Process improvements using OVM Registers and IDesignSpec

by Anupam Bakshi, CEO, Agnisys Inc.

Abstract

This paper covers the benefits of deploying an OVM-IDesignSpec based methodology for design and verification. It discusses the testbench automation provided by the OVM Register package. It shows the advantages of using the new libraries from Mentor in conjunction with a new tool called IDesignSpec from Agnisys.

This combination of new technologies enables design teams to realize cost savings, quality improvement and TTM benefits across the engineering organization.

The Problem

SoC design is fraught with many challenges, the most important among them being IP integration and verification. The IP integration challenge is compounded due to the complexity of multi-million gate designs with a variety of programming modes. This is exacerbated when the various IP providers, be they in-house or external, provide inadequate programming guidelines and documentation. The growing trend to make the IPs more reusable and configurable adds to the complexity. At the hardware level, this programming configurability is achieved by an ever increasing number of programmable registers.

While designing such a device becomes a challenge, verification becomes an even more daunting task since there could be 100s if not 1000s of registers that require verification.

Another daunting challenge is managing changes through the design process. It is typically very troublesome. Without automation, design teams fear change so much that any deemed change has to go through a formal approval process which in turn causes numerous schedule slips.

Proper specification documentation isn't just required for design but for verification, firmware, device driver, lab debug, diagnostics, application software and even technical publication. Keeping up-to-date documentation for the design is a non-trivial task. Yet it's vital to have documentation that correctly describes the device behavior.

Can good documentation solve the Problem?

Yes, but designers want to design and not document. Let's face it, documentation is a chore no one really wants to do -- but someone has to. After all who wants to spend time creating a document that becomes outdated the moment coding starts!

Can designers be forced to produce good documentation and keep it in sync with the code?

Yes, but the reason designers resent documentation is because of duplicated work. Why should they be asked to document and code the same information at the same time? Not to mention the added responsibility of managing the two separate information sources as changes take place through the design cycle.

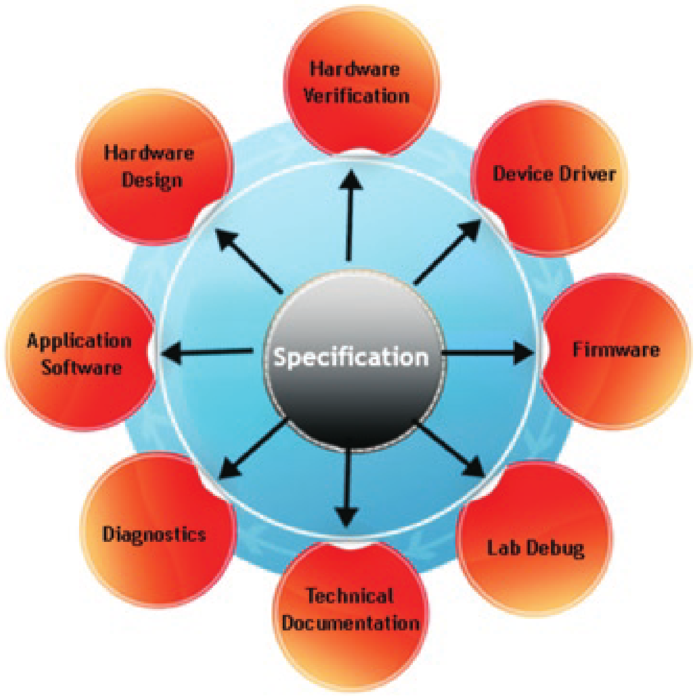


Figure 1: Specification affects everyone.

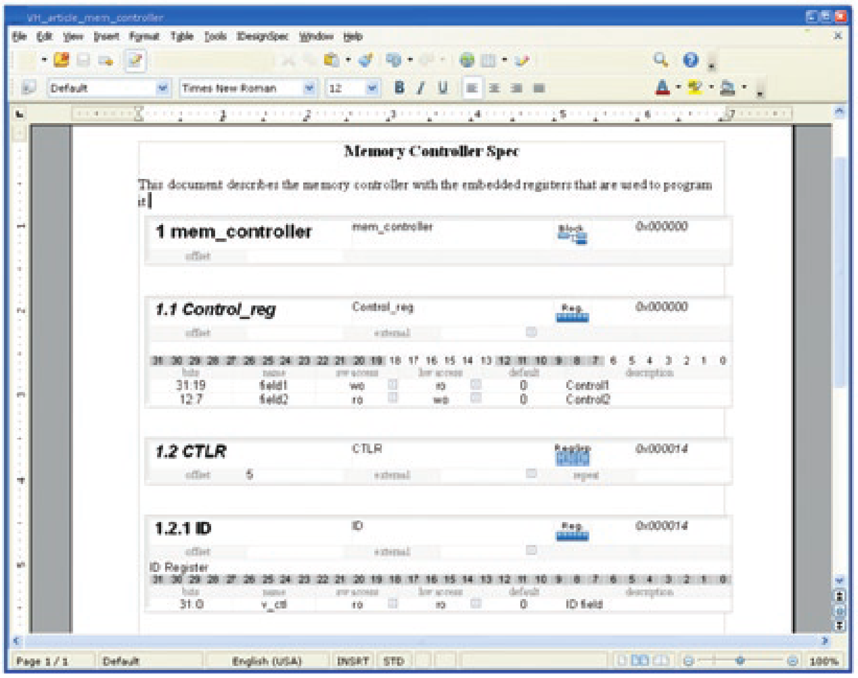
The Solution

The solution is to have a single source for the functional or design specification along with the register information. Based on our experience in working in design and verification projects we found that its no fun creating registers by hand. It is a laborious, error prone process. We use IDesignSpec ™ to capture the specification and the register information and generate all design and OVM based verification code from it. It is available as a plug-in for editors; it enables users to embed register information right inside the functional specification.

Document driven Verification Methodology

IDesignSpec provides design teams a way to capture and extract register specifications within a design specification document. This methodology fits right into any system that the users may already have in place. Most IP providers already create IP-XACT based register information. IDesignSpec can read in IP-XACT and CSV files with Register data and generate SystemVerilog files that form input to the OVM based register verification environment.

The following figure shows what a specification with embedded registers may look like in IDesignSpec. Note that the text is free form and the user is able to add any formatting or graphics enabling free expression of ideas. The proximity of the specification to the register means there is no need for excessive cross referencing.


VH_article_mem_controller
File Edit View Insert Format Table Tools IDesignSpec Window Help
Default
Times New Roman
12
Page 1 / 1
Default
English (USA)
INSRT
STD
100%

Memory Controller Spec

This document describes the memory controller with the embedded registers that are used to program it.

1 mem_controller	mem_controller	Block	0x000000
offset			

1.1 Control_reg	Control_reg	Reg	0x000000
offset		external	

bits	name	sw access	hw access	default	description
31:19	field1	wo	ro	0	Control1
12:7	field2	ro	wo	0	Control2

1.2 CTLR	CTLR	Regfile	0x000014
offset	5	external	repeat

1.2.1 ID	ID	Reg	0x000014
offset		external	

ID Register

bits	name	sw access	hw access	default	description
31:0	v_ctl	ro	ro	0	ID field

Figure 2: Specification with embedded registers in IDesignSpec.

Here is part of the OVM output generated by IDesignSpec

```
//*** This file is auto generated by IDesignSpec
 (http://www.IDesignSpec.com) . Please do not edit this file. ***
// generated by      : anupam
// IDesignSpec rev   : 1.1.0

// block-name: memcontroller

 typedef struct packed {
   bit [31:19] field1;
   bit [18:13] padding13;
   bit [12:7] field2;
   bit padding0;
 } memcontroller_Controlreg_t;


 typedef struct packed {
   bit [31:0] vctl;
 } memcontroller_CTLR_ID_t;

 class memcontroller_Controlreg extends ovm_register
       #(memcontroller_Controlreg_t);
   covergroup c;
     field1 : coverpoint data.field1;
     field2 : coverpoint data.field2;

   endgroup

   function void sample();
     c.sample();
   endfunction

   function new(string name, ovm_component p);
     super.new(name, p);
     c = new();
     WMASK = 'b11111111111111000000;
   endfunction

 endclass
```

Figure 3: Generated OVM code fragment.

This methodology works to improve the productivity of the company at three levels. Rather than working to achieve a local optimal operational efficiency for one particular group, it ensures a complete product-team wide optima.

1. Engineers become more effective as they don't have to spend time working on mundane activity like creating and verifying registers manually.

2. Design and Verification Process becomes more effective as redundancy is removed from the system and change is managed elegantly.
3. The technologies that enables this methodology (OVM and IDesignSpec) are lightweight, easy to learn and make simple things easy and difficult things (like customized OVM output) possible.

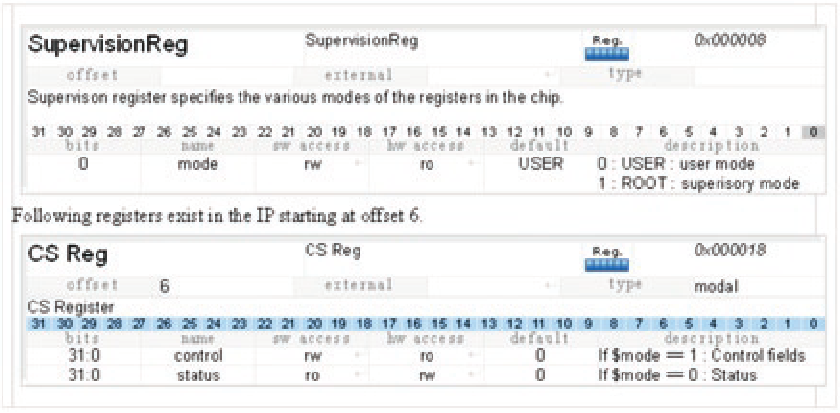
Digging deeper

IDesignSpec generates so called "regdef" files for the OVM register verification environment. Each block/IP in the specification document is transformed into a SystemVerilog Package. Each of the packages contains packed register structures, the register definitions and register instantiation. This can be used by the verification team to create register aware tests, drivers, monitors and scoreboards.

The OVM Register verification environment provides a lot of useful functionality right out of the box. Additionally, it is possible to extend the functionality.

- Register types
  - Normal registers are supported out-of-the-box, and other "quirky" registers are also supported out of the box – for example, an ID register, a coherent register and a model register are all modeled in the OVM Register kit. Additional customer specific behaviors can be easily created by simple extensions of the basic OVM Register API.

The following figure shows how a supervisory register, and model register can be specified in IDesignSpec:



SupervisionReg	SupervisionReg	Reg.	0x000008
offset	external	type	

Supervison register specifies the various modes of the registers in the chip.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

bits	name	sw access	hw access	default	description
0	mode	rw	ro	USER	0 : USER : user mode 1 : ROOT : superisory mode

Following registers exist in the IP starting at offset 6.

CS Reg	CS Reg	Reg.	0x000018
offset 6	external	type	modal

CS Register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

bits	name	sw access	hw access	default	description
31:0	control	rw	ro	0	If $mode == 1 : Control fields
31:0	status	ro	rw	0	If $mode == 0 : Status

Figure 4: Specifying some complex registers

- Register Tests
  - Write – read per register
  - Writes followed by all reads

Customized tests are easy to write – either from scratch or by extending and enhancing the existing functionality. The user has easy access to the underlying data model – the register maps, the register file and the registers themselves.

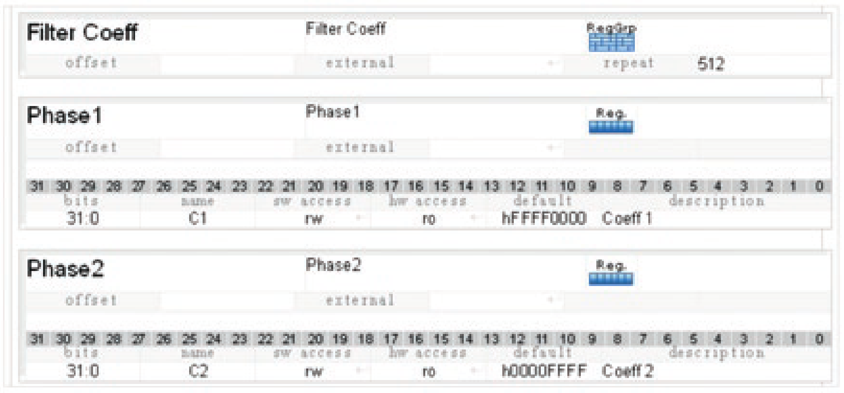
- Register Coverage
  - Register coverage is available automatically on read and write, and is customizable by the user for other interesting functional relationships.
- The OVM Register package can be used to describe various register behaviors and access policies, including:
  - Read Writable (R/W)
  - Read Only (RO)
  - Write Only (WO)
  - Read-to-Clear (CR), Read-to-Set(RS)
  - Write-1-to-Clear(RW1C), Write-1-to-Set(RW1S)

In IDesignSpec, this is simply selectable on a per-field basis.

- Hierarchical Register specification
  - Blocks
  - Register files
  - Multidimensional register structures

IDesignSpec's Chip and Block structures enable users to create a hierarchy in a linear document. RegGroup adds a dimension to the registers and one can specify a RegGroup inside another ad-infinitum. This is basically a shortcut for specifying arrays which translates into multi-dimensional register structures which are great for Video and Image processing filter applications.

The following figure on the next page shows how a register bank with 512 copies of registers is created.



Filter Coeff — Filter Coeff — RegGrp

offset		external		repeat	512

Phase1 — Phase1 — Reg.

offset		external	

bits	name	sw access	hw access	default	description
31:0	C1	rw	ro	hFFFF0000	Coeff 1

Phase2 — Phase2 — Reg.

offset		external	

bits	name	sw access	hw access	default	description
31:0	C2	rw	ro	h0000FFFF	Coeff 2

Figure 5: A group of registers

- User customizations
  - The generated “regdef” file is typically not edited manually, as doing so would violate the single-source principle, which in turn will cause the user to manually edit every time changes are made to the specification document.

IDesignSpec provides mechanisms by which users’ customizations are made to the generate code. For example, users can specify that an OVM register should be extended from a class other than the default “ovm_register” class. Despite this, if absolutely required, the “regdef” file is a readable file that can be edited and extended as needed.

Benefits

Using this tool set has enabled us to:

- **Specify Registers within functional specification**
  Keeping the register specification within the functional or design specification is very important. Changes to the functional specification usually means changes to the registers and vice versa. Having a separate tool for register management would simply exasperate the problem as cross references would need to be manually maintained and the two would constantly need to be kept in sync.

- **Dismantle information silos**
  Designers and architects are the producers of Register information, but the entire engineering team is the consumer. Starting out with a spec that is shared by the entire team is a good start, adding register information to the spec is better as the two go together and there is less need for cross referencing. Keeping a consistent methodology helps not just one product team but is fundamental to design and spec reuse.

- **Achieve global process optimization minima rather than local group minima**
  The pressure for meeting deadlines is such that in many cases each engineering team could focus on their deliverables and not think about the overall productivity improvements that could be had by having a common system in place that produces an overall advantage at the cost of one team having to go the extra mile.

- **Adapt to change rather than fight it**
  Change is inevitable in digital design. Rather than coming up with stringent regulations for avoiding it, the system should be able to accommodate it without excessive re-work. For example, if a register location changes, or its bit fields etc changes, it should be possible to recreate the tests and run them without any issue.

- **Create a single information source**
  A single source of information eliminates the need for duplication and synchronization and saves time and resources while improving quality.

Summary

This paper has shown why its important to have a consistent register methodology for getting significant ROI. It has also shown what the new additions to the latest OVM register packages are, and how to use OVM and IDesignSpec to streamline the register specification and testbench generation process.

About the author

Anupam Bakshi is Founder and CEO at Agnisys Inc.
He can be reached at ab@agnisys.us