Enhancing Design Productivity
By
Increasing Gate Production with
Behavioral Synthesis

Ron Crevier, Peter Runstadler, Paul Clemente, Kent Arnold

Raytheon Company
1001 Boston Post Road
Marlborough, MA. 01752

E-mail peter_w_runstadler@ccmail.ed.ray.com
E-mail paulc@jove.ed.ray.com

Abstract

With the advent of increasing system complexity and the need to meet shrinking schedule and cost targets, new methodologies for improved first time success to market need to be explored and developed. VHDL over the past several years has focused on the structural and register transfer level (RTL) of coding. This yields a low gates per line of code metric. If the gates per line of VHDL are to increase then methodologies based on using higher levels of abstraction will need to become more usable and widespread.

Over the past four years Raytheon has successfully used VHDL to design ASICs, modules, subsystems, and systems. During that time several different methodologies were explored. This paper will examine those methodologies and historically show a progression from the early benefits of using various RTL methods to the latest design flow using behavioral synthesis. This latest methodology allows true system design with significant reduction in the number of lines of code, manpower, and schedule. In comparison large system designs using RTL VHDL require many more lines of code, thus increasing model complexity, the risk of error and causing longer compile and simulation times. There are obvious tradeoffs and limitations when higher levels of abstraction are used and some of these details will be presented.

Section 1. Introduction

Current design methodologies do not integrate hardware and software in the early phases of the design cycle. Normally, hardware and software integration does not begin until detail design is complete. Once design is complete, the hardware and the software cannot be modified without significant cost and schedule impact. The interdependencies between hardware and software suggest integration much earlier in the design cycle. Early integration of hardware and software would uncover system errors before all hardware and software builds are complete. This early integration would reduce or eliminate the cost of second pass hardware. By using behavioral synthesis early exploration of the design space is possible. Software, as well as, hardware can be prototyped much earlier in the design process.

The key objective of this work is to significantly reduce schedule, costs, and risks associated with digital system design thereby enhancing design productivity. The main schedule savings is realized by the early integration of hardware and software through the use of behavioral modeling and virtual prototyping. In a traditional design flow, system test and integration begins after detail design but with this methodology integration can begin after the concept design review. This paper will show clear benefits in schedule and design metrics compared to previous methods and metrics collected from the work conducted at Raytheon.
Section 2. Design Methodology

Transition from specification to hardware is one of the most challenging aspects of digital system design in industry as well as Raytheon. As the complexity of designs increases, the transition becomes more difficult. A high level design flow for Raytheon's future design methodology is shown in Figure 1. The following methodology represents Raytheon's goal for establishing a new design environment. This methodology was used as the various levels of abstraction were investigated. The most important elements of this methodology are described below:

Requirements Traceability - The methodology provides a mechanism for tracing system requirements throughout the design process. Executable specifications written using VHDL are used to document and trace all requirements down through each level of the design (i.e. system, subsystem, board, ASIC). Testbenches written using VHDL are used to test that requirements are met across every level of the design (see Bottom-Up Verification below).

Top-Down Design - The methodology relies on a top down design flow from requirements to implemented hardware. From the top level system requirements specification, a series of functional requirement specifications are generated to define the system at each successively lower level (i.e. subsystem, module, ASIC). The specifications at each level are used to generate VHDL models and testbenches. The VHDL testbenches are then used to verify the models functionality and timing. The top down design, coupled with bottom-up verification (see below), ensures that the definition and allocation process to each level in the design have preserved the top level requirements.

Bottom-Up Verification - The final design implementations are verified at each successively higher level in the design flow. For example, the ASIC gate level design is first verified by simulating the ASIC using the ASIC VHDL testbench. The gate level ASIC implementation is then simulated in the context of the board design using the board VHDL testbench. Then the board is simulated in the context of the subsystem using the subsystem VHDL testbench. A thorough verification approach such as this ensures that interfaces and functionality across all levels of the digital system design are verified.

High Level Design - The use of VHDL coupled with logic synthesis enables digital design to be performed at a higher level. Designs can be modeled at high levels of abstraction and then synthesized to produce gate level implementations. The level at which designers are able to model continues to rise with the introduction of new behavioral level synthesis tools. In addition to VHDL for high level design modeling, many graphical modeling tools are now available. Some of these graphical tools produce synthesizable VHDL code in order to interface directly with logic synthesis tools. These tools automate the front end of the design process by supporting hardware-software trade-offs, design partitioning, performance analysis, and hardware-software co-design.

Seamless Design Flow - The design flow is fully automated from the initial high level model and testbench development process to the hardware implementation. Models used for simulation are linked through synthesis to the actual design implementation. The design is entered once, in the form of a VHDL model. The VHDL is verified and used as the base for the rest of the design process.

VHDL to Logic Synthesis - Technology independent register transfer level (RTL) VHDL models of ASIC and FPGA components are synthesized to gate level implementations. Logic synthesis continues to evolve as behavioral synthesis products are now entering the marketplace. Synthesis is also critical to the Virtual Prototyping technology now being developed at Raytheon (see below).

Mixed Level Simulation - The methodology utilizes mixed level simulation to verify the board design at different stages of the development process. Mixed level simulation allows models to be written at different levels of abstraction (i.e. behavioral, RTL, gate). Mixed level simulation allows the system to be simulated with a behavioral level testbench at any stage of the design process, including (and most importantly) after gate level ASIC and FPGA implementations have been synthesized.
**Virtual Prototyping** - For very large and/or complex digital systems, current design methodologies require the development of a breadboard or prototype design implementation. The prototypes are used to verify system requirements and to provide a target hardware platform for software development and validation of the system's logic design implementation. A "two pass" design approach (prototype and final design) is both expensive and time consuming. Raytheon's Electronic System's Division is addressing this issue by defining an appropriate subset of VHDL and enhancing synthesis and hardware emulation CAE tools to provide a "Virtual Prototyping" capability. This reusable functional prototyping system will allow the rapid transition from system concept and specification to a hardware implementation for requirements verification, software development and detailed logic design verification. Most of the work presented in this paper utilized virtual prototyping at the back end of the design process.

**Design for Test** - Quality must be designed-in, not screened out. Today's increasing circuit densities, high device speeds, surface mount packaging, and complex board interconnections have had positive influences on state-of-the-art electronic systems; however these factors have concurrently had adverse affects on system level test. With the marketplace demanding cost effective solutions, test groups may be tempted to choose less rigorous testing. However, a company that delivers poorly tested products will not remain competitive. Thus, an increasing number of companies are adopting DFT techniques. The result is a system delivered on time, under budget and with no sacrifice in quality.

**Section 3. Early VHDL Methodology**

The following section details the design methodology that was developed in 1992 and used on different programs. This methodology yielded a sixteen percent cost reduction, a thirty percent schedule reduction and reduced rework. This methodology is a tailored version of the methodology discussed above.

The electronic design automation (EDA) environment consisted of four main elements.

---

**Figure 1 - High Level Design Methodology**

The front end was a VHDL simulator, the second component was RTL to gate synthesis, the third component was hardware acceleration for simulation and verification, and the fourth component was system verification with emulation.

At the front end, the VHDL simulator was used to support all simulation on the workstation platform and on the hardware accelerator platform. This environment supported mixed level simulation, behavior, with RTL, or with gates. The same behavioral level VHDL testbench was used to validate the design at the various levels of abstraction, RTL or gates.

The first step of the logic synthesis (architectural synthesis) process was to follow the VHDL coding guidelines associated with the synthesis tool. This allowed the compiler the ability to "synthesize" a generic gate level implementation. This implementation was mapped to a set of primitive functions optimized for execution on the hardware accelerator. This produced a design representation and gate model that more closely represents the target foundry implementation. Another benefit was simulation execution speed: simulation is 1 - 3 X faster than the workstation, architectural synthesis is 2 - 4 X faster than gate level synthesis. All these factors contributed to faster development and validation of a design at the RTL level.

The second step in the synthesis phase consisted
of gate level logic synthesis. From the VHDL RTL ASIC design description, the synthesis tool is used to implement the design in the target foundry technology. The designs were iterated at this step until functional, performance, timing and gate count requirements were met.

The final step in synthesis started after completion of logic synthesis. Synthesis vendors provided an automatic test vector generation (ATG) capability. This allowed the insertion of boundary, partial or full scan logic into the design.

ASIC emulation was introduced into the design methodology for early hardware/software test and integration. ASIC emulation provided an effective complement to simulation, it provided an effective means to reduce overall system development cycles and greatly reduced the probability of second pass ASICs. Second pass silicon had significant impact on program cost and schedule growth risk.

Together this process formed the backbone for design, development and verification of complex digital systems. Figure 2 shows the

At the back end, the hardware accelerator box performed the conventional gate-level simulations, fault grading, back annotating of layout effects from ASIC foundries, and full-timing (dynamic) analysis. It also performed VHDL RTL acceleration that mapped VHDL descriptions at the RTL level into generic primitives. Board level simulation consisted of a blend of ASIC VHDL models and off the shelf MSI and SSI models.

Raytheon has successfully applied this methodology to the development of: eleven double sided, 8" x 8", surface mount modules, 277K gates of logic contained in eight ASIC types, 50K gates of logic contained in seventeen FPGAs, and a significant amount of off the shelf MSI and SSI components. The end result was a sixteen percent reduction in cost and a thirty percent reduction in schedule.
The methodology is successful for several major reasons:

1) Methodology dictates development of both VHDL executable specification model and testbench at each design level. This ensures that design function and performance requirements are properly flowed down and tested.

2) Methodology dictates bottom-up verification approach. Lowest level design implementations are verified in context of higher level design assemblies. This ensures design implementation compliance.

3) Design implementation via logic synthesis from verified RTL level designs greatly enhances gate level design implementation productivity.

4) Methodology allows rapid prototyping with ASIC emulation. This greatly enhances confidence in ASIC design by allowing execution of operating system and application level software on virtual ASIC implementations. This approach also allows early test and integration of both system diagnostics and operating system and/or application software. This reduces the overall system test and integration schedule.

Section 4. Early Graphical Methodology

The early phases of system development must be to understand the problem and be able to express that to others. Traditionally this has been done through textual specifications. As the complexity of systems continues to grow this fails to adequately meet these communication needs. Another form these specification can take is graphical. The power of these tools is in the ability to transition to detailed design via the automatic generation of VHDL.

A methodology has been developed at Raytheon to ensure design integrity from specification to hardware implementation for the development of both FPGAs and ASICs. Development of this methodology relies on the use of i-Logix's ExpressV-HDL as the front end requirements analysis tool. Automatic generation of Synthesizable RTL VHDL from ExpressV-HDL's statechart notation enables the exploration of multiple synthesis paths. This interactive environment for system analysis allows creation of a graphical model of the system function and behavior which can be validated via model execution. This process of formal specification of system requirements and subsequent model validation creates an executable specification. Systems are described with ExpressV-HDL using three graphical languages: module charts, activity-charts and statecharts. The module-chart represents the structural view of the system. This chart is a hierarchical system block diagram that includes data and control flow between system components. The purpose of the activity-chart is to define system processes or functions. The activity-chart allows the user to define data and control that flows between the resident functions. It is the statechart view that defines the behavior of these processes or functions. Statecharts represent the modes, or states, that a system may be in.
A written ASIC specification for a Raytheon design to be utilized in the Military VAX Computer product was the requirements specification used to drive the pathfinder. This specification describes the requirements for a CPU to peripheral bus interface function called the RDAB IC (Raytheon Digital Audio Bus Interface Chip). An implementation of this interface function would require equal portions of control and datapath logic with a gate complexity of roughly 8,000. The seamless design methodology was particularly applicable to this problem for several reasons. The requirements specification was in a state of flux and thus an ideal candidate for behavioral modeling using ExpressVHDL's statechart notation. As changes to the specification were made the statecharts could be easily modified and validated. In addition, the implementation path and target technology had not been chosen.

In order to prove the Statecharts to FPGA seamless methodology, a pathfinder was performed with using a Requirements Specification for a Raytheon design as a starting point. The path to implementation consists of behavioral modeling using statecharts, automatic VHDL generation, gate level synthesis and FPGA place & route, configuration and test. Figure 3 shows the methodology used in this pathfinder. This methodology proved successful and lead to a reduced cost and schedule to implement the target FPGA.

The next challenge was to move to a higher level of abstraction through behavioral synthesis.

Section 5. Behavioral Synthesis Methodology

Raytheon Company designs and develops complex digital systems. The algorithms used by these systems are often modeled in software prior to the start of hardware design. The algorithms are later converted from a software implementation to hardware. Raytheon's design methodology for performing this conversion to hardware currently includes a manual
translation of algorithmic models into architecture's which are then modeled at the register transfer level (RTL). The RTL models of the architecture's, which are far more complex than the original pure algorithmic models, are then used as a starting point for automated gate level implementation. Gate level implementation is accomplished with the use of logic synthesis.

Raytheon, along with many other companies currently using the same or similar top down methodologies, has discovered a number of inefficiencies with the methodology. The manual process of converting from an algorithm to an RTL description of the architecture to implement the algorithm is time-consuming. This process yields a single architecture for the design which is not easily modified if the algorithmic or performance requirements change. A new architecture must be manually generated when a requirements change occurs or if the current architecture cannot meet performance requirements. A new architecture requires a new RTL description.

The design process would be more efficient if exploration of various architectural implementations could be performed at the algorithmic level.

Optimization of the design architecture results in substantial design improvements (area, timing, power, etc.) not possible with standard logic optimization. In addition, design entry at the algorithmic level flows naturally from written system specifications. System specifications are most often written at the algorithmic level and do not define detailed logic architecture. Algorithmic modeling from a system specification is a natural translation which is less error prone, more flexible, and easier to review than modeling at the register transfer level.

Behavioral VHDL Modeling - Design Entry at the Algorithmic Level

The behavioral methodology utilizes algorithmic modeling with behavioral VHDL as the entry point in the development process. The behavioral code is a concise model of the design algorithms which requires roughly one fifth to one eighth the number of lines of code as an equivalent RTL description. The algorithms described in the behavioral VHDL code flow directly from the requirements specification. These embedded specfications provide requirements traceability from the system requirements specification.

Design entry is more efficient at the behavioral level than at the register transfer level for a number of reasons. The behavioral model flows from the original algorithm specification, thus, model generation is a less complex and faster process than RTL model generation. RTL model generation requires the complete definition of an architecture to implement a specified algorithm. The process of defining this architecture can be very lengthy and require numerous trade studies to arrive at a satisfactory result. Behavioral models are architecture independent.

In addition to being more straight forward to generate a behavioral model, changes in the algorithm are also easier to incorporate in a behavioral model. An algorithm change may require an entirely new architecture to meet system performance requirements. For an RTL model, this means a rewrite for most, if not all, of the model. Since the behavioral description does not imply a specific architecture, you can change the algorithm without nullifying the rest of the code.

Perhaps the most obvious indication of behavioral VHDL coding efficiency is that the number of lines of code required to model an algorithm at the behavioral level is close to one order of magnitude less than that required to model the same algorithm at the register transfer level. The code size reduction is due to the use of high level operations, powerful language constructs, and a lack of architecture definition. VHDL coding for behavioral synthesis with Behavioral Compiler models algorithms, instructions, and operations making use of variables and arrays, much the same as system modeling using the C programming language.

Testbench Generation

A VHDL testbench is a model which is used to apply stimulus to a Module Under Test (MUT), compare the MUT's response with an expected response, and report any differences found during the simulation. Depending on the
hierarchy of the system being developed, the MUT could be an ASIC, a CCA, a subsystem module, or the entire system. Key considerations in developing a testbench for a MUT at any level in the hierarchy are as follows:

- Testbench should be developed to verify requirements independent of a specific architecture.

- The level of response checking performed by the testbench should be well defined prior to testbench development.

- Testbench should be developed from the requirements specification independent from the design of MUT (preferably by another individual).

- As the design is translated from the highest level of abstraction to the lowest (from behavioral, to RTL, to Gates) it must be verified by the testbench. If the testbench is developed independent of a specific architecture, then the same testbench can be used to verify the design at all levels of abstraction.

Simulation of the testbench and system model is used to verify the model at each level of abstraction.

Simulation

Simulation is a key aspect of the behavioral synthesis methodology. Verification is performed throughout the development process, from the high level system model down to the gate level implementation. The behavioral synthesis simulation approach ensures that system requirements are verified at each level of design abstraction.

Complex scenarios of many seconds (real-time) of system processing are often simulated to verify the full functionality. Simulation performance is the most critical at the lower levels of model abstraction. Simulations at the RTL and gate level require orders of magnitude more events to be processed than simulation at the behavioral level. In order to make system simulation performance at lower levels of model abstraction acceptable, hardware acceleration is used.

Raytheon uses a mixed level simulation environment to verify and debug the hardware implementation of algorithmic requirements. To verify the algorithms are correct for over all system operation and that the high level system operation meets or exceeds customer expectations the behavioral synthesis methodology uses virtual prototyping.

Virtual Prototyping

Modeling complex digital systems prior to detailed architecture development is a powerful method for system exploration and performance analysis. The analysis of simulation results obtained from system models, however, can present a major challenge. The analysis is often very complex and time consuming.

Raytheon uses virtual prototyping as a new way of approaching the analysis of system simulation data. Although the term "virtual prototyping" has come to mean a lot of things to different people, for the purpose of this article it is defined as the graphical visualization of system operation. When system model simulation incorporates graphical visualization of complex data, analysis can be performed quickly at a high level. Such a visual representation of the system, or virtual prototype, can quickly reveal anomalous design behavior not easily seen while wading through huge amounts of complex and inter-related raw simulation data. Also, a virtual system can easily be verified first-hand by the customer for compliance with their vision of the product. Valuable feedback is obtained by performing this verification early in the design cycle.

Raytheon's virtual prototyping methodology extends the prototyping from pure software to a reconfigurable hardware prototype with the use of new emulation technology. Raytheon's emulation based virtual prototype provides close to target system performance. This allows integration of real time software and thorough system verification.

Synthesis

Two synthesis steps are included in the
behavioral synthesis methodology. The behavioral VHDL synthesizes to a register transfer level (RTL) of design abstraction and the RTL synthesizes to a gate-level implementation.

Behavioral synthesis is linked directly to logic (gate-level) synthesis. This allows the library specific timing and area information to be used by the behavioral synthesis tool. The logic synthesis tool can target the desired technology and resulting netlist is interpreted for logic emulation and placed in the prototyping environment. Figure 4 shows the behavioral synthesis methodology used.

Tradeoffs

The advantages and disadvantage of behavioral synthesis were compared against the features found in different automated design entry tools.

Advantages of behavioral synthesis:
- Fewer lines of code to represent the same function in RTL
- Rapid architectural exploration
- Design optimization is more effective at the higher level of abstraction
- Design entry does not imply an architecture
- Tool can aid in architecture trade studies
- Behavior will simulate faster than RTL or structural VHDL
- The methodology is moved to the next higher level of design abstraction
- Automates scheduling, hardware allocation, resource sharing and memory inferencing
- Behavioral synthesis is linked directly to logic synthesis

Disadvantages of behavioral synthesis:
- RTL and gate level debug difficult
- Tool is new to market and not widely tested
- Textual non graphical design entry
- Productivity benefit not clear for non algorithmic designs
- New coding style required by engineers
Section 6. Metrics

The behavioral synthesis methodology was compared with Raytheon's RTL VHDL based methodology in the development of a small system with high level algorithmic requirements. The following metrics show how the behavioral synthesis methodology improved productivity for the development of this system at Raytheon (Figure 5). A higher level of synthesizable model abstraction was enabled with the used of Synopsys Behavioral Compiler. This resulted in far fewer lines of code, greater than an order of magnitude increase in simulation speed, far fewer man hours of design time, and very similar design performance.

Figure 6 shows the processing speeds of the system virtual prototype at different points in the development process. The advantage of the configurable hardware based prototype ("rapid prototype" in figure) is very apparent when it is considered that full verification of this system required 300 frames of data processing. Also, because the processing times for the software virtual prototype at the RTL and unaccelerated gate levels is so long, these verification steps may be removed as behavioral synthesis technology matures.

Conclusion

The ultimate goal of introducing new VHDL based methodologies and tools in the hardware

<table>
<thead>
<tr>
<th>SYSTEM IMPLEMENTATION</th>
<th>PROCESSING TIME (sec/frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral Model</td>
<td>1,200</td>
</tr>
<tr>
<td>RTL Model</td>
<td>144,000</td>
</tr>
<tr>
<td>GATE Model</td>
<td>228,000</td>
</tr>
<tr>
<td>Gate Model on Hardware Accelerator</td>
<td>1,200</td>
</tr>
<tr>
<td>Rapid Prototype</td>
<td>0.5</td>
</tr>
<tr>
<td>Target Hardware</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 6. System Verification Speeds
design process is to enable cost and schedule savings. Previous development efforts at Raytheon which have utilized well thought out VHDL based methodologies have benefited greatly in the form of large cost and schedule reductions. Hardware development efforts at Raytheon continue to build on previous successes. The introduction of new behavioral synthesis tools promises to provide the next level in design productivity. Based on evaluation results for Raytheon's new behavioral synthesis based methodology, significant improvement in hardware development cost and schedule are possible, especially for designs with high level algorithmic requirements. Improvements are realized through design entry at a higher level of abstraction and early integration of hardware and software using virtual prototyping.

Design Efficiency Improvement

<table>
<thead>
<tr>
<th>Design Methodology</th>
<th>Manual RTL Approach</th>
<th>Behavioral Compiler Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of VHDL</td>
<td>RTL VHDL Top-Down</td>
<td>Behavioral Top-Down</td>
</tr>
<tr>
<td>Gate Count</td>
<td>23.6K</td>
<td>4.0K</td>
</tr>
<tr>
<td>Throughput</td>
<td>90K</td>
<td>~50K</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>28 cycles/frame</td>
<td>32 cycles/frame</td>
</tr>
<tr>
<td>Man hrs Detail Design</td>
<td>450 mins/frame</td>
<td>19 mins/frame</td>
</tr>
<tr>
<td>Design Duration</td>
<td>Four months</td>
<td>Three months</td>
</tr>
</tbody>
</table>

Figure 5. Design Efficiency