Design of an IP-XACT to UVM RAL Generator

[1]Arshiya Hesarur, [2]Deeksha Dev, [3]T.Nagasundaram, [4]Uma Mudenagudi
[1]M.Tech (Final year), VLSI Design and Testing, ECE, BVB College of Engineering and Technology, Hubli, INDIA
[2]B.Tech (Final year), ECE, DAV Institute of Engineering and Technology, Jalandhar, INDIA
[3]ASIC Design Verification Engineer, CVC Pvt. Ltd, Bangalore, INDIA
[4]Head of Department, ECE, BVB College of Engineering and Technology, Hubli, INDIA

Abstract— **Today's all the computing technologies originated from the foundation laid by Moore's Law according to which number of transistors on a single integrated chip doubles every year and this has led to increase in design complexity. Keeping in mind the demand of high manufacturing yield and reduced time to market window, the SoC design and verification engineers have come up with strategies like Design Re-use and Automation of Verification. Automation of IP integration and verification process requires a standard vendor-neutral and language independent mechanism for representing meta-data of system-on-chip (SoC) design which resulted in introduction of IP-XACT standard. This paper is about an automation tool developed to generate Register Abstraction Layer (RAL) by using register description of IP cores in IP-XACT schema format. This generator is designed with no prior knowledge of System Verilog/UVM. The RAL generated is then integrated in UVM verification environment to verify variety of DUTs.**

Keywords— ***UVM (Universal Verification Methodology), XML (Extensible Markup Language), IP-XACT, RAL(Register Abstraction Layer), XML Schema, EDA (Electronic Design Automation), DRC (Design Rule Check), DUT (Device Under Test)***

I. INTRODUCTION

The colossal scale of transistor on hardware chips has reached to a point where it is extremely difficult to design a complex system from scratch. SoC challenges and how to counter them is described in detail in [2]. Looking at the Design Cycle of an IC the most time consuming phase is the Verification phase. Verification of an IC consumes around 70 – 80% of the design cycle time and effort. Due to this fact industries have come up with solutions like reusing intellectual properties (IPs). Every DUT has registers and memory as their subcomponents as shown in figure 1 which are foremost to be controlled, covered and checked during verification. SoC verification has various approaches to verify DUTs one of which is Universal Verification Methodology (UVM).

This paper provides an overview of UVM, XML, IP-XACT Schema and the description of the designed RAL generator.

A. Moving to the Accelera Standard Verification Methodology – UVM:

UVM is a standard verification methodology which was created to automate verification of IC designs. It is an open source standard developed and maintained by Accellera which is adopted by multiple EDA Industries such as Cadence Design Systems, Mentor Graphics, NXP Semiconductors, STMicroelectronics, Texas Instruments, Freescale Semiconductors, Synopsys etc. By creating this standard, components become portable from one project to another, thus serves reusability concept by which one can target large designs, IP's and SoCs. It is a combination of verification guidelines and Application Programming Interface which ultimately serve to reach the automation goal in verification field.

UVM provides mapping of register content of DUT and a layer for accessing register and memory locations within DUT which is called as Register Abstraction layer (RAL). UVM requires RAL to generate register models and register transactions which are then integrated in UVM verification environment. It can be amended as many times as one wants in one's project without intervening the existing verification environment or test. Figure 2 shows the register model [4]. R1, R2…Rn are the registers containing their respective fields in a DUT which is to be verified [4].

A. Performance:

It is easy to write register classes manually for smaller designs. But writing register classes for large complex designs manually is hard, time consuming and error prone. This designed generator will automatically generate the register classes for bigger and complex designs. This automates the verification phase to some extent and much of the time and effort is saved.

II. FIRST STEP TO AUTOMATION – IP_XACT

Initially verification engineers were facing problems due to dependency of design flow on different tools. Also writing a code of more than thousands line is a tedious job. Thus a standard was required which resulted in formation of IP-XACT XML Schema.

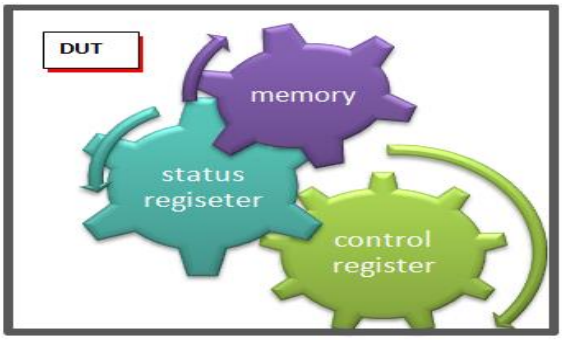


Fig. 1, DUT containing sub-components

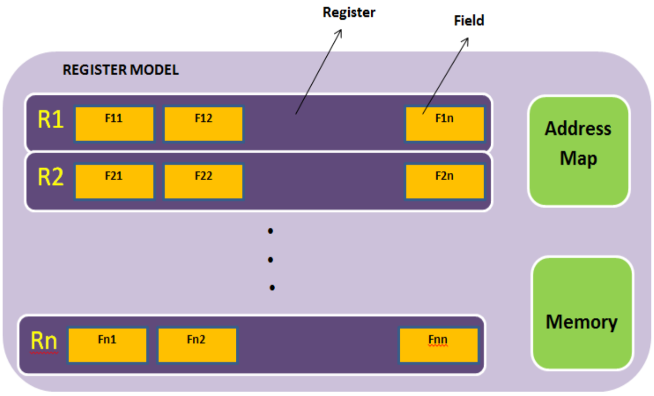


Fig. 2, Register Model view

*A. XML- e**X**tensible **M**arkup **L**anguage*

XML is a language which is capable of encoding the documents in both human readable and machine readable format. Hence it is a language which is both software and hardware independent. It is an open standard developed by W3C. XML simply carries information and gives full flexibility to users to create their own XML tags. Therefore XML is nothing but information wrapped in tags. An XML tag is the building block of XML document. It identifies data and is used to store data in an organized manner. One of the most important feature of XML is that it is “Extensible” which means XML allows us to create our own tags (self-descriptive) or languages (Ex: IP-XACT) that suits our description.

B. XML Schema:

An XML Schema is type of XML Document with certain rules and regulations imposed on the elements of an XML document. The XML document is considered as a schema only when it meets the rules and regulations defined by that schema. Thus XML Schema provides a definite structure to the XML document. XML Schema is also a recommendation of World Wide Web Consortium (W3C) [5].

C. IP-XACT:

IP-XACT is IEEE-1685-2009 standard which sets rules for representing electronic components and their designs to provide direct access to data regardless of the design language or tools. It was first released by SPIRIT Consortium which helped industry to resolve the problem of integration of IPs from multiple vendors in design flows. Mentor graphics was the first contributor to IP-XACT; other contributors are ARM, Cadence, NXP, LSI, ST, Synopsys Texas Instruments and nine others [6]. An IP-XACT view of an IP is documented using standard meta-data. The meta-data includes components and details of components such as address maps, registers and description of fields. Meta-data documents the characteristics of Intellectual Property (IP) which are required for the automation of configuration and integration of IP blocks. IP-XACT also defines an Application Programming Interface (API) to make this meta-data directly accessible to automation tools. Thus the main purpose of this standard is to provide a definite structure for Packaging, Integrating and Re-using IP within tool flows. In order to get access to the elements and text in an IP-XACT file, XML Parser is needed. An XML Parser converts any XML document to an XML DOM object. Detailed description IP-XACT standard is given in [1].

D. XML DOM Object:

XML DOM (Document Object Model) is a standard for accessing and manipulating XML documents. It views each of the XML document as a tree structure with root element, parent elements, child elements and siblings. XML DOM considers the elements, text information and attributes in an XML document as nodes. Using XML Parser one can access the nodes of an XML document. Figure 3 illustrates an example of XML document and its DOM object.

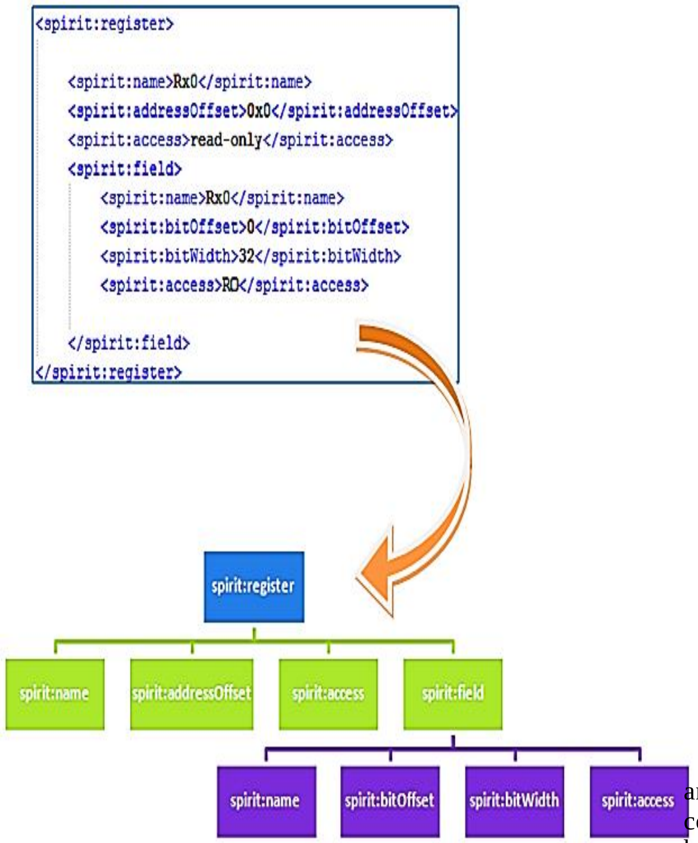


Fig. 3, DOM tree structure of an IP-XACT document

III. GENERATION OF A UVM RAL:

An automation tool that generates RAL from IP-XACT is designed. There are two files, one is the input file which is IP-XACT, containing register description of an IP core and the other one is RAL obtained as an output of generator. The source code for generator is in C language which reads XML Schema as input, parses it and then develops a new file called as "RAL.sv" as its output. Figure 4 is the block diagram of IP-XACT to RAL generator. Sequential steps followed to generate a RAL are given below:

1. IP-XACT view of an IP Core is generated (with the help of IP-specification to IPXACT tool). Additional fields access type, randomization and volatile are added to the IP-XACT document. This makes the RAL to be generated more accurate.
2. The IP-XACT file is then parsed by XML Parser defined in the generator code. Thus each node in the IP-XACT is accessed and manipulated using XML parser file.
3. After the generator code runs successfully, a UVM RAL is generated in a file named "RAL.sv". Figure 5 illustrates the IP-XACT file and UVM RAL of an SPI (Serial Peripheral Interface) core [3].

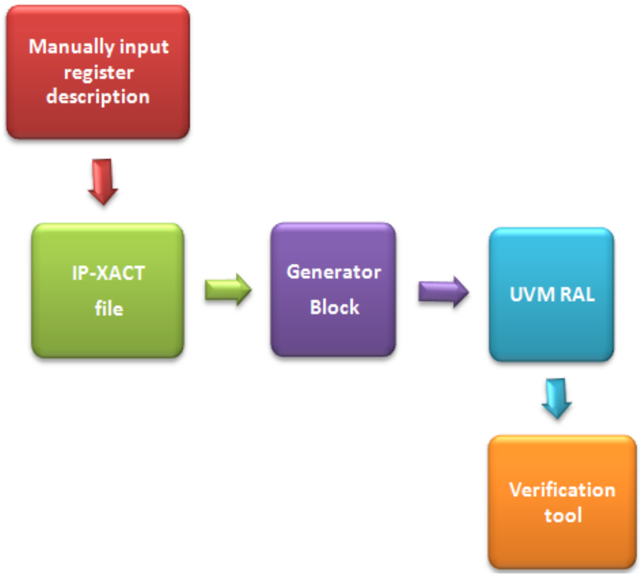
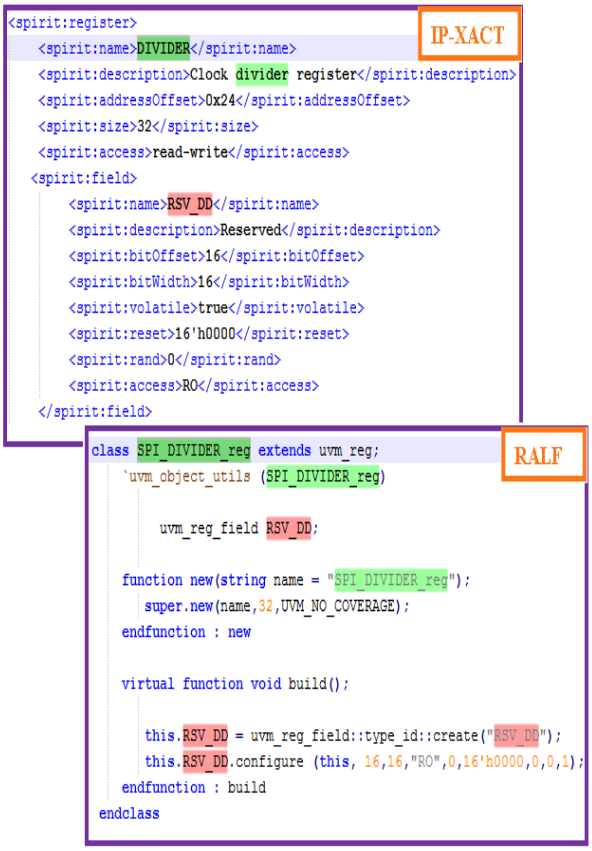


Fig. 4, Block Diagram of Automation Flow

In the output RAL, initially basic libraries were declared and then class for each register is defined. Register class contains description of configurations of each field a register has in it. After defining all the register classes one main register block is defined, this contains instances of each register class with its configuration description in it. Following is the sample showing IP-XACT to RAL generation, having register class named as "SPI_DIVIDER_reg" containing description of its field named as "RSV_DD".




```
<spirit:register>
  <spirit:name>DIVIDER</spirit:name>
  <spirit:description>Clock divider register</spirit:description>
  <spirit:addressOffset>0x24</spirit:addressOffset>
  <spirit:size>32</spirit:size>
  <spirit:access>read-write</spirit:access>
 <spirit:field>
    <spirit:name>RSV_DD</spirit:name>
    <spirit:description>Reserved</spirit:description>
    <spirit:bitOffset>16</spirit:bitOffset>
    <spirit:bitWidth>16</spirit:bitWidth>
    <spirit:volatile>true</spirit:volatile>
    <spirit:reset>16'h0000</spirit:reset>
    <spirit:rand>0</spirit:rand>
    <spirit:access>RO</spirit:access>
  </spirit:field>
```

```
class SPI_DIVIDER_reg extends uvm_reg;
   `uvm_object_utils (SPI_DIVIDER_reg)

      uvm_reg_field RSV_DD;

   function new(string name = "SPI_DIVIDER_reg");
      super.new(name,32,UVM_NO_COVERAGE);
   endfunction : new

   virtual function void build();

      this.RSV_DD = uvm_reg_field::type_id::create("RSV_DD");
      this.RSV_DD.configure (this, 16,16,"RO",0,16'h0000,0,0,1);
   endfunction : build
 endclass
```

Fig. 5, Sample of an IP-XACT file and generated RAL of SPI core

IV. CONCLUSION

Standardization is quite essential in order to increase quality of SoC designs and produce SoC systems in large quantity in small Time To Market window. In UVM, RAL is meant to describe register model of IP's which are to be verified. The key is to generate RAL automatically to deliver best timing and quality results. Thus this paper describes an automation tool which generates RAL by taking input as IP-XACT, which is a standard known for its reusability property plus vendor and language neutrality.

V. ACKNOWLEDGMENT

The authors (author[1] and author[2]) would like to thank CVC Pvt. Ltd for giving an opportunity to work as an Intern in the company and providing with the necessary tools to develop this project. The authors would also like to thank the management and team of CVC for their guidance and support.

REFERENCES

[1] IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows," IEEE Standard 1685-2009, vol., no., pp.1, 374, Feb. 18 2010.

[2] Anil Deshpande, "Verification of IP Core Based SoC's," Conexant Systems Inc. Hyderabad, India.

[3] Jain, A.; Gupta, R., "Scaling the UVM_REG Model towards Automation and Simplicity of Use," VLSI Design (VLSID), 2015 28th International Conference on, vol., no., pp.164, 169, 3-7 Jan. 2015.

[4] Mark Litterick, Marcus Harnisch, "Advanced UVM Register Modeling There's More Than One Way to Skin A Reg," Verilab GmbH, Munich, Germany.

[5] Zisman, A., "An overview of XML," Computing & Control Engineering Journal, vol.11, no.4, pp.165, 16, Aug. 2000.

[6] By Marc van Hintum, NXP Semiconductors, The Netherlands and Paul Williams, "The Value of High Quality IP-XACT XML," Mentor Graphics Consulting, Europe.