Verilog HDL vs. VHDL

For the First Time User

Bill Fuchs / President & CEO - Simucad / Chairman BoD - OVI

Introduction

The search for the perfect HDL is like the search for the perfect car, the perfect home or maybe even the perfect relationship. It probably doesn't exist. With HDLs, it is no different. The decision of which language to choose is based on a number of important baseline requirements (factors), particularly for the first time user. For argument's sake, let's define the first time user as one who has not used HDLs before and is considering the use of an HDL for their current or next design project. For the most part, the use of an HDL based design strategy should improve the first time user's productivity, although in most cases this may not be realized for some time.

Basic Factors for Choosing an HDL

Let's explore some of those factors. The most important factor is really a question, *Why should I implement an HDL in my design process, and what benefits will using an HDL provide for me?* There should be a good, solid reason for the implementation of an HDL-based design methodology. If not, the designer, the design and maybe even the company may be impacted adversely. If time to market is an important consideration then it should also be an integral part of the determination process. Although we have all read about the benefits that can be realized by utilizing an HDL, there are many issues that must be taken into account.

The primary reason for using an HDL should be an overall gain in productivity, although this may be difficult to quantify. If you can't rationally predict a gain in productivity or if a sizable loss of productivity is highly probable, don't give up the HDL investigation. Just remember, it is important to recognize that a productivity gain may not be recognizable until a number of designs have passed through the HDL based design process, and even then it may not assume an easily measurable form. Often times new HDL based designs are so different from the designs of the past that it becomes an apples to oranges comparison and productivity gains cannot be readily determined.

Do not adopt an HDL based design strategy, just to possess so called *state-of-the-art tools* or to move into the supposed world of 21st century design. Some designs are just too simple for using an HDL, and some may be inconsistent with

the strong points of HDLs. Nonetheless, the most important factor will always be, why adopt an HDL and, what are the specific benefits of using an HDL?

Ease of Use

The next most important factor is usability, which can be broken down into separate categories.

Category A, *Ease of Learning*, this relates to how easy it is to learn the language without prior experience with HDLs. It also means how simple is it to convert a designers existing logic design know-how into realizable HDL-based models in a practical timeframe.

Category B, *Ease of Use*, means once the first time user has learned the language, how easy will it be to use the language for their specific design requirements.

Category C, *Future Usability*, means although the language may be sufficient for today's requirements, what about tomorrow's requirements. This is based on the language's long term usability potential as defined by your future design requirements and not a tool vendors future positioning of the language or the tools.

Quote from E.E. Times

When I worked at HP Roseville, I remember taking my first Synopsys training class. The instructor from Synopsys kept telling us that we were making a grave mistake using Verilog and that EVERYONE who was anyone was using VHDL. (I actually was worried at the time we had chosen the wrong language, and that he was really unbiased. As I look back, I'm glad we chose Verilog, especially when teaching new engineers and when getting our SW/FW folks (who eat/sleep/breathe "C") to understand the HDL I have written. I would really encourage any new HDL designer to choose Verilog rather than VHDL, since it is much easier to learn, use and eventually master.

Scott C. Petler
Next Level Communications, Inc.

Adaptability

Another important factor is how the HDL can integrate into the current design environment and the existing design philosophy. Every user has developed certain design methodologies and strategies, which, for better or for worse, have

worked successfully for their designs. The adoption of an HDL-based methodology should afford the designer some gains in productivity but not at the expense of completely changing all of their original methodologies and strategies. Essentially, one should *not* adopt an HDL strategy at the expense of sacrificing everything else. The chosen HDL-based strategy should blend with the existing design tools and methodologies, while providing assistance in moving it into a mode where greater productivity gains can be realized. In the real world, this means taking a structural implementation, usually in gates or transistors, (switch level) and moving it up the ladder of design abstraction to a behavioral (RTL level) implementation. The HDL should support all of the necessary constructs that can enable the first time designer to use this type of methodology for his designs in a reasonable timeframe.

Quote from E.E. Times

In a previous life, I worked as an onsite applications engineer for an ASIC vendor. The customer that I supported was developing 17 ASIC's for a large program. The customer chose to develop some of the designs in VHDL and others in Verilog. All were synthesized using Synopsys. The smallest design was 15K gates, the largest was 100K gates. I interviewed the design teams to gather some interesting statistics. Conclusions were:

> ***1) Designs done in Verilog were, without fail, completed faster than those done in VHDL. (In terms of gates/manweek.)***
>
> ***2) Adding designers to VHDL and Verilog based designs SLOWED the gates/manweek metric, but adding designers to VHDL-based designs had a greater negative impact. (Probably due to data-typing issues.)***
>
> ***3) Single or dual-person design teams out performed all others.***
>
> ***The designers (80) were of various experience levels, working in groups of 2 to 10. Measuring from the end of (the) specification (cycle) to final signoff, the highest performing was a Verilog team at 1500 gates/manweek, the lowest (performing) was a VHDL team with 8 gates/manweek!***

ASIC Foundry Engineer, Anonymous

The Reality Factor

The last factor is one of general reality. Does the HDL support the specific technical methodologies and strategies that the first time user requires? Due to a first time users experience with gate and transistor level structures, this usually

means, working with structural elements and other related attributes including detailed timing constructs.

Does the HDL support timing? Are the libraries or models I use available for the HDL based tools? Are timing-based models available from multiple sources? Are tools available in my affordable price range that can support my intended use of the HDL? Are there resources to help me obtain the optimum results from my usage of the HDL? Will the HDL and these tools help me achieve my desired productivity and *"time to market"* goals?

Now that we have a preliminary set of criterion let's look at two HDLs, Verilog HDL and VHDL, and see how they measure up to this initial set of requirements.

As a first time user, it is important to understand a little background on the two languages. Often times knowing the origin gives a solid basis as to what the original developers intended when they composed the HDL.

VHDL -- Language by Committee

VHDL was developed by committee intended for documenting digital hardware behaviorally. The intent for the language was solely for the explicit purpose of documentation. It originated out of the VHSIC (Very High Speed Integrated Circuit) Program as a part of a US DOD (Department of Defense) Project in 1981. In 1983 the DOD awarded a contract to Intermetrics, IBM and Texas Instruments. It was known as VHDL 7.2 and was completed in 1985,. Its selection by the DOD as a documentation language for its digital designs has provided the initial momentum. In an effort to focus its use on practical applications and to expand beyond the use as simply a documentation language, it has been through numerous iterations. During this refinement process it has become IEEE standard 1076 in 1987. Although it was adopted by many EDA companies and carried strong support from the European electronics market, VHDL had significant deficiencies.

The deficiencies were most evident at the gate and transistor levels. In addition, there was no facility for handling timing information. Due to these crucial, limitations, VHDL did not make a major impact on the design community, although it was heavily promoted in the trade journals and by EDA companies trying to distance themselves from Cadence. The design community proposed a methodology to help VHDL move towards a more useful design language. This initial effort was called the VHDL Initiative Towards ASIC Libraries, or VITAL. VITAL was the supposed to be the specific methodology for describing VHDL gate level structures and timing. Today VHDL is going through the steps of approving Vital 2.2b as an extension and standard based on 1076. In 1992, the

IEEE formulated IEEE standard 1164 - 1992 (MVL-9) as a recommended practice for modeling guidelines for use in various simulation environments.

EDA Tool Support

Any language is only as good as the tools that support it, and VHDL is no exception. Only after VHDL was adopted by a number of the EDA companies as their exclusive HDL did the language become useful as a practical design tool. Although many of these EDA companies had their own proprietary HDLs integrated within their own simulation environments, they elected to adopt VHDL. The reason these EDA companies did not adopt Verilog HDL is that they all have a basic philosophy which states they must own all of their core technology. An HDL, any HDL would be the cornerstone of an EDA companies technology. The schematic editor would use it as would the ESDA tool, the simulation environment, synthesis tools, emulation tools, display tools, almost every phase of the design cycle would be drastically effected by the chosen HDL. These EDA companies selected VHDL because Verilog HDL was the intellectual property of Gateway Design Automation, and eventually was acquired by Cadence. Cadence's ownership of Verilog HDL made it even more essential for these EDA companies to support VHDL and distance themselves from any strategic ties to Verilog HDL and any potential competitive control from Cadence.

Quote from E.E. Times

Actually, an interesting look at VHDL vs. Verilog was accidentally done in our graduate level logic synthesis course. We recently got Synopsys Design Compiler, Synopsys VHDL and Cadence Verilog-XL. While The rest of the class did their projects in VHDL, my lab partner and I did ours in Verilog. (We learned Verilog on our own; unlike my VHDL classmates, we had no class lectures, no T/A help, no professorial help.)

The results of this were overwhelmingly in favor of Verilog as a tool to teach HDLs. Our final project, a 7500 gate, 35nsec RISC processor was ~25 pages of Verilog. The VHDL people all ended up rushing near the end to just make something which worked and could be synthesized. Several groups failed at this altogether! (Whereas our project grew so large in functionality, our only problem was finding a workstation which had enough memory to handle the synthesis of the top level design.)

The general comments in talking to the other students was they spent a majority of their timing fighting VHDL/Synopsys. We spent a majority of our time doing design work, and optimization.

Jeff Echtenkamp

Verilog HDL -- A Language for Designers

Verilog HDL comes from the commercial world and was developed as part of a complete simulation system. It was also developed to be used for describing digitally based hardware systems. Verilog HDL made its mark by allowing designers to represent their designs in the familiar method (gate and switch level descriptions) as well as abstractly or behaviorally. By allowing this top down or bottom up methodology it afforded the designers a learning period to get comfortable while at the same time achieving significant productivity gains. During Verilog HDL's formative years, it was improved on a regular basis by taking the users' input for technology requests and enhancing and modifying the language for practical designer requirements in a timely manner. When Cadence purchased the Verilog assets from Gateway in 1989, Verilog HDL and the and simulation tools in the market. Verilog was described by the users as "*the best in its class*". In 1990 Verilog HDL was placed into public domain and since then end-users, semiconductor companies and EDA companies have directly benefited from this open availability. Today there are more than 40,000 qualified Verilog HDL users; 250 ASIC, FPGA and other types of models and libraries available; more than 300 universities teach the language as part of their advanced EE programs; and now more than 75 companies offer Verilog HDL products and services.

In 1990, Open Verilog International (OVI) assumed responsibility for the public domain assets that comprise Verilog HDL. They have improved the Verilog HDL documentation set and enhanced and extended the language for use with new technologies. OVI applied for IEEE standardization which it will formally receive in December of 1995. The Technical Coordinating Committee enabled four separate technical projects in 1995. These are as follows; Verilog Analog Behavioral Language Standard, ASIC Library Modeling Standard, Delay Calculation Standard and a Verilog HDL language extension and enhancement subcommittee. All of these projects will be submitted to the IEEE within the next two years.

Quote from E.E. Times

I have spent most of my design life (last 4 years) working on VHDL designs. Recently, I have been forced into the Verilog camp by a vendor. My initial concerns that Verilog would not have the functionality that I needed have been proven wrong. Verilog does what I need better - and the simulators are faster than VHDL simulators.

Since VHDL was driven mostly by the government which has no interest in the productivity of the designers, it is not surprising to see your results from the contest.

VHDL syntax hinders progress and does not improve the robustness or quality of the design. The behavioral compilers, not VHDL, make the most sense for doing even more sophisticated design work. Please don't make be go back to VHDL!

Robert Rust
Hewlett Packard Boise Printer Division

Technical Mis-Match-up

The two languages have different technical strengths which significantly differentiates their market focus. Verilog HDL's technical strength is in its ability to represent digital hardware (and analog hardware in Verilog-A) the way a hardware designer would envision and implement the design. Verilog HDL has built-in predefined hardware net types (wire, wor, wand, tri, etc.) thus eliminating the interpretation issues inherent in VHDL. Verilog HDL also has gate and switch level modeling, enabling ASIC foundries to accurately represent their cell libraries. More complex modeling like pullups / pulldowns, dynamic charge sharing and signal strength can be accurately modeled with relative ease.

Verilog HDL affords the designer a simple language syntax and structure. This capability, unlike VHDL, allows the designer to learn the language quickly and develop more concise and effective models. The ease of use of Verilog HDL translates well into the supported simulation environments and their performance characteristics. VHDL models, on the other hand are inherently indecisive, due to programmability and therefore do not provide a framework to achieve adequate performance from their respective simulation tools. We have seen a recent test of this fact from NEC. NEC developed VHDL libraries for their complex ASICs at the gate level using the Vital 2.2b specification. NEC's experience with HDL's and model and library development is extensive. The resultant NEC, VHDL, Vital 2.2b libraries required almost 50X more memory to run than the equivalent Verilog HDL description of the same model. This translates into simulation speeds that are 50 to 100 longer than the same Verilog HDL-based simulation run. The resultant performance is commercially unacceptable.

Verilog HDL was designed with features (such as global variables) which are required to model the system's environment. Therefore, it is easy to model a "test bed" using Verilog HDL. This capability is a significant deficiency in VHDL. Verilog HDL also permits the use of monitoring code within a model to insure that errors are caught early in the design process. The modeling code has no physical analogy and simply cannot be handled easily in VHDL.

Quote from E.E. Times

My transition from VHDL to Verilog came about 2 years back when I worked on a design which was about 45K gates. I learned Verilog as the ASIC Vendor we worked with was only comfortable doing a final signoff in Verilog rather than VHDL. With the flavor of both the languages, here are my comments:

1) VHDL is a good structured HIGH level language but I feel Verilog is closer to actual hardware which is being designed.

2) As far as behavioral goes, I rank VHDL at par with Verilog, but when it comes to RTL, I consider Verilog has the edge over VHDL as far as the time to market (i.e. meeting the design schedule is concerned.)

As far as the contest goes, I think Verilog has again proved the point. Yes, with VHDL you can achieve the same target but at the cost of design time and support. In the present industry, time to market a product is the key to success. If a particular market window is missed, the ASIC and the man-months spent on it are a sheer waste. I strongly feel that given the choice and the design time I would opt for Verilog.

Subhodip Ghosh
Western Digital Corp.

Timing is Everything

Verilog HDL provides module paths and conditional specification of path delays through the specify block. This critical element of the language is important because neither lumped or distributed timing models are complete by themselves. The Standard Delay Format (SDF) in Verilog HDL provides the essential back annotation facility for loading post route delay calculations, a utility not available in VHDL. When the technical strengths of gate and switch level modeling and specific timing constructs are considered it is no wonder that almost every ASIC foundry on the planet *"sign-off"* for production on all their designs using Verilog HDL. Presently VHDL models or simulation is not used for *sign-off* by any semiconductor company. Today ASIC library support for Verilog HDL is impressive, with over 250 libraries (with full timing parameters) from more than 40 different companies, available to users of Verilog HDL. Another important fact, more than 70% of all semiconductor companies worldwide use Verilog HDL for their *"Golden Simulation"* requirements.

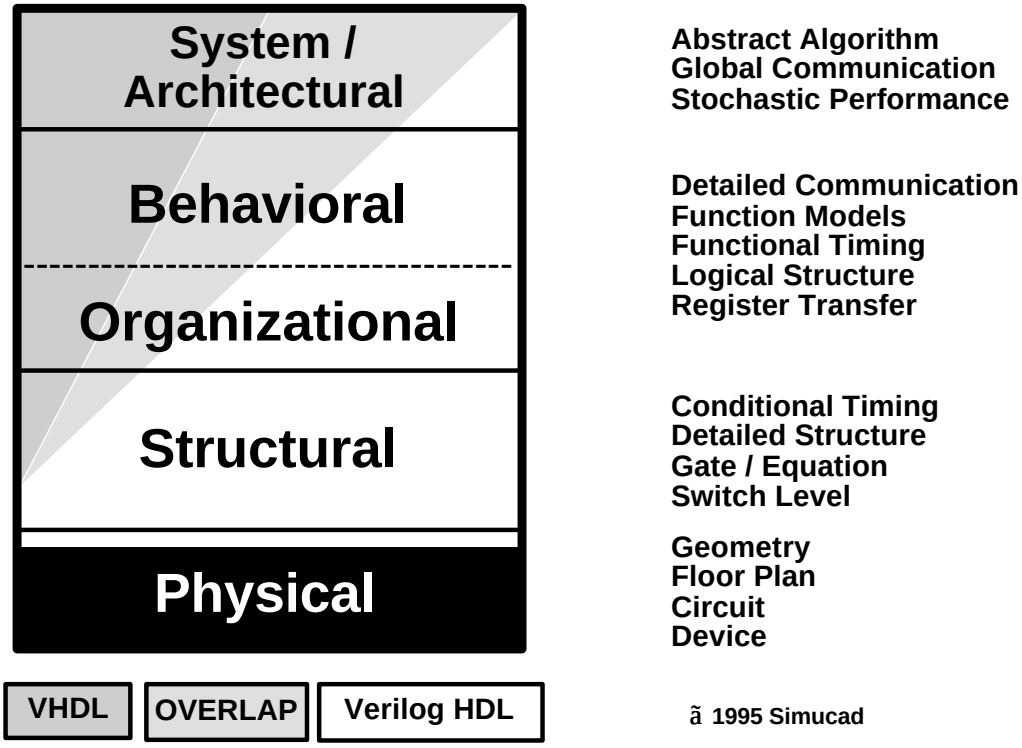
There are other areas of technical importance that affords Verilog HDL some significant benefits when compared to VHDL. The ability to reference a signal across module boundaries is necessary because it provides an arbitrary access to variables in the model. This connection can be provided without any structural requirements such as declaring ports. In addition, Verilog HDL supports "named events" that provide a much needed abstraction at the high

level where "events occur" and "events get consumed". This use of the language is extremely beneficial when modeling the "test bed environment" of a design. VHDL relies solely on communicating through signal values and the inherent inconsistencies associated with this type of process.

Verilog HDL's language supports system tasks and functions as part of the language. A designer can embed control commands into the language thereby making model development and debugging of the design much more effective and significantly more productive. VHDL simply does not support this vital debugging principle at all.

The following graph highlights the language's design space with respect to the levels of abstraction each can address.

Hierarchical Comparison of HDL Strengths



Quote from E.E. Times

It's clear from the contest that Verilog can get you to a netlist faster than VHDL - period end of story. BUT my experience has shown that the amount of time to generate a netlist is small in comparison to the over all ASIC design schedule. Verification (i.e. test bench generation) makes up most of the ASIC design schedules I put together. Verilog's C-like structure provides a very flexible environment which

integrates very smoothly into most test bench solutions. In addition, focusing on test benchs illuminates one of Verilog's best features: the PLI. I don't believe VHDL provides a PLI counterpart. Without a PLI many of the third part tools that I rely on would not be available.

At GI we have made use of Verilog's PLI for many tasks ranging from memory efficient input stimulus handling to automated test vector generation.

Rick Price
General Instrument Corp.

Synthesis Impact

The impact of synthesis on the HDL market is one of the most significant reasons for HDLs growth over the last few years. Verilog HDL has enabled this impact because synthesis tools can map directly from and to Verilog HDL without the need of a special "package". Most language statements can be synthesized, eliminating the need for large degrees of parameterization. VHDL, however, because of its programmability, must drastically limit the use of its language constructs. VHDL must be highly parameterized (limited in content) when developing models that are "synthesizable". In reality, this effectively limits the "unique programmability benefits" of VHDL. Since most people model for the purpose of implementation (physical structures) and the current shift has been toward the use of synthesis tools, the synthesis user will realize the greatest productivity benefit from the use of Verilog HDL. Although synthesis benefits VHDL too, the productivity gains are minimal, and more importantly without synthesis, VHDL's usefulness as a true design language diminishes significantly.

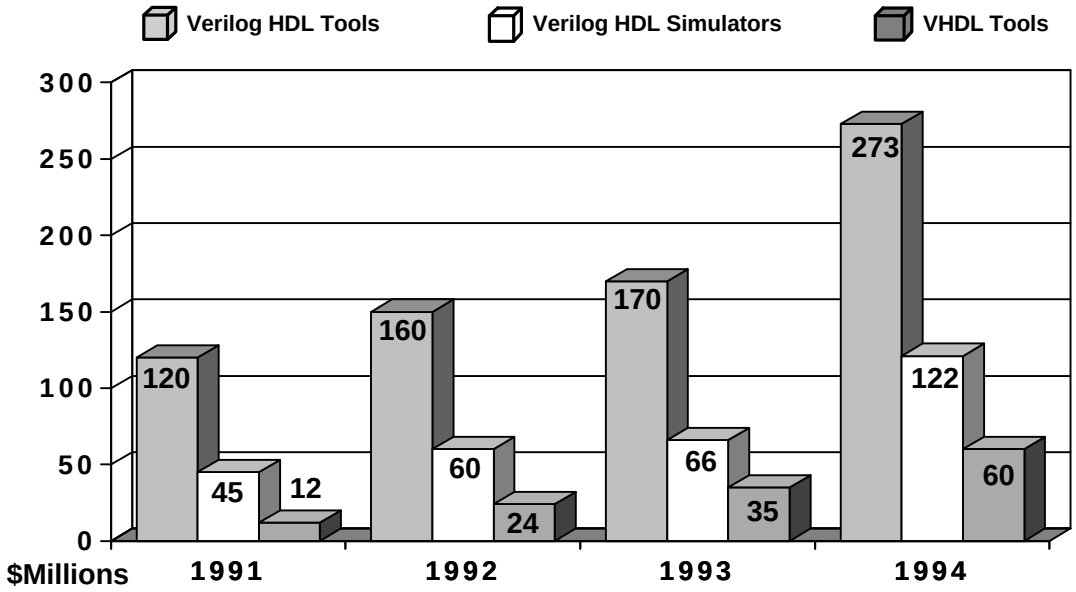
The Deep Submicron Requirement

One of the most important areas that any HDL must address is the area of the future requirements (future usability). Today, most of the large semiconductor companies are implementing or planning to do deep submicron design (less than .35 microns) within the next 12 months. For an HDL to address the needs of deep submicron design, it must absolutely be able to represent low level constructs (gate and potentially switch level models) accurately. This includes the ability to handle the very specific timing requirements for deep submicron. As a new user, although you may not see the immediate need for deep submicron in your designs, the ASIC and FPGA vendors are busy working at making this part of their next generation silicon, therefore its important to keep pace with this future trend.

Business Decision

The two languages have an entirely different appeal to different sectors of the market. First, VHDL comes from a more academic sphere of influence. Verilog HDL comes from industry and is therefore more suited to "*time to market*" based design requirements. Verilog HDL's dominance in the commercial market sector is highlighted by the great disparity in dollars spent on EDA tools from year to year. The following graph illustrates the distribution in dollars for simulation products and complete tools for each of the languages during the last four years.

Verilog HDL / VHDL Revenue Growth



Source: OVI 1995

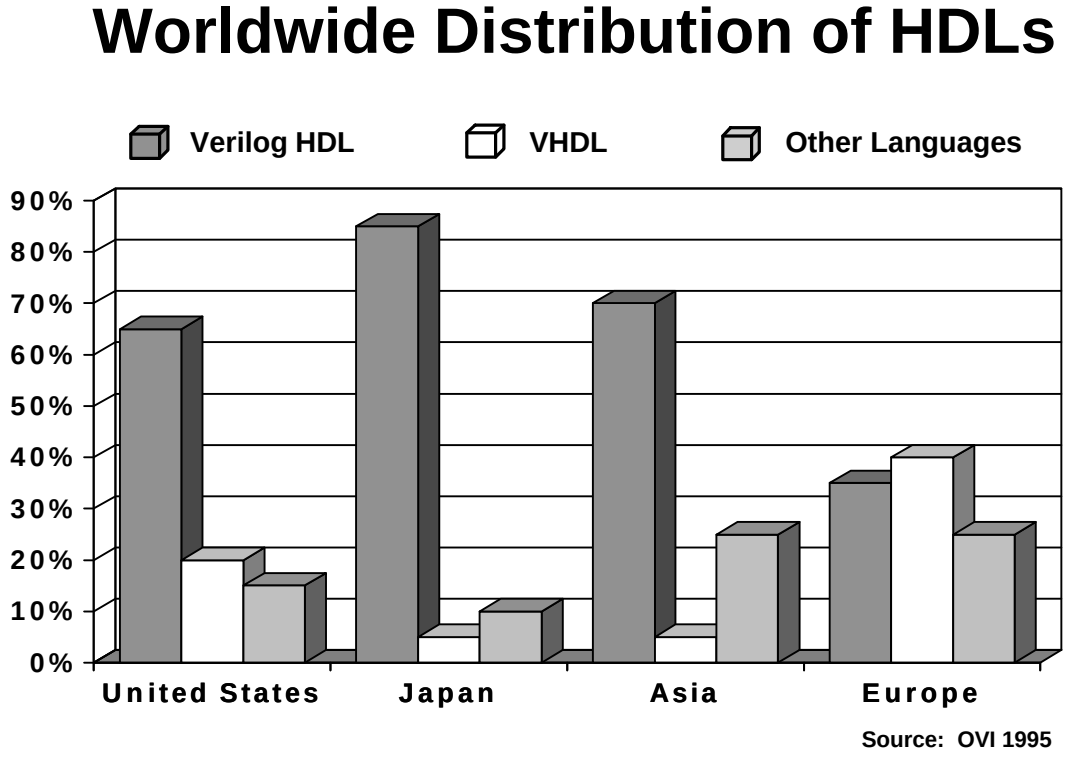
The HDL market worldwide differs in its usage of HDLs. Traditionally the US and Japan are market leaders, setting the future trends for technology throughout the world. The US and Japan established their respective HDL-based design methodologies using Verilog HDL. Asia, including Korea and Taiwan followed closely but Europe took a different path than the rest of the world with respect to HDLs.

The difference between the European market and the rest of the world is easy to understand. First, neither Gateway Design, or Cadence focused on the European market. Second, Europe is a very different market with such close borders, language diversity, different business philosophies and different business cultures. Due to this diversity and the close geopolitical borders,

Europe has developed a history and a practice of adopting formal standards. In their desire to adopt standards, they have often adopted standards that are in various stages of completeness. When it came time to adopt an HDL standard, European electronics companies, for the most part, selected VHDL. The only practical reason for this was that VHDL was an emerging IEEE standard, while Verilog HDL was the proprietary property of Cadence (Gateway).

Nonetheless, today, electronics companies can not take a design from concept to production using VHDL. For those electronics companies that have to take a design to completion, they must use Verilog HDL after synthesis. Only in this way can they complete their design process because after synthesis the design will absolutely require gate level detail and the loading of complete conditional timing information. Only the larger electronics firms can afford to use two HDLs can justify their original decision to use VHDL.

The following graph illustrates the distribution of HDLs worldwide as of 1995.



Conclusion

To summarize, VHDL was developed (for the US DOD) to provide a consistent modeling language for the documentation of digital hardware designs. The language was never intended to be used to do actual design. However, to maintain a *supposed* competitive advantage, individual EDA companies exerted

considerable influence, resources and dollars to force the language to become a design language. These same EDA companies implemented their own semi-unique versions of the language at different stages during its development. This means VHDL models that were developed on one system, may not run on a different system. The language is difficult to learn and even more difficult to use. It is extremely verbose, especially at the gate level, when timing information is specific and considerable. VHDL's verbosity causes severe memory problems when trying to simulate medium to large designs. ASIC vendors have been very reluctant to provide VHDL gate level libraries that include full timing because of the size of the models and the abnormally long simulation times associated with validating a relatively simple design. The framers of VHDL were driven by the US DOD, which has no material interest in design productivity. VHDL's complex syntax interferes with design productivity and does not offer any strategic advantage that would improve the quality of the design. This essentially undermines the basic strength of VHDL, productivity achieved via a methodology based on top-down-design.

Verilog HDL has been developed and will continue to evolve to address the needs and commercial applications of the design community that has made it the most successful language in use today. The design community has invested almost 20 billion dollars in Verilog HDL and related tools over the last 8 years. The ability to address higher level language constructs are well supported in the language, along with its rock solid structural (gate and switch level) strengths. As long as designers and their companies have to get high quality innovative products to market in the time sensitive world in which we all compete, Verilog HDL will continue to be the dominant solution. Almost every major computer manufacturer, system developer, ASIC and semiconductor manufacturer uses Verilog HDL as their modeling language.

For the first time HDL user the selection of Verilog HDL as your modeling language will be a very wise decision. It will mean there are a number of tools available from schematic entry to synthesis to simulation at various price ranges and on numerous platforms from PCs to mainframes. There are also numerous libraries available from a variety of sources that support full timing based models with all the necessary delay functionality required to meet your critical design needs. There is also a vast resource of Verilog HDL engineering talent that has had experience using the language for practical commercial design to provide critical assistance if it becomes necessary. There are many HDLs you can choose from but only one that has proven time and time again that it is the only choice for real designs.