A Review: Digital System Design by VHDL

Manoj Mali
Asst.Prof
Electronics & Communication Department,
Shri USB College of Engineering, Abu Road
Manoj3045@gmail.com

Himanshu Trivedi
M Tech Student
Electronics & Communication Department,
Geetanjali Institute of Technical Studies, Udaipur
Himanshu22693trivedi@gmail.com

***Abstract* - VHDL is the VHSIC Hardware Description Language. VHSIC is an abbreviation for Very High Speed Integrated Circuit. It can describe the behaviour and structure of electronic systems, but is particularly suited as a language to describe the structure and behaviour of digital electronic hardware designs, such as ASICs and FPGAs as well as conventional digital circuits.**
VHDL is a notation, and is precisely and completely defined by the Language Reference Manual (LRM). This sets VHDL apart from other hardware description languages, which are to some extent defined in an ad hoc way by the behaviour of tools that use them. VHDL is an international standard, regulated by the IEEE. The definition of the language is non-proprietary.
VHDL is not an information model, a database schema, a simulator, a toolset or a methodology! However, a methodology and a toolset are essential for the effective use of VHDL.
Simulation and synthesis are the two main kinds of tools which operate on the VHDL language. The Language Reference Manual does not define a simulator, but unambiguously defines what each simulator must do with each part of the language.
VHDL does not constrain the user to one style of description. VHDL allows designs to be described using any methodology - top down, bottom up or middle out! VHDL can be used to describe hardware at the gate level or in a more abstract way. Successful high level design requires a language, a tool set and a suitable methodology. VHDL is the language, you choose the tools, and the methodology

Keywords - VHDL, Data Flow, Behavioural, Structural

I. Introduction

VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design. This collection of simulation models is commonly called a *test bench*.

VHDL has constructs to handle the parallelism inherent in hardware designs, but these constructs (*processes*) differ in syntax from the parallel constructs in Ada (*tasks*). Like Ada, VHDL isstrongly typed and is not case sensitive. In order to directly represent operations which are common in hardware, there are many features of VHDL which are not found in Ada, such as an extended set of Boolean operators including **nand** and **nor**. VHDL also allows arrays to be indexed in either ascending or descending direction; both conventions are used in hardware, whereas in Ada and most programming languages only ascending indexing is available.

VHDL has file input and output capabilities, and can be used as a general-purpose language for text processing, but files are more commonly used by a simulation test bench for stimulus or verification data. There are some VHDL compilers which build executable binaries. In this case, it might be possible to use VHDL to write a test bench to verify the functionality of the design using files on the host computer to define stimuli, to interact with the user, and to compare results with those expected. However, most designers leave this job to the simulator.

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

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).

II. Different modeling style

A. *Data Flow Style*

– It describes how the data flows from the inputs to the output most often using NOT, AND and OR operations.

Example:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity and_gate is
    port(
        a : in STD_LOGIC;
        b : in STD_LOGIC;
        dout : out STD_LOGIC
        );
```

```
end and_gate;

architecture and_gate_arc of and_gate is
begin

   dout <= a and b;

end and_gate_arc;
```

B. ***Behavioural Style***

It describes how the output is derived from the inputs using structured

Example:

File : JK Flip Flop.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity jk_flip_flop is
    port(
        j : in STD_LOGIC;
        k : in STD_LOGIC;
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        q : out STD_LOGIC;
        qb : out STD_LOGIC
        );
end jk_flip_flop;

architecture jk_flip_flop_arc of jk_flip_flop is
begin

   jkff : process (j,k,clk,reset) is
   variable m : std_logic := '0';
   begin
      if (reset='1') then
         m := '0';
      elsif (rising_edge (clk)) then
         if (j/=k) then
            m := j;
         elsif (j='1' and k='1') then
            m := not m;
         end if;
      end if;
      q <= m;
      qb <= not m;
   end process jkff;
   end jk_flip_flop_arc;
```

C. ***Structural Style***

It describes how gates are interconnected similar to schematic approach.

Example:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity adder_4bit is
     port(
         a : in STD_LOGIC_VECTOR(3 downto 0);
         b : in STD_LOGIC_VECTOR(3 downto 0);
         carry : out STD_LOGIC;
         sum : out STD_LOGIC_VECTOR(3 downto 0)
         );
end adder_4bit;

architecture adder_4bit_arc of adder_4bit is

Component fa is
    port (a : in STD_LOGIC;
          b : in STD_LOGIC;
          c : in STD_LOGIC;
          sum : out STD_LOGIC;
          carry : out STD_LOGIC
          );
end component;

signal s : std_logic_vector (2 downto 0);

begin

    u0 : fa port map (a(0),b(0),'0',sum(0),s(0));
    u1 : fa port map (a(1),b(1),s(0),sum(1),s(1));
    u2 : fa port map (a(2),b(2),s(1),sum(2),s(2));
    ue : fa port map (a(3),b(3),s(2),sum(3),carry);

end adder_4bit_arc;
```

III. ADVANTAGES & CONCLUSIONS

In this paper, we introduced VHDL programming and all the different modeling style

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. 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 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).

A 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.

REFERENCES

[1] Why should I care about Transparent Latches?". Doulos. Retrieved 22 December 2012..
[2] "Clock Generation". Doulos. Retrieved 22 December 2012.
[3] Peter J. Ashenden, "The Designer's Guide to VHDL, Third Edition (Systems on Silicon)", 2008,
[4] Bryan Mealy, Fabrizio Tappero (February 2012). Free Range VHDL
[5] www.coel.ecgf.uakron.edu
[6] ww.w.vhdlbynaresh.blogspot.in