

A HIGH-LEVEL VHDL SIMULATOR

E. Casino, P.Sánchez and E.Villar
Dept. of Electronics, University of Cantabria
Avda. Los Castros s/n, 39005 Santander, Spain

Abstract

High-level synthesis is a mature research area with a high industrial potential and high-level systems are seen to be an emerging key technology in digital design. VHDL, being standard, may facilitate the acceptance of these new CAD tools. Although VHDL is a very rich and powerful description language at the RT and logic levels, it presents some limitations when it is used at high level. In this paper a description is given of the high-level simulator SIVa. It accepts the same VHDL descriptions and the same semantics as PSAL2, the high-level synthesis tool developed at our department, thus overcoming the problems caused by using a logic simulator at the high level. High-level simulation represents a first step in the development of multi-level simulators to cover all levels from high to logic level.

Section 1. Introduction

VHDL allows the modeling of hardware using structural, data-flow and behavioral descriptions. Thus, the language can represent the design at several stages of the synthesis process. For this reason, VHDL offers the possibility of a unified multi-level user interface, where the same language is used to represent the system at different design steps.

As automation of design stages progresses from layout to logic level, the increase in complexity of VLSI circuits means the effort of design shifts to higher levels of abstraction, thus increasing the demand for high level synthesis tools [1]. These tools obtain the digital system structure which best implements the desired behavior while satisfying the set of constraints imposed by the designer. The principal advantages of high-level synthesis are a reduction in the design time, a reduction in the number of errors by means of the formalization and automation of the design process, and the possibility of a rapid evaluation of multiple design alternatives as well as a useful documentation of the system. High-level synthesis is a mature research area with a high industrial potential [2] and high-level systems are seen to be an emerging key technology in digital design [3]. VHDL, being standard, may facilitate the acceptance of these new CAD tools.

The input description to the high-level synthesis tool uses the sequential statements of a process to describe the algorithm performed by the system. Only the data dependencies are relevant and no kind of timing information is needed [4-6].

Although VHDL was developed initially as an event-driven simulation language for modeling purposes, its flexibility and the strong demand for a standard design language have forced its use outside this initial scope. The two main directions in which attempts are being made to extend VHDL are new applications and new abstraction domains. Of the new applications, synthesis and formal verification stand out. With regard to new

abstraction domains, it is intended that the language cover analog description and simulation at the electric level and high-level description and simulation at algorithmic level.

Although VHDL is a very rich and powerful description language at the RT and logic levels, it presents some problems when it is used at high-level. One of the problems to solve is simulation because a logic event-driven simulator (the canonical VHDL simulator) has to be used in order to verify the abstract behavior of the input algorithm, in which time is meaningless.

In high-level simulation the data dependencies described in the algorithmic description have to be verified. The simulator has to execute the sequential statements one by one. Each statement corresponds to an execution step [6] without any timing information. Only the data transformations between objects (variables, signals and constants) contained in the statement have to be performed.

In this paper, section 2 describes the high-level simulation methodology, whose application to multi-level simulation is commented on in section 3. Section 4 then presents a description of the newly-developed high-level simulator SiVa, which accepts the same VHDL descriptions and the same semantics as PSAL2, the high-level synthesis tool developed at our department [5]. Finally, a summary of some results is given in section 5.

Section 2. High-level simulation

Let us suppose the VHDL algorithmic description of the digital filter of Figure 1. This description could be used as input for a high-level synthesis tool which would produce a register transfer architecture.

In order to verify the correctness of the initial description, it has to be simulated. However, the high-level description cannot be simulated directly with the canonical logic VHDL simulator because it uses delta delay timing and, in some cases, no signal is assigned a value. Furthermore, different values are generally required each time an input signal is used and, at the same time, an output may be assigned different values in the same δ cycle.

If input and output signals are used in the ports of the entity and they are read and written in the process, some kind of handshaking technique using wait statements can be used in order to avoid the delta delay timing. This approach has been proposed in several papers [7-8] and provides an apparently correct result, at least from a functional point of view. Suppose, for instance, that two wait statements are added to the description in Figure 1, as shown in Figure 2. The corresponding logic simulation might be that of Figure 3.

A problem appears when the code between two wait statements is scheduled. The time between them was incorrectly simulated and there is no match either in time or in clock cycles with the simulation at register-transfer level. This should be avoided in high-level simulation and has to be resolved in multi-level simulation.

The cause of the mismatch is that at the high level, the domain timing specifications for a circuit may generally be known, but the functional timing specifications of the element are not yet known. Any time (absolute time or number of clock cycles) assigned to them arbitrarily will be incorrect. In addition, handshaking is always asynchronous and most of the complex digital systems currently being developed are synchronous.

In the previous example, the functional delay between 'go' taking the value '1' and the output being assigned the value '-60' has been evaluated as 0ns. That is obviously

incorrect since the code between the reading of the input value and the writing of the output result requires several clock cycles.

A first solution is based on the modification of the description used for synthesis so that it can be simulated with the canonical logic simulator. In a previous paper [5], the simulation of high-level behavior using commercial VHDL simulators was analyzed and some strategies were suggested. A more complete solution is based on the development of a specific high-level simulator.

```

entity dfilter is
  port ( input: in integer;
         output: out integer);
end dfilter;

architecture behavior of dfilter is
begin
  p1: process
    type coef_arr is array (integer range 1 to 12) of integer;
    constant c: coef_arr := (0,1,2,3,4,5,6,7,8,9,10,11); -- coefficient array
    variable s1, s2, s3, s4, s5, s6: integer; -- cstate variables
    variable x1, x2, x3, x4, x5, x6: integer; -- temporary variables
    variable v1, v2: integer; -- common subexpression variables
  begin
    s1 := 0;
    s2 := 0;
    s3 := 0;
    s4 := 0;
    s5 := 0;
    s6 := 0;
    loop
      x0 := input;
      x1 := s1;
      x2 := s2;
      x3 := s3;
      x4 := s4;
      x5 := s5;
      x6 := s6;
      v1 := x4 + x2 - (x0 - x1)*c(7);
      v2 := x5*c(7) + x6 - x4;
      s1 := v1*c(12) + v1*c(12) + x1;
      s2 := x2 + x5*c(4) - x5*c(2);
      s3 := x4*c(8) + x3;
      s4 := (x1 - x3 - x5)*c(6) + x4;
      s5 := v1*c(11) + v2*c(10) + x5;
      s6 := x1*c(3) - x5*c(1) - x6;
      output <= s5;
    end loop;
  end process;
end behavior;

```

Figure 1: A VHDL high-level description

The main difference between the high-level and the logic simulator concerns the way in which signals are assigned values. In high-level simulation, signals are to be treated in the same way as variables. No unnecessary wait statements have to be included.

```

entity dfilter is
  port ( input: in integer;
        output:out integer;
        go:   in bit);
end dfilter;

architecture behavior of dfilter is
.
.
.
  loop
    wait until go = '1';
    x0 := input;
.
.
.
    output <= s5;
    wait until go = '0';
  end loop;
.
.
.

```

Figure 2: A high-level description with wait statements

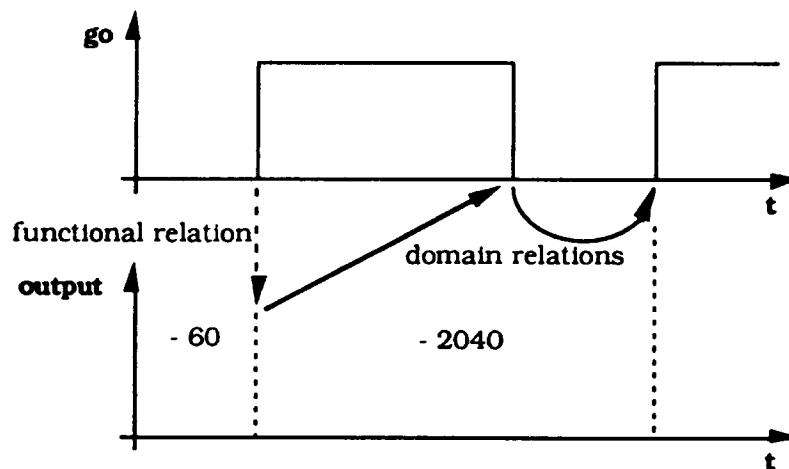


Figure 3: Simulation result

Each time an input value is needed, the simulator will read it from an external sequence generator. This may occur in the same δ cycle. The portion of sequential code in which an input signal has to maintain its value should be defined by the user. Output signals can also be assigned different values during the same δ cycle. The corresponding sequence of outputs will be produced. Whenever the user so desires, the execution can be stopped and the values in the variables and signals displayed.

This requires a modification in the simulation kernel which can be achieved by adapting the signal assignment functions in the C code generated from the VHDL description. The input description is compiled by the CLSI's VTIP analyzer [9], which generates the machine-processable descriptions in the Design Library System (DLS). From the DLS descriptions, the retargetable VHDL code generator (RVCG) [10-11]

generates C code that implements the dynamic semantics of VHDL descriptions. The C program is compiled and linked with the simulator kernel libraries which has been modified for high-level synthesis. This provides a very flexible workbench where new simulation algorithms can be developed.

The high-level simulator will allow the verification of the algorithmic description prior to the synthesis process. This simulation becomes especially important in a complete silicon compilation design system where the integrated circuit is obtained directly from the high-level description.

Section 3. Multi-level simulation

The extension of VHDL to description and simulation at high level is very similar to the extension of the language down to the description and simulation of circuits at the electrical level. In both cases it is possible to think of the description of the analog circuit or the high-level algorithm using entities written in standard VHDL but with different semantics. These subparts interact with the fully standard part through the interface of the entities but the internal behavior and simulation algorithms can now be different.

The main difference between simulation at different abstraction levels concerns how time is handled. In electric simulation the differential equations which define the relations between voltages and currents in the circuit are resolved in a continuous time simulation. The logic abstraction provides a mechanism to reduce the valid signal values to a small set (0, 1, X, Z, etc.) and the time employed by the signal to change from one value to another is modeled by discrete delays. At the RT level a functional simulation is carried out and the delays are not taken into account. At logic and RT level, event-driven, discrete-time simulation is performed. At the algorithmic level, time is meaningless and can be ignored; only the data dependencies are relevant. At this high level there is no relation between the evolution of the values of the variables and time. At the RT level, there is an evolution of the values of the signals in time. The changes correspond to the active transitions of the clock signal and no delay is assumed. At the logic level, discrete delays are usually taken into account. At the electric level, the signal varies in a continuous time over a continuous range of values.

The extension of VHDL up to high-level description and simulation as well as down to the analog level would allow VHDL to be truly used as the multilevel representation in the evaluation of the design from the high-level algorithm down to the electric circuit. Multilevel analysis aids, such as design verification tools or multilevel simulators could be developed. The latter would allow designers to exploit the advantages of high-level simulation (speed and ability to handle complexity) and low-level simulation (accuracy) [6]. They would also allow optimization of the implementation of a hardware algorithm in order to improve its behavior as a part of a previously designed system with fixed timing behavior.

Section 4. SIVA: The high-level VHDL simulator

As commented before, SiVa has been developed from the kernel of a canonical logic VHDL simulator [10-11]. From this base, several modifications and extensions have been made which allow direct simulation of descriptions used as input in high-level synthesis.

The most important modification affects the way values are assigned to signals. All action of the driver has been eliminated so that the value of the signal changes immediately assignment is made. This, in fact, eliminates all time relations between the different assignments to the signal but maintains data dependencies.

Another important change affects the way the simulator processes the sequence of values in the input signals and generates the sequence of values in the output signals. Both inputs and outputs have to be signals and are declared in the port_clause of the only permitted entity.

Data input is carried out each time a signal is read. The value is taken from an input file previously prepared by the user and declared in the simulator command mode with the order:

```
input "file_name".
```

The input file is a text file with the following format:

```
#signal input_signal_1:
  value_1;
  value_2;
  value_3;
  ....
#endsignal

#signal input_signal_2:
  value_1;
  ....
#endsignal

  ....
  ....

#end
```

where input_signal_n is the name of each of the input signals and the value_n is one of the values it takes.

The input file is compiled (by the ic program) and produces a file in a format that can be interpreted by the simulator. This compiled file is the file that must be declared with the input order of the simulator command mode.

When no values are found for a given signal, either because they have all been used or because the signal is not included in the input file or this file does not exist, the simulator requires the user to introduce a valid value for the signal. Should this not be done, the signal will remain at the previous value. If there are no remaining values for a given signal, the following command can be used:

```
restore signal1 signal2
```

which resets the list of values for each of the signals indicated. If no signal is specified, the command affects all signals in the file.

The output sequence is stored in the file declared by the command:

```
output "file_name"
```

If none is specified, the standard output is taken by default. The file is updated each time an input is read or a value is assigned to an output signal, and a line is added with the name of the signal and its new value. If an error occurs in an input, this is also reflected in the output file.

The simulator has been provided with a command, "assign object=value", which allows assignment of values to signals and variables at any moment in the stepwise simulation. This command, together with "examine object", both of which are included in the logic simulator, are particularly useful in the debugging of VHDL programs.

There are a number of constraints in the high-level VHDL description that SiVa is capable of simulating. The most important affect the use of VHDL itself in high-level synthesis [4-5]; others, the type of VHDL description accepted by the logic simulator used in the development of SiVa.

Section 5. Results

The current version of SiVa has been developed in C on a SUN Sparc WS. In order to evaluate its characteristics, the digital filter shown in Figure 1 was used as input for the high-level simulation. The input sequences used are those given in Figure 4:

```
#signal input:
    1;
    34;
    -13;
#endsignal

#signal go:
    '0';
    '0';
    '0';
    '1';
    '0';
    '0';
    '1';
#endsignal

#end
```

Figure 4: Input file

The corresponding output file is shown in Figure 5.

Conclusions

The high-level simulator SiVa, has been described. It accepts the same VHDL descriptions and the same semantics as PSAL2, the high-level synthesis tool developed at our department, thus overcoming the problems caused by using a logic simulator at the high-level. SiVa facilitates the verification of the algorithmic description prior to the synthesis process. This simulation becomes especially important in a complete silicon compilation design system where the integrated circuit is obtained directly from the high-level description and represents a first step in the development of multi-level simulators to cover all levels from high to logic level.

```

Input signal 'go':
'0'
Input signal 'go':
'0'
Input signal 'go':
'0'
Input signal 'go':
'1'
Input signal 'input':
1
Output signal 'output':
-60
Input signal 'go':
'0'
Input signal 'go':
'0'
Input signal 'go':
'1'
Input signal 'input':
34
Output signal 'output':
-2040

```

Figure 5: Output file

References

- [1] R. Camposano: "From behavior to structure: high-level synthesis", IEEE Design & Test of Computers, October, 1990, pp. 8-19.
- [2] R. C. Sarma; M.D. Dooley; N.C. Newman & G. Hetherington: "High-level synthesis: Technology transfer to industry", Proc. of the 27th ACM/IEEE Design Automation Conference, 1990, pp.549-553.
- [3] G. de Michell: "High-level synthesis of digital circuits", IEEE Design & Test of Computers, October, 1990, pp. 6-7.
- [4] R. Camposano; L.F. Saunders and R.M. Tabet: "VHDL as input for high-level synthesis", IEEE Design & Test of Computers, March, 1991, pp. 43-49.
- [5] L. Berrojo; P. Sanchez & E. Villar: "High-level synthesis and simulation with VHDL", Proc. of the Second European Conference on VHDL, Stockholm, September 8-11, 1991, pp.62-69.
- [6] D.E. Thomas & J.A. Nestor: "Defining and implementing a multilevel design representation with simulation applications", IEEE Trans. on Computer -Aided Design, V.CAD-2, N.3, July, 1983, pp.135-145.
- [7] A. Postula: "VHDL specific issues in high-level synthesis", Proc. of the Second European Conference on VHDL, Stockholm, September 8-11, 1991, pp.70-77.
- [8] R. Cottrell: "Chip design using Silicon 1076", Proc. of the Second European Conference on VHDL, Stockholm, September 8-11, 1991, pp.306-312.
- [9] "VHDL analyzer user's manual", CLSI, November, 1990.
- [10] "Retargetable VHDL code generator. Product specification", CLSI, June, 1990.
- [11] "Retargetable VHDL code generator. VHDL kernel interface specification", CLSI, October, 1990.