CSRCompiler and a Total Register Automation Methodology

Richard Weber and Jamsheed Agahi
Semifore, Inc.

Executive Summary

Semifore's CSRCompiler provides the scalable infrastructure to automate all aspects of the hardware/software interface (HW/SW interface) requirements throughout the entire design organization, enabling best practices, and removing technical and business risks. The Semifore Total Register Automation Methodology propagates the HW/SW interface information to ensure successful product delivery in an efficient, functionally correct, and timely manner.

The HW/SW interface has both grown in complexity and become an essential design component used by architects, RTL implementers, verification engineers, software developers and technical writers. Each discipline requires access to the HW/SW interface information in a specific format to complete their deliverables. In development organizations the HW/SW interface specification and implementation is typically very dynamic, and requires multiple iterations based on feedback from the various teams.

Given this exponential increase in complexity and the need for better communication between the different teams, Semifore created CSRCompiler, the single source CSRSpec language, and the associated Total Register Automation Methodology specifically to provide the following benefits:

- **Advanced single source specification language**
- **Early engagement by all teams in the definition phase**
- **Instantaneous access to customized views for all teams**
- **Elimination of errors due to manual processes**
- **Continuous and fast iteration capability**

Challenges of the Hardware/Software Interface

Over time, the HW/SW interface has grown significantly in size, complexity, and importance - growing from a few 8-bit registers in the ubiquitous UART to thousands of registers in sophisticated networking devices. Many of today's products have a very large software component to implement required functionality in a design which creates a heavy communication requirement for continuous feedback loops on changes between the Software, Verification, and Hardware RTL teams.

The effort and resources for the development of the software component of a product has exceeded the corresponding effort in development of the hardware. The commercial availability of a product is now largely dependent on the completion of the associated software.

HW/SW interface complexity also affects design verification. The RTL verification team requires the information in exacting detail to verify correctness of the interface itself and the hardware function controlled by the interface.

The traditional practice has been for the software, verification, and documentation teams to independently generate their required HW/SW interface information based on a design specification, which is time consuming and error-prone. This approach has the added disadvantage that, as the design goes through refinement iterations, each team must again independently propagate those changes to their respective environment. This propagation process involves either manual intervention or the use of internally-developed scripts.

But manual efforts or even internally developed scripts not only fail to keep up with the size and complexity of the HW/SW interface, they also put project completion at risk.

Manual entry has the following disadvantages:

- **High potential for errors**
- **Serial process**
- **No automated error checking**

Internally developed scripts have these disadvantages**:**

- **Limited functionality and excessively long run times**
- **Not sophisticated enough to ensure correctness and quality of results**
- **Difficult and time consuming to develop, and expensive to maintain**
  - **Undocumented use model, features and commands**
  - **Knowledge base tied to developer, not institutionalized inside the company**
- **Poor utilization of expensive engineering talent**

All these activities divert resources which would be better spent on the core product development.

Typical Product Design Cycle Involving the Hardware/Software Interface

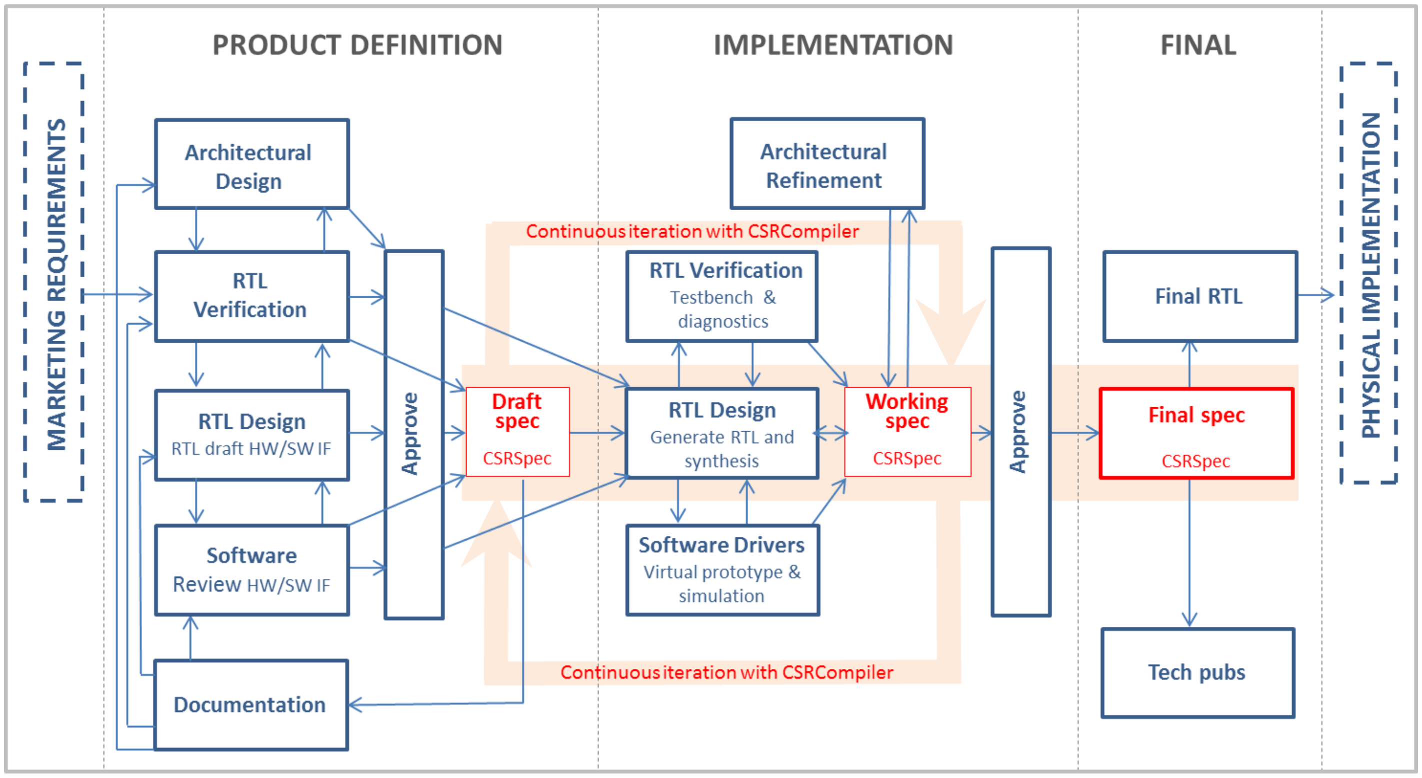
A project goes through initial product definition and RTL implementation before arriving at the back-end physical implementation. The initial definition defines the product from marketing requirements document (MRD), and sets the starting point for front-end implementation. A strong initial starting point is required to provide a foundation for a successful design implementation and eventual product release.

During the initial draft of the design architecture resource requirements of the RTL design, RTL Verification, and Software development teams are estimated. The teams iterate on the definition of the product until a consensus is reached among the teams (Hardware, Verification, Software and Documentation) that the design implementation can begin.

During design implementation, detailed RTL implementation, RTL verification, and software development is performed. Changes are often introduced from new marketing requirements, refactoring due to challenging constraints, or errors in the product definition or initial implementation. It is not unusual to have to give up functionality during this phase to ensure deadlines are met. The teams need to iterate on the RTL implementation and software driver development until design implementation is complete and satisfactory design verification has been accomplished. The design is then ready for physical implementation.

The Semifore software infrastructure, comprising CSRSpec and CSRCompiler, was developed specifically to improve every step of this process.

Semifore Total Register Automation Methodology flow



Product Definition Phase

During the product definition phase the basic parameters and architecture of the product are defined. The Architecture Team specifies the basic layout of the product, and makes the initial decisions on how the features will be partitioned between hardware and software. This initial partitioning provides the foundation for the HW/SW interface.

Semifore Methodology – Best Practices:

- ✓ **Productively involve representative members from Architecture, RTL implementation, RTL verification and, especially, Software development teams.**
- ✓ **Capture the hardware/software interface features using a single source: CSRSpec.**
  The RTL implementation team specifies the microarchitecture of the product. A large part of the microarchitecture involves the detailed specification of the HW/SW interface. The RTL implementation team captures the intended address map layout, including the software accessible registers and fields, using the CSRSpec language which was specifically designed to define and capture all the requirements of the HW/SW interface in a single source file. All future modifications will be made using this single source.
- ✓ **Engage the Software Team from the outset**:
  This will enable the software team to analyze the choices made by the Architectural and RTL implementation teams and to determine if the product objectives with regard to features, performance and schedule can be met. It is essential that the software team substantiates that the HW/SW interface is architected in a way to make it possible to develop drivers in an efficient way.

Implementation Phase

During the implementation phase most of the actual work is performed. The RTL implementation team creates the RTL design. Since the features of the HW/SW interface are captured in the single source CSRSpec language, the CSRCompiler automatically generates the relevant RTL implementation.

The RTL design team receives feedback from the RTL verification team about the completeness and quality of the design compared to the required product features and performance. In response to feedback from the software and verification teams, the RTL team has to make changes to the specification of the hardware/software interface which will have to propagate back to all other teams.

CSRCompiler provides the continuous iteration capability that the inevitable multiple changes demand. The architectural team continues to verify the design's compliance with the feature and performance requirements. The architectural team can use finer detail as it becomes available from the RTL implementation team.

Semifore Methodology - Best Practices:

- ✓ **Use the CSRSpec language to capture, in a single source, the features of hardware/software interface.**
- ✓ **Use the CSRCompiler to automatically generate the RTL associated with the address map registers using the CSRSpec input file.**
- ✓ **Have as much work as possible performed in parallel, especially hardware implementation and software development.**
- ✓ **Gain early access to software header files from the CSRCompiler for virtual prototyping.**

Benefits include:

- ✓ **Terse input expanding to functionally correct, synthesizable RTL output**
- ✓ **Syntactically and semantically correct target output generation**
- ✓ **Immediate availability of all appropriate views in correct format**
- ✓ **Fast turn-around times, allows multiple iterations per day**
- ✓ **CSRSpec language provides functionality not available in other formats such as SystemRDL, IP-XACT, spreadsheet**
- ✓ **Configurability and customization of target outputs**
- ✓ **Parallel implementation**

RTL Verification

Given today's feature-packed and complex designs, verifying the RTL design becomes especially challenging. The introduction of the various SystemVerilog-based verification methodologies has met this challenge to some extent, but the various components of the testbench (and its subsequent modifications) and the tests still have to be created by the verification team. Automating the generation of the register abstraction model will greatly enhance the efficiency of the verification team.

The CSRCompiler generates, from the single source CSRSpec language, the HW/SW interface register abstraction layer based on the UVM register class library. VMM and OVM are also fully supported. This model may contain any combination of the following:

- **coverage model**
- **sequences**
- **constraints**
- **backdoor paths**

As the HW/SW interface changes, using the CSRCompiler to generate the relevant register abstraction model guarantees that the verification team has instantaneous access to both the up-to-date RTL model, and the resulting changes for the testbench. This eliminates working with a stale design, or discovering later that certain changes in the design were not reflected in the testbench.

Software Development

The software component of any product development today far exceeds the effort required for RTL design. Serial development of the HW/SW interface does not support today's time to market imperatives. The objective should be to engage the software team in the architectural phase of the product definition, as well as provide the team with early access to the HW/SW interface information in order to enable virtual prototyping. The CSRCompiler generates the 'C' header file, reflecting the software view of the HW/SW interface, based on the single source CSRSpec file. It also automatically generates the RTL model of the interface, thus enabling the software team to build a virtual prototype of the design while the RTL design is being completed.

Benefits include:

- ✓ **Fostering engagement and communication during the architectural phase**
- ✓ **Allowing early development of device drivers**
- ✓ **Having early access to the RTL model to perform virtual prototyping**
- ✓ **Enable full system bring up soon after silicon is available**

Implementation Documentation

The technical writing team develops the product documentation. The CSRCompiler automatically generates the detailed address map register documentation in HTML, Microsoft Word, Adobe Framemaker or DocBook XML. During the iterative process engineering talent can use either interactive web browsing or Word documents to view design status or information.

Whenever the design team makes changes to the design the new versions of the documentation files are immediately available. The single source CSRSpec language provides the flexibility to customize partitioning of the documentation to manage access control internally for design teams. The technical writing team has the ability to override the descriptions in the documentation and can filter the content based on internal consumers or external customers.

Conclusion

As market requirements for products with ever greater performance drives increases in design complexity, the data required for the definition of the Hardware / Software interface has grown exponentially. It is imperative that design organizations find some form of automation to help their design teams navigate the cross team input and feedback loops needed for the definition, specification, and implementation for products. Lack of robust standards to assist design teams in developing a consistent methodology to turn marketing requirements into a delivered product also challenges design organizations to create a design flow that can deliver products on schedule with all of the features, functions, and performance per market requirements. This is even true for products that will be available by reuse of internal design content, or other third party IP, in the next design.

However very few design organizations inside of electronics companies have the in-house expertise to complete the software development for this design niche, or have the money and time to develop their own solutions within the product design project schedule. Internal scripts provide partial translation solutions, which are often not documented, are unsupported and thus difficult to use, maintain, or enhance in long run. Automation software products, such as CSRSpec and CSRCompiler used in the Semifore Total Register Automation Methodology must be employed to capture the HW/SW interface features of the design in a single source. This enables communication between the RTL, software, and verification teams, and fosters parallel software/hardware development, resulting in system bring-up soon after silicon is available.

CSRSpec and CSRCompiler allows for a significant increase in the multiple Iterations of the implementation phase, and provides all teams with virtually instantaneous access to the up-to-date HW/SW information that provides the design teams to meet ever constricting development and market delivery schedules.