

The Merging of Present Day Tool Capability with VHDL

Ron Werner
Computer Integrated Design and Manufacturing
Motorola
Government Electronics Group
Scottsdale, Az

Abstract

As we look into the future, we see that VHDL modeling for digital simulation will be readily available. Tools will provide capability for creating VHDL models and for interfacing with other tool environments. Today, many tools in support of VHDL are in their infancy. Several interfaces still need to be created, and it will take time to prove that these interfaces and tools are production ready.

VHDL tools need to merge and interface with tools that are presently being used to design complex digital circuits in a production environment. The efficiency and support of present tools are key factors in determining the best level of VHDL integration to fit your companys design environment. This paper will attempt to bring to light those nasty questions that many VHDL advocates attempt to avoid when discussing VHDL within a specified methodology. With this information, users will be in a much better position to plan the implementation of VHDL and thereby achieve many of the targeted benefits. To meet increasing demands for shortened design cycles, higher quality and productivity, VHDL **must** be incorporated into our design environment rather than being avoided!

Future design methodologies

Presently, many design groups are working in an autonomous environment, and often, an individual is responsible for almost the entire design cycle. Several factors are at work that will change this design paradigm:

- Increasing logical functionality.
- Core blocks are increasing in size and complexity.
- Decreasing Silicon geometries are increasing component densities.
- Manufacturing techniques need to accommodate advanced packaging (e.g. MCM).
- Cycle time reduction initiatives.
- Pursuit of Six Sigma quality goals.

As the above indicates, increasing design complexities will mandate a paradigm shift in our design environment and methodology.

In the future, design groups will operate in a homogeneous team environment with heterogeneous tools. This heterogeneous environment will be one that incorporates use of different software tools, hardware description languages, and workstation platforms. Interoperability is a **key** factor in the success of this heterogeneous environment. VHDL will play a major role in accomplishing the transition (interoperability) by providing a Hardware Description Language that will convey information about the operation of devices. Standard components will be described in VHDL, and models will be developed before or concurrent with silicon. Model availability must be the norm, to secure this environment.

Methodology in Motorola

The four areas in which methodology is being set in place at Motorola are:

Conceptual design.

This is a very abstract definition of what the system should entail. Requirements are defined and traced through subsequent design activities. The concept is decomposed to unit definitions.

Unit design.

In the unit design phase, engineers are decomposing the high level of abstraction into lower levels. The end result will be a partitioning of the design into functions that could represent LSI (ASIC) functions and board (standard component) functions.

Board/LSI design.

Standard components and LSI descriptions are used to define the board level designs. Fault grading and manufacturing test/capabilities will have to be well defined and developed. Design for testability will play a key role in this phase.

Decomposition of behavior for LSI will continue until a synthesizable RTL code is created. LSI designers will select ASIC vendor/vendors and develop the required net list and test vectors to meet manufacturing and design functionality requirements.

Manufacturing.

Manufacturing personnel normally have the end responsibility to properly test components and systems to insure that they can meet Six Sigma quality standards. Their first concern is to insure that manufacturing defects have not been injected and to provide confidence in the level of functional correctness.

All designs will follow this methodology. As the design grows in its functionality and complexity, so will the number of engineers/groups grow. Often a design group will be focused around each of the major categories as listed above.

Between each of the methodology areas there are large amounts of information that need to be exchanged. It is intended that a majority of the information exchange will be accomplished by way of VHDL. Below, we will discuss in greater detail how design information is passed to subsequent design groups and how back of annotation information is handled to keep each group synchronized.

Specialized tools

A myriad of software tools have been generated that enhance an engineer's capabilities. These tools are designed to address very specific niches that have direct impact on the selected

methodologies and/or technologies. The following is a list of tools selected at GEG to aid in accomplishing our design goals and addressing the above methodology. Many of these tools are already in use across Motorola.

Ascent logic / RDD – Requirements tracking.

SES workbench – Digital data flow design evaluation.

Comdisco / SPW – Digital signal processing design.

Cadence – Schematic capture and layout of digital systems.

GenRad – Digital design verification through use of software simulation.

Mentor – Schematic capture, design verification and layout of LSI.

Synopsys – Primarily used for synthesis and secondarily for VHDL simulation.

Excluded from this list are software development tools and hardware test tools.

All of the above tools are used within various groups in Motorola. For each of the methodologies and specific tools, there are proven paths for specific task completion. Interface software tools have been developed that aid in getting information into and out of specific tools. These proven paths are used today to insure that we can produce good silicon and that our software models accurately represent device characteristics.

Investigation of the above list indicates that within each of the tools, the methodology for integration, interoperability and the ability to transfer design information is well defined. Outside of that specific environment the information transfer becomes more clouded.

Interface and methodology interaction

Motorola has internally agreed that its standard hardware description language will be VHDL. VHDL is a very robust language with very few restrictions placed on the user in relation to methodology or language structures. In theory, all aspects of the VHDL language may be applied in a variety of ways and consistent, equivalent results should be achieved across all supporting simulations. Present day established CAE vendors have produced products that fulfill the immediate needs of their customers. They cannot afford to just dump their old system and move to a new one. These vendors must establish a migration path from what they presently have to the new target system. This, in many cases, is like a ball and chain attached to older CAE vendors. This ball and chain has positive benefits to users but the caveat is that longer development time will result as new languages are implemented.

So, a major question is, how is interoperability going to be maintained? By reviewing VHDL standards, there are two basic standards which attempt to get a handle on the interoperability issue. They are IEEE MVL9 and EIA Component Model Standard. The intent of these standards is to aid in insuring model interoperability of components. A software model could replicate the operation of a device at a level of abstraction from just above the RTL level, down to the gate level. Over this range of abstraction, all bits are known and defined. Note: this level of abstraction may not allow for direct synthesis of VHDL code.

VHDL facilitates model simulation at a very high level of system abstraction. At high levels of abstraction the user is concerned about philosophy, overall system characteristics and development of requirements. Some characteristics of this VHDL code may include enumerated types, variable character lengths, etc. GEG's experience has established that deriving standard pack-

ages and methodologies to insure efficient transfer of design information is a significant task. For one project, Motorola has invested several man months in developing packages and a methodology for its system level VHDL models.

It was determined that the most expedient path to incorporating VHDL in our system methodology was to concentrate on the EIA model component standard. Because we have been a participant in the development and review of the EIA standard from the beginning, our needs have been addressed.

Tool interface and hazards

With the above methodology in mind, each of the tool vendors, as listed under specialized tools, were asked to address issues of interfacing and utilizing VHDL. This was intended to be a leading question to be followed with specific questions as to how they were going to solve the task. We have reviewed and evaluated the VHDL issues and below is a brief statement about each of the tools.

RDD – VHDL output is not supported at this time. Engineers will be required to manually create information required by follow-on design groups. The engineer must act as the interface, therefore must understand both environments.

This tool may be the weak link and could require the most effort of all the interfaces.

SES workbench – VHDL output is supported, but it cannot be read back into the system. A VHDL testbench is also an output. With the testbench, one can run the simulation utilizing a favorite simulator and compare the results to insure that they are equivalent between two systems. The VHDL that is produced does not meet the EIA standard and is normally at a high level of abstraction.

Additional decomposition of this model would be required to create an RTL synthesizable model. This effort could be significant and will depend a great deal on knowledge of the system level designers assumptions. The decomposition will be primarily a manual process. Traceability could be maintained and decomposition can be completed in an homogeneous environment.

SPW – VHDL output with an associated testbench is supported, but VHDL cannot be read back into the system. By utilizing the testbench you can run the simulation using your favorite simulator and compare the results to insure that they are equivalent between the two systems. The VHDL that is produced does not conform to the EIA standard.

There are two levels of VHDL that can be outputted from SPW. These are:

- Non-synthesizable VHDL code
- Synthesizable RTL VHDL code.

Non-synthesizable VHDL would contain such things as floating point math and other algorithms, with features that synthesis cannot support. If these constructs are used, an engineer will need to manually decompose this model to create a synthesizable output.

By decomposing non-synthesizable SPW constructs into those constructs identified by

Synopsys and SPW for synthesis, one can output VHDL code that can be read by the synthesis product. At this level of decomposition, simulators that can handle VHDL should be able to simulate the design. One needs to be aware that the VHDL testbench does not conform to the the same level of abstraction as the VHDL model. Therefore, a simulator that can handle synthesizable VHDL may not be able to properly simulate the device with the testbench provided.

GenRad – VHDL at the RTL level, plus some additional constructs, are supported today.

Some of the constructs used by SES workbench are not supported by GenRad. The result is that GenRad cannot simulate SES workbench designs.

With an understanding of translation variations, the SPW code could be simulated by GenRad.

Steps need to be taken, in the creation of the testbench to ensure that unsupported constructs are not used.

GenRad's product has the most VHDL construct coverage of established CAE companies who provide VHDL simulation capabilities.

People are successfully using the GenRad VHDL today.

Mentor – A subset of VHDL 1076 is supported. Constructs used by SES and SPW are not supported at this time.

People are successfully using the Mentor VHDL today, within the defined subset.

Synopsys – This is a simulation tool that was created for the sole purpose of simulating VHDL structures. VHDL outputs from SES and SPW can be read into the Synopsys simulator and simulated. Synopsys also supports a synthesis product that is another product line within their organization.

Tool notes and vendor Interactions

The above list represents only a small portion of available tools. Each of the vendors were requested to interact with other vendors. During each validation effort, problems were discovered and modifications were put in place. The end result was a product that the user could use. Many of the transformations were transparent to the user.

With the relationships that developed between vendors, we can expect tighter integration of the products, as new releases are introduced.

Tool summary/disclaimer

All of the companies have indicated that they have plans for enhancement of their product in regards to VHDL capabilities.

When selecting tools, it is important to determine what the target VHDL is. You have to make a consumer decision based on general tool performance versus quality and transportability of VHDL.

Summary

I assume that eventually all VHDL constructs will be fully supported in all digital software tools sets. We will have the capability of generating VHDL output and the capability of reading in VHDL input. Models will be available that describe design information in VHDL and interface tools will be productized that will construct VHDL and extract information readily from VHDL code. This is not the case in today's marketplace.

In a company as large and diverse as Motorola, and a market place that is even more diverse, we cannot afford to wait until everything is ready. EIA, IEEE and other standards are now being developed that will have significant impact on our methodologies. The tools that are presently available are usable with caveats. Once these caveats are acknowledged, work-arounds can be devised and useful work accomplished. Those projects that start on the learning curve now will be positioned for early achievement of their design goals and, at the same time, they will be able to influence tool creation in support of their target methodologies and data interchange standards.

Required action

Each of us must:

Apply pressure to our vendors to insure their compliance to standards. It should be the responsibility of the vendor to provide utilities that will treat translation of information transparently, to meet the above standards.

Test available standards in your area to insure your requirements can be met.

Use the tools available and apply pressure to vendors for tool improvements.

When a vendor indicates support of VHDL, don't blindly accept their claims, but rather explore what is actually meant by their claim (i.e. what caveats are involved and what are the limitations that exist in the system).

Require vendors to provide utilities that will expedite establishing interfaces into your environment.

We must strongly encourage EDA tool suppliers to meet the EIA modeling standard for components.

Ending statement

Today, vendors are providing software tools that allow you to integrate VHDL into your design environment. We have been able to demonstrate that there is cost benefit in using VHDL to interface between different tools and groups. Using a standard hardware description language does come with caveats and effort needs to be invested to work the deficiencies and achieve positive results.

If all users were to become more VHDL literate and use VHDL in their present tool environment, then more effective guidance and direction would be available to insure successful implementation of VHDL. This will provide a win-win situation for all parties.

Acknowledgments

I would like to recognize the following for providing support and assistance in the preparation of this document:

Linda Werner, Bob Stephens, Ron Lieberman, John Scruggs, Harv Rakestrow, Andrew Guyler.