**An Introduction to VHDL**

**Introduction**

VHDL (VHSIC Hardware Description Language) is becoming increasingly popular as a way to capture complex digital electronic circuits for both simulation and synthesis. Digital circuits captured using VHDL can be easily simulated, are more likely to be synthesizable into multiple target technologies, and can be archived for later modification and reuse.

In his introduction to A VHDL Primer (Prentice Hall, 1992), Jayaram Bhasker writes, "VHDL is a large and complex language with many complex constructs that have complex semantic meanings...". This statement, with its possibly record-breaking three instances of the word "complex", reflects a common and for the most part correct perception about VHDL: it is a large and complicated language.

VHDL is a rich and powerful language. But is VHDL really so hard to learn and use? VHDL is not impenetrable, if you follow well-established coding conventions and borrow liberally from sample circuits such as those found in this introduction.

**A Brief History Of VHDL**

VHDL (which stands for VHSIC Hardware Description Language) was developed in the early 1980s as a spin-off of a high-speed integrated circuit research project funded by the U.S. Department of Defense. During the VHSIC program, researchers were confronted with the daunting task of describing circuits of enormous scale (for their time) and of managing very large circuit design problems that involved multiple teams of engineers. With only gate-level design tools available, it soon became clear that better, more structured design methods and tools would need to be developed.

To meet this challenge, a team of engineers from three companies -- IBM, Texas Instruments and Intermetrics -- were contracted by the Department of Defense to complete the specification and implementation of a new, language-based design description method. The first publicly available version of VHDL, version 7.2, was made available in 1985.

**IEEE Standard 1076-1987:-** In 1986, the IEEE was presented with a proposal to standardize the language, which it did in 1987 after substantial enhancements and modifications were made by a team of commercial, government and academic representatives. The resulting standard, IEEE 1076-1987, is the basis for virtually every simulation and synthesis product sold today. An enhanced and updated version of the language, IEEE 1076-1993, was released in 1994, and VHDL tool vendors have been responding by adding these new language features to their products.

**IEEE Standard 1164:-** Although IEEE standard 1076 defines the complete VHDL language, there are aspects of the language that make it difficult to write completely portable design descriptions (descriptions that can be simulated identically using different vendors' tools). The problems stems from the fact that VHDL supports many abstract data types, but does not directly address the problem of characterizing different signal strengths or commonly used simulation conditions such as unknowns and high- impedence. Soon after IEEE 1076-1987 was adopted, simulator companies began enhancing VHDL with new signal types (typically through the use of syntactically legal, but non-standard enumerated types) to allow their customers to accurately simulate complex electronic circuits. This causes problems because design descriptions entered using one simulator were often incompatible with other simulation environments. VHDL was quickly becoming a non-standard.

To get around this problem of non-standard data types, another standard was developed by commitee and adopted by the IEEE. This standard, numbered 1164, defines a standard package (a VHDL feature that allows commonly used declarations to he collected into an external library) containing definitions for a standard nine-valued data type. This standard data type is called standard logic, and the IEEE 1164 package is often referred to as the standard logic package, or sometimes MVL9 (for mutivalued logic, nine values).

The IEEE 1076-1987 and IEEE 1164 standards together form the complete VHDL standard in widest use today. (IEEE 1076-1993 is slowly working its way into the VHDL mainstream, but does not add significant new features for synthesis users.)

**VITAL Initiative:-** The VITAL initiative (VHDL Initiative Toward ASIC Libraries) is an effort to enhance VHDL's abilities for modeling timing in ASIC and FPGA design environments. VITAL borrows liberally from existing methods for timing annotation used in Verilog HDL. Specifically, the VITAL standard (standard 1076.4, which as of this writing is in balloting phase) describes a method for annotating delay information using the same underlying tabular format as specified in Verilog. The adoptance of this standard will make it much easier for ASIC and FPGA vendors to create timing-annotated netlists and other data describing the detailed behavior of their devices.

**What Is VHDL?**

VHDL is a programming language that has been designed and optimized for describing the behavior of digital circuits and systems. As such, VHDL combines features of the following:

**A Simulation Modeling Language:-** VHDL has many features appropriate for describing (to an excruciating level of detail) the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. Features of VHDL allow electrical aspects of circuit behavior (such as rise and fall times of signals, delays through gates, and functional operation) to be precisely described. The resulting VHDL simulation models can then be used as building blocks in larger circuits (using schematics, block diagrams or system-level VHDL descriptions) for the purpose of simulation.

**A Design Entry Language:-** Just as high-level programming languages allow complex design concepts to be expressed as computer programs, VHDL allows the behavior of complex electronic circuits to be captured into a design system for automatic circuit synthesis or for system simulation. Like Pascal, C and C++, VHDL includes features useful for structured design techniques, and offers a rich set of control and data representation features. Unlike these other programming languages, VHDL provides features allowing concurrent events to be described. This is important because the hardware being described using VHDL is inherently concurrent in its operation. Users of PLD programming languages such as PALASM, ABEL, CUPL and others will find the concurrent features of VHDL quite familiar. Those who have only programmed using software programming languages will, however, have some new concepts to grasp.

**A Test Language:-** One of the most important (and under-utilized) aspects of VHDL is its ability to capture the performance specification for a circuit, in a form commonly refered to as a test bench. Test benches are VHDL descriptions of circuit stimulus and corresponding expected outputs that verify the behavior of a circuit over time. Test benches should be an integral part of any VHDL project, and should be created in parallel with other descriptions of the circuit.

**A Netlist Language:-** VHDL is a powerful language with which to enter new designs at a high level, but it is also useful as a low-level form of communication between different tools in a computer-based design environment. VHDL's structural language features allow it to be effectively used as a netlist language, replacing (or augmenting) other netlist languages such as EDIF.

**A Standard Language:-** One of the most compeling reasons for you to become experienced with and knowledgable in VHDL is its adoptance as a standard in the electronic design community. Using a standard language such as VHDL will virtually guarantee that you will not have to throw away and re- capture design concepts simply because the design entry method you have chosen is not supported in a newer generation of design tools. Using a standard language also means that you are more likely to be able to take advantage of the most up-to-date design tools, and will have access to a knowledge-base of thousands of other engineers, many of who are solving problems similar to your own.

**How Is VHDL Used?**

VHDL is a general-purpose programming language optimized for electronic circuit design. As such, there are many points in the overall design process at which VHDL can help.

**For design specification:-** VHDL can be used right up front, while you are still designing at a high level, to capture the performance and interface requirements of each component in a large system. This is particularly useful for large projects involving many team members. Using a top-down approach to design, a system designer may define the interface to each component in the system, and describe the acceptance requirements of those components in the form of a high-level test bench. The interface definition (typically expressed as a VHDL entity declaration) and high-level performance specification (the test bench) can then be passed on to other team members for completion or refinement.

**For design capture:-**  Design capture is that phase in which the details of the system are entered (captured) in a computer-based design system. In this phase, you may express your design (or portions of your design) as schematics (either board-level or purely functional) or using VHDL descriptions. If you are going to be using synthesis technology, then you will want to write the VHDL portions of the design using a style of VHDL that is appropriate for synthesis.

The design capture phase may include tools and design entry methods other than VHDL. In many cases, design descriptions written in VHDL are combined with other representations, such as schematics, to form the complete system.

**For design simulation:-** Once entered into a computer-based design system, you will probably want to simulate the operation of your circuit to find out if it will meet the functional and timing requirements developed during the specification process. If you have created one or more test benches as a part of your design specification, then you will use a simulator to apply the test bench to your design as it is written for synthesis (a functional simulation) and possibly using the post-synthesis version of the design as well.

**For design documentation:-** The structured programming features of VHDL, coupled with its configuration management features, make VHDL a natural form in which to document a large and complex circuit. The value of using a high-level language such as VHDL for design documentation is pointed out by the fact that the U.S. Department of Defense now requires VHDL as the standard format for communicating design requirements between government subcontractors.

**As an alternative to schematics:-** Schematics have long been a part of electronic system design, and it is unlikely that they will become extinct anytime soon. Schematics have their advantages, particularly when used to depict circuitry in block diagram form. For this reason many VHDL design tools now offer th ability to combine schematic and VHDL representations in a design.

**As an alternative to proprietary languages:-** If you have used programmable logic devices in the past, then you have probably already used some form of hardware description language (HDL). Proprietary languages such as PALASM, ABEL, CUPL and Altera's AHDL have been developed over the years by PLD device vendors and design tool suppliers, and remain in widespread use today. In fact, there are probably more users of PLD-oriented proprietary languages in the word today than all other HDLs (including Verilog and VHDL) combined.