**The Basics of VHDL**

INTROUDCTION TO UHDL

VHDL (VHSIC (Very high Speed Inegrated Circuit) Hardware Description Kanguage) is one of the industry standard hardware description languages. The goals of this handout are to provide some basic features of the BHDL language and its use in specifying and simulating digital Systems. VHDL features can be learned similar to any other ptogramming languages such as C. Howver, the topics will be presented in all the ,modules and in the laboratories from a design Perspective rather than a language theory perspective. VHDL models of simple circuits will be used to iontroduce feature of the language.

VHDL is a language to describe digital system. VDHDL captures the basicfundamentals and concepts of digital systems. Therefore, learning it will reinforce the theory of digital system design. Addtionally, VDDL (along with Verilog) is used as a standard description for communication between design automation (CAD) tools.

VHDL is hardware description language that can be used to model a digital system at many levels of abstraction, ranging from the algorithmic level to the gate level.

The VHDL language can be regarded as on combination of following language.

Sequential – Language

Concurrent Language

Net list-Language

Timing Specification

Waveform generating Language

**History**

VHDL was originally developed at the behest of the us department of defense in order to document the behavior ogf the asic that supplier companies were including in equipment . that is to say , vhdl was developed as an alternative to huge , complex manuals which were subject to implementation specific details

The idea of being able to simulate this documentation was so obviously attractive that logic simulators were developed that could read the vhdl files . the next step was the development of logic synthetic tols that read the vhdl , and output a definiyion of the physical implementation of the circuit . modern synthetic tolls can extract ram , counter , and arithmetic blocks out iof the code and implement them according to what the user specifies . thus , the same vhdl code be synthesized differently for lowest area , lowest power consumption , highest clock speed , or other requirements

VHDL borrows heavily from the Ada programming language in both concept ( for example the slice notation for indexing part of a one – dimensional array ) and syntax . VHDL has constructs to handle the parallelism inherent in hardware designs , but these constructs ( processes ) defer in syntax from the parallel constructs in Ada ( task ) . like Ada , VHDL is strongly typed and is not case sensitive . there many feature of VHDLl which are not found in Ada , such as an extended set of Boolean operators including **nand** and **no**r in order to directly represent operations which are common in hardware . VHDL also allows arrays to be in either direction ( ascending or descending )because both conventions are used in hardware, whereas Ada (like most programming languages) provides ascending indexing only. The reason for the similarity between the two languages is that the Department of Defense required as much of the syntax as possible to be based on Ada, in order to avoid re-inventing concepts that had already been thoroughly tested ion the development of Ada.

The initial version of BHDL, designed to IEEE standard 1076-1987, included a wide range of data types, including numerical (integer and real), logical (bit and Boolean), character and time, plus arrays of bit called bit\_vector and of character called string.

A problem not solves by this edition, however, was “multi-valued logic”, where a signal’s drive strength (none, weak or strong) and unknown values are also considered. This required IEEE standard 1164, which defined the 9-value logic types: scalar std\_ulogic and its vector version std\_ulogic\_vector.

The second issue if IEEE 1076, in 1993, made the syntax more consistent, allowed more flexibility in naming, extended the character type to allow ISO-8859-1 printable characters, added the xnor operator, etc.

Minor changes in the standard (2000 and 2002) added the idea of protected types (similar to the concept of class in C++) and removed some restrictions from port mapping rules.

In addtition to IEEE standard 1164, several child standards were introduced to extend functionality of the language, IEEE standard 1076.2 added better handling to real and complex data types. IEEE standard 1076.3 introduced signed and unsigned types yto facilitate arithmetical operations on vectors. IEEE standard 1076.1 (known as NHDL-AMS) provided analog and mixed-signal circuit design extensions.

Some others standards support wider user of VHDL, notably VITAL (VHDL Initiative Towards AISC Librarries) and microwave circuit design extension.

In June 2006, VHDL Technical Committee of Accellera (delegated by IEEE to work on next update of the standard) approved so called Draft 3.0 of VHDL-2006. While maintaining full compatibility with older versions, this proposed standard provides numerous extension that make writing and managing VHDL code easier. Key changes include incorporation of child standards (1164, 1076.2, 1076.3) into the main 1076 standard, an extended set of operators, more flexible syntax of ‘case’ and ‘generate’ statements, incorporation of VHPI (interface to C/C++ languages) and a subset of PSL (Property Specification Language). These changes should improve quality of synthesizable VHDL code, make testbenches more flexible, and allow wider use of VHDL for system-level descriptions.

In February 2008, Accellera approved VHDL 4.0 also informally known as VHDL 2008, which addressed more than 90 issues discovered during the trial period for version 3.0 and includes enhanced generic types. In 2008, Accellera released VHDL 4.0 to the IEEE for balloting for inclusion in IEEE 1076-2008. The NHDL standard IEEE 1076-2008 was approved ny REVCOM in September 2008.

**Design** :

VHDL is a fairly general-purpose language, and it doesn’t require a simulator on which to run the code. There are a lot of VHDL compilers, which build executable binaries. It can read and write files on the host computer, so a VHDL program can be written thgaat generates another VHDL program to be incorporated in the design being developed. Because of this general-purpose nature, it is possible to use VHDL to write a testbench that verifies the functionality of the design using files on the host computer to define stimuli, interacts with the user, ands compares results with those expected. VHDL is a strongly typed language.

It is relatively easy for an inexperienced developer to producer code that simulates successfully butthat cannot be synthesized into a real device, or is too large to be practical. One particular pitfall is the accidental productiob of transparent latched rather than D-type flipe-flops as storage elements.

VHDL is not a case sensitive language. One can design hardware in a VHDL IDE (suXilinx or Quartus) to produce the RTL schematic of the desiredcircuit. After that, the generated schematic can be verified using simulation software (such as ModelSim) which shows the waveforms of inputs and outputs of the circuit after generating the appropriate testbench. To generate an appropriate testbench for a particular circuit of VHDL code, the inputs have to be defined correctly. For example, for clock inout, loop process or an iterative statement is required.

The key advantage of VHDL when used for system design is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires).

Another benefit is that VHDL allows the description of a concurrent system (many parts, each with its own sub-behaviour, working together at the same time). VHDL is a Dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially., one instruction at a time.

A final point is that when a VHDL model is translated into the ‘gateds and wores” that are mapped onto a programmable logic such as a CPLD or FPGA, then it is the actual hardware being configured, rather than the VHDL code being ‘executed” as if on some form of a processor chop.

Advantages

The key advantage of VHDL, when used for systems design, is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires).

Another benefit is that VHDL allows the description of a [concurrent system](http://en.wikipedia.org/wiki/Concurrent_system). VHDL is a [dataflow language](http://en.wikipedia.org/wiki/Dataflow_programming), unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, one instruction at a time.

VHDL project is multipurpose. Being created once, a calculation block can be used in many other projects. However, many formational and functional block parameters can be tuned (capacity parameters, memory size, element base, block composition and interconnection structure).

VHDL project is portable. Being created for one element base, a computing device project can be ported on another element base, for example VLSI with various technologies

### Synthesizeable constructs and VHDL templates

VHDL is frequently used for two different goals: simulation of electronic designs and synthesis of such designs. Synthesis is a process where a VHDL is compiled and mapped into an implementation technology such as an FPGA or an ASIC. Many FPGA vendors have free (or inexpensive) tools to synthesize VHDL for use with their chips, where ASIC tools are often very expensive.

Not all constructs in VHDL are suitable for synthesis. For example, most constructs that explicitly deal with timing such as wait for 10 ns; are not synthesizable despite being valid for simulation. While different synthesis tools have different capabilities, there exists a common *synthesizable subset* of VHDL that defines what language constructs and idioms map into common hardware for many synthesis tools. IEEE 1076.6 defines a subset of the language that is considered the official synthesis subset. It is generally considered a "best practice" to write very idiomatic code for synthesis as results can be incorrect or suboptimal for non-standard constructs.

**VHDL program for bcd to seven segment display**

Library ieee;

Use ieee.std\_logic\_1164.all;

Use ieee.std\_logic\_arith.all;

Use ieee.std\_logic\_unsigned.all;

Entity bcd\_u is

Port ( sel : in std\_logic\_vector(3 downto 0 );

Q : out\_logic\_vector(6 doento o ));

End bcd\_u

Architecture Behavioural of bcd\_u is

Begin

With sel select

Q<=”1111110”when”oooo”

“0110000”when”0001”,

“1101101”when”0010”,

“1111001”when”0011”,

“0110011”when”0100”,

“1011011”when”0101”,

“1011111”when”0110”,

“1110000”when”0111”,

“1111111”when”1000”,

“1111011”when”1001”,

“0000000”when others;

End behavioral;