Leveraging IP-XACT standardized IP interfaces for rapid IP integration

David Murray
Duolog Technologies
David.Murray@duolog.com

Simon Rance
Duolog Technologies
Simon.Rance@duolog.com

***Abstract* - IP integration has continued to be a key challenge in SoC development. Increases in IP reuse, IP configurability and system complexity within tightly bound schedules have compounded the problem of IP integration. One fundamental aspect of improving the IP integration process is the provision of standardized interfaces. Traditionally signal naming standardization practices have been utilized to provide consistent views of hardware interfaces for the sub-system and SoC integration activities. However IP reuse in a context of a wide variety of IP sources and coding styles reduces the efficiency of flows dependent on signal naming convention, and it is not feasible to redefine signal names as there are many dependencies on this RTL view to change. A solution to this problem can be realized through the IP-XACT standard, which provides mechanisms for the standardization of IP interfaces. IP-XACT is a standard (IEEE1685-2009) that defines an XML structure for packaging, integrating, and reusing IP within tools flows. This white paper will focus on the IP interface standardization mechanisms available in IP-XACT and provides detail on the constructs involved including bus interfaces, bus definitions and port mapping. It will also provide an overview of the standardization of Hardware/Software interfaces including memory-maps, registers and bitfields.**

This paper will then detail how this standardized data, or metadata, can be used in various different flows. Firstly, before delivering IP metadata to integration teams, it's vital to ensure that the IP metadata is coherent with the RTL implementation. This paper details how IP verification flows, simulation or formal-based, can help to qualify the IP-Metadata and also to verify design intent. An integration flow will also be presented that shows how this interface standardization streamlines IP integration and can result in a 10x improvement in integration productivity. This productivity improvement is realized through the use of interface-based connectivity synthesis as well as the utilization of standardized bus definitions and interface standardization. The paper will conclude with a look at some of the key guidelines to consider when standardizing IP interfaces and will outline some key recommendations.

Keywords—IP Reuse; IP Integration; IP-XACT; Rules-based integration

I. INTRODUCTION

The integration of semiconductor IP continues to be a key challenge in SoC development and is quickly becoming one of the main chip design challenges. Yesterday's IP integration challenges comprised of integrating only internal known IP. Although these challenges were not that long ago, they seem so distant compared with today's challenges comprising of integrating not only internal known IP, but also unfamiliar external IP from other internal organizations or third party IP.

The SoC revolution that is driving the mobile electronics market and enabling the various lifestyle trends, is being enabled by the adoption and integration of many complex semiconductor IP's. Where would we be without the commercial IP market? Certainly our computing devices wouldn't be so compact, mobile and multifunctional and our home devices and appliances certainly wouldn't be controllable from our phones. The adoption and integration of commercial IP reduces the cost and time-to-market of SoCs while dramatically raising the bar of innovation versus the competition.

As IP adoption and reuse become more mainstream in SoC realization, IP integration is increasingly seen as a key challenge and a growing contributor to the overall cost of SoC development [1]. There are many factors that combine to compound the problem of IP integration.

- The increase in SoC design complexity means more IP blocks and sub-systems to integrate together.
- The constant drive to reduce SoC development schedules and costs, without impacting quality, has led to pressure to reuse internal and third-party IP and to integrate these IP as quickly and efficiently as possible.
- IP's are becoming more complex and configurable and can have thousands of ports and hundreds of different configurations.
- Design teams are not being scaled to the same level as the problem, and so bigger problems have to be managed by fewer people.
- The poor adoption of standards and methodologies for IP integration is making efficient and reusable integration more difficult.

The result is a poor quality IP integration process that has been identified as one of the main chip design challenges [2][3][4]. A solution is needed that provides a balance between abstraction and automation while enabling IP configurability, IP quality, and predictable IP integration.

II. Lack of Standardization

The semiconductor and EDA industries are not new to the benefits of standardization. The mainstream adoption of HDLs in the early 1990's was followed closely by the publishing of several internal and external naming convention guidelines. There are many on the web today but Xilinx for instance produces coding guidelines with port naming conventions[5].

While coding and naming conventions guidelines do reap benefits there are some downfalls. For naming conventions to work as intended, all IP being used in the system must adhere to them. Sounds simple enough if the design team is small and all the IP is developed locally within that team. However, the reality is that today's SoCs consist of new, legacy and external IP. The chances that legacy or external IP has followed the coding and naming conventions guidelines set forth by the integration team are slim to none.

The IP integration reality is that most of the IP integration tasks are almost entirely manual. Scripts and the use of naming conventions can help automate some of the IP integration. However, with the levels of complexity and configurability in today's IP's continuing to increase at alarming rates, scripts cannot keep up and constantly require huge amounts of maintenance. Scripts are typically only familiar with IP created and managed locally. Bring one external IP from another team within the same company or from a third party vendor and immediately, that script is broken and requires some maintenance. Bring in multiple external IP's to integrate and not only are your scripts now broken and useless, but now you have a new problem, the familiar script maintenance problem. The results are that scripts and naming conventions cannot maintain standardization or automation throughout the entire integration task.

According to Gartner, the semiconductor IP market is forecasted to grow from $2B to $3B by 2016. A 50% increase in just a few years shows the level of growth in this market. Although positive news, this increase will only push the IP integration task more to the forefront of front-end chip design. Semiconductor IP vendors are not necessarily integrating the IP themselves, however the IP consumers will continue to mount pressure on them to supply high quality IP that will predictability integrate into their design as seamlessly as possible. There is a need for some standardization for IP quality, integration and predictability. Part of a solution is for the IP provider to explore implementation feasibility of RTP during IP development and ensure the quality and predictability of the outgoing IP by providing it in a standardized package. In turn, the IP integrator/consumer can use the same information packaged as a programmable specification as well as a quality measure of incoming IP for the purposes of IP integration.

Typically, most IP's in today's SoC designs are connected via interfaces. These bus interfaces are usually either standard or proprietary buses and are connected via some type of interconnect or Network-on-Chip (NoC). The remaining connections are typically considered non-interfaced based connections or ad-hoc connections. Although ad-hoc connections will exist, upon closer examination you will most likely find that not all of them are just one-to-one ad-hoc connections and several similar pins will connect to the same place such as an interrupt controller. These pins or signals can be potentially grouped together at a higher level of abstraction such as an interface which can then be connected accordingly. The secret sauce to improved interface based IP integration is to us a higher level of abstraction. This higher level of abstraction is simply mapping a collection of ports to interfaces enabling a higher level of abstraction from ad-hoc based IP integration to predictable interface-based IP integration.

This higher level of abstraction of an interface can contain many logical ports and their necessary attributes and parameters for configurability. Using this abstraction, IP interfaces can be fully standardized providing the IP integrator with the information needed to seamlessly integrate all the necessary IP's in the system in an automated fashion. This not only enhances productivity and reduces time-to-market, but provides a repeatable IP integration model with predictability and quality.

III. What is IP interface standardization?

Interface standardization is the standardization of both hardware and Hardware/Software interfaces on an IP. Hardware interfaces typically consist of RTL ports and Hardware/Software interface are typically memory mapped interfaces to Hardware/Software structures such as registers, memories or bridges. Within this white paper, the scope is focused on how the interface mechanisms (ports, registers) themselves as opposed to defining a protocol e.g. AMBA, PCI etc.

IV. Why seek IP interface standardization?

One of the key benefits of having standardized IP interfaces is that of interoperability. Interoperability is the ability of making systems and organizations to work together[6]. In the case of SoC design, it involves the sharing of data or processes across different organizations and disciplines. For instance, as outlined in Fig. 1 a standardized definition of an IP hardware interface can be can be shared between IP providers, IP consumers and the EDA community

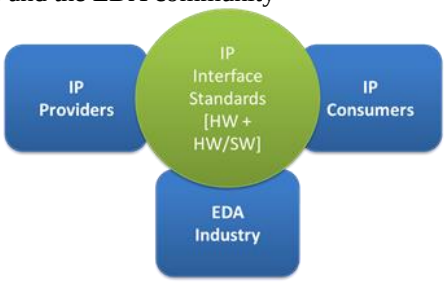

Fig. 1. Shared usage of IP interface standard definition

Even within these organizations this information is shared and consumed by different teams and disciplines e.g. technical

documentation, System modeling , hardware design, System modeling verification.

A Hardware/Software interface can in addition be consumed by software disciplines including firmware development, Hardware/Software integration, Hardware/Software validation, chip debugging etc.

Another key benefit of IP interface standardization is that it establishes and upholds higher levels of compatibility and quality.

1) Leveraging interface standards for IP Creators

IP creators can leverage interface standards directly when designing and creating their IP. A standards-driven approach can define their interface to be standards-compliant and utilize automation to create documentation, design implementations or verification environments. The standards have indirect benefits when an IP is used in a system as the consumer is also utilizing the standard.

2) Leveraging standards for IP Consumers

IP consumers can leverage IP interface standards significantly especially when it comes to IP integration and verification. IP blocks with known good interfaces are easier to integrate both from the hardware and software side.

3) Leveraging standards by EDA industry

While interoperability requires a level of standardization – this is in itself not enough. In order to reap real benefits in today's highly complex and automated design flows, this standard must be highly 'automatable' and easy to process. Both IP creators and consumers would not benefit without significant levels of automation which can be provided by the EDA industry.

V. IP-XACT - THE IP INTERFACE STANDARD

One of the main standards to emerge to solve the problem of IP interface standardization is IP-XACT[7]. IP-XACT was developed by the SPIRIT consortium to enable sharing of standard component descriptions from multiple component vendors. IP-XACT is a '*Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows*'. Currently Accellera manages this definition of this standard (IEEE-1685-2009) under the IP-XACT Technical committee.

IP-XACT provides a schema for the definition of IP component and design metadata and has a mechanism to standardize the view of an IP and its interfaces. Because IP-XACT defines an XML schema this is a format that is very easy to process and provides significant automation enablement. IP-XACT, through interface standardization had the ability to make IP more 'integration-ready', and could result in a 30% improvement in the time and cost of SoC integration [1].

The next section of this white paper will detail the different mechanisms used in IP-XACT to define interfaces in a standardized and highly automated format.

VI. STANDARDIZING HARDWARE INTERFACES USING IP-XACT

From an RTL perspective the hardware interface is typically represented on the lowest level by hardware ports with a given name, direction (usually inputs, outputs and bidirectional ports) and a width (a single bit or a vector). These ports can often have a functional grouping that represent a specific functional interfaces e.g. APB™, AHB™, AHB-Lite™, AXI™, ATB™, LPI™, AXI4™, APB4™, ACE™ and ACE-Lite™[1], JTAG, UART etc. While VHDL and Verilog don't formally describe this grouping, new languages such as SystemVerilog model this concept.

In order to standardize ports and interfaces, IP-XACT has several a few simple concepts. These are described in Fig. 2

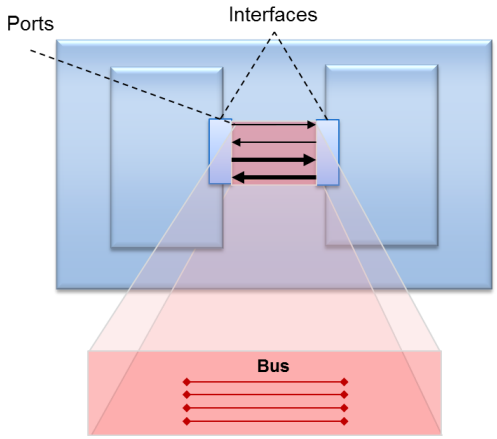

Fig. 2. IP-XACT Concept of a Bus

In focusing on point-to-point connectivity IP-XACT has some well-defined concepts. An interface on a component that contains a well known set of ports is called a **bus interface**. The set of ports that can be defined on that interface is called a **bus**. Like ports, bus interface can generally have high level transactional or data flow characteristics and behave as masters or slaves. In these cases there may have different composition of ports and may have different variants like direction and size.

In IP-XACT the majority of these variants can be described in an IP-XACT **bus** definition – which consists of a **bus_definition** and and **abstractionDefinition.** While these two describe the characteristics of a bus they are typically referred to as a **bus definition**.

[1] APB, AHB, AHB-Lite, AXI, ATB, LPI,, AXI4, APB4, ACE and ACE-Lite are the trademarks of ARM Limited in the EU and other countries.

From a hardware interface standardization perspective, if we can define a standard set of ports that should appear on different interfaces and provide a machine-readable definition of how these relate to the actual physical implementation then we can enable high levels of quality checking and automation.

A. Defining Busses

In IP-XACT a bus definition defines the high-level name and characteristics. It describes the name of the bus and other characteristics of if it is addressable and the number of master/slaves the bus supports. The real detail however is left to the AbstractionDefinition which defines the port contents of interfaces in different roles. Fig. 3 shows the typical information in an IP-XACT AbstractionDefinition for a single address port called ADR on a simple addressable bus.

Port Definition		
Name		ADR
Description		Address
Type		Wire
Default		0
Qualifier		isAddress
On Master	**Presence**	Required
	Direction	out
	Width	
On Slave	**Presence**	Required
	Direction	In
	Width	
On System	**Presence**	illegal
	Direction	
	Width	

Fig. 3. Example of an AbstractionDefinition Port definition

Every port that can exist in the bus is defined using a name, description the type of port (in this example wire, the default of the port and some additional qualifiers to identify signals such as address, data, clock and reset. Fig. 4 shows an example of the full definition of an AbstractionDefinition.

Name	Description	Type	Default	Qualifier	On System Pre	On Slave	On Master		
						Presence	Presence	Direction	Width
ADR	Address	Wire	0	isAddress	Il	Required	Required	out	
WDATA	WriteData	Wire	0	isData	Il	Required	Required	out	32
RDATA	ReadData	Wire	0	isData	Il	Required	Required	in	32
WR_N	Write Enable	Wire	1		Il	Required	Required	in	1
ERR	Error	Wire	0		Il	Required	Optional	in	1
						Optional			

Fig. 4. Example of full AbstractionDefinition for the bus

These Bus Definitions and AbstractionDefinitions can be used to create a standard definition of the busses in a system.

B. Using these bus definitions to standardize Hardware interfaces

Once defined in IP-XACT these bus definitions can be used in conjunction with IP-XACT component descriptions to describe hardware interfaces. There are two main tasks needed to achieve this:

- Define the Physical Ports.
- Map the physical ports to the standardized definition

In IP-XACT physical ports are used to representing RTL interfaces. In an IP-XACT component document they are described under a *component/model/ports* node. Attributes such as name, direction, vector definition can be described here. Fig. 5 shows an example of a physical/RTL ports being represented in IP-XACT

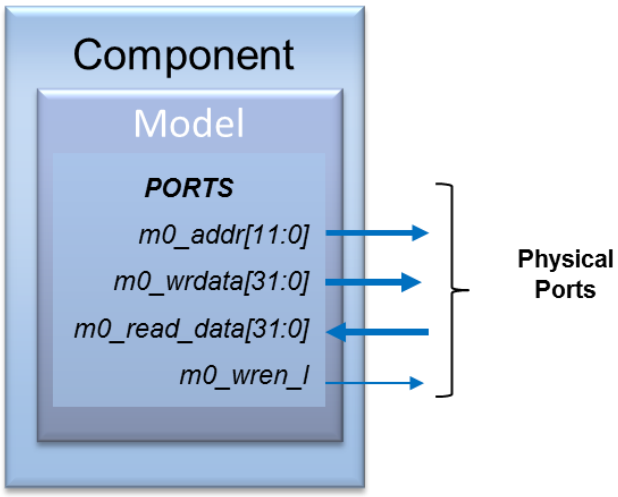

Fig. 5. IP-XACT Representations of physical ports (RTL)

In order to assign these ports to an interface in IP-XACT, a **bus interface** needs to be defined. Bus interfaces are described under *component/busInterfaces* and have the following functions:

- They can link to a bus definition /AbstractionDefinition pair
- The define the role of the interface e.g. if it is a Master/Slave/System and if it is mirrored
- They define the mapping between physical (RTL) and logical ports (Bus/Abstraction definition) which can be seen as a standard view.
- It is also possible to associate a memory map with an interface.

Once a bus interface is linked to a Bus/Abstraction Definition there are constraints brought in by this association:

- For the given interface role (master/slave/system) there may be a mandatory or illegal ports on this interface
- The direction and size of the ports themselves may be constrained.

These constraints can be utilized when defining a mapping or they can be used to check that an existing mapping is correct using IP-XACT Semantic-Consistency checks or SCRs. Physical ports are mapped to the bus definition/abstraction definition logical ports using a **port map** (*component/busInterfaces/busInterface/portMaps/portMap*)

Fig. 6 shows an IP-XACT representation of a master bus interface of a hypothetical simple processor bus (SPB)

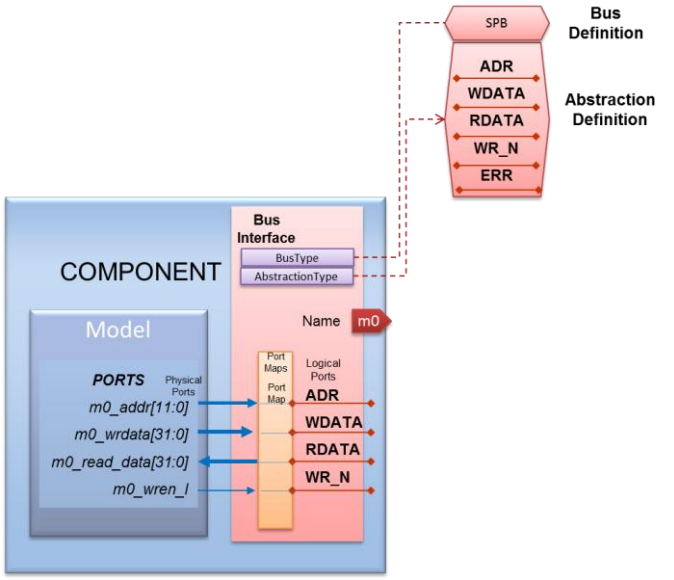

Fig. 6. IP-XACT Representations of a master bus interface

Once this mapping has been defined a full suite of IP-XACT SCR checks can be run on the IP-XACT component for example:

- Check that all required ports in a bus definition have been mapped
- Check that all directions of all ports are consistent with the bus definition
- Check that all widths of all ports are consistent with the bus definition
- Check that no illegal ports have been mapped

If these pass, the interfaces contained in this IP-XACT component definition can considered to be compliant to the standard. Even though the RTL file has port names such *m0_wren_l* by viewing through the IP-XACT definition we can determine that this in fact means the *WR_N* port of interface *m0*. This and the fact that IP-XACT is a standard XML structure allows for standardized design flow enablement and leverage by IP creators and consumers as well as the EDA industry in general.

VII. Standardized Hardware/Software interfaces using IP-XACT

IP-XACT also allows the standardization of Hardware/Software interfaces. Fig. 7 shows an IP-XACT representation of a leaf IP with registers. The Hardware/Software interface description in IP-XACT is quite comprehensive and can be used to define register/field access types (read/read-write/write-only etc.) access side behavior (e.g. read-to-clear), reset values.

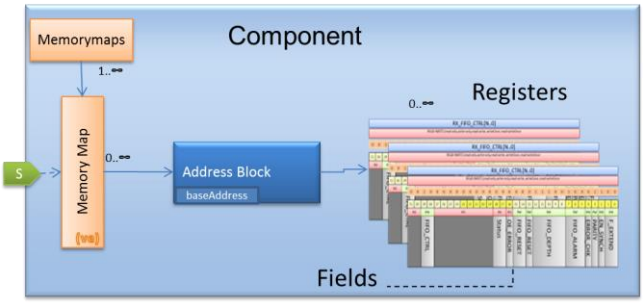

Fig. 7. IP-XACT representation of an IP with Hardware/Software interface (registers)

IP-XACT can also represent address mapping through Decoders/Bridges/Bus interconnect as shown in Fig. 8.

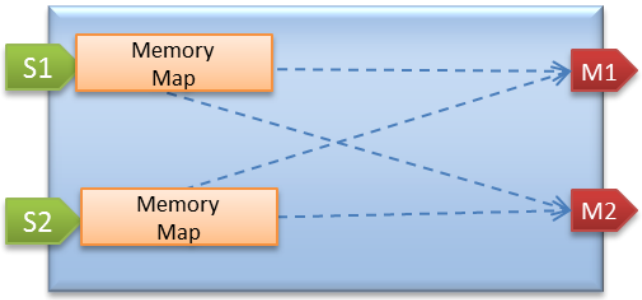

Fig. 8. IP-XACT representation of an IP as a bus interconnect

It is also possible to describe memories and different memory structures. This topic has been discussed in other publications [10][11] which describe how by having this formal specification (IP-XACT) as an industry-wide standard, the EDA industry provides highly automated solutions to ensure improved quality and efficiency.

VIII. DESCRIBING DESIGNS IN IP-XACT

Before looking at how we leverage these standardized interfaces, it is also important to highlight some additional features of IP-XACT that can provide further capabilities.

An IP-XACT design can describe a hierarchical system and associated connectivity as described in Fig. 9.

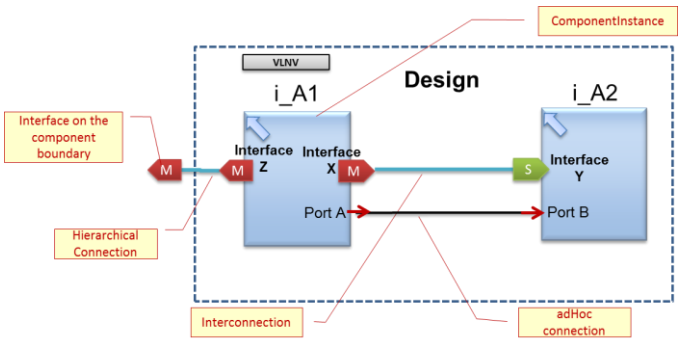

Fig. 9. IP-XACT representation of a design

In an IP-XACT design file component instances reference IP-XACT component definitions. Interconnections are connections between bus interfaces of systems and hierarchical connections are connections between instance interfaces and component interfaces. Ad-hoc connections are port-level connections.

The fact that this type of connectivity can be defined in IP-XACT format allows a highly interoperable method of automation. In particular this instance and connectivity information can be utilized to:

- Generate a design netlist
- Provide an address path and associated calculations
- Provide metadata to streamline verification flows

In the next section of this white paper we will look at how we can leverage IP-XACT in more detail.

IX. LEVERAGING IP-XACT STANDARDIZED IP INTERFACES FOR RAPID IP INTEGRATION

A. Provision of Integration-Ready IP

One of the key fundamental solutions for rapid IP integration is the provision of high-quality, integration-ready IP with standardized IP interfaces. While IP-XACT can offer a standardized view of an IP, it can be considered design collateral additional to RTL. It is therefore essential to ensure that:

1. The IP-XACT description adheres very rigorously to the IP-XACT standard
2. The IP-XACT is fully coherent with the RTL view.

For (1), the IP-XACT specification contains a list of checks to ensure that the IP-XACT document is semantically correct. There are known as IP-XACT SCRS (Semantic Consistency Rules) and can be run from an IP-XACT Design Environment.

A more challenging aspect however is (2) where we need to ensure that the IP-XACT is coherent with the RTL. For these quality aspects and challenges the IP/EDA industry has come up with several solutions that utilize the interoperability of IP-XACT to address these challenges and provide high-quality, integration-ready IP.

Fig. 10 shows an IP-XACT based IP flow that involves interoperability between several different companies.

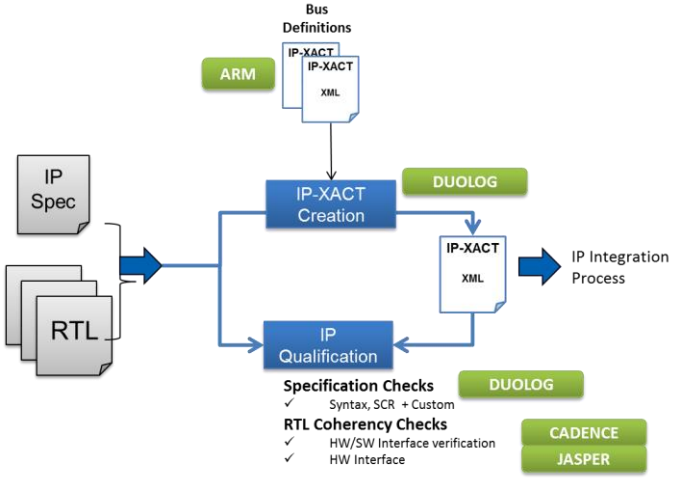

Fig. 10. IP-XACT Interoperablity across multipe vendors

Within this flow:

- ARM provides industry standard IP-XACT AMBA Bus definitions[12]
- Duolog's Socrates solution [13] addresses the challenges of creating IP-XACT interface descriptions (both Hardware/Software registers as well as hardware interface only) as well as running IP-XACT SCRs on any IP-XACT files.
- A UVM simulation-based solution[14], running on Cadence Incisive Enterprise Simulator can leverage IP-XACT to produce UVM test-cases that automatically verify the behavior of registers to ensure they conform to the specification (IP-XACT)
- A formal verification-based solution from Jasper called the JasperGold® CSR Verification App[15], formally verifies control and status registers from an IP-XACT specification.
- In addition to Hardware/Software interface verification, Jasper also provides intelligent proof-kits[16] for many standard protocols (e.g. AMBA, AXI, PCI-Express). As an IP-XACT interface provides a reference to a bus definition and standardized map to the RTL ports, it is possible to

automate the instantiation and connection of these proof-kits to the RTL.

Delivering high-quality integration ready-IP is essential for rapid IP integration and this solution shows that IP-XACT can deliver interoperable flows across multiple vendors.

B. IP integration

IP-XACT standardized interfaces can be utilized to accelerate IP integration. IP-XACT can provide a definition of all of the connectivity of a full hierarchical system at interface and port-levels and with specific semantics and rules about how interfaces infer port-level (adhoc) connectivity. Fig. 11 shows how physical (RTL port-level) connectivity can be inferred from an IP-XACT interconnection between 2 bus interfaces.

Fig. 11. Inferring physical connectivity from an IP-XACT design reppresntation

It is therefore possible to have a netlisting flow that can infer an RTL netlist from an IP-XACT definition which can remain at a higher level than RTL.

A case study [18] shows how using Duolog's Socrates Weaver the RTL creation of an ARM-IP Based system (Fig. 12) can be accelerated by a factor of 10x-15x using a rules-based integration methodology that leverages standardized IP interfaces using IP-XACT.

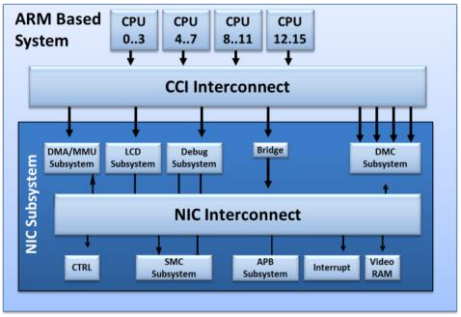

Fig. 12. Sample ARM-IP based system

This methodology directly utilizes the IP-XACT bus interface definitions and mapping to bus definitions to easily select and connect different interfaces and ports together. An output of this process is an IP-XACT representation of the system that can also be used by verification flows to validate connectivity integration or facilitate other verification targets.

C. Integration Verification

IP-XACT can enable verification in a number of different ways. Firstly, IP-XACT through its semantic definition can be validated against RTL. Jasper through its JasperGold® Connectivity Verification App can validate IP-XACT against an RTL netlist.

It is also possible to traverse the integrated design and elaborate the processor views of the system and calculate register physical addresses. The previously described IP qualification capabilities can be utilized again at sub-system or chip-level to validate system memory map consistency.

Another interesting case of verification enablement can be found using Cadence's Interconnect Workbench[19] which enables teams to rapidly generate performance analysis testbenches and optimize optimize performance of SoCs incorporating ARM CoreLink™ CCI-400™, NIC-400™, NIC-301™ and ADB-400™ system IP[2].

In Fig. 13 an ARM, Duolog, Cadence multi-vendor IP-XACT enabled flow is presented. This shows a flow where ARM provides bus definitions and IP-XACT descriptions of their IP, that have been automatically generated from ARM CoreLink™ AMBA® Designer. Socrates Weaver, from Duolog, can read the IP-XACT (that defines standardized IP interfaces) and connect the system using rules-based integration. An RTL netlist and corresponding IP-XACT file are generated in the process.

Cadences' Interconnect Workbench reads the IP-XACT file, extracts interface and connectivity information and uses this to automatically generate an IP-XACT testbench which reduces the time and effort commonly needed to create a test environment that previously required several weeks.

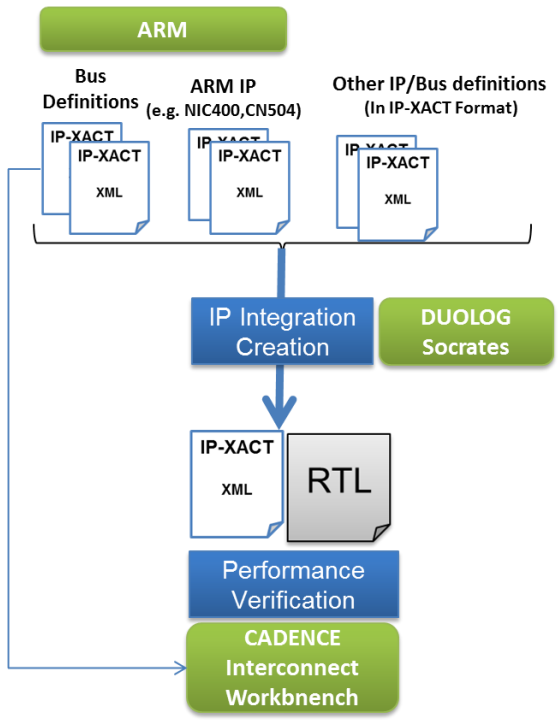

Fig. 13. IP-XACT Interoperablity across multipe vendors

X. SUMMARY

In summary, the growth of 3rd party IP usage and the lack of hardware and Hardware/Software interface standardization are causing a lot of integration issues which remain a key challenge of SoC design. IP-XACT has emerged as a clear leader for the IP interface specification standards and has comprehensive capabilities when it comes to defining the intricacies of RTL port interfaces or Hardware/Software elements such as registers and memory maps. IP-XACT adoption in the industry has led to high-value multi-vendor, interoperable flows that provide integration ready-IP, accelerate IP integration and enable high levels of verification automation.

XI. FUTURE

As IP-XACT gets more utilization in the industry there are more enhancements required. The Accellera IP-XACT Technical Committee is currently in the process bringing proposals to IEEE for creation of IEEE-1685-2009.

XII. ACKNOWLEDGMENTS

Thanks our industry partners who have collaborated with us in the past number of years to make these highly interoperable IP-XACT Flows. In particular Paul Martin and William Orme from ARM, Norris Ip and Shaun Giebel from Jasper, and Nick Heaton and Adam Sherer from Cadence.

[2] ARM® CoreLink™ CCI-400™, NIC-400™, NIC-301™ ADB-400™, and AMBA® are the trademarks of ARM Limited in the EU and other countries.

REFERENCES

[1] R. Goering, Cost of IP integration is rising dramatically, 2010: http://www.cadence.com/Community/blogs/ii/archive/2010/03/29/isqed-keynote-putting-some-numbers-to-cost-aware-design.aspx?postID=43255

[2] R. Goering: EDAC- CEOs Identified integration as key challenge in IC design, 2012 http://www.cadence.com/Community/blogs/ii/archive/2012/03/05/eda-ceos-speak-out-3d-ics-ip-integration-low-power-and-more.aspx?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+cadence%2Fcommunity%2Fblogs%2Fii+%28Industry+Insights+Blog%29&utm_content=Google+Reader

[3] R. Goering, EDA CEOs speak out on IP integration , 2012 http://www.cadence.com/Community/blogs/ii/archive/2012/03/05/eda-ceos-speak-out-3d-ics-ip-integration-low-power-and-more.aspx?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+cadence%2Fcommunity%2Fblogs%2Fii+%28Industry+Insights+Blog%29&utm_content=Google+Reader

[4] R. Wawrzyniak, Integration in the top 9 chip design Challenges, EBN 2012 http://www.ebnonline.com/author.asp?section_id=1102&doc_id=243335

[5] Xilnx Coding Guidelines: http://www.mrc.uidaho.edu/mrc/people/jff/240/241/xilinx_conventions.pdf

[6] Definition of interoperability: http://en.wikipedia.org/wiki/Interoperability

[7] IP-XACT Technical Committee : http://www.accellera.org/activities/committees/ip-xact

[8] ARM, CoreLink System IP & Design Tools for AMBA http://www.arm.com/products/system-ip/amba/

[9] http://en.wikipedia.org/wiki/Interoperability

[10] D. Murray, S.Boylan: Addressing Hardware/Software Interface Quality through Standards, DVCON 2012.

[11] D. Murray: Verification and Automation Improvement Using IP-XACT, DVCON 2012.

[12] ARM IP-XACT 1.4 AMBA Bus Definition https://silver.arm.com/download/eula.tm?pv=1202309

[13] Duolog Socrates : http://www.duolog.com/products/socrates/

[14] D.Murray, A. Sherer, Automated UVM Solutions to Common Hardware/Software Integration Problems - , Joint Webinar, 2011

[15] Jasper : Connectivity Verifcation App : http://www.jasper-da.com/products/jaspergold-apps/connectivity-verification-app

[16] Jasper Intelligent Proof Kits : http://jasper-da.com/products/intelligent-proof-kits

[17] Jasper: Control and Status Register Verification App. http://www.jasper-da.com/products/jaspergold_apps/csr_verification_app

[18] D.Murray, S. Boylan: Lessons from the field – IP/SoC integration techniques that work, DVCON 2014.

[19] Cadence Interconnect Workbench: http://www.cadence.com/cadence/newsroom/press_releases/pages/pr.aspx?xml=102913_iwb