

# On Simulating Parallel Algorithms with VHDL

Stavros Souravlas, Efthimios Kotsialos, Athanasios Margaritis, and Manos Roumeliotis

University of Macedonia, Applied Informatics Department,

156 Egnatia St. 54006, Thessaloniki, GREECE

email:{sourstav,ekots,amarg,manos}@uom.gr

## Abstract

Hardware Description Languages (HDL) like VHDL are widely used to design and simulate with programmable logic devices. Simulation of very large scale integrated digital systems (VLSI) is of great importance as it assures system correctness and maximizes system performance ([4], [5], [9]). The utilization of parallel simulation introduces the problem of how to partition the logic gates and system behaviors of the circuit among the available processors in order to obtain maximum speedup. This paper presents how a series of sequential statements like VHDL *processes* and *signals* can be used to simulate parallel message broadcasts. Processes and signals are the most important VHDL parts for simulation.

## I. INTRODUCTION-RELATED WORK

Hardware Description Languages (HDL) like VHDL are widely used to design and simulate with programmable logic devices. Simulation of very large scale integrated digital systems (VLSI) is of great importance as it assures system correctness and maximizes system performance ([4], [5], [9]). The field of parallel simulation introduces the problem of how to partition the logic gates and system behaviors of the circuit among the available processors in order to obtain maximum speedup [7]. Performance of VHDL simulation is a critical issue in electronic circuit design and is hard to achieve due to the complexity of the language and the different abstraction levels [2].

Several efforts have been made in the field of parallel VHDL simulation. Lungeanu and Shi [4] provided a mechanism for parallel and distributed VHDL simulation using the PDES (parallel discrete-event simulation) paradigm. More specifically, they present how VHDL signals and processes are mapped to PDES to form a VHDL kernel.

Martin et al. [10], focused on the problem of integrating parallel VHDL with a parallel SPICE (analog circuit) simulator called *Xyce* into a common design framework. Willis and Li [17] described VHDL compilation techniques which facilitate parallel or distributed simulation by embedding evaluation scheduling in the emitted code.

Krishnaswamy et al. [8] presented approaches and algorithms for parallelization of compiled event driven VHDL simulations on shared-memory multiprocessors (SMP). Two approaches for parallelizing event driven simulation are presented: the first is based on list scheduling of the graph derived from static simulation while the second approach is based on graph partitioning.

Ghosh [6], [16] presented P<sup>2</sup>EDAS algorithm for managing asynchronous, concurrent, and accurate simulation of behavior models on parallel processors. The P<sup>2</sup>EDAS algorithm represents a significant advancement in that it permits the use of any number of propagation delays for every path between the input and output of every hardware entity.

Price et al. [12] describe how the VHDL hardware description language can be used to aid the analysis, design and implementation of adaptive array beamformers for use in teleconferencing environments using parallel architectures. They state that VHDL provides a number of benefits to designers of such systems including improved vision of algorithm architecture, a faster design cycle and a more parallel and implementation-orientated design.

McBrayer and Wilsey [11] present a method through which parallel logic gates in a VHDL description can be combined into single processes. More specifically they propose to compile parallel processes from a logic circuit into a single combined Logical Process. Thus, instead of  $n$  parallel LPs executing events and exchanging event information, they propose to statically compile the objects together to reduce the number of LPs executing in parallel.

Ramstein et al deal with a CAD tool dedicated to the design and the simulation of specific Array Processor architectures. These architectures are described into a specific notation which includes major characteristics of the VHDL syntax. A preprocessor generates full standard VHDL code describing the behavior of the designed architecture. An original application to image processing is also given: the design of a specific architecture for the extraction of regions of interests.

Schumacher and Nebel [14] offered a solution on how to successfully deal with the anomaly that arises from the use of inheritance in parallel hardware systems. The basic idea was to separate the synchronization code (protocol specification) and the actual behavior of a method. Method guards which allow a method to execute if a guard expression evaluates to true are proposed to model synchronization constraints. It is shown how to implement a suitable re-schedule mechanism for methods as part of the synchronization code to handle the case that a guard expression is evaluated to false.

## II. VHDL SIMULATORS REVIEW

The simulator is a program that models the response of a system to an input stimuli [1]. The most important aspects concerning the simulator are the *processes* and the *driving signals*.

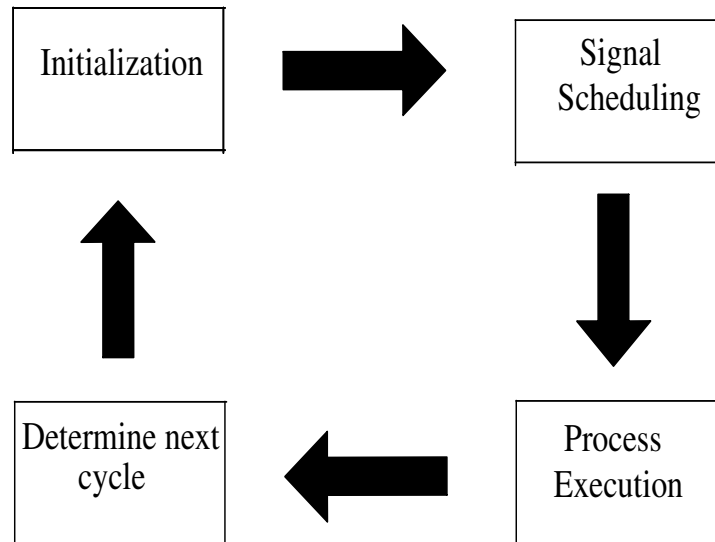


Fig. 1. VHDL Simulation Cycle

A VHDL **process** is a construct for embodying algorithms. A process is used to describe the behavior of a part of the design using a series of sequential statements. In a parallel processor system, communication is performed via message passing; at any instance of time, a number of processors exchange messages, that is, many processes are running. In our modeling approach, a set of parallel running processes will be mapped to a set of VHDL processes. A VHDL process has two states: active or suspended. The processes to be executed are put into a process queue. Once a process is selected for execution, it is put to active state and all the sequential statements inside are executed one by one. As soon as the last statement is executed, the process reaches the "suspended" state. A process becomes active again only when there is a change of value in any of its *driving signals*.

A **driving signal** is a special VHDL element that can cause a process to execute. A process associated with a signal is executed if the signal changes value. The list of signals that may cause a process to run is called **sensitivity list**. It is obvious that more than one signal may cause a process to execute and there may be many processes running, triggered by the same signal. It is important to note that VHDL signals do not change value like variables do in a high-level programming language. A new value is *scheduled* to be assigned to a signal one **delta delay** after the current simulation time. This means that if the simulation time is  $T_c$  then a signal scheduled to change value will not be updated until  $T_{c+\delta}$ . Delta delays can cause the simulator to repeat the series of statements inside a process more than once.

A *simulation cycle* occurs every time the simulation time advances. In every simulation cycle, it is determined which signals are scheduled to change value. These signals will cause the execution of some processes as part of a simulation cycle. During a simulation cycle, a number of delta delays may occur, depending on the number of signals that change value during the cycle.

Sim cycle	$\delta$	$P_0$	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	stg0	stg1	stg2
1	+0	1	1	0	0	0	0	NULL	NULL	NULL
	+1	1	1	0	0	0	0	$P_0, P_1$	NULL	NULL
2	+0	0	0	1	1	0	0	$P_0, P_1$	NULL	NULL
	+1	0	0	1	1	0	0	$P_0, P_1$	stg0	NULL
	+2	0	0	1	1	0	0	$P_2, P_3$	$P_0, P_1$	NULL
3	+0	0	0	0	0	1	1	$P_2, P_3$	$P_0, P_1$	NULL
	+1	0	0	0	0	1	1	$P_2, P_3$	$P_0, P_1$	stg1
	+2	0	0	0	0	1	1	$P_2, P_3$	stg0	$P_0, P_1$
	+3	0	0	0	0	1	1	$P_4, P_5$	$P_2, P_3$	$P_0, P_1$

Fig. 2. Simulation Cycles & Signal updates

As shown in Fig.1, a VHDL simulation cycle consists of the following phases:

- *Initialization phase*: All signals are assigned an initial value.
- *Signal scheduling*: Signals are scheduled future values at specific times.
- *Process execution*: All processes that include in their sensitivity list signals that change values during the current simulation cycle are executed.
- *Determine next cycle*: The simulation time of the next cycle is determined. It is either the time that a signal gets a new value or a delta cycle or multiple delta cycles.

### III. SIMULATING MESSAGE PASSING PATTERNS

We now describe how we can use the VHDL tools to simulate a parallel message passing algorithm. Assume that a communication scheduling algorithm that allows pipeline based processing of messages has been developed. Also, assume that the pipeline strategy has three stages,  $stg0$ ,  $stg1$ , and  $stg2$ . To make a VHDL description for the message passing problem we first need to activate transfer processes by assigned transfer signals and then to identify a set of variables that act as stage registers.

When a transfer process executes, a message from a node is passed to the staging register of the pipeline. Suppose that there are 6 parallel processes that can execute at a time:  $P_0, P_1, P_2, \dots, P_5$ . Each of the transfer processes is sensitive to a change of value that occurs in a signal of an originally defined array of bit or logic\_vector type. We name this array *transfer\_processor\_list*. In other words, this processor list is a bit vector with its values being 1 if the corresponding processor is to send a message during a simulation cycle, or 0 if the corresponding processor remains idle. The decision on the processes that will execute is taken by the scheduling algorithm that successively updates the bit vector. A new simulation cycle begins when the *transfer\_processor\_list* gets a new value or when a previously executed process suspends. Each simulation cycle consists of a number of *delta cycles*. Each delta cycle updates the value of the stage registers.

Figure 2 illustrates the simulation cycles created and the changes occurring in each cycle, for the six processes. Initially, all signals that may activate processes are set to zero. Assume that the algorithm decides that processes  $P_0$  and  $P_1$  must send a message at a time. This means that the 6-bit signal *transfer\_processor\_list* transitions to "110000" (the first two bits change value) during the first simulation cycle. This causes two transfer processes  $P_0$  and  $P_1$  to execute. After a  $\delta$  cycle, the first stage of the pipeline (which is modeled by stage register *stg0*) is updated with its new value, that is, it handles the proper data for the messages from  $P_0$  and  $P_1$ . In Figure 2, we show that by assigning the values  $P_0$  and  $P_1$  to *stg0*.

The scheduling algorithm keeps on generating interprocessor communications. Assume that  $P_2$  and  $P_3$  are scheduled to execute. The value of the signal *transfer\_processor\_list* changes from "110000" to "111100" (see Figure 1) during simulation cycle 2. Stage register *stg1* will now be responsible for the execution of  $P_0$ ,  $P_1$  (previously handled by *stg0*) while stage register *stg0* is responsible for the newly triggered processes. This is shown in Figure 1, by assigning the value *stg0* to *stg1* and the values  $P_2, P_3$  to *stg0*. These value changes occur within two more  $\delta$  cycles.

Finally,  $P_2$  and  $P_3$  are scheduled to execute. The *transfer\_processor\_list* transitions from "111100" to "111111". The last stage register *stg2* now becomes responsible for the communications previously handled by *stg1*, while the stage register *stg1* is responsible for the communications previously handled by *stg0*. These changes require three  $\delta$  delays +0,+1,+2. During the last  $\delta$  delay, the stage registers have been assigned their proper value, that is,  $stg0 \leftarrow P_4, P_5$ ,  $stg1 \leftarrow P_2, P_3$ , and  $stg2 \leftarrow P_0, P_1$ . A VHDL sample code for 3 writing processes is given in Listing 1. Recall that the transfer processor list that is in the sensitivity list of the processes is updated by a message scheduling algorithm.

```

p0: process (transfer_processor_list0)
begin
for i in N downto 0 loop
if transfer_processor_list0(i)='1' then
stg0[i]<=mem[i] after  $\delta$  ns;
end if;
end loop;
end process;
p1: process (transfer_processor_list1)
begin
for i in N downto 0 loop

```

```

if transfer_processor_list1(i)='1' then
stg1[i]<=stage0[i] after  $\delta$  ns;
stg0[i]<=mem[i]after  $\delta$  ns;
end if;
end loop;
end process;
p2: process (transfer_processor_list2)
begin
for i in N downto 0 loop
if transfer_processor_list2(i)='1' then
stg2[i]<=stage1[i]after  $\delta$  ns;
stg1[i]<=stage0[i]after  $\delta$  ns;
stg0[i]<=mem[i]after  $\delta$  ns;
end if;
end loop;
end process;

```

**Listing 1:** VHDL Sample Code of transferring processes

#### IV. CONCLUSIONS

This paper is the beginning of a research on how to model parallel algorithms and applications with a hardware description language like VHDL. We initially focus on producing the VHDL source that describes the necessary operations required by a parallel algorithm. Our key idea is to map parallel executing processes to VHDL processes and use signals drivers for process activation. The signal values are updated by the message passing algorithm.

#### REFERENCES

- [1] James R. Armstrong and F. Gail Gray, "VHDL Design Representation and Synthesis - Second Edition", Prentice Hall PTR, 2000.
- [2] Alessandra Costa, Alessandro de Gloria, Paolo Faraboschi, Mauro Olivieri, "An evaluation system for distributed-time VHDL simulation", *ACM SIGSIM Simulation Digest*, *Proceedings of the eighth workshop on Parallel and distributed simulation*, Volume 24 Issue 1, July 1994.
- [3] Frederic Desprez, Jack Dongarra, Antoine Petitet, Cyril Randriamaro, Yves Robert, "Scheduling Block-Cyclic Array Redistribution", *IEEE Transactions on Parallel and Distributed Systems*, Vol.9, NO.2, February 1998, pp.192-205.
- [4] Dragos Lungeanu, C.J. Richard Shi, "Parallel and Distributed VHDL Simulation", *IEEE Design, Automation and Test in Europe (DATE '00)*, pp. 658, March 27 - 30, 2000 Paris, France.
- [5] Dragos Lungeanu, C.J. Richard Shi, "Distributed Event-Driven Simulation of VHDL-SPICE Mixed-Signal Circuits", *IEEE International Conference on Computer Design: VLSI in Computers & Processors (ICCD'01)*, pp. 0302, September 2001.

- [6] Sumit Ghosh , "P<sup>2</sup>EDAS: Asynchronous, Distributed Event Driven Simulation Algorithm with Inconsistent Event Preemption for Accurate Execution of VHDL Descriptions on Parallel Processors ", *IEEE Transactions on Computers*, Vol. 50, No 1, January 2001, p. 28-50.
- [7] K.L. Kapp, T.C. Hartrum, T.S. Wailes, "An improved cost function for static partitioning of parallel circuit simulations using a conservative synchronization protocol", *IEEE Ninth Workshop on Parallel and Distributed Simulation (PADS'95)* , p. 78, June 14 - 16, 1995 Lake Placid, New York .
- [8] V. Kirshnaswamy, G. Hasteer, and P. Banerjee, "Automatic Parallelization of Compiled Event Driven VHDL Simulation", *IEEE Transactions on Computers*, Vol. 51, No 4, April 2002, p. 380-394.
- [9] Tun Li, Yang Guo, Si-Kun Li, "Design and Implementation of a Parallel Verilog Simulator: PVSim ", *IEEE 17th International Conference on VLSI Design* , p. 329, January 05 - 09, 2004 Mumbai, India.
- [10] D. E. Martin, P. A. Wilsey, R. J. Hoekstra, E. R. Keiter, S. A. Hutchinson, T. V. Russo, and L. J. Waters, "Integrating Multiple Parallel Simulation Engines for MixedTechnology Parallel Simulation", *IEEE 35th Annual Simulation Symposium*, pp 0045, April 2002.
- [11] T.J. McBrayer, P.A. Wilsey, "Process combination to increase event granularity in parallel logic simulation ", *IEEE 9th International Parallel Processing Symposium*, p. 572, April 25 - 28, 1995 Santa Barbara, CA.
- [12] Tony. P.W. Price, David. M. Howard, Andy. M. Tyrrell, and Alwyn. V. Lewis, "Adaptive Microphone Array Beamforming for Teleconferencing Using VHDL and Parallel Architectures", *IEEE Seventh Euromicro Workshop on Parallel and Distributed Processing* , pp 13, February 03 - 05, 1999 Funchal, Portugal, 2002.
- [13] G. Ramstein, O. Deforges, P. Bakowski, "A Design Tool for the Specification and the Simulation of Array Processors Architectures - Application to Image Processing: The Extraction of Regions of Interests", *IEEE, The International Conference on Application Specific Array Processors (ASAP'95)* , p. 322, July 24 - 26, 1995 Strasbourg, France.
- [14] Guido Schumacher, Wolfgang Nebel, " Object-Oriented Modelling of Parallel Hardware Systems ", *IEEE, Design Automation and Test in Europe (DATE '98)* , p. 234, February 23 - 26, 1998 Paris, France .
- [15] Kevin Skahill, "VHDL for Programmable Logic", Cypress Semiconductor, 1996.
- [16] Peter A. Walker, Sumit Ghosh, "Asynchronous, distributed event driven simulation algorithm for execution of VHDL on parallel processors", *Proceedings of the 32nd ACM/IEEE conference on Design automation*, January 1995.
- [17] J. Willis, Zhiyuan Li, Tsang-Puu Lin, "Use of embedded scheduling to compile VHDL for effective parallel simulation", *IEEE European Design Automation Conference with EURO-VHDL '95*, p 400, September 18 - 22, 1995 Brighton, Great Britain.