Register Management with CSRCompiler™

1 Introduction

The CSRCompiler™ is an information processing system that manages a design's register and memory specification. The high-level architectural view of this information is an address map, sometimes referred to as a memory map. An address map is a list of offsets linked to objects. These objects can be memories, registers, or other address maps. The address map specifies and organizes the software accessible objects within a design. The complete register specifications include implementation information about the function of the registers and fields and any side-effects due to software access. An address map example is shown in Figure 1.

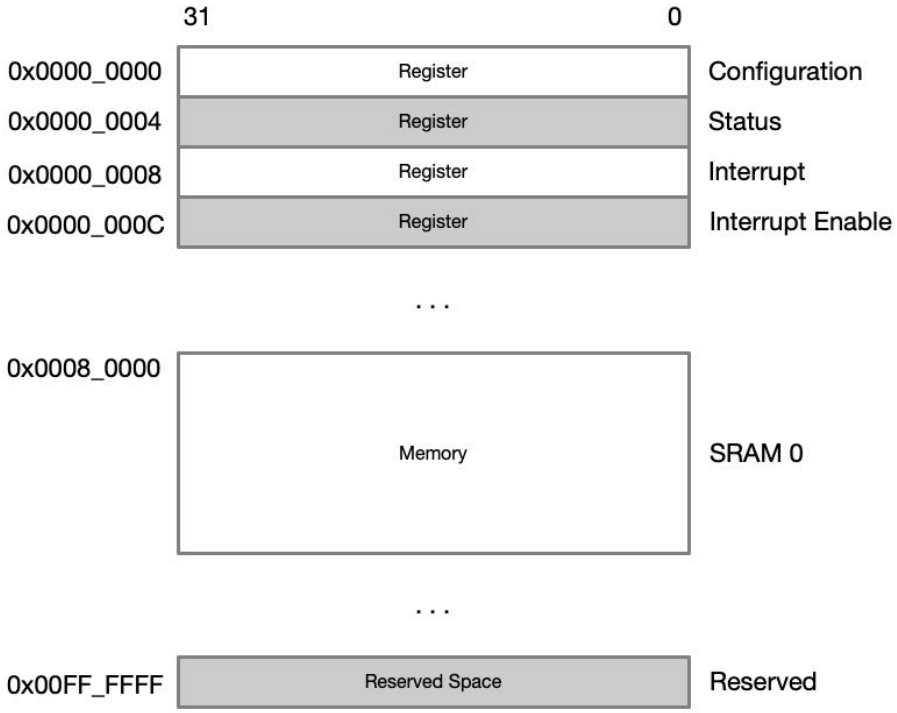

Figure 1: An Address Map Example

Address maps often contain other address maps, called hierarchical address maps. Leaf address maps refer to maps that are a child of another address map. The complete address map of a design is called a full or top-level address map. Figure 2 is an example of a full map with hierarchical and leaf maps within it.

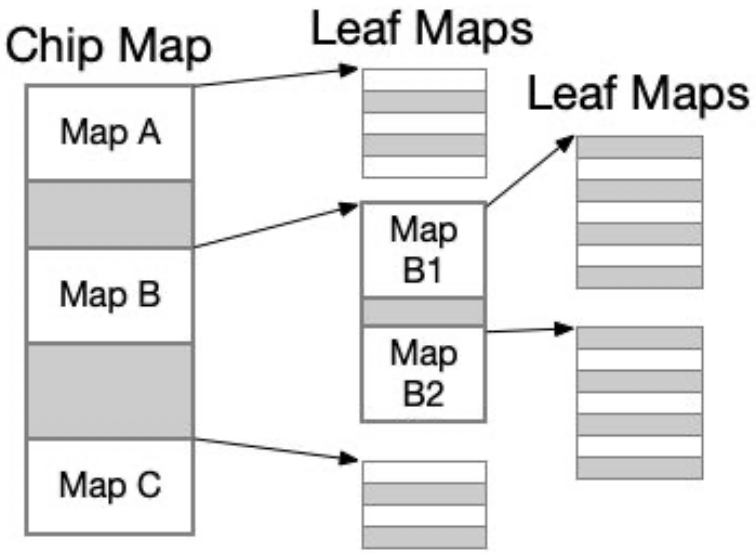

Figure 2: Hierarchical Address Map

2 Who Needs Address Map Information?

The whole team. Each discipline involved in a design needs current information about the address map. While everyone on the team can benefit from up-to-date documentation, each group requires the address map knowledge in a different format. The design team needs synthesizable RTL, the verification team needs the RTL and the UVM class definition or header files, and the software team needs the C header files. At the same time, the technical publishing team needs a well-formatted document to use in the final manual to provide to the end-user, which may be different from the HTML. CSRCompiler can provide the required format for each team's needs.

3 Inputs to CSRCompiler

CSRSpec™

The CSRSpec™ language is a domain-specific language for capturing address map information. It provides a single source for specifying the address map architecture of a design. Each object can have a set of properties that document or define the object's behavior. CSRSpec also has object templates to provide consistent, re-usable, and parameterizable architectural definitions. The language is a super-set of the various input formats typically used for register implementation.

CSRSpec Address Map Object

The CSRSpec language provides a container object for address maps. This container is specified with the keyword `addressmap`. The RTL implementation of an address map contains the bus interface, address decode, a read mux, registers, and fields. The RTL implementation of an address map is a Verilog module or a VHDL entity. An address map is represented in a C header file as a set of text macros and/or a set of structs.

CSRSpec Register Object

The CSRSpec language provides the `register` container object for fields. It has an address and contains fields. The RTL implementation of a register concatenates the fields with zeros provided for undefined positions. A register is represented in a C Header file as a set of text macros providing the reset values and a struct member.

CSRSpec Field Object

The CSRSpec language provides the `field` object to represent a register field. A field is a single bit or a consecutive group of bits that detail a single function. A field has a position within a register. The RTL implementation of a field depends on the specific architecture and any optional properties that may affect its function. A field is represented in a C header file as text macros that provide the position, reset value, masks, and access macros. An optional representation is a bitfield declaration in the struct for the register.

CSRSpec Property

Within CSRCompiler, property refers to a specification element for an object. It just says something about the object in question. For example, this field is an interrupt. These properties specify the behavior and features of an object within CSRSpec. In CSRSpec, there are over 250 properties. With properties, an `addressmap` object can have various bus protocols, including most AMBA buses. Properties can also add pipeline stages to the input and output ports.

A property can also specify a field type. Each field type implies a specific hardware behavior.

Certain properties facilitate behaviors that are required by software. For example, software may require atomic access to a register to avoid loss of information or accuracy. CSRSpec provides properties for specifying atomic access to a register.

IP-XACT

CSRCompiler can read and write IP-XACT. IP-XACT is an XML format used as an interchange format for IP. IP vendors provide it to their customers to specify the information about an IP block. It contains bus protocol information, memories, registers, fields, and ports.

SystemRDL

The CSRCompiler can process and produce SystemRDL files. The syntax and style of SystemRDL is similar to CSRSpec. The SystemRDL standard has the objects: `addrmap`, `regfile`, `reg`, and `field`. These objects map to CSRSpec `addressmap`, `group`, `register`, and `field`, respectively.

Spreadsheet

The CSRCompiler can read data from typical spreadsheet applications like Microsoft Excel. Each row of the spreadsheet represents one object in the register architecture, and each column represents a property of the object. The first row of the spreadsheet contains the titles of the columns. The column title specifies which property is defined in the column. The columns may be in any order and can have

custom titles.

4 RTL Implementation

Implementing the RTL of an address map needs to include a bus interface, address decoder, any optional error handling, read and write access, and any hardware-accessible ports shown in Figure 3. CSRCompiler is provided with enough detail to create a synthesizable RTL implementation of the address map, including these features. The generated RTL may be in Verilog, SystemVerilog, or VHDL. It includes the AMBA bus family, Wishbone_3b, Avalon_3_2, OCP_2_1 and a simple basic protocol that can bridge to a custom protocol unique to a design.

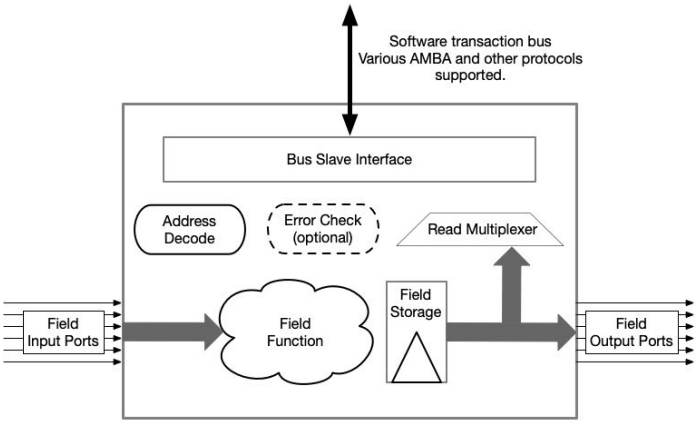

Figure 3 CSRCompiler RTL Implementation

5 Conclusion

We have explored the details of how to use the CSRSpec language and CSRCompiler to implement the address map and associated collateral reliably. Semifore focuses exclusively in this area. We are well-known for supporting the implementation and well-documented interfaces between the hardware and software for complex designs. We welcome the opportunity to discuss the needs of your design team and how we can help them be more productive. CSRCompiler creates a solid design foundation, allowing the design team to innovate confidently.

If you'd like to explore the possibilities, please contact us.

Semifore, Inc.
(650) 960 0200
sales@semifore.com
www.semifore.com