system and the relationships among its components. The need

to use reliable machine translation of a design from a system level to a netlist level was recognized many years ago, but the first attempts were not successful. At the time the technique was called Behavioral Synthesis and, as the name implies, was trying to do too much. The word behavior, in fact encompasses algorithms, control systems, and necessary instruction sequences that can only be expressed in software. Yet the pioneering behavioral synthesis tools only addressed hardware and did not benefit from any descriptive language designed to aid their task. The result was that those tools were difficult to use and produced sub-optimal results.

The introduction of SystemC by Synopsys was a turning point in this market. Finally designers could use a dialect of the C language to represent intended hardware functions and those description could be synthesized into a hardware description language (HDL) already commonly used, like Verilog or VHDL. The evolution and growth of a new approach to HLS has been remarkable. Not only the tools themselves have become more powerful and generate optimized hardware representations, but the language they can process has increased in scope. Although SystemC is still very popular among designers, many HLS tools now can accept a much larger subset of C or even C++ and can differentiate input that can be implemented in hardware from that which cannot, at least not yet.

The result is that hardware/software co-design can now take place. To be sure other development contributed to this breakthrough, but HLS is and has been, the key component of progress contributing to lower the cost of electronic system development.

Together with articles from Atrenta, Forte Design Systems, and Mentor Graphics, about HLS this issue also includes an article containing the transcript of the testimony by my friend Dr. Dieter Ernst to the US-China Economic and Security Commission of the US Congress. The testimony is relevant because it shows that development and innovation is now a global affair, not just a local one. For most of its existence the EDA industry has seen leading edge tools developed in the United States, but this is changing. Both captive engineering groups, working for US based companies, and indigenous startups in developing countries are now contributing key innovative tools to our industry.

One cannot say if this situation happened by design or as a consequence of better electronic communications and the global financial systems, but one can say for sure that there is no turning back the clock.

---

## **The Maturation of High-level Design**

**John Sanguinetti, Forte Design Systems**

As an industry, we have been doing hardware design at a higher level of abstraction than RTL for nearly 10 years now, and of course there were earlier attempts which had varying levels of success. There have been a lot of different approaches taken, and a

fair number of different design tools marketed. Now, we can draw some conclusions about what higher level design means, what works, and what doesn't.

### Higher-level means abstraction

"Level" in "higher-level design" refers to a level of abstraction. When we refer to "higherlevel design", we mean "something more abstract than RTL". The definition of a given level of abstraction is often not agreed upon for some time as the practicality of implementation gets sorted out. Indeed, for a long time in the 1990's, the definition of RTL was "whatever Design Compiler accepts as input".

We can now say that "higher-level" design fundamentally is defined by abstraction along two dimensions, time and space. The language used to describe high-level synthesis tools refers to scheduling and allocation as the two fundamental activities, and these are just the implementation of the time and space abstractions, respectively.

### Time abstraction

The time abstraction is easy to see. Instead of RTL code, where every cycle is explicitly identified, HLD code simply specifies the logical operation required, and the amount of time that it takes is not specified explicitly. Figure 1a shows a small bit of code that is untimed. Figure 1b shows the same operation with the timing elaborated.

```
typedef unsigned char bit8;
unsigned func( bit8 a, b, c, d, e )
{
    unsigned y;
    y = ( ( a * b ) + c ) * ( d * e );
    return y;
}
```

[ Figure 1a. Untimed code ]

It is easy to see that an HLS tool could produce the schedule of figure 1b from the input of figure 1a. It is also easy to see that the same result could be obtained in a more compact manner by giving a pragma to the HLS tool as in figure 1c.

```

typedef unsigned char bit8;
unsigned func( bit8 a, b, c, d, e )
{
    unsigned y;
    unsigned t1, t2, t3;

    t1 = a * b;
    wait();    // next clock edge

    t2 = d * e;
    t3 = t1 + c;
    wait();    // next clock edge
    y = t2 * t3;
    return y;
}

```

[ Figure 1b. Cycle-accurate code ]

```

typedef unsigned char bit8;
unsigned func( bit8 a, b, c, d, e )
{
    // pragma LATENCY 3
    unsigned y;
    y = ( ( a * b ) + c ) * ( d * e );
    return y;
}

```

[ Figure 1c. Untimed code with a pragma ]

Timing abstraction has been a primary feature of HLD from the beginning, as the benefits are obvious. However, in most real designs, there is some aspect of timing that must be explicitly controlled, usually at points of interaction with other modules or functions. As a result, tools which were developed that implement only untimed code, using straight C/C++ as input, had practical difficulties interfacing the modules they produced with other parts of a system.

It is now apparent that for general use, it is necessary for an HLS tool to support a mixture of untimed and cycle-accurate code in the same module. This can be called “protocol-accurate”, and is illustrated in figure 2.

```

{ // pragma PROTOCOL("get");
  // this block is cycle-accurate
  wait();
  in_rdy = 1;
  do { wait(); } while ( !in_vld);
  a = in_a; b = in_b;
  in_rdy = 0;
}

// this block is untimed
val = func(a, b, c, d, e);

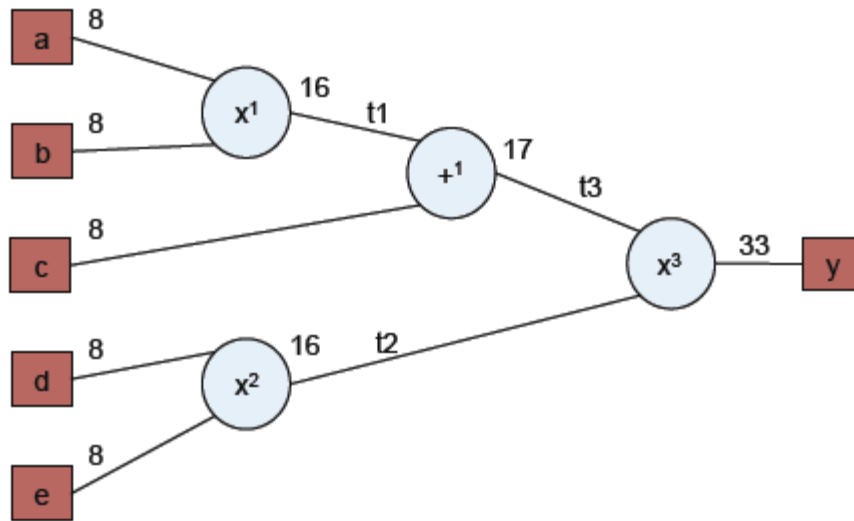
{ // pragma PROTOCOL("put");
  // this block is cycle-accurate
  do { wait(); } while ( !out_rdy);
  out_data = val;
  out_vld = 1;
  wait();
  out_vld = 0;
}

```

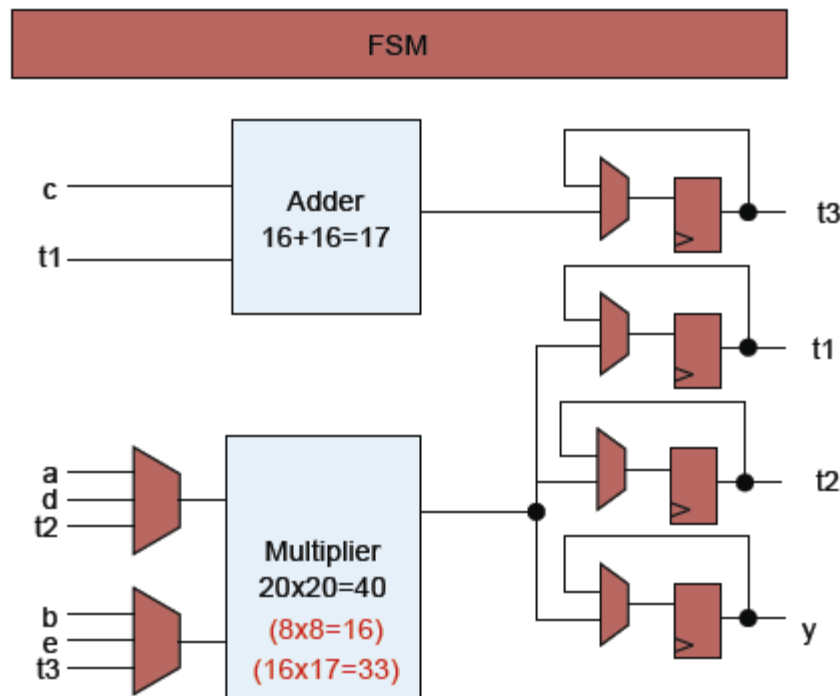
[ [Figure 2. Protocol-accurate code](#) ]

### Space Abstraction

The second major abstraction provided by “high-level design” is space. That is, how many hardware resources, and of what type, are used to implement the function described. As with time abstraction, the designer shouldn’t have to care which adders, multipliers, muxes, and registers, or how many, are used to implement the described computation. Again like the time abstraction, the HLS tool has many degrees of freedom, and the use of additional information in the form of pragmas can be used to effect the resulting implementation. Figure 3a shows the graph of the computation of the function in figure 1a. Figure 3b shows one possible mapping of the operations onto hardware resources.



[ Figure 3a. Graph of figure 1a computation ]



[ Figure 3b. A possible implementation ]

Figure 3c shows the schedule that would be required to implement the function using the hardware resources of figure 3b. This illustrates the interaction between the time and space abstractions.

If untimed code is given to an HLS tool unconstrained, it can always produce a correct, though not always good, result. However, if constraints are given, it is quite possible to over-constrain the problem such that there is no implementation that satisfies the desired functionality and the given constraints.

In practice, there are few pragmas that are useful concerning space. Sometimes, the

designer knows what kind of adder or multiplier he wants to use, but most often that kind of choice is best left up to the HLS tool. However, there is one pragma that is very useful, and that is an indication to the HLS tool that a given computation should be done in a single data-path component. Datapath synthesis tools have been around for a long time, and are pretty mature. Combining a datapath synthesis capability with HLS is very effective at creating custom parts that can be scheduled in a shorter time and using less space than if a collection of discrete smaller units were used. In our example here, that pragma could be applied to the function of figure 1a and the HLS tool would create a single part to do the entire computation, using as little time as possible (which could be several cycles).

Operator	# Needed	Cycle 1	Cycle 2	Cycle 3
$16+16 = 17$	1		$t3=t1+c$	
$20 \times 20 = 40$	1	$t1=a*b$	$t2=d*e$	$v=t2*t3$

[ Figure 3c. Schedule of computation ]

### Other abstractions

Though time and space are the two fundamental abstractions provided by what is termed “high-level design”, there are several other abstractions which have found their way into general use. These abstractions are implemented by extension[1] of the input language, which is readily available with the class structure of C++ and, thus, SystemC. The most useful of these extension abstractions is encapsulation. In SystemC, it is easy to create a class which implements a port, complete with protocol-accurate access methods. The port can then be instantiated as an object and the methods simply referenced in the code being synthesized. Space restrictions preclude a more thorough discussion of this type of abstraction, but suffice it to say that the use of encapsulation makes the time and space abstractions more effective.

### Conclusion

High-level design is all about abstraction. While it has not always been obvious just what constitutes a level of abstraction higher than RTL, the picture has now come into much clearer focus. The fundamental abstractions of HLD are time and space, and the task of creators of the tools to be used in HLD has been to expose those abstractions in the most natural way possible.

[1] extension is a term coined by Jack Dennis in a paper which describes the different forms of creating levels of abstraction: Dennis, J. 1975. *The Design and Construction of Software Systems*. In *Lecture Notes in Computer Science*, 30, Springer-Verlag, New York.

# Taking the Load Off RTL Design and Verification

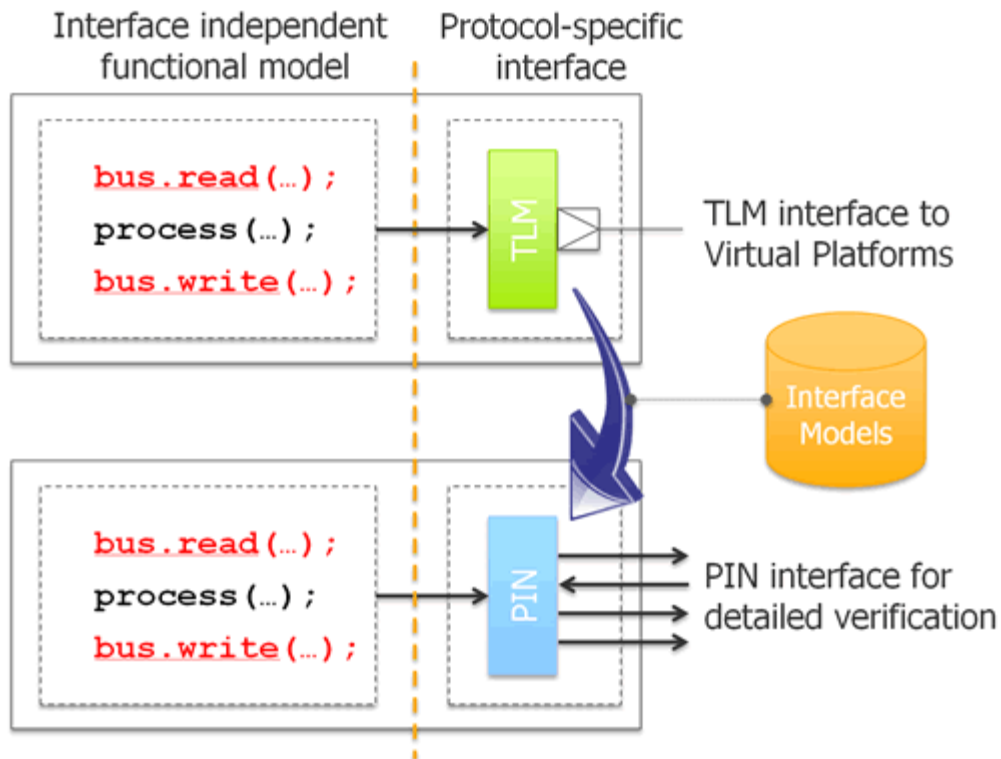
Shawn McCloud, Mentor Graphics

For the future of electronic design to flourish as it has thus far, the loci of activity must shift up a level of abstraction. Only then can productivity and quality increase to match the complexity and intricacy of the systems we're likely to see. This shift requires model convergence and a reduction of verification. Two recent innovations in the ESL space make both of these possible today: TLM Synthesis and the delivery of an ESL verification flow. The biggest value of these two innovations is to move validation from the RTL to high-level models implemented in C++ and SystemC.

TLM Synthesis bridges the gap between two different modeling worlds. One is transaction-level models, used to assemble TLM platforms for system-level analysis and verification, and the other is high-level synthesis (HLS) models, used primarily to optimize and automate RTL generation. Up to now these two models have been developed and used in isolation, requiring redundant modeling efforts and incurring redundant verification, model mismatches, and wasted resources. The net is higher engineering cost and longer development cycles.

This dual modeling status quo exists because these two modeling worlds have very different requirements. The biggest difference is that transaction-level models focus on high-speed simulation in order to handle 10, 20, or 50 million gate platforms upwards of 50 MIPS; whereas HLS models must possess the bit-by-bit, pin-level accuracy required to produce high-quality hardware implementations. In fact, the essential difference between a transaction-level model and an HLS model is centered on the interfaces: TLM tends to move packets of data across a transaction boundary versus the individual pin-level wiggles at the signal level used in HLS. This is also where we find the opportunity for a solution.

With TLM Synthesis technology, the core functionality of the hardware and the I/O are compartmentalized so that changing the I/O does not affect the core. Logically, the I/O has two views: an HLS view and a TLM view, which are collected in a *Multi-View IO* library of complex interfaces. For example, if a designer wants to connect a subsystem with an ARM processor-based platform using an AMBA AXI interface, he or she simply plugs in the AMBA-AXI Multi-View IO and selects the TLM view, when working in the TLM domain, and then switches to the the HLS view when it is time to implement the subsystem using HLS.



[ Figure 1. The Multi-View I/O library of standard interface models facilitates conversion of transaction-level interfaces to the pin level while functionality remains unchanged. ]

Using TLM Synthesis to connect the TLM models with the HLS models provides numerous modeling benefits, including more reusability and less modeling effort. But it also delivers significant verification benefits. If design teams are going to move design to a higher abstraction, they need to verify the design at that same level of abstraction. The Multi-View IO components provide the necessary link to verification IP, allowing users of TLM Synthesis to drop in VIP of complex bus interfaces and pair them up with the appropriate Multi-View IO component. This allows them to verify that the resulting implementation is functionally correct using all of the advanced verification technologies, such as constrained random testing, assertion-based coverage, monitoring, and scoreboarding.

It also provides an integrated flow between ESL implementation and ESL verification. Thus the TLM Synthesis methodology using Multi-View IO components not only produces a common, single source model that can be shared between the HLS and TLM worlds, but also shifts validation to the C++/SystemC level.

To see why this is the case, it's important to clarify the difference between validation and verification. Validation asks whether the design matches the specification: does it work as specified. Verification checks to see whether the implementation process has introduced any errors: does the implementation function differently than what was already validated.

Validation presents a formidable challenge because it's not possible to run every possible data combination at slow RTL simulation speeds. Thus, in the conventional flow, validation occurs after or concurrent with implementation, typically using emulation, event driven compute farms, or FPGA hardware prototypes. This is way too late, forcing companies to dedicate up front redundant verification cycles and many man hours to validate whether their RTL does what it is supposed to do

as compared to the spec.

The solution is to move validation to the start of the design cycle before starting implementation. This has the enormous advantage of informing us that what is implemented meets the specification. This is possible because of the speed at which TLM models execute, making it practical to fully validate that the modeled system matches the design goals and specifications. This validated system model now becomes the working specification, which, importantly, is executable. The advantages are significant: greater predictability, higher productivity, and faster time to tape outs. Further, software designers can start software development months before the hardware is ready because the executable reference platform is available months in advance.

To achieve the highest productivity gains, it is critical that the models used for design validation are also implementation ready. We call this HLS-ready TLM platforms. This requires that model partitioning is done in such a way that the model is TLM Synthesis ready and the verification goals are predictable. Thus, pre-synthesis, validation tasks will not need to be repeated. In other words, partitioning must be done correctly to enable verification sign off of the high-level model prior to synthesis. Thus, the value of the system-level modeling proposition is to pull validation up front by creating an executable specification that is TLM Synthesis ready.

These executable specs can also be used as reference models in future designs. Validation in the presence of the software typically uses a TLM platform. A lot of pieces of that platform can be reused, such as star IP, while some pieces need to be designed from scratch. The latter are the targets for high level synthesis. Again, for reasons of productivity, to retain the benefits of not having to develop and verify multiple models, the single source models must be virtual platform and HLS ready.

The Mentor Graphics TLM Synthesis unified methodology with the Multi-View IO Library, as supported by Catapult® C Synthesis and Vista™, makes a major shift to higher level design and verification possible with model convergence and validation at the C++/SystemC level. Among the benefits contributing to significant advances in productivity are reusability, reduced modeling effort, and reduced verification.

#### **About the Author:**



Shawn McCloud is the Product Line Director for the Mentor Graphics high-level synthesis technology. He joined Mentor Graphics in 1994 after several years as a senior system architect responsible for RISC and CISC based micro-processor design. Shawn received his B.S. degree in electrical and computer engineering from Case Western Reserve University.

---

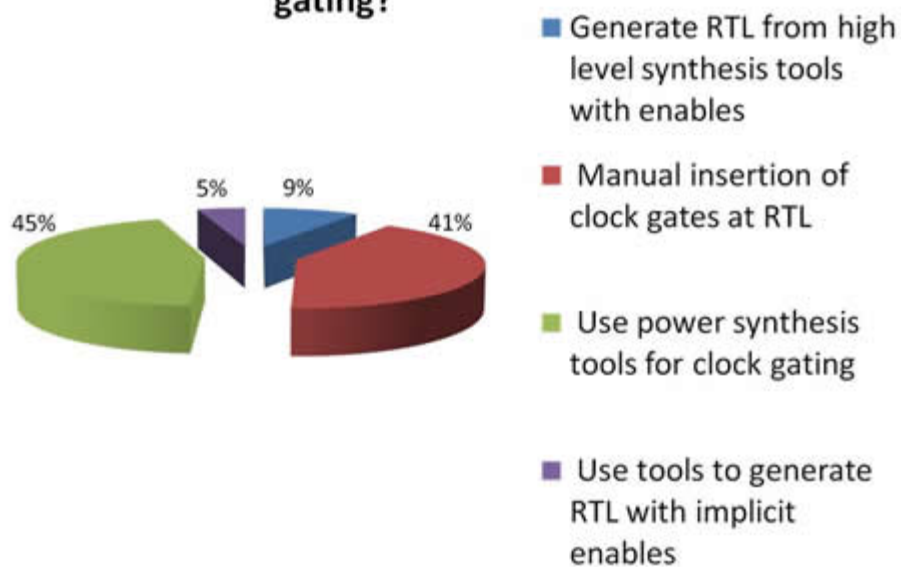
## **Power Optimization for High-level Synthesis**

**Kiran Vittal, Director of Product Marketing Atrenta Inc.**

High-level synthesis is now seeing wider adoption and applicability. It can be used to design subsystems comprised of algorithm and control logic including interfaces, finite state machines, and data paths found in the most complicated designs.

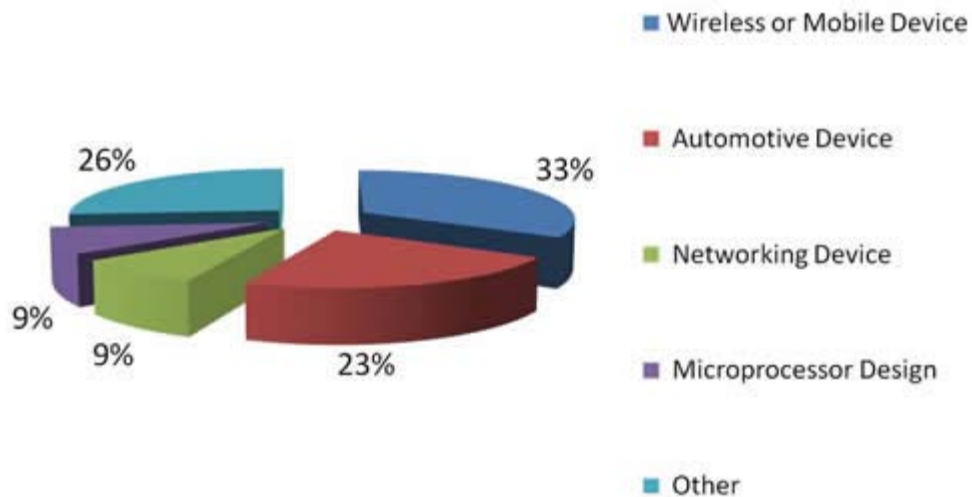
High-level synthesis tools available in the market today start with highly abstract ANSI C++ source. Through an interactive synthesis process, designers can automatically evaluate different candidate designs for area, performance, and power consumption and make important design decisions at a highly abstract level. It's very important to do power analysis and optimization as the RTL for different architectures of these complex systems is being created. Today's advanced high-level synthesis tools automatically apply common techniques to save power, including clock gating. For the designer to make the most informed choice, these tools should integrate an industry-standard power estimation tool to measure the power for the different architectures.

### What solution do you use to address clock gating?



[ Figure 1a: Atrenta seminar survey data – March 2011 ]

## What is the end-application of your chip/design?



[ Figure 1b: Atrenta seminar survey data – March 2011]

In a recent survey of 80 designers attending Atrenta seminars in India and Taiwan, 9% of the respondents said they use high-level synthesis tools and 33% of the total respondents designed chips for wireless or mobile applications. Meanwhile, 41% of the respondents inserted clock gates at RTL, and 45% used power synthesis tools to insert clock gates at the netlist level to save power.

Although the use of high-level synthesis tools is growing, designers have already adopted early design analysis solutions like Atrenta's SpyGlass® for lint, clocks, testability, constraints, physical, power analysis and optimization at RTL.

Atrenta has formed partnerships with leading high-level synthesis tool providers under its SpyLinks™ industry alliance program. The program ensures that the RTL output from the partners' tools is validated with the SpyGlass RTL analysis platform and meets quality requirements on a wide range of issues such as coding standards, synthesizability, simulation readiness, clock management and power management.

The integration of these high-level synthesis tools with Atrenta's SpyGlass lint and power solutions enables quick assessment of the power for different architectures.

Figure 2 shows a high-level synthesis tool flow that generates RTL from a C-based specification. The integrated high-level synthesis and SpyGlass environment helps the designer validate that the RTL meets the required coding guidelines with lint checks. The designer then runs SpyGlass Power estimation to measure the power for different architectures and chooses the best architecture that meets the area, performance and power criteria. SpyGlass Power reduction can further optimize the power in the generated RTL by using formal analysis techniques to find new clock-gating opportunities and generate a new power optimized design using the "RTL AutoFix" capability. SpyGlass Power is also clock domain crossing (CDC) aware and will only create opportunities that

are safe from metastability issues.

The power-optimized RTL can also be verified to meet functionality requirements by running simulation. SpyGlass sequential equivalence checking (SEC) can also be performed against the original “generated RTL” to make sure that the new reduction opportunities have not broken functionality.



[ Figure 2: Integrated high-level synthesis & power optimization flow ]

Example results from running SpyGlass Power estimation invoked completely from a leading high-level synthesis environment are as follows:

Fmax	Multipliers	Throughput	Total Power	Area Score
50 MHz	8	20 ns	0.61 mW	13,402
100 MHz	4	20 ns	2.43 mW	8,390
200 MHz	2	20 ns	3.58 mW	5,162
200 MHz	8	5 ns	2.30 mW	13,402
400 MHz	8	2.5 ns	4.54 mW	15,714

[ Figure 4 - Example results from running SpyGlass Power estimation]

From these results, the fourth iteration is selected as it is closest to target goals with the best area, performance and power.

NOTE: The RTL for the 200 MHz / 5 ns result was essentially the same as the 50 MHz solution, with everything occurring in one clock cycle. The 400 MHz implementation needed faster multipliers which pushed the area up as synthesis used faster multiplier components.

As seen by the above data, optimal results can be achieved by using a tool like SpyGlass Power in conjunction with high-level synthesis. The clock gating enables created during high-level synthesis

can be validated by SpyGlass and additional power-saving opportunities are found using formal techniques. The newly generated RTL from SpyGlass now has the best area and performance as well as requires the least power. This hybrid strategy is an effective approach to early power optimization.

#### References:

Atrenta Announces "SpyGlass" Clean Flow with Leading ESL Synthesis Providers

<http://www.atrenta.com/atrenta-news/70.news>

Power Opto and Linting in CatapultC and SpyGlass

<http://www.deepchip.com/items/0478-08.html>

Power analysis of clock gating at RTL

[http://i.cmpnet.com/eetimes/news/online/2010/06/Atrenta\\_Power.pdf](http://i.cmpnet.com/eetimes/news/online/2010/06/Atrenta_Power.pdf)

© 2011 Atrenta Inc. All rights reserved. Atrenta, the Atrenta logo, and SpyGlass are registered trademarks of Atrenta. SpyLinks is a trademark of Atrenta Inc. All others are the property of their respective holders.

---

## United States Innovation Within Global Innovation

### Gabe Moretti

This newsletter is not, and will not be, a vehicle to present political ideas, positions, or commentary. Having said this, the forward looking nature of the newsletter makes it necessary to consider not just the technical aspect of progress, but also the financial and political environment within which such progress can be made. No one will argue against the statement that we now live in a global economy, and that what goes on or is proposed in the US must be evaluated against the backdrop of global conditions and evolution.

The subject of this newsletter High Level Synthesis represents one of the areas of research and development most important to the ability of the electronic industry to progress toward true system level design unencumbered by manual translation of such design into an implementable representation.

To this purpose I am going to reprint with permission the testimony given by my friend Dr. Dieter Ernst before the US-China Economic and Security Review Commission of the 112 Congress, on June 15, 2011.

Dr. Dieter Ernst is a Senior Fellow at the East-West Center in Honolulu, Hawaii. Dr. Ernest is a former senior advisor to the Organization for Economic Cooperation and Development in Paris, a

former research director of the Berkeley Roundtable on the International Economy, University of California at Berkeley, and a former professor of International Business at the Copenhagen Business School.

Dr. Ernst has co-chaired an advisory committee of the U.S. Social Science Research Council to develop a new program on innovation, business institutions, and governance in Asia. He has also served as scientific advisor to several institutions, among them the Organization of Economic Cooperation and Development, the World Bank, the National Bureau for Asian Research, the U.N. Conference on Trade and Development, and the U.N. Industrial Development Organization. He holds a Ph.D. in economics from the University of Bremen.

### **The Testimony**

Thank you, Chairman. And I'm really delighted to have the opportunity to testify today on China's indigenous innovation policies and possible challenges for America. And as you can see from my institution affiliation, this topic is of great interest to the East-West Center.

Based on our research, I would like to start with a simple statement: Fears that China's innovation policy constitutes an immediate threat to U.S. leadership in science and technology are a bit exaggerated. In my statement, I present data that show China's quite substantial achievements in a relatively short period of time in terms of developing innovative capabilities. These achievements are impressive. But if you compare these data, and in particular look at data that try to bring out qualitative aspects of innovative capabilities, you see that China still has a very persistent innovation gap relative to the U.S., and even relative to the European Union and Japan.

And so the issue is not an immediate threat to the existing American innovation system. The issue is much more that what we see happening in China and in other emerging economies should be taken as a wake-up call for the U.S., a wake-up call for America. We need to look first, at what are the fundamental issues that we need to address in our trade diplomacy vis-a-vis a country like China or India.

We need to take a hard look and ask: Have we done enough? That was one of the questions I heard in the earlier session. Have we done enough or can we actually do better? And I would argue we can do substantially better even with limited resources, and that's what I'm trying to explain within the written statement. We can do better on the international front with regard to our economic diplomacy.

And the second element, of course, is we need to reconsider what we can do actually at home. What would we need to do in order to build on existing strengths of the American innovation system and adapt it and adjust it to the challenges that we are facing. The fundamental challenge is that not only production but also the development of new technologies, i.e., innovation, are being rapidly internationalized.

In the new world of global production and innovation, even the U.S. can no longer do things on its own. We are part of global production and innovation networks. It's not just China that is part of that. We're also part of those networks. And so that's one important global transformation that needs to be reflected in our continuous debates about what we as a government and what we as American companies can do to adjust to this changing reality. And part of that new reality, of course, is that

we have now new players, who seek to reshape the dynamics of global competition. China is one of them. And these new players start with very different institutions. They are on a very different level of economic development, and so it shouldn't be a surprise that they use policies that are not in complete compliance, to say the least, with what we consider to be the rules of the game.

This is not a value statement . It 's just a statement of fact. I mean their approach, the Chinese approach, is different from ours. And so if we want to achieve something, we really need to understand the subtleties of the Chinese approach, and I would say it's important to understand that there are different stakeholders and actors in China.

These different Chinese actors have conflicting interests. And so maybe a proactive and smart trade diplomacy can play a little bit on this fragmentation of China's innovation system, trying to build coalitions with stakeholders in China that are much more interested in fostering a more open system. This probably would help us strengthen our policy response on trade conflicts that result from China's innovation policy. In terms of strengthening our domestic innovation system, a defining strength of the American system is that it has thrived on a decentralized market-led system, and that system has produced a treasure trove of innovations.

America's innovation system is still alive and well. However, given the global transformations that I alluded to before, we need to complement this system of decentralized market-led innovation with reinvigorated public-private partnerships like, for instance, DARPA or SBIR, The Small Business Innovation Research program. We need to combine market-led innovation with robust public-private partnerships. If we would do that , if we really would just look again at a white sheet of paper and identify what we could actually do within the tight limits that we have, I am confident that we could solve some of the challenges that result from China's innovation policy.

Thank you very much.

## References

To read more from Dr. Ernst you can read his prepared statement for the commission beginning on page 59 of [this document](#).

Two more documents from Dr. Ernst are relevant to the topic:

[Indigenous Innovation and Globalization: The Challenge for China's Standardization Strategy](#), and [China's Innovation Policy Is a Wake-Up Call for America](#).

