THE USAGE OF VHDL IN THE EUROPEAN SPACE AGENCY

Peter Sinander
ESA/ESTEC WSM, Postbus 299, 2200 AG Noordwijk, The Netherlands
Phone: +31-71-653367, fax: +31-71-654295, e-mail: psi@wd.estec.esa.nl

ABSTRACT

Being a standardised Hardware Description Language supported by most CAD tool vendors, VHDL is well suited for describing digital circuits, particularly when more than a hundred companies from fifteen different countries work together. This paper outlines how VHDL is used in European Space Agency *activities, in particular focusing on the need for modelling guidelines and an approach to use VHDL for board-level simulation. Concerning the latter, various techniques are described including methods for protecting the design information in a VHDL model, such as source code encryption and distribution of compiled models. While only an overview is presented here, complete information is freely available on the* Internet.

1 INTRODUCTION

The *European Space Agency* (ESA) is an organisation with 14 European member states, and with Canada as an associate member. The purpose of ESA is to promote cooperation among the member states in space research and technology and their space applications, in particular for scientific purposes and space application systems. Since the charter is to develop the space industry of the member states, all major technical work is performed by external companies under ESA contracts and supervision.

ESA has several establishments in Europe, of which one is the *European Space Research and Technology Centre* (ESTEC) located in the Netherlands. The *Microelectronics and Technology* section (WSM) at ESTEC deals with design methodology in the field of microelectronics, development of Integrated Circuits (ICs) and related topics.

Electronic equipment and components are an increasingly important part of spacecraft, as in almost all other complex systems. Due to the harsh environment in space - in particular radiation effects - and the fact that it is practically impossible or extremely expensive to service a spacecraft after it has been launched, special system concepts, ICs and process technologies are frequently employed. Such special ICs are developed by the space industry, in Europe often under ESA contract. Three main categories of devices can be distinguished:

- ASICs (*Application Specific Integrated Circuits*), typically used only for one specific application;
- ASSPs (*Application Specific Standard Products*), i.e. an ASIC that is made available to other users than the designer;
- Transfer of existing commercial ICs for comparatively complex devices such as the SPARC microprocessor and the ADSP21020 Digital Signal Processor.

2 ASICS AND VHDL

2.1 Development of ASICs

Most of the serious errors encountered in ASIC developments originate from incorrect specifications or incorrect implementation of the specification. To allow the functionality of a device to be independently evaluated, in ESA developments before the detailed design is started a VHDL model is normally required to allow the functionality to be simulated, e.g. by ESA or by another company.

2.2 Selecting VHDL

Since the design of ASICs is performed by various companies, ESA must allow different design tools to be used; it is neither desirable nor feasible to force all companies to use tools from a specific vendor. In the beginning of the nineties when the ASIC design process was being formalised for ESA developments, VHDL was the only HDL (Hardware Description Language) supported by multiple tool vendors and was therefore the natural choice.

VHDL ’93 [VHDL93] is the baseline today since there is a desire to reduce the long-term maintenance costs for the VHDL models; after being delivered ESA cannot require that the original model developer shall update the model when needed. While it was initially intended to use VHDL ’87 as a baseline, this was changed to VHDL ’93 when it became clear that it would be necessary to update practically every VHDL ’87 model, since there are minor syntax and semantic differences. Although these changes are technically of minor nature, it would be a major task to update and subsequently verify a large number of VHDL models originating from several companies.

However, many VHDL tool vendors still only support VHDL ’87, some even supporting constructs deemed illegal by the VASG (VHDL Analysis and Standardization Group) [Interpret]. One example is the *Std.TextIO.EndLine* function, which is illegal VHDL but still is supported by at least one simulator. Several ESA developments therefore use tools based on VHDL ’87, with an update to VHDL ’93 being performed just before delivery.

There is no requirement on which gate-level simulator to be used, be that *Cadence Verilog-XL*, *Mentor QuickSim* or any other gate-level simulator, provided that it is supported by the ASIC foundry. The choice of whether to use stand-alone or integrated VHDL and gate-level simulators is deferred to the designers.

The number of tools supporting the Verilog HDL has increased after it was released as a non-proprietary language in an effort to counter the increasing popularity of VHDL. Nevertheless VHDL is regarded as a good choice, especially in view of its capability for high-level abstractions. It is however possible that such features will be added to Verilog in the future, in the same way as analog capabilities similar to what is being developed for VHDL-A [VHDL-A] have been announced.

3 VHDL Modelling Guidelines

3.1 The need for Modelling Guidelines

Companies working with ESA have varying experience, ranging from large multinational corporations with decades of relevant experience to small companies entering new business areas. In comparatively new fields such as ASIC design and VHDL modelling it is therefore often necessary to actively support the development work. ESA thus tries to establish concepts and methodologies helping companies to gain proficiency and to increase their efficiency. Implanting such knowledge into the companies is beneficial to ESA, ultimately reducing development risk, cost and schedules.

In the last five years ESA has received VHDL models developed by contractors. For each delivery of a VHDL model new error types and unusual coding styles that should be avoided have been identified. Some examples are:
- No or confusing indentation;
- Comments irrelevant or missing;
- Simulator dependent code, such as illegal VHDL being accepted by the contractor's simulator;
- Obscure and/or unnecessarily complicated code;
- The model has the wrong functionality, but was not sufficiently verified so the error was not discovered;
- Run-time code errors not being discovered due to insufficient verification.

Although some of these coding anomalies might be regarded as minor, problems in terms of long-term maintenance, code reuse etc. are significant when considering a large number of models and with many different companies being involved, either as model developers or model users. A similar problem exists for multinational and multi-site companies, and sometimes even between different departments within the same company. In all situations where VHDL is employed for portability reasons, modelling guidelines are necessary as well.

practical and ready-to-use, since this gives the best effect; "*many designers prefer a small well-defined box to a large but unknown freedom*". But it is crucial to avoid insignificant issues considered as almost fundamental by some designers. A typical example is whether to use upper- or lower-case characters for reserved words; this actually unimportant issue can cause endless discussions.

Specific guidelines for Logic Synthesis have not been incorporated since ESA does not mandate Logic Synthesis for ASIC design; this is the choice - and potentially the problem - for each company. In cases where the modelling is not directly related to ASIC design it is even not desirable. Furthermore different Logic Synthesis tools have different limitations and capabilities, and the status of tools available on the market is constantly changing. In summary there are few specific requirements on models for component simulation, allowing a flexible selection of design tools and design methodology.

The requirements for models for board-level simulation are more elaborated, as described in the following section.

Finally, system-level simulation covers many areas such as protocols, algorithms and models for performance evaluation. Some system-level concepts have not been tried yet, at least not using VHDL. There are therefore not very detailed requirements for this type of models.

Summary of Concept:

- High readability and clarity;
- Avoid obscure constructs;
- Only use portable constructs;
- High level of verification;
- Common characteristics and interfaces for models for board-level simulation.

General requirements supporting almost all simulators and synthesis tools, written in a ready-to-use style.

3.4 Using VHDL for Board-level Simulation

Board-level simulation can be defined as simulating the functionality of a printed circuit board built with standard components, possibly incorporating ASICs and ASSPs as well. Due to simulator performance reasons board-level simulation is normally limited to the digital domain.

A major issue for board-level simulation is the availability of simulation models of the components used on the board. Despite commercial models being available for many standard components, increasingly often ASSPs, ASICs and other unusual devices are used. Hardware modellers can solve this problem, though they are expensive, more complicated to use and have limitations in the number of devices that can be used simultaneously.

VHDL models are therefore an interesting alternative when no models are available, which is the typical case for almost all components used onboard spacecraft. Major issues when using VHDL for board-level simulation are to have a **common model interface**, to implement **timing modelling** (setup and hold times, output delays etc.) and **handling of unknowns** for the model inputs and outputs, and **simulation performance**.

In the ESA *VHDL Modelling Guidelines* the concept for timing modelling is based on the IEEE Vital (VHDL Initiative Toward ASIC Libraries) activity [Vital]. This allows the Vital subprograms to be reused, saving coding effort as well as potentially offering speed optimisation; several simulators already offer optimised versions of Vital subprograms. Full Vital compliance has not been achieved; a different technique for the selection of the simulation conditions (e.g. min. or max. delay) has been specified:

- Vital is based on using an external delay calculator, where the actual timing values for a specific simulation condition are back-annotated using the SDF (Standard Delay File) format. The timing values are specific for each component instance;
- It was felt that users of board-level simulation prefer a less complicated mechanism such as simply using a single generic to change the simulation condition for all components together.

As a support to the *VHDL Modelling Guidelines*, the document *VHDL Models for Board-level Simulation* [BoardLevel] is currently under preparation. While the guidelines largely contain requirements, this document will show how they can be implemented in a practical way, explaining the underlying issues and trade-offs. The main scope is VHDL models for board-level simulation, but many techniques can be used also for other types of modelling. Some examples of the contents are:

- How to efficiently code models with timing checkers and X-checkers only activated when necessary;
- Techniques for improving simulation performance;
- Verification of the models including methods for assessing the verification coverage;
- Automatic testbenches allowing repeatable verification, with cycle-by-cycle checks using MISR (Multiple Input Shift Register) techniques (detailed on the next page).

Both of the previously described ESA documents are freely available to anybody finding them useful, not only to companies involved in space activities. Some examples where the ESA *VHDL Modelling Guidelines* have been used outside the space community are:

- VHDL teaching at the Technical University of Twente, the Netherlands, and at the Technical University of Vienna, Austria;
- Parts have been incorporated in the book *VHDL Coding Styles and Methodologies* by Mr B. Cohen, to be published by Kluwer Academic Publishers in 1995;
- Applied Micro Circuits Corporation (AMCC), a manufacturer of ASICs and ASSPs located in California, are basing their internal VHDL modelling guidelines on it;
- There are plans within the *European CAD Standardisation Initiative* (ECSI) to use it as a starting point for an international standard.

The ESA documents are available on the *Internet* in PostScript format at the address given in the *References* section of this paper.

4 DISTRIBUTING VHDL MODELS FOR BOARD-LEVEL SIMULATION

By the middle of 1995 ESA will have received around twenty VHDL models of complex ICs for spacecraft electronics. The next step will be to actively introduce the concept of board-level simulation to the companies working with ESA. Not only will board-level simulation improve the design quality and reduce the development schedules, it also will give new companies increased possibilities to design spacecraft electronics, thus increasing the competition as well as the competitiveness of the equipment suppliers. The major task will be to ensure the availability of simulation models.

Realising that the market for simulation models for space-specific components is too small to be of economic interest to established modelling companies as *Synopsys/Logic Modelling Corporation* (LMC), the strategy is to reuse the VHDL models developed when the components are designed. By using VHDL the effort to support several platforms and simulators is greatly reduced, since VHDL models require no or only minor modifications for each new simulator. This trend can be seen also for commercial models, e.g. LMC offers VHDL models of lower complexity components. Commercially C models are often used for complex components, mainly for performance reasons; if a relatively large number of models can be sold it is economically possible to support the additional cost associated with creating, porting and maintaining a C model.

4.1 Source Code Encryption

While distributing VHDL source code would require a minimum effort since the user can be made responsible for the adaptation for different simulators if necessary, such an approach is not acceptable for reasons of protecting the design information:

- The company that designed the component is understandably often unwilling to let the information become available to its competitors, especially in those cases where the VHDL code is synthesizable;
- The availability of VHDL source code would encourage the redevelopment of similar devices leading to increased costs for ESA as well as significantly decreasing the interest for a foundry to support a device as an ASSP.

```
prOceSS BEgIN waiT ON llllll1ll ; If llllll1ll = ’1’ anD llllll1ll’eVENt AnD
llllll1ll’LaST_VAlue = ’0’ ThEN iF ( ll1l11lll = ’1’ ) ThEN lll11llll <=
ll11l11l1 ; END IF ; lll11l1l1 <= ll1ll1l11 ; ll1llllll <= l1ll1l1l1 ;
ll1llll1l <= ll1ll11ll ; if ll111l1ll = ’0’ THEn l11l11ll1 <= l11l111l11 ;
ElSE cAse l11l11ll1 IS whEn l11111l1l1 => if ll1l1lll1 = ’1’ oR l1llll11l =
’1’ oR llll111ll = ’1’ oR l1111llll = ’1’ oR ll1lll111 = ’1’ then IF
ll1ll1l11 = ’1’ tHEn l11l11ll1 <= l11111ll11 ; eNd iF ; end if ; wHeN
l11111ll11 => l11l11ll1 <= l1l11l1l11 ; WhEn l1l11l1l11 => If ll1ll111 l =
’1’ Then l11l11ll1 <= lll1llll11 ; eLse l11l11ll1 <= llll1l1ll1 ; enD If ;
whEn llll1l1ll1 => IF ll1ll111l = ’1’ Then l11l11ll1 <= lll1llll11 ; eLsIF
l1l1l11ll = ’1’ THEN l11l11ll1 <= l11l111111 ; end iF ; whEN l11l111111 =>
iF ll1ll111l = ’1’ OR l1l1l11ll = ’1’ thEN l11l11ll1 <= lll1llll11 ; eNd iF
; WHEN lll1llll11 => l11l11ll1 <= l11111l1l1 ; whEN l11l111l11 =>
l11l11ll1 <= ll11l11ll1 ; WHEn ll11l11ll1 => l11l11ll1 <= l11111l1l1 ; wheN
oThERs => l11l11ll1 <= l11111l1l1 ; eNd case ; ENd if ; EnD If ; enD PrOcESS;
```

Encrypted VHDL source code

A first level of protection can be achieved by **source code encryption** or scrambling. With this technique comments are removed, identifiers are replaced with meaningless names and code structure such as indentation is removed. While the encrypted source code is difficult to read directly, it can be automatically processed to increase the readability. Worse is that the model is still identical, i.e. a synthesizable model will still be equally synthesizable.

In one case as an experiment, encrypted VHDL source code has been distributed in combination with a non-disclosure agreement. While the tool *Krypton* is available from the French company *LEDA* [Krypton], for this first experiment a simple encryption tool was developed in-house using *TurboPascal* on an IBM PC.

4.2 Compiled VHDL models

The solution envisaged for future larger scale model distribution is to supply the models in a compiled format, supporting those simulators allowing sufficient protection of the VHDL code. A further requirement is that it must be possible to remove most of the remaining source code information.

To increase the level of protection, in some cases the compilation will be combined with source code encryption, to rename units and signals internal to the model. In specific cases the VHDL models can first be modified to be more difficult to synthesize, for example by eliminating hierarchy and by merging processes. An additional benefit would then be increased simulation performance.

```
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading WORK.tctestdef(body)
# Loading WORK.ptdtestbench(functional)
# Loading PTD_LIB.l1l1lll11
# Loading PTD_LIB.ll111ll1l(body)
# Loading PTD_LIB.ptd(boardlevel)
# Loading PTD_LIB.lllllll1l(body)
# Loading PTD_LIB.l1l1ll1ll(lllllllll)
# Loading PTD_LIB.l11ll1l11(lllllllll)
# Loading PTD_LIB.ll1l1llll(lllllllll)
# Loading PTD_LIB.lll111l1l(lllllllll)
# Loading PTD_LIB.llll111l1(lllllllll)
```

Simulation of an encrypted and compiled model

The *Model Technology V-System* simulator compiles to an intermediate format consisting of low-level RISC-like instructions. The format is claimed to be identical for all supported workstations and operating systems, including the Windows version for the IBM PC. The format seems to be stable between minor releases, but may change for major releases. When supporting the IBM PC the file names for the compiled models and any files used by the models have to respect the IBM PC DOS *8+3* format.

Synopsys VSS has two intermediate formats; one interpreted (must be present) and one where the VHDL code is translated to C and subsequently compiled (optional). The interpreted format is relatively simple to decode, but from version 3.1 it is possible to encrypt it, which should make reverse-engineering more difficult. The format for each minor release, platform and operating system seems to be incompatible, complicating the task of distributing compiled models.

The *Mentor QuickVHDL* simulator core is identical with the one from *Model Technology*, though it can be a problem to identify what version numbers correspond. In principle it should therefore be possible to use the same format for both simulators.

Cadence Leapfrog is a candidate for future evaluation, since most of the companies currently working with ESA would then be supported. This simulator compiles into low-level instructions in the same way as the one from *Model Technology*.

4.3 Experiences

Work is currently ongoing to demonstrate the feasibility of distributing compiled VHDL models. We have created scripts that automatically compile a VHDL model for a specific simulator version and purge all unnecessary source code information. Automatic testbenches are then used to verify the compiled model without manual inspection being necessary. When the approach has been proven it is the intention to let an external company take over the distribution on a commercial basis. The first models in compiled format have already been delivered and successfully used by one company.

An activity demonstrating board-level simulation was started in 1994, and is foreseen to be completed in 1995. We are also investigating the possibilities to introduce full board-level simulation for one of the spacecraft currently being designed.

5 SUMMARY

VHDL is an ideal language for hardware modelling when interoperability is required. In this paper several aspects of interoperability have been discussed, as experienced in the activities of the *European Space Agency*. However, VHDL alone is not sufficient and the need for modelling guidelines has been described. It is believed that establishing guidelines accepted internationally will be useful due to the ever-increasing international cooperation between companies. The ESA *VHDL Modelling Guidelines* could be a possible starting point for such modelling guidelines.

Thereafter several modelling aspects when using VHDL for board-level simulation were discussed. As one example a verification technique based on MISRs (Multiple Input Shift Registers) was described in detail.

Finally, the issue of protecting the design information when distributing VHDL models was presented. Two possible methods to protect the information - source code encryption and compiling models - have been described and some limitations and practical problems highlighted.

Rather than here presenting detailed information from the ESA documents out of context, mainly the concepts have been outlined. The reader is however encouraged to retrieve the complete information, which is freely available on the *Internet*.

REFERENCES

[VHDL93] *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076-93, IEEE, New York, USA, 1994

[Interpret] *IEEE Standards Interpretations: IEEE Std 1076-87, IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076/INT-1991, IEEE, New York, USA, 1992

[VHDL-A] *VHDL-A Design Objective Document*, version 2.1, IEEE PAR 1076.1, 1994. URL ftp://nestor.epfl.ch/pub/vhdl/standards/ieee/1076.1/requirements/DOD_v2.1_draft.txt

[EIA567] *VHDL Hardware Component Model Specification*, EIA Standard 567-A, EIA, Washington DC, USA, 1994

[StdDevKit] *Std_DevelopersKit v2.0 manuals*, The VHDL Technology Group, Pennsylvania, USA, 1995. URL ftp://supportnet.mentorg.com/pub/mentortech/tdd/manuals/std_developerskit.ps.tar.Z

[ModGuide] *VHDL Modelling Guidelines*, ASIC/001, European Space Agency, The Netherlands, 1994. URL ftp://ftp.estec.esa.nl/pub/vhdl/doc/ModelGuide.ps

[StdLogic] *IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164)*, IEEE Std 1164-93, IEEE, New York, USA, 1993

[Vital] *VITAL Model Development Specification*, version v2.2b, IEEE PAR 1076.4, 1994. URL ftp://vhdl.org/vi/vital/Spec/v2.2b/vital22b.ps

[BoardLevel] *VHDL Models for Board-level Simulation*, WSM/SH/010, European Space Agency, The Netherlands, 1995. URL ftp://ftp.estec.esa.nl/pub/vhdl/doc/BoardLevel.ps

[MathPack] *Package Math_Real*, version 0.9, IEEE PAR 1076.2, 1995. URL ftp://vhdl.org/vi/math/package/math_head.3.24.95.vhd

[Krypton] *Non-Reversible VHDL Source-Source Encryption*, K. O'Brien & S. Maginot, Proceedings of Euro-VHDL '94, 1994, pp. 480 to 485