sponsored by

Gabe on EDA • EDAmarket

Assembling the Future

A Newsletter About the Design and Production of Electronics

ISSUE 018 • AUGUST 2012

SUBSCRIBE

PDF and Archives

twitter

In this issue:

The Growth of High Level Synthesis

Gabe Moretti

High Level Synthesis (HLS) was baptized with a bad name: Behavioral Synthesis. The term behavioral does not suggest a scientific nature to most observers, and in fact there was really nothing in the technology to suggest the intent to describe how a particular design would behave. Not that its present name is much better, but it is an improvement.

The result was that designers did not trust it. Of course the name was not the only culprit. Early tools implementers were too ambitious and tried to immediately solve problems that were too complicated. The results were a collection of issues that spelled the end of behavioral synthesis. It was too difficult to use and often provided results that were wrong in the context of the intended design.

High Level Synthesis connotes something that is above a normal or low level synthesis, if one wants to use semantics. And very few people would agree that RTL synthesis is either low level or normal. Yet the industry has not found a better name for describing the transformation of functional

descriptions into one that is synthesizable into one that uses only primitives supported by a semiconductors foundry. I found it interesting that Calypto (www.calypto.com) on its web site calls the technology ESL Synthesis. This might be a better description of what this class of tools do then HLS. . The input is certainly at the system level, and certainly describes an electronic function. Is ESS taken as a three letter acronym? The tools in this class do in fact that a portion of a description of an electronic system and reduce it to a lower level of abstraction.

A design can be described in terms of algorithms, structure, inputs and outputs. HLS deals with the algorithmic part of the design, and the major issue to the ability to fully synthesize all of the algorithmic part of a design is that not all algorithms are mathematical and deterministic. Some depend on the environment in which they execute, making them, guess what, behavioral. These blocks of designs are the most difficult to synthesize because in many cases, the tool is not given all of the information about the state, or states, of the surrounding system.

Yet, even with all these issues, HLS has finally established itself as a recognized science and a growing segment of the market is a strong statement to its health and prospects. If there was any doubt that the name "Behavioral Synthesis" or even "High Level Synthesis" create confusion in the minds of those outside of the industry, the article by David Pursley of Forte will clarify the issue. of course David is attracted by the challenges this segment of the industry presents, and, having left, has come back to fight the good battle once more.

The article by Jack Erickson of Cadence provides a very good picture of the state of the HLS (it is too late to correct this to ESS, I fear) market. Japanese designers have bee nat the forefront of using C as a design language, and thus, not surprisingly offer a very good group to study present and potential opportunities for HLS tools. The article contains a few surprises, or may be I should call them oportunities fro reflection. For example, low power is not a primary concern. Is it because the tools are not mature enough?

A return to EDA and a Renewed Commitment to High-Level Synthesis

David Pursley, Director, Product Management, Forte Design Systems

Committing oneself to EDA, especially to an EDA startup like the one I joined in 1998, is an interesting career choice, to say the least. You are committing to long weeks, long trips and it is a family commitment, not an individual commitment. You have an unclear future, but you also have a gut instinct that what you are working on will change the state of art in the digital design world.

These were the early days of High-Level Synthesis, then called Behavioral Synthesis, and success with the tools available was as much art as science. Still, users who persevered were successful, and the technology grew and improved. Eventually, through acquisition and merger, I found myself

at Forte Design Systems, where I am again, but I'm getting ahead of myself.

Since these challenges are endemic in any start-up atmosphere, it isn't the unique part of the EDA commitment. What's unique is that you are working on something you cannot see, cannot touch and, more often than not, cannot explain to laypersons. I was confident my wife never actually understood what we did at Forte or how Cynthesizer, our System C High-Level Synthesis tool, was used. In fact, I'm confident that is still the case, although again I'm getting ahead of myself.

After eight years at Forte, I left to work in the Embedded Computing Technology industry. Our company produced rugged embedded motherboards for everything from slot machines and vegetable sorters to battleships and fighter aircraft.

Suddenly, I was able to explain what I did. My wife understood what I did. My friends understood what I did. I could see products we were designing and selling, and I could see them being used. True, we weren't changing the state of the art in digital design, but there's no real EDA equivalent to seeing your embedded computers boards installed on an Unmanned Aerial Vehicle or the back of a HUMVEE.

Yet, the hunger to change the state of the art remained.

As it happened, earlier this year I found myself inside of Best Buy, and it hit me. Everywhere I looked, I could see "Cynthesizer inside." True, it's not actually written on any of the products on display, but I knew from my friends that Forte products had been used to build chips in physical things I could touch and see throughout the store. Televisions, camcorders, Blu-Ray players, gaming machines, handheld devices, network equipment and more contained ASICs in whole or in part created with Forte products. As I later found out, even some embedded boards from my previous company had "Cynthesizer inside" some of the chips.

As you might guess, one thing led to another and I'm now back at Forte and believe we are playing a small part in changing the start of the art. Here are a few of the ways state of the art for digital design has changed in the last 10 years ago:

- Designers now have the ability to accurately simulate at a high level of abstraction, and then synthesize code directly from the simulation model. The same testbench can (and should) be used on both the behavioral and RTL models. If not, how do you know you are synthesizing correct behavior? All of EDA, like much of life, is Garbage In-Garbage Out.

- In order to maintain the high level of abstraction, some High-Level Synthesis tools support simulation and synthesis of complicated data structures as well floating point, fixed point and complex datatypes. At one point, all of these were considered unsynthesizable constructs for High-Level Synthesis.

- Most users have gone beyond synthesizing dataflow pipelines, arguably the simplest application for High-Level Synthesis. These users are now using it on blocks such as switches, processors and other types of control blocks. Real designs also include control-dominated designs and mixtures of the two. If High-Level Synthesis cannot be used across the design spectrum, it can never become part of mainstream corporate EDA flows.

- Speaking of real designs, when was the last time you designed an ASIC with only one

block? Cynthesizer, for example, allows designers to simulate and synthesize multiple blocks. This includes the implied constraint of being able to do multi-level simulation (TLM, pin-accurate, cycle accurate SystemC or RTL).

- As a side-effect of the above two points, High-Level Synthesis tools support complex, real-world communication paradigms between blocks, ranging from simple handshakes to shared memories to arbitrated bus interfaces. Again, these can (and must) be accurately simulated at the highest level of abstraction in order to verify not just the correctness of the dataflow but also the communication.

- The best tools allow other best-in-class tools to be used together through an API, including simulators, synthesis tools and power tools. It also includes a number of options to tailor the output style for specific downstream flow.

- By far, the largest testament to changing the state of art is that there are now chips where all of the design work was done at the behavioral level. Formerly, hardware designers had been forced to use RTL and now get to use High-Level Synthesis. This is the definition of "changing the state of the art."

By the way, you may have noticed I didn't include the "standard" High-Level Synthesis values of productivity and quality of results. In my mind, those are the minimum ante to even get in the High-Level Synthesis game. It isn't changing the state of the art of digital design; instead, it is a requirement in order to be able to attempt to change the state of the art.

To go back to the original premise of committing to EDA, are Forte and Cynthesizer changing the state of the art for digital design? For a few dozen companies, they are. For many more, not yet. This is the challenge of cutting edge products in EDA, whether in a large company or small. And this is the challenge we accept.

Japan C-to-Silicon User Group Survey Results Provide Insight for HLS Worldwide

Jack Erickson, Product Marketing Director, Cadence

To help ease the adoption of high-level synthesis (HLS), Cadence has been holding its C-to-Silicon Compiler user groups in regions throughout the world. The most recent user group was held in Japan. Japan has been at the forefront of HLS adoption - C/C++/SystemC. With many customers in the rest of the world in the early stages of production adoption, we can learn from Japan's years of experience.

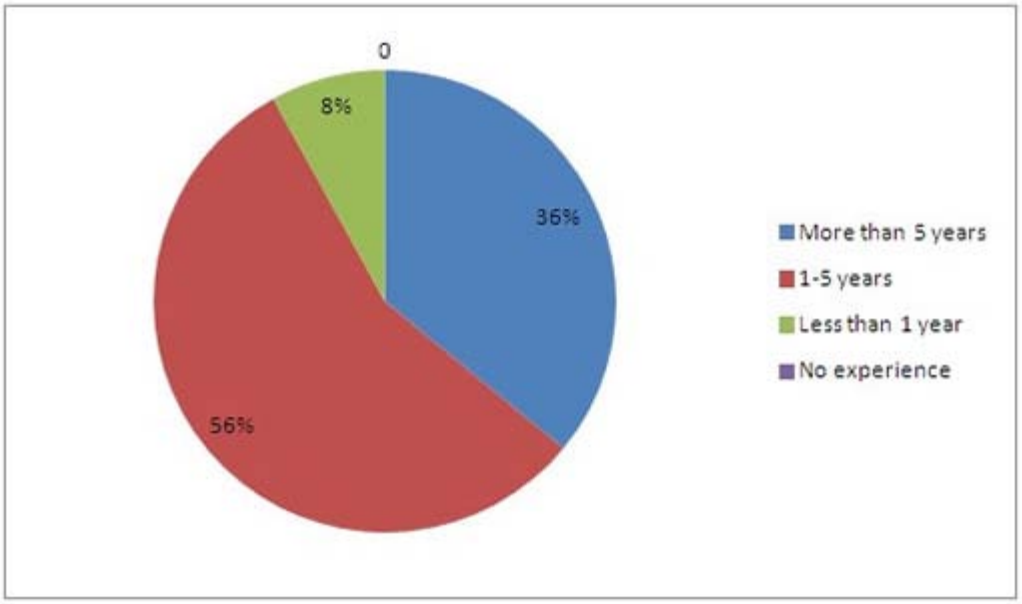
As part of the event, we surveyed the attendees to gain some insight on where they are using Cadence® C-to-Silicon Compiler, HLS and verification in general. Based on what we have seen on

the ground from our engagements in other regions, it seems that there are a lot of parallels and thus lessons that can be taken from Japan's early lead.

Most of the survey questions covered in this article originate from the general HLS portion of the survey that was answered by the 50+ attendees of the general session in the afternoon. There was one exception (explained below). These attendees represent a broad sample of Japanese customers - from the traditional large semiconductor/IDM houses to the more consumer-focused system houses. Attendees of the general session were not necessarily C-to-Silicon Compiler users.

Background: amount of HLS experience

First of all, this audience all had some amount of experience with high-level synthesis:

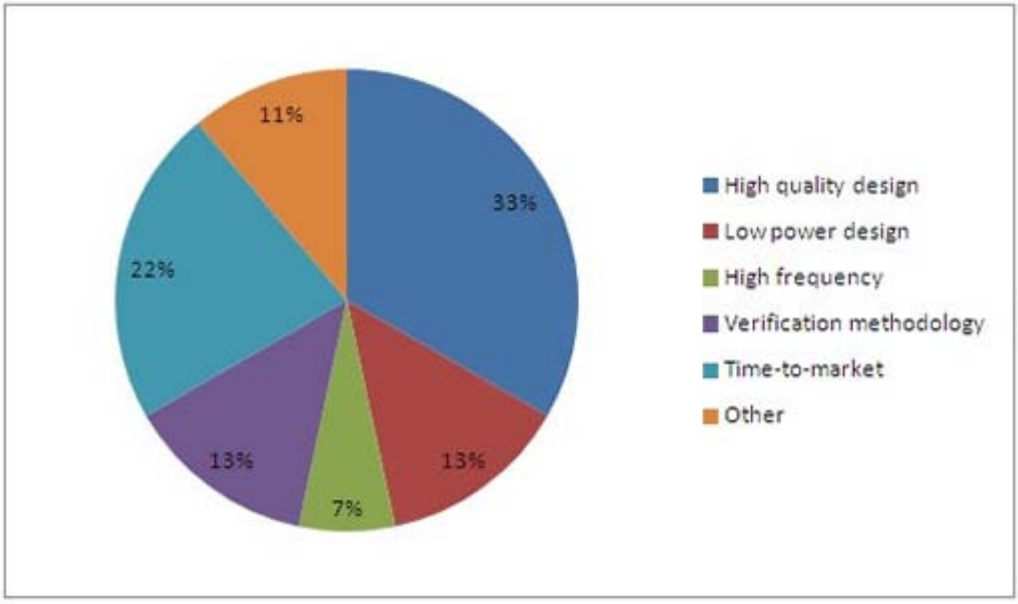


[Figure 1]

Analysis: We assumed that anybody with more than one year of experience has used HLS for at least one production project. So 92% of those surveyed have used HLS in production. It's also quite possible that some of those who have been using HLS for less than one year have production experience. Because HLS is a new methodology, we find that users are not proficient until they have done at least one production project, but their expertise accelerates quickly after that. Additionally, those with more than 5 years experience must have experience with other HLS tools (since C-to-Silicon Compiler is a relatively new tool to the market, having been commercially available since 2009).

Background: design goals

Just to get some insight on what the primary drivers are in terms of design goals, we asked "In your field, what is considered the most important technical attribute?"

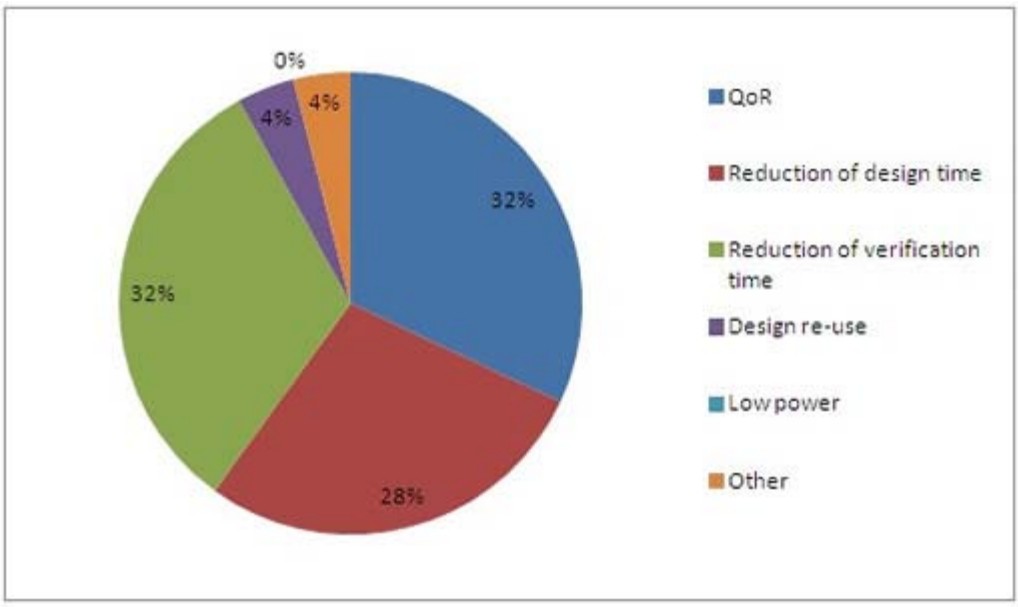


[Figure 2]

Analysis: This is pretty evenly distributed, and a couple of the responses under "Other" stated that many of the design goals were equally important so they could not decide. "High-quality design" is a loaded category—this can mean both functional quality (verification) as well as good quality of results (QoR). Verification methodology could likely be added into this category. The next highest score was for time to market. So these customers need to get to market quickly with a high-quality (functional and QoR) design. No wonder they are using HLS!

Why HLS?

Related to the drivers of design goals, we asked "What is the most important reason for adopting HLS?"

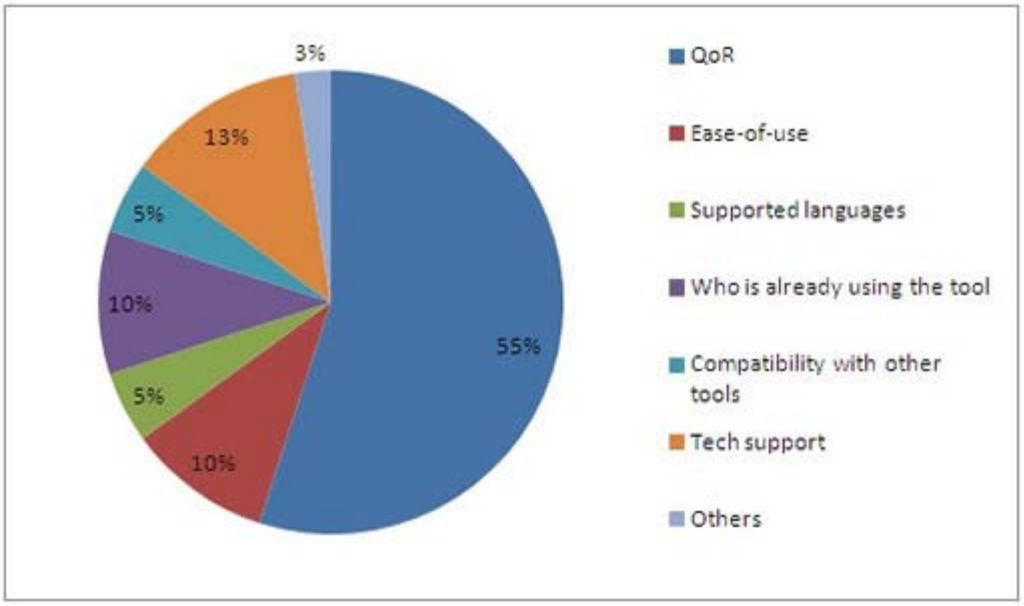


[Figure 3]

Analysis: It is not a surprise that most chose reduction of design/verification time. It was somewhat surprising that only 4% chose design reuse, because we hear from a lot of customers that easy re-targeting of pre-verified IP is a big value of HLS. Perhaps this is more often a secondary driver, since the survey asked them to select only one answer. You may be surprised to see that QoR scored so highly—but remember that HLS allows users to very quickly explore a wide range of micro-architectures to see which best meets their QoR goals. To do that with RTL would require many manual re-writes, so typically there is much less exploration of options.

Tool selection criteria

More specifically, when comparing HLS tools, what is the most important factor?



[Figure 4]

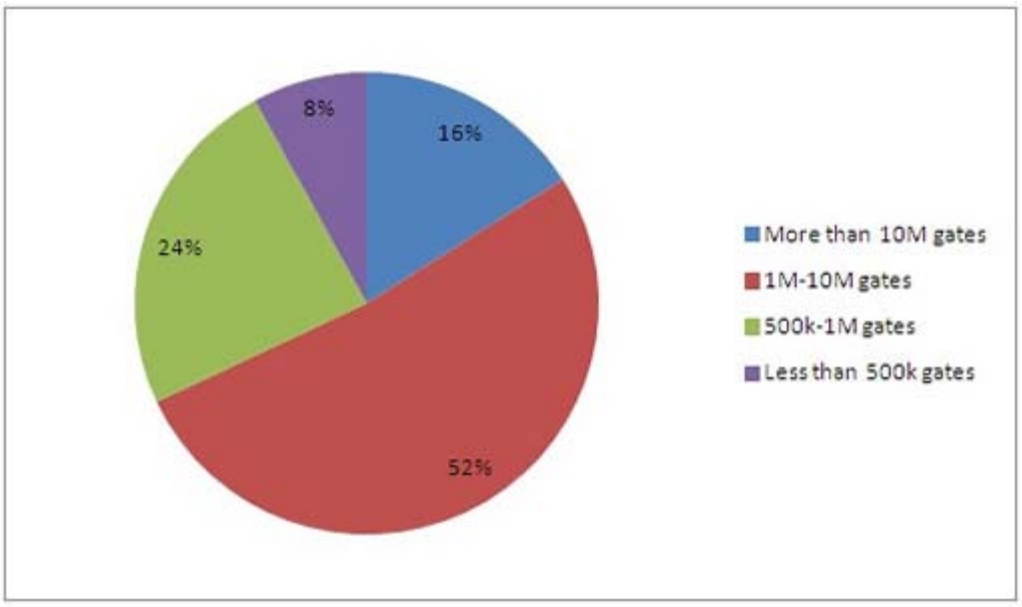
Analysis: More often than not, the first thing prospective customers look at is QoR—whether it can be at least close to that of hand-written RTL. After all, productivity benefits are not worthwhile if users cannot achieve the same high-quality design. As mentioned in the previous analysis, often users can achieve much better QoR if they can find a better micro-architecture. In some cases, however, a design may have fewer micro-architecture options, or a very experienced designer might be able to quickly find one that satisfies the QoR goals. Therefore, the HLS tool also needs to be able to generate RTL for that micro-architecture that will deliver results out of RTL synthesis that are comparable to what a designer could do.

This is where there is a big benefit to a tool like C-to-Silicon Compiler, which embeds production RTL synthesis to guide its scheduling and resource allocation. After QoR comes "Tech support" then "Ease of use," which is not surprising given that this is still a new methodology for the majority of the market. "Who is already using the tool" scoring so highly is also interesting—it shows that this technology is beginning to move from early adopters to early majority.

What size designs?

We are constantly asked what size designs C-to-Silicon Compiler can handle, and what the sweet

spot is. Here is some data. Again, this was asked of the general session audience so it is a sample across different HLS tools:

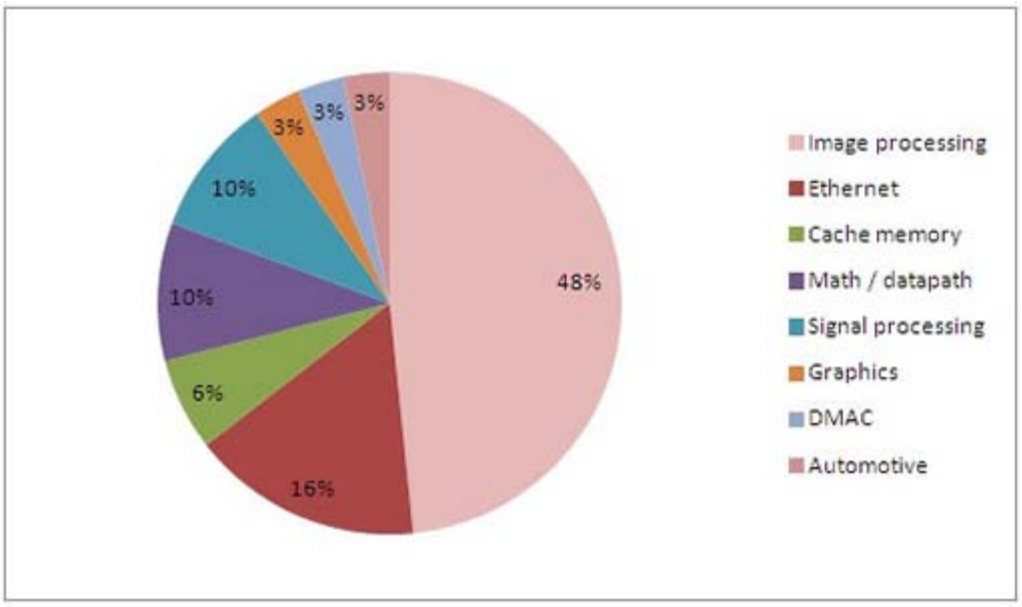


[Figure 5]

Analysis: It seems like 1M to 10M gates is somewhat of a sweet spot, though there is a significant percentage of "More than 10M gates" as well as 500K-1M gates. What does this tell us? That it probably depends on the user's design and how tasks are partitioned. We have customers who use C-to-Silicon Compiler for large subsystems and for small blocks or specific algorithms. Sometimes we see customers only use it for small blocks early in their adoption cycle, as they get more comfortable with the language, tool, and methodology. As they grow the amount of SystemC design code they have in-house, they synthesize larger chunks of the design. As with RTL, the partitioning will probably ultimately settle on functional partitions as well as companies' organizational responsibilities.

Application space

This is another question we often get. What application space are you using C-to-Silicon Compiler? Note that this question was asked specifically of the morning C-to-Silicon user-only audience.



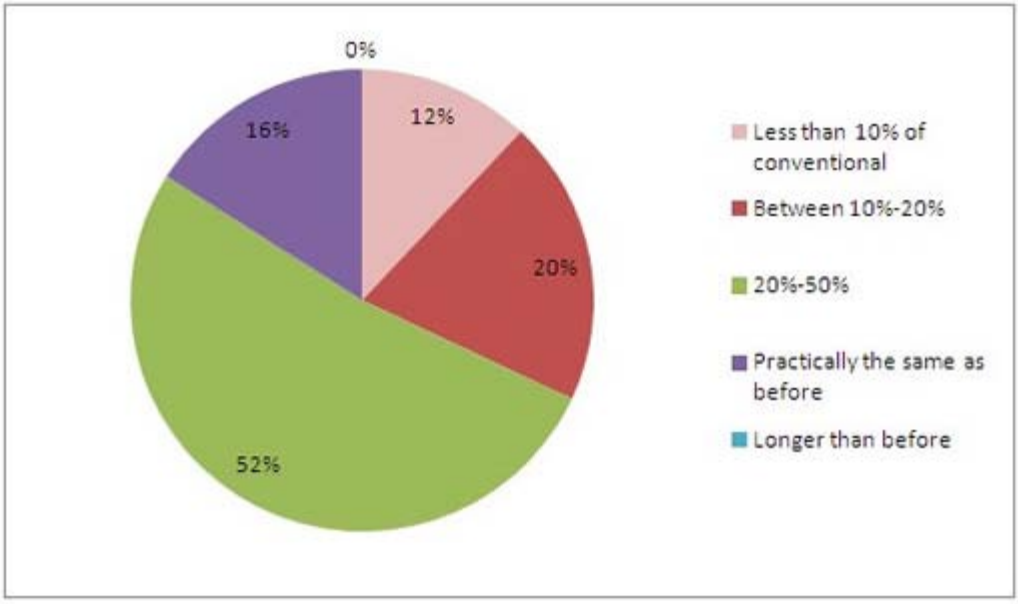
[Figure 6]

Analysis: There are some clear sweet spots here: image processing and various types of signal processing. However, we are seeing more and more adoption of C-to-Silicon Compiler on a wider variety of designs—here we see some Ethernet designs, cache control, DMA control, and even automotive. This reflects what we see in other regions as well. Design teams typically start out using C-to-Silicon Compiler where they anticipate getting the most benefit—algorithm-heavy designs—then as they learn how to get the most out of it, they apply it to a wider variety of designs.

Note that this question was also asked of the general session audience. The answers from the non–C-to-Silicon Compiler users were all either image or signal processing. So in that sample, those applications overwhelmed the others and made for less interesting analysis (other than that other HLS tools seem to be staying more in the HLS sweet spot whereas C-to-Silicon Compiler is being used on a broader array of applications—this could be due to tool limitations or simply that the other tools are used more by algorithm architects than for production hardware design).

Design cycle reduction

Since the primary goal for most customers is to reduce design and verification turnaround time, we asked how much HLS has shortened their design cycle:

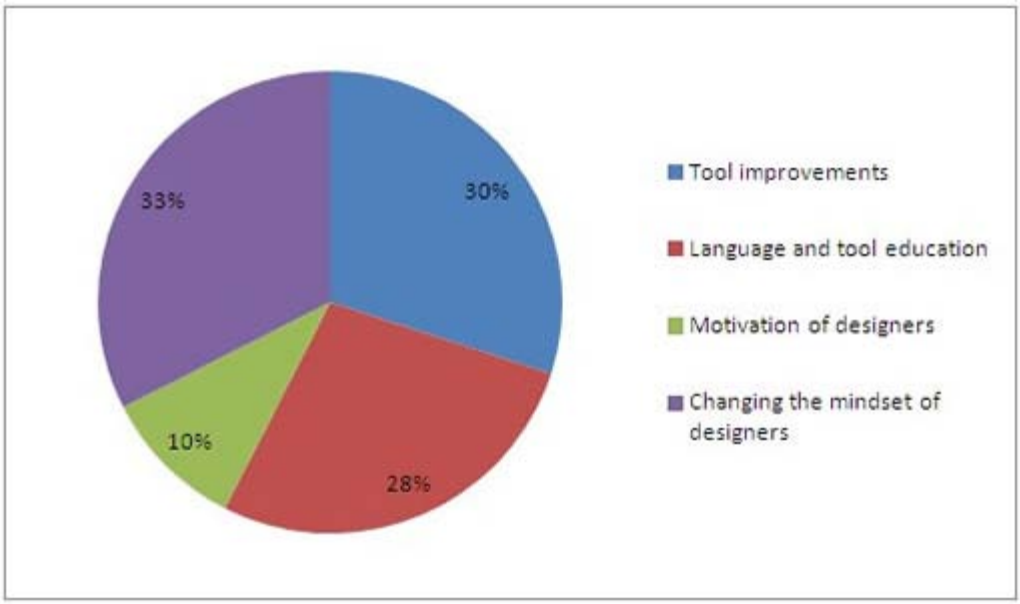


[Figure 7]

Analysis: Nobody's design cycle got longer! Typically we see this metric improve with every design project as customers gain experience. The first project will often be less than 10% improved or even practically the same, since there is a ramp associated with learning the new language, how to get the most out of an HLS tool, and how to most efficiently apply the user's verification methodology. Subsequent projects do not have this ramp and thus see bigger improvements. And when users get to the point of being able to reuse IP without having to re-architect it for different requirements (high frequency vs. low power) or for a different process, then users tend to see very large benefits. Today, the number we typically see is a 30%-50% reduction, and that is right in line with these results.

What is most important to grow adoption?

Looking toward the future, we asked what is the most important point for the widespread use of HLS in the future?

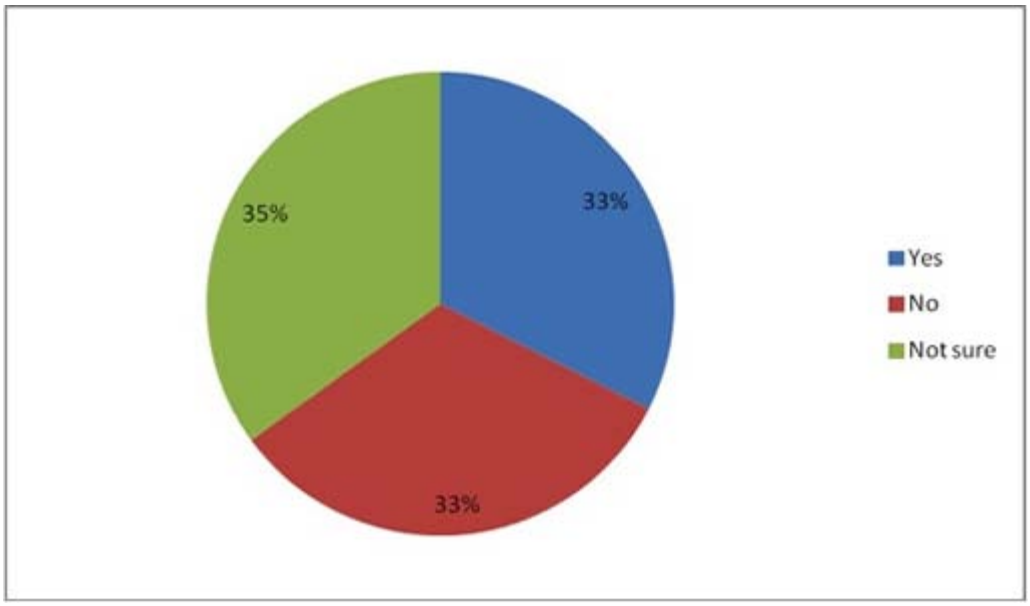


[Figure 8]

Analysis: This reflects very well what was discussed at the HLS panel at the Design Automation Conference (DAC 2012) in San Francisco. While of course tool improvements are needed (we are still early in the technology cycle here), the biggest need by far is education. We find that once designers learn the language and are comfortable with C-to-Silicon Compiler, it's almost universal that they prefer this methodology to writing RTL. So education addresses the "motivation" and "mindset" issues as well.

Will HLS surpass RTL in the short term?

Finally, the "money question"—do you think that within 5 years, HLS technology could surpass RTL design?



[Figure 9]

Analysis: Pretty evenly split! What is more interesting is to examine some of the explanations in the "Not sure" category. Some cited the learning curve while others felt that some design types are still better-suited for RTL. However, the majority of these responses cited the large amount of legacy RTL. In fact, this is probably the reason behind a lot of the "No" responses as well, because 5 years is a short amount of time for HLS usage to surpass the amount of RTL reuse. We see customers typically adopt C-to-Silicon Compiler for projects where they are developing new IP. Often that's a block or subsystem that goes into an SoC dominated by reused and third-party RTL IP. Once they get to a point where the reused IP needs to be re-architected, they re-code it with SystemC. But until then, they reuse what is already verified.

Conclusions

For customers outside of Japan, this is some great insight from a region that has been at the forefront of high-level synthesis adoption. The survey answers generally reflect what we see in the customers we are working with outside of Japan. This should provide some insight into the potential

benefits and challenges associated with adopting high-level design, synthesis, and verification.

It is clear that customers are seeing real benefits once they move up the learning curve, and over the past year or so we have finally been seeing this accelerate outside of Japan as well. Let's give thanks to our Japanese colleagues for blazing the trail in this area, and let's learn from their experience to accelerate the path to the benefits that they are realizing.