Fully Synthesizable Microprocessor Core via HDL Porting

Microprocessors integrated in superchips have traditionally been ported from third-party processor vendors via artwork. A new methodology uses hardware description language (HDL) instead of artwork. Having the HDL source allows the processor design to be optimized for HP's process in much the same way as other top-down designs.

by Jim J. Lin

The level of integration has been rapidly increasing with advances in semiconductor technology. Many HP design groups have capitalized on this capability to create highly integrated ASICs, or superchips. Superchips that integrate conventional ASIC logic, microprocessors, embedded RAM and ROM, and other megacell functions save cost, power, board space, and inventory overhead and increase I/O performance at the same time. This industry-wide trend has put an increased burden on ASIC suppliers to come up with megacells that are off-the-shelf, proven, and testable. As an ASIC supplier to other HP divisions, we at HP's Integrated Circuit Business Division (ICBD) licensed several early microprocessors that customers demanded and artwork-ported them into our process. All superchips built at ICBD today contain artwork-ported microprocessors.

However, artwork porting has its limitations. It often does not yield the best area possible in a given technology. In addition, the process technologies of processor vendors tend to diverge from ICBD's technology for the next generation. The testability of these processors is also a problem in a superchip because they require access to their functional pins to run parallel vectors. Multiplexing the processor pins with ASIC pins is a level of complication preferably avoided. Customers also demand some controllability of the exact microprocessor configuration that goes into the superchip. For example, customers may choose to increase or decrease the size of the cache that's included with the microprocessor after profiling target code under different cache configurations. Making such changes at the artwork level is a major undertaking and often requires a lot of work for the processor vendor as well. Presilicon verification is also virtually nonexistent and the design often requires several mask turns. Finally, the processor is technology dependent and requires almost as much effort to go into a different process as the original port.

One methodology that successfully addresses these issues is HDL (Hardware Description Language) synthesis. A number of underlying technologies make this methodology work. The increased density in our standard cell technology allows the implementation of dense data path functions and is area-effective in general. ICBD's standard cell design flow for top-down design methodology is robust and mature. Synthesis is becoming more and more powerful and is capable of highly complex designs. ICBD's RAM generation is also a key component that delivers good performance for cache applications. Processor vendors in the embedded market have shifted their design paradigms as well. They are no longer custom designing everything and are using HDL and synthesis more and more.

The methodology of porting cores using HDL synthesis incorporates an existing standard tool flow. This is a major advantage. An efficient tool flow is essential in today's ASIC market. Significant effort has gone into making ICBD's as efficient as possible to handle the high-integration market. The goal for this methodology is to leverage the standard tool flow as much as possible. In essence, processor HDL is transferred from the processor vendor. After necessary changes in certain configurations at the HDL level, the HDL is verified through functional simulation. It is then fed into the synthesis tool to obtain a netlist that is subsequently fed into the conventional tool flow.

The rest of this paper will discuss in more detail the methodology and how it was used to implement the Coldfire 5202 microprocessor from Motorola in a test chip.

Methodology Overview

Since the processor cores developed from this methodology are going to be used in superchips, they need to be designed with ease of integration and customer needs in mind. Testability, customizability, technology independence, minimum die size, and thorough presilicon verification are all goals that are important to delivering a successful core for superintegration.

The testability of a processor core has different constraints than a standalone processor because the functional pins are not visible when the core is integrated on a superchip. Pins of processors have traditionally been multiplexed out in test mode, which takes additional effort, and fault grading the functional vectors is not always easy. Our new methodology uses full-scan test patterns that require only a few scan ports that are needed anyway for other ASIC logic and can be effectively fault graded to determine the quality of the vectors. This approach to testing is compatible with HP testing standards and minimizes the cost of testing.

Having the HDL for the processor means that changes to the processor can be done at the source level rather than the artwork level. While changes to the instruction set architecture are nontrivial and are not recommended by the processor vendor, changes to the cache and bus configurations are within the realm of possibility. Customers can often save area by cutting out an unused block on the processor or reducing the cache size. The functionality of the design can be verified before silicon to bolster confidence for first-time success in the customized processor.

Our methodology also enables technology independence through logic remapping. The HDL can be simply recompiled to target a different technology, assuming that the technology has a standard cell library capable of synthesis. This is true for porting a processor to a new technology at ICBD, a second source, or a dual source.

Area is a very important consideration as well. If the area of the synthesized core is not competitive with custom-laid-out processors, customers will not adopt this strategy regardless of how good the methodology. Standard cell density has increased to the point where even data path blocks like ALU, barrel shifter, and register files can be implemented in a reasonable amount of area. Another flexibility in using standard cells is that a processor core can be compiled with the desired target frequency. If the nominal frequency is faster than the target, cells can be sized down to save some additional area.

Having the HDL source for the processors also means that the design can be simulated, both at the RTL (Register Transfer Language) level initially and at the gate level at the end. Vectors can be run to verify the functionality and timing of the design.

To ensure functionality, three different approaches can be used, either alone or in combination. RTL and netlist verification can run precaptured vectors from the processor vendor. These vectors are diagnostics, benchmarks, or random instructions that processor vendors themselves use. The netlist can also be compared with the vendor's design using formal verification methods. Lastly, an environment in which random instructions are generated can be set up locally to subject the design to new random instruction testing.

Timing is verified with a combination of static and dynamic timing analysis. An even greater advantage for the customer is the ability for the entire superchip to be simulated in a timing-accurate fashion since the processor is in the same library as the ASIC logic. Previously, such system-level simulation was only possible using a hardware modeler or a software model that was not timing-accurate.

Design Flow

An ICBD design automation group has a series of supported design tool flows. A modified version of their Standard Tool Flow 2 (STF2) is used to carry out our methodology. In this way, our methodology leverages a proven methodology for doing HDL-based design. Synthesizing microprocessor cores becomes an extension of the current capability.

This paper focuses only on those aspects of the methodology that are unique to porting microprocessor cores. The process is illustrated in Fig. 1. The methodology incorporates standard ICBD tool flows. For Verilog-based designs, the STF2 is used. For VHDL (Very High-Speed Integrated Circuit Hardware Description Language) designs, an HP proprietary VHDL tool flow is used. These flows are simply encapsulated as design processes in Fig. 1.

Inputs. The processor vendor needs to have the following list of items to feed into our tool flow:

- Design Specifications. Timing, functionality, and pin descriptions of the processor. Most of this information can be obtained from a data book if available. For newer cores a data book may not be available, but internal documentation that will eventually be part of the data book will suffice.
- Behavioral HDL. A Verilog or VHDL model of the processor core. This HDL model does not necessarily have to be synthesizable. As long as it models the cycle-to-cycle behavior of the design, it can be made synthesizable with some rewriting of the HDL.
- Functional Vectors. Verification vectors in Verilog or VHDL format. These are run on their respective simulators. They can also be translated so that they can be run on the testers as well. These vectors enable presilicon verification.
- Synthesis Scripts (optional). These are used to synthesize the HDL into standard cells. They are only available for cores that have been previously synthesized.

Outputs. For use in superchip integration and product prototyping, the ICBD CPU team provides design groups and customers with the following:

- Megacell. Processor core with all requirements for inclusion in the HP intellectual property library. Deliverables include ERS, gate-level netlist, behavioral model, data sheet, etc.
- Test Chip Data. Mask, packaging, test vectors, and test program input for test chip fabrication and test. A core without a test chip will not have this.

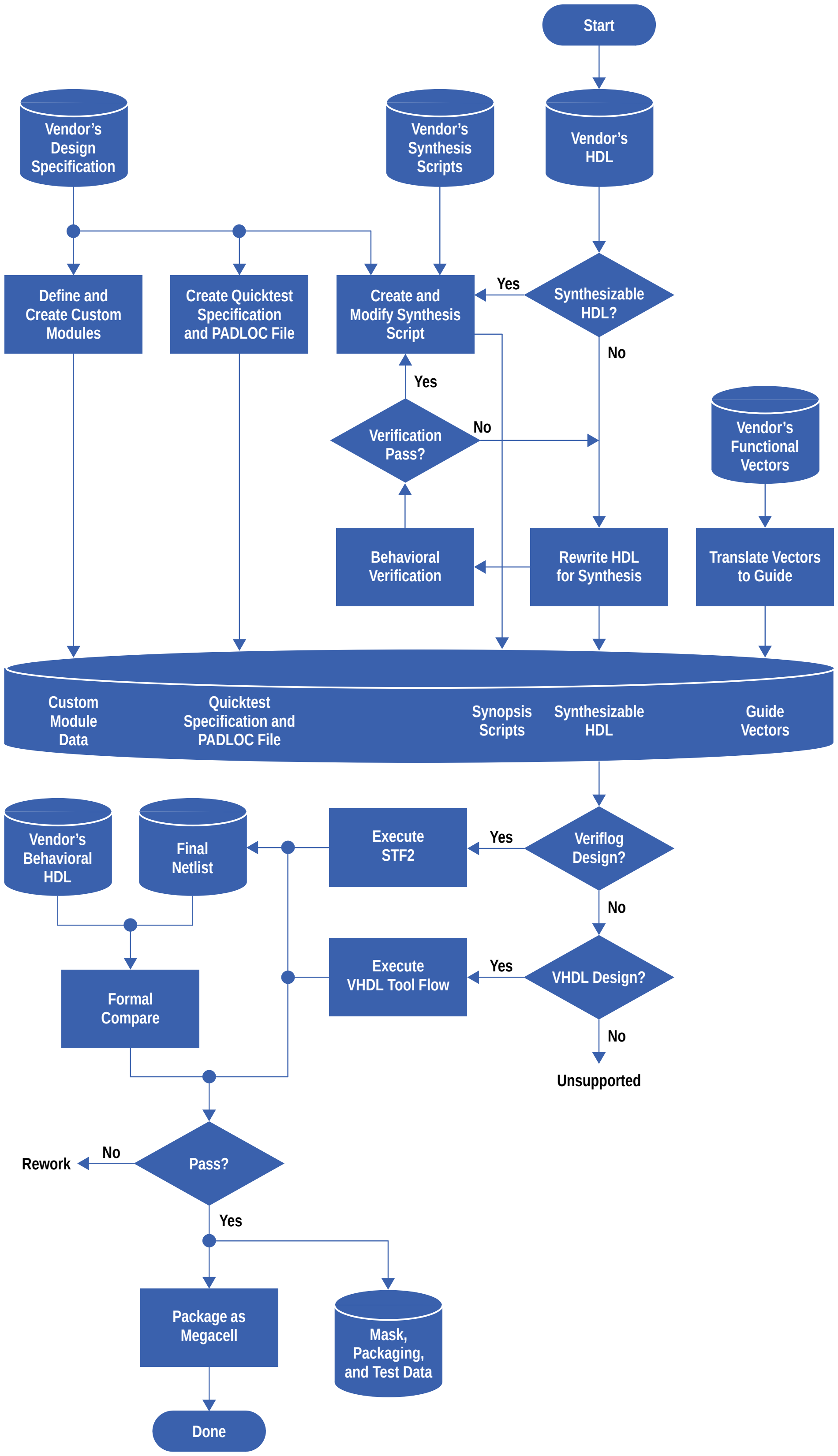

Fig. 1. *Processor core tool flow.*

Rewrite HDL for Synthesis. The HDL that is transferred from the processor vendor may not be synthesizable since it may have been written only as a model, not as a synthesis source. Current synthesis tools have limitations on the type of HDL constructs allowed and yield very poor-quality circuits for HDL not written with synthesis in mind. ICBD has a set of HDL coding guidelines that need to be followed when rewriting the HDL. This step may warrant some iterations in synthesis to explore the optimal mapping of specific behavioral constructs. Regardless of what changes are made to the HDL or the quality of synthesis achieved, the changes should not alter the functionality. In fact, any changes should be thoroughly verified by running regression vectors as discussed next.

Behavioral Verification. Even though behavioral verification is a part of STF2, this portion of the flow focuses on the extra verification that results from recoding for either customization or synthesizability. The verification is an extensive simulation of the altered HDL code with the vendor-provided vectors. In the future, this step may be augmented by formal verification. This subject will be revisited when the verification of the test chip (discussed later) is analyzed.

Create and Modify Synthesis Scripts. The purpose of this task is to create Synopsys Design Compiler scripts that can be used to compile the standard cell portion of the core consistently and systematically. As mentioned earlier, there may not be existing synthesis scripts if the processor has never been synthesized before. In other cases, there will be a full suite of scripts along with design constraints. In still other designs, portions of the control logic will have synthesis scripts and the data path will not. This represents the design methodology of many processor vendors.

In case synthesis scripts need to be created, design specifications and the actual HDL are the best sources. ICBD has a generic synthesis script that can be used as a template to help create these scripts. Even when all of the scripts exist, modifications are often needed to make them work in ICBD's environment and libraries. The processor vendor's approach to synthesis may not be the most efficient approach in ICBD's technology. Furthermore, the vendor's approach may not use the most up-to-date synthesis technique available in the latest release of the synthesis tool. Many trial-and-error runs of different approaches may be needed to determine the best synthesis approach.

Create Quicktest Specification and PADLOC File. Quicktest generates a test template given a target tester. The PADLOC (pad location) file is an input to Quicktest and other tools such as the router. This step is only needed if a test chip is planned for the particular processor core. The need for a test chip is determined on a core-by-core basis. The first core in an architecture, a core with major customization, and a customer need for prototyping are all reasons for a test chip.

The Quicktest specification is used to generate a test program for the processor core test chip. The Quicktest specification documentation describes how to create the Quicktest file from the design specification. The PADLOC file is used to place the test chip pads around the core logic. It contains placement data for the pad ring. The PADLOC specification document describes how to create a PADLOC file from the design specification.

Identify and Create Custom Modules. Not all blocks in a microprocessor can be implemented as standard cells, although the list of such blocks is becoming shorter and shorter. This task identifies all blocks within the core that should be implemented as custom logic and creates the identified blocks along with all the models and information required to use them in the downstream standard cell design methodology.

To identify blocks that should be custom, the HDL and design specification must be studied along with the design goals. Blocks are custom-designed to meet area or performance goals unachievable with standard cells. Typically these blocks will be limited to memory arrays. However, they may also include structured data path logic for implementing highly regular or speed-critical circuits, such as a large multiport register file, multiplier, or barrel shifter. This step may involve feasibility studies in which candidate blocks are partially designed or estimated for both standard cell and custom implementations. The following items must be produced for each block selected for custom implementation:

- Verilog or VHDL model with timing
- Synopsys timing with pin timing
- Cell3 LEF file (Cell3 is an automatic place-and-route tool from Cadence)
- Artwork database
- Sunrise (an automatic test vector generator from Sunrise) or ATG (an in-house HP tool similar to Sunrise) fault model.

Translate Vectors. By translating simulation-based verification vectors into Guide format, the standard path to testers and later portions of the tool flow is established. Guide is an in-house HP tester independent vector translation tool. This step may require that custom programs or scripts be written and supported to translate from unknown formats. Therefore, it is advisable to require the processor vendor to supply vectors in some known format like Verilog, for which there is a clear path to Guide. The vectors are needed for functional and diagnostic purposes only. Manufacturing test will not run these vectors and will rely on I_{ddq}, stuck-at, and at-speed scan testing.

Execute Standard Tool Flow 2 (STF2). As mentioned above, Standard Tool Flow 2 is a design flow supported by an ICBD design automation group. It includes Verilog simulation, Synopsys synthesis, Cell3 place and route, ATG and Sunrise full-scan testing, artwork and mask generation, and Quicktest and Guide test program and vector creation. There is extensive documentation on the entire tool flow. A variation of STF2 is STF3, which supports partial-scan testing. This may be an option for cores that would realize significant area savings from it without the loss of appreciable test coverage. The VHDL tool flow is derived from an HP proprietary VHDL design flow. So far, no core has been developed using this design flow.

Package Core as Megacell. The final step in the process is to create the data necessary to offer the core as a megacell to HP customers and ICBD design centers. The requirements for this type of product are currently being defined. The release of any core will adhere to the standards set up and provide all the models, documentation, and support required.

Test Chip Experience

The Coldfire test chip (Fig. 2) is a test chip implementing the Coldfire 5202 processor from Motorola. Coldfire is a new line of embedded microprocessors that improves performance over the 68000 architecture while maintaining compatibility with most of the 68000 instruction set and minimizing area. The Coldfire 5202 has a 2K-byte 4-way set-associative cache, a debug unit, and JTAG capability (IEEE 1149.1 boundary scan test capability) along with the core as implemented by Motorola.

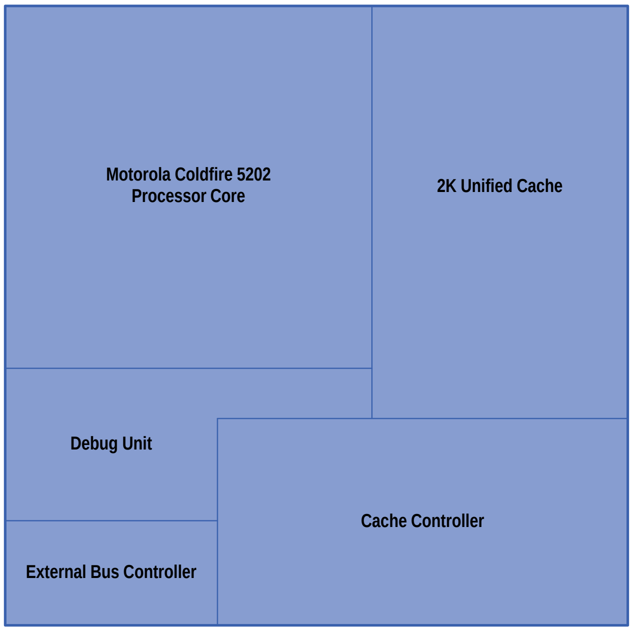

Fig. 2. *The Coldfire test chip floor plan.*

Design Transfer

The Coldfire test chip team received a brief course on the architecture and a tape of the HDL source for the Coldfire 5202. The tape contained synthesizable HDL for every block in the design. The cache memory blocks and some tag maintenance logic were custom-designed at Motorola and had only a behavioral representation. Each synthesizable block also had a constraint file used in Synopsys. There was also a top-level synthesis script. This represented all that was needed to get started with the port. Subsequently, Motorola has sent test vectors and fielded questions from ICBD. The level of support at Motorola has been very good.

Making the Coldfire Test Chip

The Coldfire test chip has been targeted as a test vehicle for this methodology and offers early prototypes for customers interested in using the Coldfire 5202. The Coldfire test chip is implemented in a 0.5-μm technology and takes advantage of the process's increased density. A fairly high frequency of 50 MHz was targeted to show the scalability of the methodology.

Custom Modules. The team knew at the outset that Motorola had designed custom cache RAMs and tag logic and that custom designing caches for every core would not fit into our methodology. The generated WEST SRAM available from an HP RAM group provided the solution. These RAMs are designed for ASIC integration and good performance. However, to use these RAMs, there were two extra requirements that were specific to this cache. First, it had to be byte-writable, that is, each byte of a multibyte word must be written separately. The second requirement is that the bits of one column needed to be reset in one cycle. This is used to invalidate the cache valid bits at startup or in case of an invalidate instruction. The RAM group took the first request and incorporated byte writability (actually bit writability as implemented in the artwork) into the generator. The one-cycle invalidate was not changed in the generator. Instead, the valid bits were turned into flip-flops on this test chip for schedule reasons. This incurred an area penalty and will hopefully be fixed in the RAM in the future.

Rewrite HDL. The interface to the cache RAM was asynchronous in the Motorola implementation. The WEST SRAM is synchronous. This means that addresses, data, and other control signals need to be set up before and held until after the rising edge of the clock. Motorola latches the various signals on the rising edge of the clock before feeding them to the RAM. With this scheme, the signals would have missed the setup time on the WEST SRAM. If no latch is used, then the hold time cannot be met. As a result, negative-level-sensitive latches are used to provide the necessary hold requirement. The control signals that Motorola uses are sufficient to generate the WEST SRAM control signals. The 4-way set-associatively can be easily implemented as four data RAMs and four tag RAMs, each representing one way in the cache organization.

Another change made to save some area was the removal of the JTAG block. The JTAG methodology does not fit into the superintegration process since this process has its own mechanism of doing boundary testing. The JTAG logic is a fairly modular block that can be stripped out with minimal perturbation to the rest of the design. The necessary logic needed in place of the JTAG block is encapsulated in its own level of hierarchy. The necessary JTAG-like functionality has been replaced by a scan wrapper that better fits ICBD's superchip test methodology.

One final change is in the area of testing. Motorola uses a test mode called ad-hoc mode to test the cache RAM circuitry. ICBD uses BIST (built-in self-test) instead. As a result, the logic that makes the cache RAM controllable and observable and interlocks the pipeline is no longer needed. By rendering the ad-hoc mode inactive, all logic associated with this test mode can be minimized.

Synthesis Script Modification. The synthesis script that came from Motorola put highly detailed constraints on each block. The reasons are twofold. Motorola was concerned with the speed of the synthesis job if the entire design were compiled at once and wanted to be able to do most of the compilation at the block level. Secondly, Motorola had good ideas about where Synopsys should be spending its time and wanted to influence the tool in that direction. A makefile generator was used to piece together the numerous block-level compiles and their respective constraint files. This approach does not necessarily yield the best design in ICBD's technology, as was found during initial synthesis trials using Motorola's script. The block-level constraints were often unrealistic and Synopsys was spending optimization cycles on the wrong circuits. As a result, the synthesis scripts were overhauled to put constraints only at the top level, that is, the I/O specifications of the chip.

A hierarchical compile at the top level replaced the block-level compile. As a result, Synopsys has more freedom in partitioning the time. The compile time is long but not intolerable. The entire synthesis job from reading in HDL to outputting an optimized netlist at 50 MHz takes 48 hours on an HP 9000 Model 755 server. For fast turnaround needs, such as testing a quick fix of a bug, block-level constraints obtained from hierarchical characterization and the write script of the previous synthesis run can be used to compile at the block level. To get even better area, the entire design has been compiled with the hierarchy flattened so that interblock optimization can be performed during synthesis.

Verification. All the changes mentioned earlier have been simulated extensively to make sure that the desired functionality is achieved without having broken some other part of the design. Once the functionality is determined, then the HDL is synthesized to obtain a netlist from which both static timing and dynamic timing analysis are done. The majority of the emphasis in timing verification centers on static timing analysis. Synopsys' timing analyzer is used to generate timing reports. False paths and multicycle paths have been carefully reviewed to make sure that there is no escaped path in the report. Both maximum and minimum paths are reported to expose possible setup and hold violations.

The vectors run on the design include benchmark, diagnostic, and RIS vectors from Motorola. Motorola has developed a sophisticated random instruction sequence (RIS) generator that can be tuned to generate instructions in an area of interest along with random interrupts and exceptions to perturb the processor. In future Coldfire cores, the ability to generate RIS vectors will be incorporated into the verification process. This time, Motorola has generated all the RIS vectors and sent them to ICBD. Verification using a more formal method of binary decision diagram comparison has also been pursued using Motorola's in-house tool. This step will not be available for every core since most processor vendors do not support this methodology.

Simulation of the netlist had some hurdles. One is the inability of the netlist to reset properly. This problem has its roots in the way the reset logic was done in the HDL. Instead of using an explicit reset inference on all flip-flops, the reset logic became part of the input logic. Depending on where the reset was structured in the logic, it might or might not cause a particular flip-flop to reset correctly. In fact, this problem is more general. Every time a reset-like signal is used, unknown states (Xs) are not guaranteed to be suppressed. Unknown states are periodically introduced into the design by captured vectors that use uninitialized memory for operations. For example, an uninitialized stack memory may be used to fill a cache line and pushed back to memory. Granted, this is a mere simulation issue. However, it makes verification harder because only after these issues are fixed can other real problems be visible, because problems that would have been masked can then be caught.

Motorola will restructure their HDL to avoid this problem in the future. However, for the Coldfire test chip, several steps have been taken to remedy the problem. Every flip-flop and latch in the design is reset using a force-and-release pair upon startup. When unknown states are introduced into the system, the pattern is intercepted and given a random value instead. Since unknown states are essentially conditions in which the state of one or more bits is unknown, randomizing these patterns effectively gives an unknown pattern without the simulation nightmare.

Test Strategy. To make the test work with STF2 as mentioned earlier, the core is full-scan except the register file, the instruction buffer, and the latches in front of the cache RAM. The latches can also be tested using the latest methods from ATG and thus offer virtually no degradation of the test coverage. The cache RAM has BIST circuitry testing all eight RAMs in parallel. The BIST mode is encoded in the test mode pins available on the Coldfire 5202. A limited number of functional vectors run in verification are also ported to the testers.

Technology Independence. The entire core is technology independent. The only technology dependent portion of the Coldfire test chip is the pads. Since the prototypes are targeted to be used in Motorola's 5V evaluation boards, the 3V and 5V pads in the HP CMOS14 library are used. Since only I/O pads exist, input and output only pads are made by tying off the appropriate enable signals. The pads are instantiated only for the test chip. Synthesizable HDL versions of the pads do exist and can be synthesized to buffers when the megacell becomes available.

Results

The Coldfire test chip is the first trial of the proposed methodology. The performance target of 50 MHz has been met with no custom cells or modules. Both small die size and high test coverage were achieved by this chip. Higher row utilization is only limited by extreme congestion spots like the barrel shifter, register file, and pipeline control block. An even smaller version is possible with a few modifications in key areas. In addition, future versions are not expected to have the overhead of the invalidate registers implemented as flip-flops. The gates may also be resized to meet the actual target frequency.

The desired changes have been successfully implemented. Motorola's custom cache was turned into synthesizable control and generated WEST SRAM. The JTAG has been removed with minimum changes to the original HDL. BIST and test circuitry have been added. All of these changes have been verified at the functional and netlist levels. Being able to make changes at these levels and maintain high confidence in the design is an invaluable advantage with this approach that would not have been possible with artwork porting.

Data management that is needed to maintain the coherency of the design is an important aspect of the project that cannot be overlooked. Problems in this area occurred fairly early in the project. Scripts were written to make use of lineup files, that is, lists of designs with specific revision numbers that go together for a particular simulation or synthesis run. Changes that are not yet released are made in private directories that can be part of a private lineup file. The massive verification effort requires jobs to be run at every available time, using every available open Verilog license. Scripts have also been written to use HP Task Broker to get maximum efficiency of the available resources.

Conclusion

Porting processor cores using the new ICBD methodology of standard cell synthesis has been shown to be a viable alternative to the traditional artwork port. HDL porting has the advantages of testability, technology independence, customizability, efficient area use, system simulation capability, and presilicon verification. It is also a straightforward methodology to support since virtually all components of it are already in use in the HP Standard Tool Flow 2.

The approach has its disadvantages. It cannot be applied indiscriminately on any processor core. Many cores designed today still do not have synthesizable HDL. The synthesizability of the core may also run the gamut from being very easy to extremely difficult, depending on a host of issues such as clocking strategy, coding style, and architecture complexity. The need for customization puts even higher expectations on the quality of the HDL. Trying to change the functionality of a design written with raw Boolean equations and flip-flop instantiations is almost as daunting as editing a netlist. Therefore, the selection of a microprocessor vendor may depend on the vendor's design methodology. For cores that do not have synthesizable HDL, artwork porting may still be the only option.

HDL porting will become increasingly feasible with better synthesis tools and denser and faster technology. The advances in these two areas have now reached a threshold at which implementation of entire microprocessor cores with standard cells compiled using HDL synthesis is practicable. As more processors are designed using HDL and synthesis, this methodology will become more general. As the speed of the technology increases, the level of processor performance achievable using this methodology also increases. Silicon compilation is slowly becoming a reality. IC porting in the future should reach a level similar to porting software today, as designs are targeted to different technologies with a few changes in the synthesis and constraint scripts.

Acknowledgments

The author would like to thank the entire Coldfire team: Mark Reed, Paul Chenard, Tom Thatcher, and especially Jay McDougal for his work on synthesizable core methodology and synthesis techniques. Thanks also go to Tim Brown and Neal Jaarsma for looking into the superintegration and test aspects of the core. The team appreciates the support from the Motorola Coldfire team, especially Jeff Freeman and his timely responses and advice.

▶ Go to Next Article

▶ Go to Journal Home Page