Board-Level Simulation using VHDL Models

Sandi Habinc and Peter Sinander

European Space Agency, ESTEC WSM, Postbus 299, 2200 AG Noordwijk, The Netherlands
sandi@ws.estec.esa.nl, psi@ws.estec.esa.nl

Abstract

Board-level simulation is a necessity for successful development of printed circuit boards built with complex components such as microprocessors, ASICs and ASSPs. In this paper an approach to prototyping will be presented based on VHDL models for board-level simulation. Examples of industry activities using such models for simulation of spacecraft electronics will demonstrate the feasibility of this approach. The importance of optimisation on the source code level will be shown with practical examples on simulation performance improvement. Distribution of models will be discussed, emphasising on protection of design information and model availability.

1. INTRODUCTION

VHDL is a language for specification, design and simulation of digital electronics, including Application Specific Integrated Circuits (ASICs), board designs and subsystems. VHDL stands for VHSIC Hardware Description Language, where VHSIC stands for the Very High Speed Integrated Circuit project initiated by the US Department of Defence. In 1987 VHDL became IEEE standard 1076[1].

In European Space Agency (ESA) developments a VHDL model is normally required when an ASIC is designed to allow the functionality to be independently verified by simulation. A logical follow-on activity is to model and simulate complete board designs using VHDL. Such an approach will be presented hereafter.

2. WHAT IS BOARD-LEVEL SIMULATION?

Board-level simulation can be defined as simulating the functionality of one or several printed circuit boards built with standard components, possibly incorporating ASICs and Application Specific Standard Products (ASSPs). Board-level simulation is also known as rapid or virtual prototyping and sometimes system simulation. The purpose of board-level simulation is to verify the correct behaviour of the board design, e.g. that the components operate in the selected operating modes as intended. When board designs contain processors it is possible to perform verification of the hardware and software interaction, such as verifying that ASIC registers can be programmed and software drivers work properly etc. In addition, the performance of the processor board could be evaluated. Board-level simulation will also give some information about timing correctness, though it should not replace worst-case timing analysis for high-reliability applications such as spacecraft.

To avoid unreasonable expectations from board-level simulation it is necessary to understand what it does not comprise. It does not comprise simulation of system performance, including aspects such as throughput and latency, where neither accurate data nor clock behaviour is considered being essential. Neither does board-level simulation comprise explorative simulation for defining system baselines.

By employing board-level simulation, simulated integration and test of printed circuit boards can be performed early in the development without any hardware manufacturing. The manufacturing can therefore be postponed until the specifications have settled and all interfaces have been verified. Board-level simulation supports a top-down methodology, allowing simulation of boards not implemented, enabling the designer to work with incomplete specifications of the own system or component, and facilitates early verification of the functionality. In large developments with boards being developed by several different subcontractors, the designer can deliver a board design with simulation models for early system verification.

When designing an ASIC, its operation in a board design can be verified before manufacture. If models for board-level simulation are provided before the first silicon for an ASIC or ASSP, significant savings in schedule can be obtained for the first board designs using them.

Models for board-level simulation allow the designer to simulate situations that are difficult to capture in real hardware resulting in a more thorough verification of the board design. It also provides the designer with unlimited probing and acquisition points, not always possible to realise for the hardware, and can also provide visibility into the internal state of components, such as registers and state machines.

Models for board-level simulation can provide limited simulation support during parallel development of software and hardware. This type of simulation usually takes long time to perform when using such models but delivers high functional accuracy. By carefully selecting which software parts to simulate, the simulations can be reduced to manageable times. It is perhaps not feasible to boot an operating system, but all the firmware could be verified using board-level simulation. Board-level simulation enables hardware and software designers to work together in an early stage and to solve interfacing problems before the hardware manufacturing, which is especially important when ASICs are involved due to large schedule and non-recurring engineering cost impact. This type of simulation will become more interesting with the continuously increasing speed of workstations and simulators.

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 special devices are used. VHDL models are therefore an interesting alternative when no models are available, which is the typical case for almost all components used on-board spacecraft. 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. VHDL has therefore been chosen as the modelling language for models for board-level simulation to be used in developments funded by ESA.

3. FUNCTIONAL ACCURACY

A model for board-level simulation is characterised by modelling of the component behaviour, simulation performance, and ease of use for board designers. The behaviour of the model as seen from the outside should be the same as for the actual component and should include the full functionality. Specific test modes only used for manufacturing test need not be implemented. The interface signals of the model should have the same waveform behaviour as the component it represents. Models intended for board-level simulation need to have a common model interface, implement timing modelling (setup and hold times, output delays etc.) and handling of unknowns for the model inputs and outputs. This also applies to VHDL models for board-level simulation.

Bus functional models, sometimes called bus interface models, are considered being reduced models for board-level simulation, modelling only timing and behaviour of the component interfaces. These are typically used for very complex devices such as processors in which case instruction execution, interrupts etc., are not available in the model. The timing and format of output drivers for data/control/addresses etc. are modelled as accurately as possible, while the internal functionality of the component is not necessarily modelled at all. Using bus functional models does not provide the full potential of board-level simulation, only modelling the interfaces.

Models for board-level simulation have to implement the functional behaviour of components accurately enough to

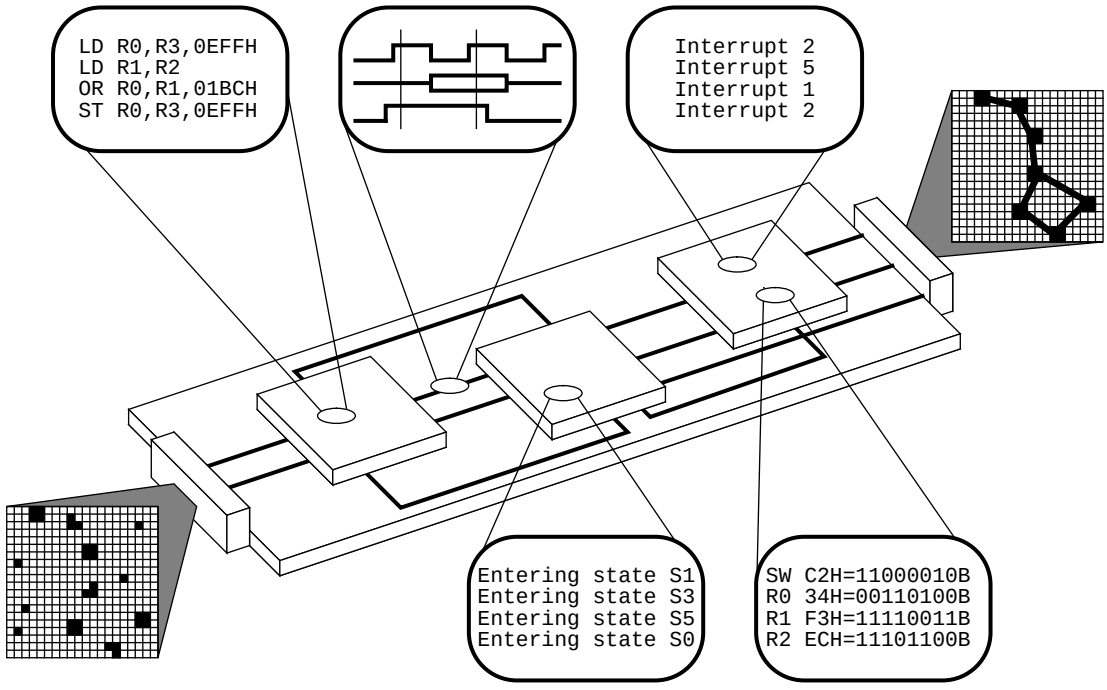

Figure 1: *The designer's view of a board design when using board-level simulation.*

allow board designs to be verified for functionality and timing. Simulating boards using models with high accuracy will reduce the number of errors found on the manufactured board. However, an error in a model can lead to design errors, e.g. in an ASIC or a printed circuit board, not being detected alternatively being introduced in the board design.

There are two major approaches to modelling; independently develop the model from a functional specification or data sheet, or enhance an existing model on the Register Transfer level (RTL) or higher. The first approach is necessary when no RTL model is available to the board-level developer, or when only a gate-level model is available.

Care should be taken when a model is developed using only a data sheet as input, since the component is not always described in a data sheet as actually being implemented. Details might be left out and errors can have been introduced, and there are more chances for misunderstandings. The information in the data sheet could have been simplified, e.g. the description of an interface protocol may be more constrained than the actual design requirements, which would result in a not fully correct model. When a model for board-level simulation is being developed in parallel with the component and the two are compared with each other by means of simulation, the component development could most likely benefit from the independent interpretation of the specification made by the board-level developer.

The simulation performance should be assessed when transforming an RTL model to a model intended for board-level simulation, since RTL models do normally not have sufficient simulation performance. Also, RTL models are normally written for synthesis, which imposes conflicting requirements on the VHDL code with respect to models for board-level simulation. The main development effort would then be to optimise the code for simulation performance, to implement the model interfaces and to develop the verification test stimuli.

Occasionally there are inconsistencies between the behaviour that is described by the RTL model and the data sheet. It may be that some functionality of the device has been simplified or omitted in the data sheet, e.g. proprietary design features. The more accurately the model reflects the actual hardware, the more likely is it that problems in a board design will be detected while simulating it. Also, the inclusion of unsupported or undocumented functionality of the component in the model for board-level simulation could simplify its comparison versus an RTL model during verification, using the same set of stimuli.

4. SIMULATION PERFORMANCE

The performance of present workstations and VHDL simulators provides a means for simulating board designs comprising several complex components such as microprocessors and ASICs. However, to be able to tap this simulation performance the models have to be efficiently coded for simulation. An absolute requirement on simulation performance for models intended for board-level simulation cannot easily be defined, although unnecessarily slow or cumbersome implementations should be avoided.

The recommendations presented below are based on experiences with modelling and using models for board-level simulation, and on results from an ESA activity on simulation performance. This is not an exhaustive list of issues to be addressed when a model is tuned for simulation performance. It should also be remembered

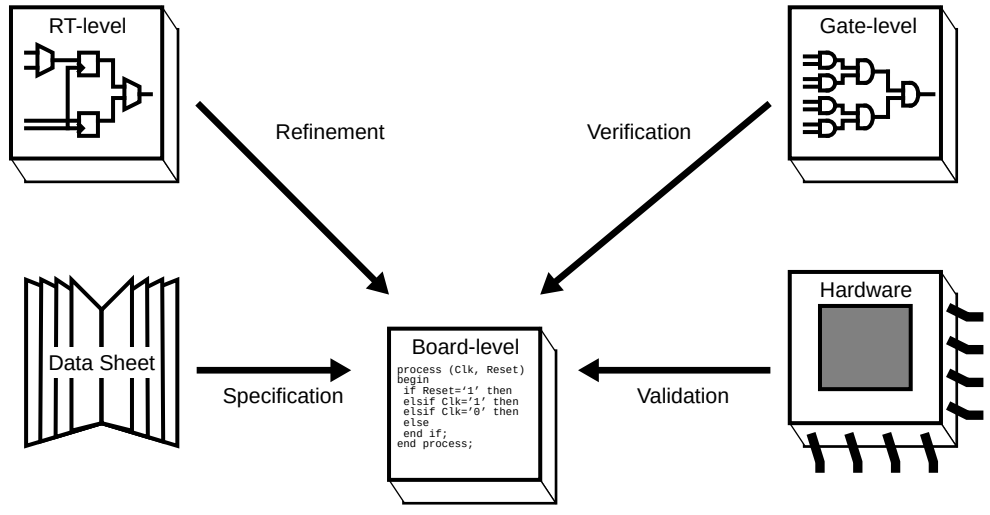

Figure 2: *Sources of information for modelling.*

that each suggestion might not be true for all situations and simulators. More detailed information on modelling for simulation performance can be found in the article *Using VHDL Models for Board-Level Simulation*[5] in the 1996 fall issue of the *IEEE Design & Test of Computers* magazine.

The major contributors to the time spend simulating VHDL models are signals, processes and concurrent statement, which even in optimised models account for 80% to 90% of the total time. Variables should therefore be used instead of signals wherever possible, since each signal requires specific handling (event scheduling) which takes more memory storage and instructions to execute. In fact, results show that signals significantly contribute to the simulation time compared to variables.

Each process invocation has a cost in terms of simulation performance and in principle the number of processes should therefore be kept small. An observation made is that functionality sensitive to the same signals should be grouped in the same process, reducing the number of processes to invoke for each signal event. Following this approach, all functions related to the same clock region should be grouped in one process. Another reason for merging processes is that the number of signals used for the communication between them is reduced.

A fundamental rule when modelling for simulation performance is to only execute code when necessary. Therefore, conditional statements should be used to reduce unnecessary execution of code. An outer conditional statement should reduce the necessity to evaluate enclosed conditional statements. This is done by using nested if and case-statements, ordered so that the branches with the highest probability are executed first. The efficiency of the ordering of conditional statements can be assessed by analysing the results from a code coverage utility. The performance can thereafter often be improved by reordering the code or modifying the structure of conditional statements. Results have shown that an assignment of a signal is between one and two orders of magnitude more costly in terms of simulation time compared to reading a signal or variable in an if-statement. This suggests that one can use large structures of if-statements to prevent a signal to be unnecessarily assigned, and still gain in simulation performance.

Many rules and techniques that apply to writing optimised software, such as loop unrolling, code in-lining etc., do also apply to models with good simulation performance since VHDL has many characteristics of a programming language. Some VHDL simulators have less built-in optimisation capabilities than state of the art compilers for software, and it is therefore often beneficial to manually perform optimisation at the source code level. In addition, calls to subprograms declared in packages are difficult to optimise by the analyser since the package body can be reanalysed without demanding that the code where the call is made from is reanalysed as well. It is therefore necessary to manually replace subprogram calls by in-line coding in the source code when optimising a model for simulation performance.

Writing efficient VHDL models requires knowledge of which constructs are fast, and which ones should be avoided. While experienced model writers often have a feeling for efficient coding, there is currently little exact information available. The model writer therefore needs a “cost list” for different VHDL statements in terms of simulation performance and memory usage. Having such information based on quantitative measurements and not on assumptions, can help the model writer to choose between coding variants for the best performance. As a first step to such a “cost list” an activity was initiated to measure the cost of several basic low-level VHDL constructs such as variable and signal assignments. After obtaining the final results from the study above, it should be possible to extract more exact information about what VHDL constructs simulators optimise poorly.

5. EXAMPLES OF OPTIMISATION FOR SIMULATION PERFORMANCE

When developing an ASIC or other component using VHDL and synthesis, the objective is the resulting component and not the VHDL model as such. The VHDL model is often split into parts allowing several designers to be involved concurrently, each working on a separate part. To facilitate both the design and verification, each part is then further broken down into code chunks of manageable size. Any optimisation performed is targeted to the synthesis results, to reduce area and/or increase the design speed as required. Sometimes functionality is also added for testability reasons. Not being a primary objective, the resulting simulation performance is often far below a model coded for optimal simulation performance. While acceptable for the simulations during the development, optimisation of the simulation performance is necessary to allow the model to be used for board-level simulation.

In order to estimate the possibility to automatically transform a VHDL RTL model into a functionally identical model with better simulation performance, a series of experiments where performed on a model from an ASIC development activity, including:

- Unnecessary signals were removed;
- Hierarchy was removed so all processes were contained in the same architecture;
- All clocked process were then merged into one, with only a single clock edge condition statement.

In this experiment the performance increased by up to a factor of two, although a large number of concurrent processes and signals was not optimised due to constraints on the effort available to continue the experiment. The above optimisation steps were implemented in a way allowing automatic translation of a VHDL model according to a set of transformation rules. Such a tool could be implemented as a VHDL-to-VHDL translator, or alternatively, be implemented directly in a simulator as one optimisation step. The latter method would of course be preferable, since besides being invisible for the user, also the designers of the ASIC would benefit from the simulation speed-up.

Another example is an ASSP for telemetry that has been modelled on several abstraction levels and the simulation performance has been monitored throughout the development of the different models. The ASSP was designed using Verilog written on the register transfer level, although many blocks resembled functions such as flip-flops and low level multiplexers. The design was then translated to VHDL using semi-automatic conversion. The purpose of the translation was to enable ESA to independently verify the design on a VHDL simulator.

A model for board-level simulation was developed for the ASSP following the ESA guidelines, the model comprising multiple processes representing the major blocks in the component. Further optimisation was done by capturing the synchronous functionality of the ASSP in one process and all asynchronous interfaces in another process. A memory under the exclusive control of the ASSP was integrated in the process containing synchronous functions, removing any overhead in terms of signal assignments for memory interfacing, and also removing the memory model out of the simulation. The increase of simulation performance between the different abstraction levels was quite remarkable. The model intended for board-level simulation simulated approximately one hundred times faster than the register transfer level model, which is a performance increase that makes board design simulations feasible. It shall be mentioned that the RTL model was only two times faster than a gate-level model run on a conventional non-VHDL simulator. The RTL model was however written on a low abstraction level and the translation from Verilog was perhaps not the most efficient one. An RTL model written directly in VHDL should normally be much faster than the gate-level model and not fully that much slower than the model for board-level simulation. The model with only two processes and with the memory integrated was yet five times faster than the model intended for board-level simulation. Normally this ASSP is used in a constellation where there are four to eight of these components, and consequently the performance of the model has a great impact on the total simulation time.

6. VERIFICATION

Verification of models intended for board-level simulation is performed to ensure that the model correctly represents the actual hardware what concerns functionality and timing. Two categories of verification can be identified; to ensure that the model has the same functionality as the actual hardware (performed during model development), and to verify that the model works for a certain combination of simulator and platform (performed by the user with test stimuli provided by the model developer).

The first category of verification should be performed at the end of the model development to ensure that the model intended for board-level simulation reflects the functionality of the component. The method outlined here is based on the existence of actual hardware or a low-level design representation such as a gate netlist for comparison. When this is not available, the verification should be based on information found in a data sheet or similar, though this increases the number of potential errors and misunderstandings regarding the functionality. Emphasis should then be put on performing the verification independently from the model development. The verification should include all functional test stimuli used during the component development. As a last

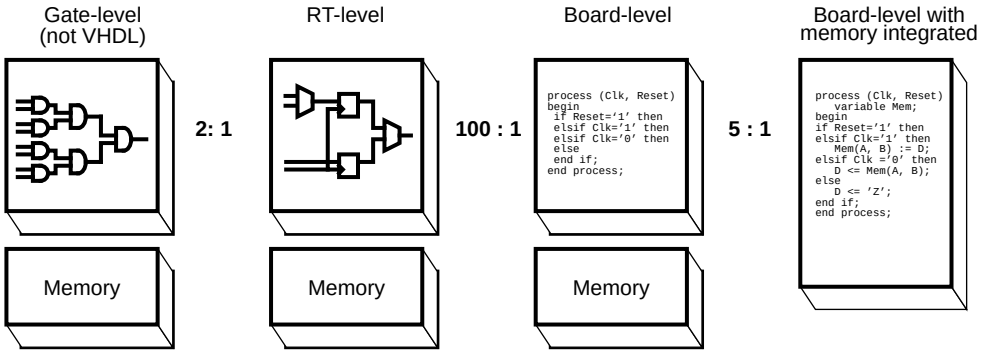

Figure 3: *An example of simulation performance speed-up due to increase of abstraction level*

verification, the model should be simulated when placed in a typical board design to detect any problems not covered by other test stimuli. The test stimuli should be suitable as a maintenance vehicle for verifying the model after modifications. As for all verification activities, test stimuli should be developed by somebody not involved in the development of the model to avoid masking of errors.

The second category of verification should be performed by users after installing a model in their particular simulation environment to ensure that it will operate correctly. This should be done by using one or more test stimuli provided by the model developer, reporting whether the test has passed or failed. The reason for performing a verification when a model is installed is that there are still some differences between VHDL simulators and different releases of the same simulator. In addition, users tend to suspect the models first in case of a problem during simulation. Test benches that support automatic verification and require a minimum involvement by the user are more likely to be used, potentially reducing the amount of problem reports not related to the model itself being handled by the model developer or distributor.

A method useful for verifying that the exact behaviour of a model has not changed when delivered to the user is to sample the outputs on the test object and compress the acquired data using a Multiple Input-Signature Register (MISR) which is compared to a predetermined signature to determine whether a test has passed or failed. The benefit of compressing output data is the elimination of large reference files. This approach could also be used for regression tests during all types of model an ASIC developments. Such data compression is most feasible for data that have been synchronously sampled, and are therefore best used for test benches verifying the functional behaviour on a per clock cycle basis. The MISR should have sufficient number of register stages to allow for long test stimuli without significant risk of error masking. It should therefore be structured as a primitive binary polynomial implementing a maximum length Linear Feedback Shift Register (LFSR). More information about MISR and LFSR can be found in the text book *Built-In Test for VLSI*[6].

The efficiency of a test stimulus can be evaluated using quantitative measurement methods such as code coverage. A code coverage utility can help verifying that a test stimulus executes every source code line of a model, but does not tell how much of the model functionality has been covered. The purpose of the verification should always be to verify the complete functionality of the model, not to solely satisfy the code coverage goal since it is merely an inexact measurement of the verification efficiently.

7. MAINTENANCE OF MODELS

During the last five years ESA has received VHDL models developed by contractors. For each delivery of a model, new error types and unusual coding styles that should be avoided have been identified. 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. In all situations where VHDL is employed for portability reasons, modelling guidelines are necessary as well.

Before creating ESA specific guidelines, a search was done for existing modelling guidelines that could be used, fully or at least as a starting point. Unfortunately, those guidelines found were not considered as very useful, containing mainly general and high-level requirements. Such guidelines would require each user to spend significant effort for the analysis of requirements and for developing the technical solutions needed. In addition, despite several years of existence none of these guidelines seem yet to be widely accepted by the design community.

The first basic modelling guidelines were established in 1992, based on our own modelling experiences obtained during several years. They were introduced for contracted developments and were updated every time a new VHDL model was received. After several such overlapping iterations, in March 1994 a major draft was circulated for review among most of the companies working with ESA. The ESA *VHDL Modelling Guidelines*[3] document was issued in September 1994. It is used as a requirement document in many ESA contracts involving VHDL modelling. In line with the philosophy of not mandating specific design tools, the VHDL Modelling Guidelines can be used with practically any VHDL simulator or synthesis tool. The guidelines have been written to be 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.

As a support to the VHDL Modelling Guidelines when developing models for board-level simulation, the ESA document *VHDL Models for Board-level Simulation*[4] has been prepared. While the guidelines largely contain requirements, this document shows how they can be implemented in a practical way, explaining underlying issues and trade-offs. The main scope is VHDL models

for board-level simulation, but many techniques can be used also in other types of developments, such as verification of ASICs.

For models intended for board-level simulation the concept for timing modelling is based on the IEEE Vital[2] (VHDL Initiative Toward ASIC Libraries) activity. 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 since a different technique for the selection of the simulation conditions (e.g. minimum or max.imum 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. 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.

8. DISTRIBUTION OF MODELS

In 1996 ESA will have received around twenty VHDL models of complex ICs for spacecraft electronics. A logical step would be to actively introduce the concept of board-level simulation to the companies working with ESA depending on their interest. 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 is 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, our 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. Synopsys Logic Modelling offers VHDL models of lower complexity components.

Commercially, C models have often been used for complex components due to historical reasons and claimed performance gains. However, the cost for scheduling model external signals in current simulators is the major limiting factor what concerns simulation performance. Until significant improvements in the signal scheduling have been introduced, there should be little difference in simulation performance between VHDL and C based models in a VHDL simulator.

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 would not be 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 synthesisable. 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 foundries to support the devices as ASSPs.

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 synthesisable model will still be equally synthesisable. In one case as an experiment, encrypted VHDL source code has been distributed in combination with a non-disclosure agreement. While encryption tools are commercially available, for this first experiment a simple encryption tool was developed in-house.

```
proCess bEgiN WAIT on llllll1ll ; if llllll1ll =
‘1’ AnD llllll1ll’EvEnt ANd llllll1ll’laST_VaLUE
= ‘0’ THeN l1l1111l1 <= l11llllll ; IF l1l1ll1l1
= ‘0’ thEn l1111llll <= “00111111” ; lll1l1ll1
<= “11111111” ; ElsIf ( ll1l111l1 = ‘1’ anD
ll1111l1l = ‘1’ ) ThEn l111ll1ll <= l111ll1ll +
“00000001” ; eNd IF ; iF ( l11l1l11l = ‘1’ ) THEn
ll11llll1 <= l1111llll ; eNd if ; ENd iF ; EnD
iF ; eND pRocESS;
```

Figure 4: *Encrypted VHDL source code*

The solution envisaged for potential future model distribution is to supply the models in an analysed 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 could 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 synthesise, for example by eliminating hierarchy and by merging processes. An additional benefit would then be increased simulation performance.

While models could be freely distributed both in source code and analysed format, such a service would not be feasible since the cost for maintenance would not be covered. It is preferred that all models are bundled and

sold in a package including maintenance, being handled by a commercial company.

Scripts have been created that automatically analyse a VHDL model for a specific simulator version and purge all unnecessary source code information. Automatic test benches are then used to verify the analysed model without manual inspection being necessary.

The final decision whether to launch a service for this type of model availability will depend on many factors, such as the market size and expectations, and the existence of a suitable distribution company, the latter currently being the major problem.

9. EXPERIENCE USING MODELS

An activity to demonstrate board-level simulation was started in 1994. The aim was to design and simulate a fictive computer for spacecraft data handling. The board design incorporates a processor of the MIL-STD-1750 type, memories, a bus interface for accessing other units on-board the spacecraft, four ASSPs implementing the up-link and down-link communication protocols, and some glue logic. Models for the processor, memories and glue logic were written based on their data sheets. These models were written for high simulation performance. In addition, very basic software for the verification of the board design was written in assembler.

As a result of the board-level simulation around ten errors in the hardware design and in the software were detected. The hardware errors ranged from devices not being reset and buffers being incorrectly activated to identification of omissions in the board specification. While the software intentionally was simple, several errors in the interaction between the hardware and the software were discovered and corrected.

In addition to demonstrating the concept of board-level simulation, several issues related to the development of complex models for board-level simulation were identified. One finding is that at the time of writing it seems difficult to reach more than one thousand simulated instructions per second for a timing accurate VHDL model of a processor. The reason for this limit is seemingly due to limitations in the implementation of the signal scheduling of current VHDL simulators. As reference, an instruction simulator in VHDL, i.e. without time notion or signal scheduling, has roughly one order of magnitude higher performance, and an instruction simulator written in C has up to two orders of magnitude higher simulation performance.

Another finding is that developing complex models from their data sheets requires large efforts for the modelling as well as for the verification. As an example, for the processor model the original verification was extensive including test stimuli for each instruction as well as verification using part of the functional production test vectors. Nevertheless, almost one hundred errors were found when ESA later validated the model with respect to the original design database of the device using pseudo-random sequences of in total several million instructions.

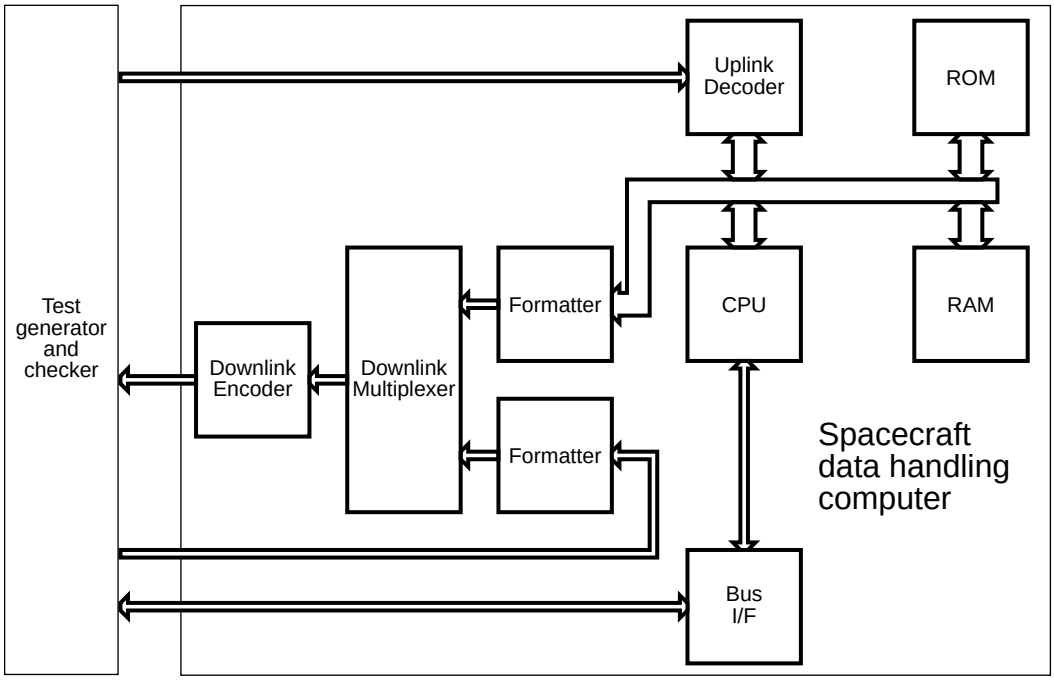

Figure 5: *A fictive computer for spacecraft data handling*

The activity had to be closed with several errors remaining.

In another case, a European space company has successfully demonstrated the benefits of using models intended for board-level simulation while designing an ASIC to be used for a number of satellites. The ASIC interfaces several components using different protocols in an on-board data handling system, and performs conversion between these protocols. The design was verified by simulating not only the board on which the ASIC was located, but also some components on other boards with which the ASIC interfaces. Two of these components were represented in the simulation as models developed according to the guidelines for board-level simulation established by ESA, of which one model has been developed by a third company and the other was an ESA in-house development. The simulations of the ASIC and these models revealed two errors in the ASIC design that could be corrected prior to manufacturing. The errors were located in the ASIC subblock interfacing the models developed for board-level simulation and were caused by undocumented component behaviour. The modelling of the reset sequence for the two models and the start-up procedure for one of the protocols enabled the designers to discover the errors, demonstrating the importance of modelling for functional accuracy. The availability of these models also made it possible for the company to generate test vectors for the printed circuit board containing the ASIC, to be used by their automatic test equipment during the manufacturing of the board.

During the last year, models for board-level simulation have been used by ESA when performing independent verification of ASIC designs. In one case an ASIC has been successfully simulated together with two different microprocessor models, peripherals and multiple data handling bus components, in total comprising eight different models. Being already developed, the effort for using these models was minor. Furthermore, large parts of previously developed test stimuli were reused.

10. SUMMARY

In this paper the current usage of VHDL in ESA has been presented. One logical step after using VHDL for ASIC design has been identified as board-level simulation. A definition of board-level simulation has been presented together with its benefits and limitations. Its major benefit being verification of printed circuit boards or subsystems without the need for manufacturing any hardware.

The importance of accurate models has been described, identifying two different approaches to model development; refinement of an existing RTL model, or modelling using a data sheet or a specification as input. Recommendations have been given on modelling for simulation performance, based on experience with development of models for board-level simulation and results from an ESA study. Some examples of simulation performance improvement have also been presented.

Two categories of model verification have been discussed; to ensure that the model intended for board-level simulation has the same functionality as the actual hardware, and to verify that the model works for a certain combination of simulator and platform.

A major issue for board-level simulation is the availability of simulation models of the components used on spacecraft. Issues concerning the distribution of VHDL models without revealing proprietary information have been discussed, suggesting encryption and compilation as techniques for protection.

Finally, successful industry activities using models for board-level simulation have been presented. The experience from these activities demonstrates the feasibility and benefits of board-level simulation for spacecraft electronics.

11. REFERENCES

1 IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-93, IEEE, New York, USA, 1994

2 IEEE Standard VITAL ASIC Modelling Specification, IEEE Std 1076.4-95, IEEE, New York, USA, 1995. URL: http://vhdl.org/vi/vital

3 VHDL Modelling Guidelines, P. Sinander ASIC/001, European Space Agency, Noordwijk, The Netherlands, 1994. URL: ftp://ftp.estec.esa.nl/pub/vhdl/doc/ModelGuide.ps

4 VHDL Models for Board-level Simulation, S. Habinc WSM/SH/010, European Space Agency, Noordwijk, The Netherlands, 1996. URL: ftp://ftp.estec.esa.nl/pub/vhdl/doc/BoardLevel.ps

5 Using VHDL for Board-Level Simulation, S. Habinc & P. Sinander, IEEE Design & Test of Computers, New York, USA, Fall 1996. URL: ftp://ftp.estec.esa.nl/pub/vhdl/doc/BoardSim.ps

6 Built-In Test for VLSI: Pseudorandom Techniques, P. H. Bardell et al., John Wiley & Sons, New York, USA, 1987

More information on microelectronics and VHDL simulation can be found on the ESA web pages at URL: http://www.estec.esa.nl/wsmwww.