EVALUATING VHDL-BASED ASIC SYNTHESIS TOOLS

W. A. Hanna, R. A. Kaskowitz, L. N. Wallace
McDonnell Douglas Electronic Systems Company
P.O. Box 426
St. Charles, MO 63302-0426
Phone (314) 925-4059

Abstract --- This paper describes an evaluation of VHDL based design tools, and using synthesis vs. schematic capture-macrocell approach to FPGA design. This is becoming more important as development time is decreasing and the need to use ASICs in systems is increasing. The risk of committing to an ASIC technology for a project that may or may not go into production, can be mitigated by using FPGAs. Designing with VHDL allows flexibility if the decision is made to migrate the design.

Introduction.

ASIC Technology covers a broad spectrum of Integrated Circuit (IC) products including: Read Only Memory (ROM), Programmable ROM (PROM) , Programmable Array Logic) (PAL), Programmable Logic Device (PLD), Gate Array (GA), Field Programmable Gate Array (FPGA), Standard Cells, Compiled ICs, and Custom Chips. Lower end products, such as PALs, PLDs, and GAs of less than 2000 gates, are designed using tabular formats or boolean expressions to program devices; e.g. the ABEL language approach to designing PALs/PLDs, or they may use a schematic approach with defined macrocells.

The sea of gates device introduced in the mid '70s had a few hundred 2-Input NAND gate equivalent complexity. The GAs in use today can reach hundreds of thousands 2-Input NAND gate equivalent complexity. This reflects a 3 orders of magnitude increase in complexity. Compounding this exponential increase in device complexity is the drive to get a product on the market faster (6-12 month development cycle as compared to 18-36 month 10 years ago)!

For the large complex designs, a formal language description is more effective than using the macrocell approach. It is desirable to design at a higher level of abstraction and synthesize at the functional and/or data flow level. Synthesis Computer Aided Design (CAD) Tools are employed to do the detailed implementation.

By using the same standard language to target the variety of available device types and technologies, more flexibility is afforded in changing implementation strategies for different stages of a product life cycle. The choice of CAD tools is also more flexible, as it is possible to move a design with minimal translation.

Top Down Design Methods

Our military customer is mandating VHSIC Hardware Description Language (VHDL)[1] as the language in which to deliver data about the function and performance acceptance of ASICs [2]. The use of VHDL will allow portable models and multiple implementation possibilities, features that are required to reduce the risk of inserting a technology that may not be available for the life of a program. All customers desire high quality and high performance hardware delivered on time, and within tight budget constraints, and they want assurances of a successful outcome prior to committing themselves to a design.

The high non-recurring foundry cost associated with prototyping of ASICs has restricted their use in multi-stage contract applications. Behavioral simulation in the early design stages has proven to be a valuable aid in validating design concepts without committing to the time and expense of prototyping in hardware. [3]

The traditional approach of designing circuits by schematic capture takes too long to iterate for large designs as the management of changes becomes a problem. This form of design is often vendor and technology dependent, unless considerable effort is made to remain generic. [4] The drawbacks of designing in generic fashion with schematic approaches is that many of the macro or standard cell offerings from foundries are not standard to the industry, which precludes their use in a generic library. This means that a designer might not take advantage of a macrocell available for a given technology, which might be the best solution. This method also requires mapping of generic primitives to the target library primitives, which may not always have one to one correspondence. The VHDL-Based Synthesis method affords many benefits over both schematic capture with target technology primitives and generic primitives:

0-7803-0768-2/92$03.00 ©IEEE

1) Standard HDL for portability and broad applicability.
2) Designing at the High level(behavior) first, allows design alternatives to be compared early.
3) Design at the functional (black Box)/data flow Register Transfer Level (RTL) level, reduces design time when using
4) CAD synthesis tools to perform the detailed gate level design.
5) Automatic synthesis allows vendor and technology independence, without the burden of maintaining generic components or forgoing use of non-standard but high performance cells.
6) IC fabrication technology obsolescence becomes less of an issue.

Many new tools have come on the market in the last two years, based on VHDL, promising to make top down design an affordable and practical reality. Some of the established structural simulators have adopted VHDL as a front end to their existing tools. We have experience and investment in the use of structural simulators, and want to continue to take advantage of their performance benefits, but we also want to include automatic logic synthesis and behavioral modeling capabilities. We undertook an exercise in 1991 to evaluate for ourselves some of the tools which promise such productivity increases and shorter development times. We employed a VHDL synthesis design methodology, in parallel with more conventional methods, in the design of a bus interface chip for one of our products. The target technology was an FPGA which might be migrated to a GA or Std cell, if the design went into large scale production.

VHDL tools

Our first investment in VHDL was a Personal Computer(PC) based simulator, which proved to be a valuable, and low cost, learning vehicle for the transition to the VHDL language. We used it to describe and simulate the overall circuit behavior. The PC-based tool was very basic, but with it we were able to accomplish the task of simulating the behavior of our target design (550 lines of VHDL code).

We divided the design up into smaller modules, which were described more fully and simulated individually. These were treated as processes within the overall entity. By using a divide and conquer approach we could design all of the modules using the PC. We reached the limit of the PC capability when we began simulating the entire circuit together. This was a hardware resource limitation, rather than software.

The design was then moved and simulated on two different workstation-based VHDL simulation/synthesis systems. There were three primary tasks required to convert from the PC based VHDL simulator to the first workstation tools:

Statistics

ITEM	EFFORT	REMARKS
Interface Chip:		
Synthesis Effort	2 months	Shorter than Macrocell based design
Lines of code	550 lines	VHDL Behavioral description
PC-W/S conversion	100 Hours	Long due to conversion to language subsets
W/S tool 1 to W/S tool 2	16 Hours	More efficient (emerging synthesis subset)
Equivalent gate count (core logic)	3000-4000	Depended on speed vs area optimizations Does not include I/O.
First FPGA translations	4 hours	Correct function : didn't meet speed required
Routing iterations	many	Due to timing requirements exceeding FPGA speed capabilities ; Signal routing is many times logic delays and changes each iteration.

First , workstation tool conversion required some time to become familiar with the particular tool set operation and interfaces.

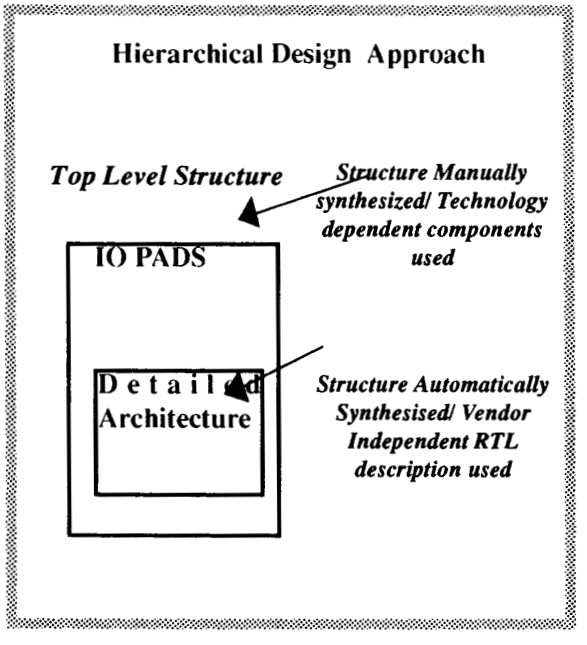
Second, we located and modified VHDL constructs (e.g. clk'EVENT) which were not supported by the vendor at that time. This sometimes required using special purpose, vendor specific, functions to replace the unsupported VHDL constructs.

Third, there was some conversion required from purely behavioral constructs to the RTL constructs which were synthesizable by this logic synthesis program.

Some VHDL constructs would simulate correctly, but could not be synthesized. For example, counters and incrementers had to be converted from code using decimal integer arithmetic, which works for simulation to operations using only binary arithmetic, which is required for synthesis. One line addition operations (S <= A + B) had to be converted to special add functions in one of the tool.

The transfer, conversion, simulation and synthesis took approximately 100 hours. The RTL descriptions were simulated and automatic synthesis was performed for several target technologies (FPGA, GA, and Std Cell). The structural VHDL generated by the synthesis step was simulated and compared to the RTL simulation results.

We then began evaluating a second workstation (ws) based VHDL CAD tool set. Transferring the description took only 2 working days which we attributed to the model being in the synthesizable style[4]. We were able to simulate and synthesize the chip and compare simulation results from the two tools using comparison utilities. The optimized design is in the range of 3000-4000 gates based on power-speed optimizations. This does not include I/O.



We programmed an FPGA which is in use in the subsystem prototype. Although the synthesis didn't take full advantage of all the available FPGA logic, the quick turn around for programming made this an interesting rapid prototyping concept. We were also able to quickly implement structural logic designs in several GA technologies from different IC vendors. The GA implementation easily meets the speed required.

Problems Encountered/Resolved

We used VHDL in a design targeted to FPGAs so that we could learn the use of VHDL, and would be able to migrate the design to a Standard ASIC in later production. Due to timing complexity, the design required manual optimization to fit the FPGA. This was primarily required to improve the timing to meet the performance constraints. A very small number of logic changes were made to the synthesized logic to improve performance.

IO Pads Special Case The FPGA Input/Output pads required special handling. Both logic synthesis packages which we evaluated treated IO as special cases. Current sink and source requirements, rise and fall transition times, power and ground pad placement, noise considerations, and other IO specifications are difficult to describe in VHDL and are often layout as well as logic family dependent. ASIC vendors often have multiple pad options for each IO function, and the synthesis programs had not incorporated the capability to select the appropriate IO pads. After investigating several alternatives, we elected to use a design style which instantiates the IO as components. This means that the IO portion of the design becomes ASIC vendor specific, which we wanted to avoid as much as possible. To minimize that weakness, we used a hierarchial design approach in which the IO pads and the detailed architecture were instantiated in a top level structure. Therefore only this file contained vendor specific information and the detailed architectures remained vendor independent. Changing vendors would only require modifying one relatively small file.

Immaturity of the language We encountered problems in translating VHDL descriptions because the simulator vendors had partial language implementations. The first synthesis attempt was not easy to complete due to partial language implementation. There is also a learning curve for the "synthesis subset" of VHDL. This is perhaps not so much an immaturity of the language as it is an immaturity of general knowledge of it's practical use. We utilized the information available on how to use the synthesis subset [5],[6]. At this point in time there is not a Standard subset of VHDL constructs which are synthesizable. This requires detailed knowledge of the synthesis tools to be used and might limit transportability.

Immaturity of libraries The immaturity of technology libraries was seen initially, for example inout port mode was not supported in the first library we used.

In the case of the FPGA library, in addition to the problems already mentioned the macros which are available in the vendors own design tool are not part of the VHDL-based synthesis library. This limits the efficiency of automatic synthesis for FPGAs, as compared with an experienced designer working with special purpose tools.

Standard Logic Translations between Standard logic functions and types was an issue. Although work towards developing the IEEE 1164 Standard is nearing completion, the support is still in the form of a manual translation. Each simulator used different built in logic types and state strength value systems. Early versions of IEEE 1164 packages vary in the functions contained.

Schematic Requirement By tradition or by contract, schematics are still the format that designers use when presenting a design . We found the schematics produced automatically were inappropriate for release to a customer. This required re-entering the schematic, at a higher level of structural abstraction than the automatic synthesis created. This is an area of potential improvement.

Pending Issues

Libraries A major issue for logic synthesis is ASIC library support. First the ASIC vendor of your choice must support the synthesis vendor of your choice. The level of library support will have a major impact on the resource utilization and performance of the synthesized ASIC. For example, the initial library available for the FPGA's were low level logic elements such as AND, OR, XOR, etc. Logic synthesized from these elements did not make optimum use of the Configurable Logic Blocks (CLB) which are the basic building blocks of the FPGA. Functional designs are created, but the performance and perhaps the size of the FPGA are not as good as results obtained through special purpose tools. Higher level libraries which contain larger blocks such as register, adders, Arithmetic Logic Unit (ALU)'s, and processor blocks will improve the optimization achieved by logic synthesis. In the case of FPGA's, libraries which are optimized to the CLBs should improve the synthesized results.

Integration of tools The conversions required could be minimized by further Standardization, particularly on standard logic types and conversion functions. The speed of developing interfaces between hardware vendor libraries and synthesis tools is a bottleneck which will hopefully improve as new standards develop.

General Conclusions

Conclusions are based on our own experience in using VHDL, top-down design methodology, specific ICs technologies, the design/synthesis tools we used, and the platforms available to our team. Some of our findings are swayed by our particular business and might not have general validity. We believe, based on our experience with the several tools which we evaluated and purchased that VHDL-Based Description and Synthesis is an effective approach to the design of large complexity ICs and should be extended to board level designs to reap more benefits at the system level where design decisions have the most impact.

We verified that the use of VHDL based tools made it somewhat easy to migrate the design to different CAD tools and platforms allowing us to capitalize on the strengths of each one, or allowing designers to work with multiple entry tools on the same design. The use of synthesis allowed us to quickly generate multiple implementations which allowed comparison of foundries, and technologies in days, rather than in weeks. The inclusion of FPGAs in the synthesis process helps to mitigate the risk perception of using ASICs and allows a common methodology to be used for either type of technology, reducing investment in special purpose tools.

We found that all of the tools we used in the evaluation had useful features. We are convinced of the benefits of VHDL synthesis design style and the general validity of VHDL as a standard for electronic hardware description.

References

[1] IEEE Standard VHDL Lnaguage Refernece Manual, IEEE INC, New York, 1988

[2] USGovernment OMB, "DI-EGDS-80811.VHDL DATA ITEM DESCRIPTION", Approved 5/11/89

[3] G. Shao , W.A. Hanna, "Soft Prototyping in the Design of Military Electronics," NAECON89, Dayton, Ohio, May 1989

[4] W.A.Hanna, G. Shao, and S.E. Thielker, " Generic ASIC Design: A Sensible Methodology for Rapid Insertion of VHSIC/VLSIC Technology", NAECON88, Dayton, Ohio, May 1988

[5] J.S. Lis, "Structured Modeling for VHDLSynthesis," Technical Report 89-14, University of California at Irvine

[6] S. Carlson, Introduction to HDL-Based Design using VHDL. California. , Synopsys Inc., 1991