WHITE PAPER

Solving Next Generation IP Configurability

David Murray, IP Tooling Architect
IP Tooling, Systems and Software,
ARM – david.murray@arm.com

Simon Rance, Senior Product Manager,
IP Tooling, Systems and Software,
ARM – simon.rance@arm.com

ABSTRACT

IP configurability has continued to be a key challenge in IP development and system integration. Increases in IP reuse, IP configurability and system complexity within tightly bound schedules have compounded the problem of IP configurability and system integration for optimal end applications.

We will look at how configurability is modeled in design flows and try and understand where our current design flows are limited. As well as looking at the standard RTL languages, IP-XACT will also be included in this.

This white paper will preset the various types and complexities of IP configurability and how these can be managed effectively using IP-XACT and other means. It will also provide an overview of what is currently being done in the industry and demonstrate different types of configurability solutions.

This paper will then detail a system integration solution with configurable IP. The proposed solution can provide a standard mechanism to define conditionality and model configurability that can be leveraged across IP providers, IP consumers and EDA solution providers.

Keywords—IP configurability; IP Reuse; IP Integration; IP-XACT

INTRODUCTION

IP configurability is evolving due to today's highly complex systems and competitive IP market. IP providers have to deliver reasonable cost, high quality and highly configurable IP to meet various end applications. This poses a unique challenge to the IP provider because they need to create and deliver IP without knowing how that IP will be integrated and configured in the IP consumers system. The IP needs to be highly configurable to suit a wide variety of end applications.

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 and highly configurable 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.

The configurability of semiconductor IP continues to be a key challenge in IP development and is quickly becoming not just a hardware and software design challenge, but also a challenge for verification. Verifying all the combinations of IP configurations is a very challenging task for not only the IP provider, but also the IP consumer.

Yesterday's IP configurability challenges comprised of offering simple RTL configurability for hardware design using ifdef parameters; although these challenges were not that long ago, they seem so distant compared with today's challenges comprising of a high level of intricate configurability on both the hardware and software sides.

As IP adoption and reuse become more mainstream in SoC realization, IP configurability is increasingly seen as a key challenge in enabling various next generation end applications using the same IP. There are many IP configuration challenges.

- Industry standards do not cover the full scope of IP configurability that is needed
- Parameterization complexity goes well beyond industry standards like IP-XACT
- Configurability is becoming IP specific
- IP's are becoming more complex and configurable and can have thousands of ports and hundreds of different configurations.
- Detailed documentation is required to describe the IP configurability so that the IP consumer can optimally configure the IP and integrate into the end application
- The poor adoption of standards and methodologies for IP configurability 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. A solution is needed that provides a balance between abstraction and automation while enabling IP configurability, IP quality, and predictable IP integration.

WHAT IS CONFIGURABLE IP?

A static or fixed IP is an IP that has static Hardware interface and design. There are no configurability options. The IP interface and logic are fixed and not parameterizable. It can be used 'off-the-shelf' and is not dependent on any other part of the system. An example of static IP would be a simple UART interface with no configurability. A configurable IP is an IP that can have different configuration options. These options can be a range of different complexity from simple configurable port widths to configurations of different internal logic, hardware interface and HW/SW interface. From an IP delivery perspective the configurability options can have a dramatic effect on the IP testbench and test environment, documentation, software drivers and physical constraints. In summary adding configurability to an IP can have a dramatic effect on the development complexity of the IP.

Examples of configurable IP

One example of configurable IP would be the DMC400 from ARM. As defined in DMC400 r2p1 TRM [2], this system has:

- A configurable number of ACE-Lite interfaces (1,2 or 4)
- A configurable number of memory channels (1 or 2)

However, each of these interfaces are themselves configurable. For instance each ACE-Lite interface has the following configuration options:

- Data bus width (64, 128, 256 bit)
- Address bus width (32, 40, 64 bit)
- ID bus width (4-24 bit)
- Read burst acceptance capability (16, 32, 64)
- Read hazard acceptance capability (8, 16)

- Read hazard buffer implementation (RAM or synthesized registers)
- Virtual Network existence (true/false)

And for each one of the memory interfaces, the following are configurable.

- Read queue depth (16, 32, 64 bursts)
- Write buffer depth (16, 32, 64 bursts)
- Write buffer implementation (e.g. RAM, or synthesized registers)
- Number of chip selects (1, 2)
- DFI data width (32, 64, 128)
- Maximum DFI burst length (4, 8)
- Enable or disable memory SECDED (true/false)

Similarly Xilinx provide a PCI express v2.2 as an Integrated Block for use with Xilinx Zynq®-7000 All Programmable SoC, and 7 series FPGA families.[3] This PCI express IP Supports 1-lane, 2-lane, 4-lane, and 8-lane operation. Other configuration options are PCIe Device / Port Type, PCIe block location, maximum link speed, AXI interface width and AXI interface frequency.

There can be a high-level of interdependency within these parameters e.g. A 4-lane configuration with link speed of 2.5 Gb/s can only support AXI interface width of 64 bits but a 4-lane configuration of link speed of 5 Gb/S can support 64 or 128 bits. Consequently the recommend frequency for the former v's latter is 250 MHz v's 125 MHz.

From the fixed IP to the highly configurable ones there are a wide range of configurability requirements that need to be managed.

EVOLUTION OF IP CONFIGURABILITY

IP configurability has evolved substantially in recent years given the growth of the semiconductor IP (SIP) market. The evolution of IP configurability spans configurability supported by standards and some requiring more advanced techniques. IP configurability can affect component ports, interfaces, instances, registers etc. The right approach to IP configuration depends on the complexity of the IP that requires configuration. The various types of configurability are:

- Fixed
- RTL parameters
- General parameters
- Conditionality
- Generation

On the low end of complexity lie the fixed and RTL parameters types of configurability. On the high end of complexity lie the conditionality and generation types of configurability.

Fixed type of configurability essentially is a hardened configuration fully supported and described in Verilog, VHDL, and IP-XACT. IP Configurability utilizing RTL parameters offers configuration through IP component instantiation and parameter passing (configurable elements). These are fully supported in Verilog (parameter), VHDL (generic), and IP-XACT (ModelParameter). IP Configurability utilizing general parameters offers a standard approach to parameterization using non-RTL parameters. These general parameters are described using the Accellera IP-XACT standard (IEEE-1685-2009). IP Configurability utilizing conditionality essentially offers conditionality using *'define* in Verilog, and *generate* in VHDL. IP-XACT IEEE-1685-2009 does not offer support for conditionality although native support called 'Conditionality' is part of the upcoming version of IP-XACT (IEEE-1685-2014).

IP Configurability utilizing generation characteristics offers multiple layers of configurability. Multiple layers of configurability can be described such as a parameterizable number of parameterizable interfaces (e.g. 2 AXI interfaces – one has read data size of 32 and the other has read data size of 64). The output of this configurability is generated Verilog, VHDL or IP-XACT. This 'generation' type truly is the next generation of IP configurability.

IP-XACT – THE IP & IP INTERFACE STANDARD

One of the main standards to emerge to solve the problem of IP and 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 the definition of this standard (IEEE-1685-2009).

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. Since 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 and the ability to make IP more 'integration-ready', has the potential to achieve 30% improvement in the time and cost of SoC integration [6].

The IP-XACT forms that are standardized include: components, systems, bus interfaces and connections, abstractions of those buses, and details of the components including address maps, register and field descriptions, and file set descriptions for use in automating design, verification, documentation, and use flows for electronic systems. A set of XML schemas of the form described by the World Wide Web Consortium (w3c) and a set of semantic consistency rules (SCRs) are included. A generator interface that is portable across tool environments is provided.

The specified combination of methodology-independent meta-data and the tool-independent mechanism for accessing that data provides for portability of design data, design methodologies, and environment implementations [7].

An IP-XACT design can describe a hierarchical system and associated connectivity as described in Fig. 1.

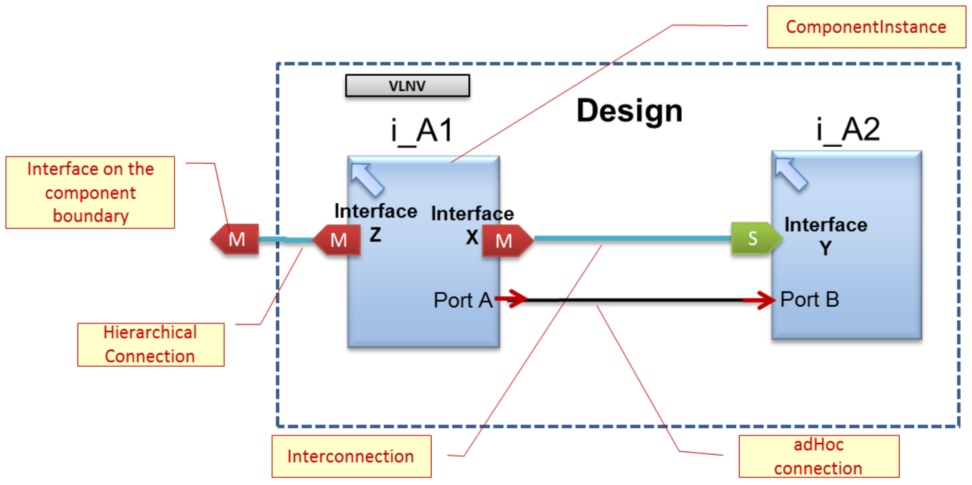


Fig. 1. IP-XACT representation of a design

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 instances interfaces and component interfaces. Ad-hoc connections are port-level connections.

The fact that this type of connectivity can be defined in IP-XACT 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

HOW IS CONFIGURABILITY CURRENTLY MODELED IN RTL AND IP-XACT

In this next section we will look at how configurability is modeled in design flows as well as outlining where our current design flows are limited.

Modeling Fixed IP

Fig. 2 shows an IP with a fixed interface. The data input port is a static 32 bits. This It can be used 'off-the-shelf' and used without additional parameterization.

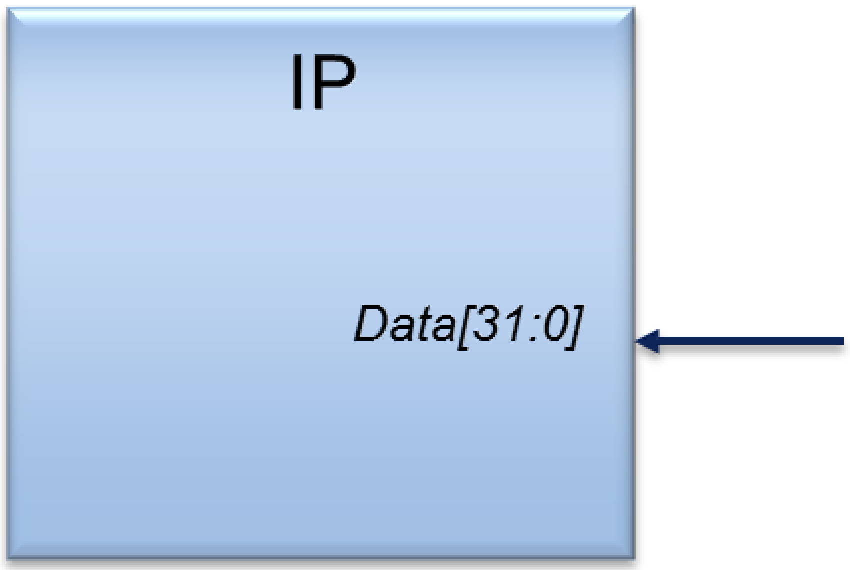


Fig. 2. Example of an fixed IP interface

Modeling Fixed interface but different port widths

Fig. 3 shows a fixed interface but with a parameterizable port width. This level of configurability can be handled in a number of ways natively handled to RTL.

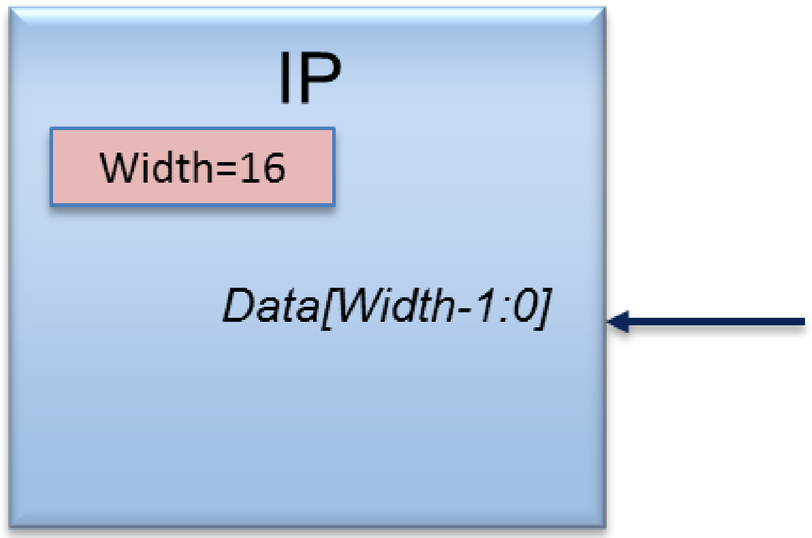


Fig. 3. Example IP with fixed interace with parameterizable port width

In some cases, this configurability is handled through constants values contained in include files or packages. Another method is to use parameterization mechanisms such as VHDL Generics or Verilog parameters to define this configurability. The advantage of parameterization mechanisms over the use of constants/packages is that parameterization allows the same component to be used multiple times in a single design with different sets of parameters. VHDL Generics or Verilog parameters are fully supported in IP-XACT (using the *modelParameters* construct found under *component/model*) and RTL so there is no specific design processing required. IP-XACT doesn't support packages or include files.

For designs there are many different levels of complexity but their effects can be broken into two main categories:

1) Modeling changes in the HW interface of a component

Many configuration options for configurable IP will have an effect on the hardware interface. In the example of the DMC-400, we see that there is a configurable number of ACE-Lite interfaces and memory channels. This will typically have a significant impact on the ports that appear on the IP interface. Another example is shown in Fig. 4 where the DMA interface, scan interface and an interrupt port are conditional on configuration parameters. This characteristic of the existence of a port or interface can be defined as *conditionality*.

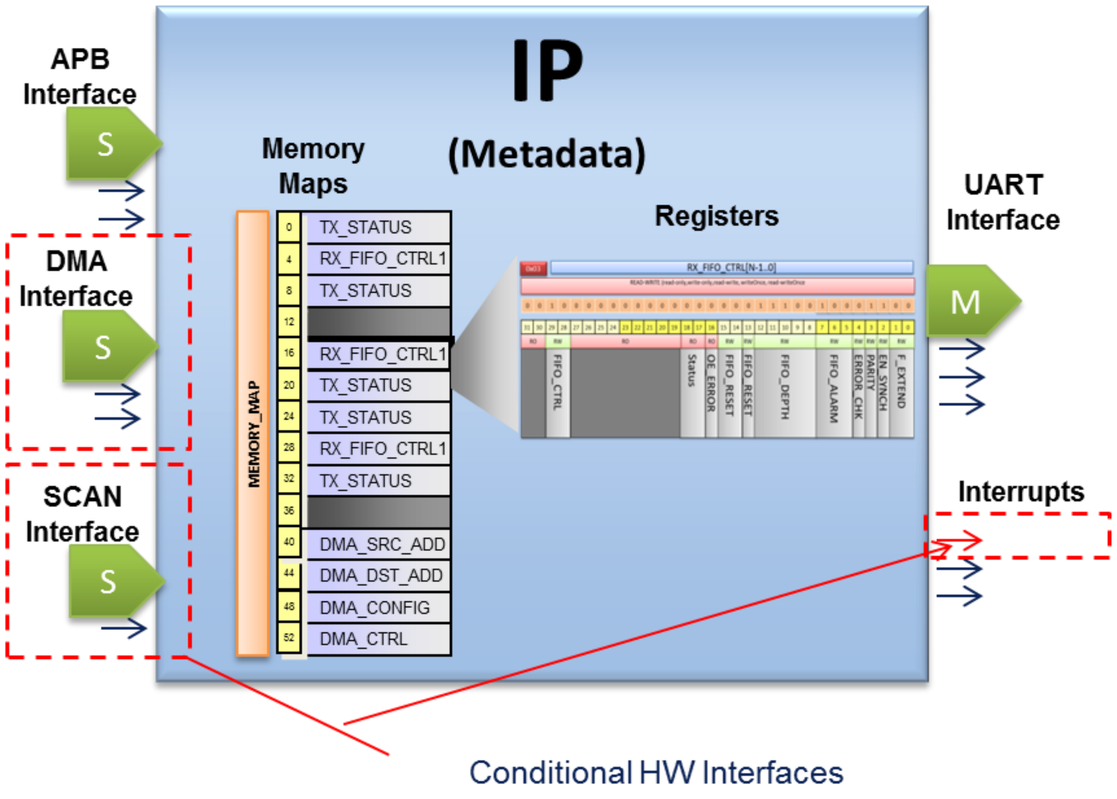


Fig. 4. Example IP with fixed interace with parameterizable port width

Port conditionality has different modeling support depending on the format.

- VHDL does not support port conditionality natively as ports that are declared in VHDL can't dynamically appear and disappear.
- While port conditionality is not a native feature of Verilog it can be implemented through the use of the *'ifdef..'endif* compiler directives. These directives can be put around port declarations, port input/output/inout declarations and when Verilog is compiled the interface can take on the interface constraints.
- Port conditionality is not natively supported in IP-XACT

2) Modeling changes within a components design

In general, if we have to define and constrain the configurability of a design we need to be able to have conditional instances and connectivity. The instances themselves may be configurable.

- VHDL has some limited support for modeling this configurability. The VHDL **generate** statement can be used around concurrent statements to give a type of limited programmability. A typical use case here is to use a generate statement to instantiate an array of components. Inside a VHDL architecture however, signal declarations cannot be put inside the generate statement. It would be possible to parameterize the sizes of signals and the connectivity could be indexed within a generate statement. Thus, this mechanism could be used from some specific use cases. From the last scenario, conditionality is not supported on VHDL ports so overall this is poorly supported in VHDL
- While instance and connectivity conditionality is not a native feature of Verilog, like port declarations, it can also be implemented through the use of the *'ifdef..'endif* compiler directives. These directives can be put around Verilog assign statements or processes and when Verilog is processed the design can take on the intended function.
- Instance and connectivity conditionality is not natively supported in IP-XACT.

3) Modeling changes within a component's HW/SW interface

While Modeling changes within a component's hardware/software interface is similar to the previous scenario, there is a use-case that may require configurability of an IP interface while black-boxing the design. While port/interface conditionality has been considered, there may be a requirement for memory-map configurability, especially when using IP-XACT as an IP specification. For example, the actual DMC-400 configuration register structure may change depending on the configuration settings. While this could be analogous to the HW interface configurability, it can be seen as independent. IP-XACT contains register/field descriptions and it is possible to have a parameterizable size, reset etc. However, it is not possible to parameterize register or bitfield offsets or to have register or bitfield existence be conditional. Parameterization using IP-XACT can be accomplished using IP-XACT model parameters, component parameters or register parameters.

LIMITATIONS AND ISSUES WITH CURRENT MODELING

When analyzing the issues with current modeling it is useful to identify the key organizations and roles that utilize IP. Fig. 5 shows an IP concept shared across between IP providers, IP consumers and the EDA community. In order to truly leverage reusable IP, these three communities and their internal teams need an interoperable solution for configurability.

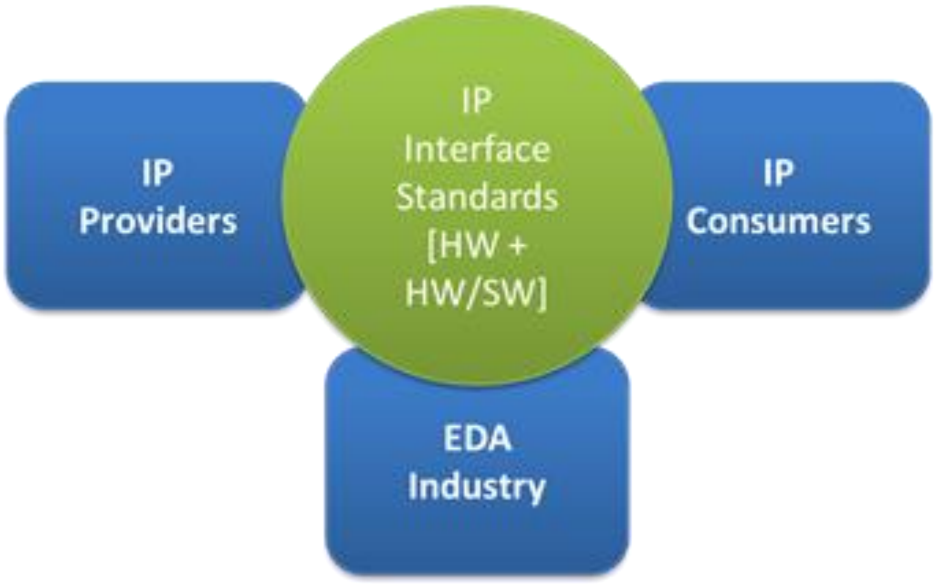


Fig. 5. Organizations sharing IP metadata

Interoperability of IP is enabled when an IP provider can provide IP to an IP consumer and utilize standard EDA flows. From the analysis described in the previous section the following capabilities can be summarized in Fig. 6.

		VHDL	VERILOG	IP-XACT
Fixed Interface		✓	✓	✓
Configurable IP Interface	Port Widths	✓	✓	✓
	Port Existence	✗	✓	✗
	Registers Widths	N/A	N/A	✓
	Registers Existence	N/A	N/A	✗
Configurable IP design	Configurable Instances	✗	✓	✗
	Configurable Connections	✗	✓	✗

Fig. 6. Example IP with fixed interace with parameterizable port width

Verilog, using the *'ifdef* directive, can be used to model most of the RTL configurability. However for VHDL and IP-XACT, having a standard way of expressing even basic port existence is limited. For VHDL, a component entity corresponding to a configuration would need to be somehow generated. Similarly any IP-XACT files that describe the IP would also need to be transformed to align it to its RTL equivalent. Is there a way to model conditionality and will this result in a next generation IP configurability solution?

SOLUTION #1 – MODELING CONDITIONALITY IN IP-XACT – IP-XACT++

One of the first areas to investigate a solution would be if we could somehow extend a standard IP-XACT definition to model conditionality. At its core, the concept is very simple. In IP-XACT we could implement basic conditionality using the following mechanisms:

- Define a boolean attribute (e.g. *isPresent*) on an element (such as a port) that can specify if the element is present or not
- This attribute can be an expression that utilizes IP-XACT parameters and these expressions are resolved using standard IP-XACT methods
- Provide the ability to post-process this IP-XACT file to remove any elements that have the *isPresent* attribute set to false.

The isPresent attribute would need to be stored in an IP-XACT vendorExtension assigned to the element and to differentiate these extensions we will tag this solution as IP-XACT++.

Fig. 7 shows an example of this process:

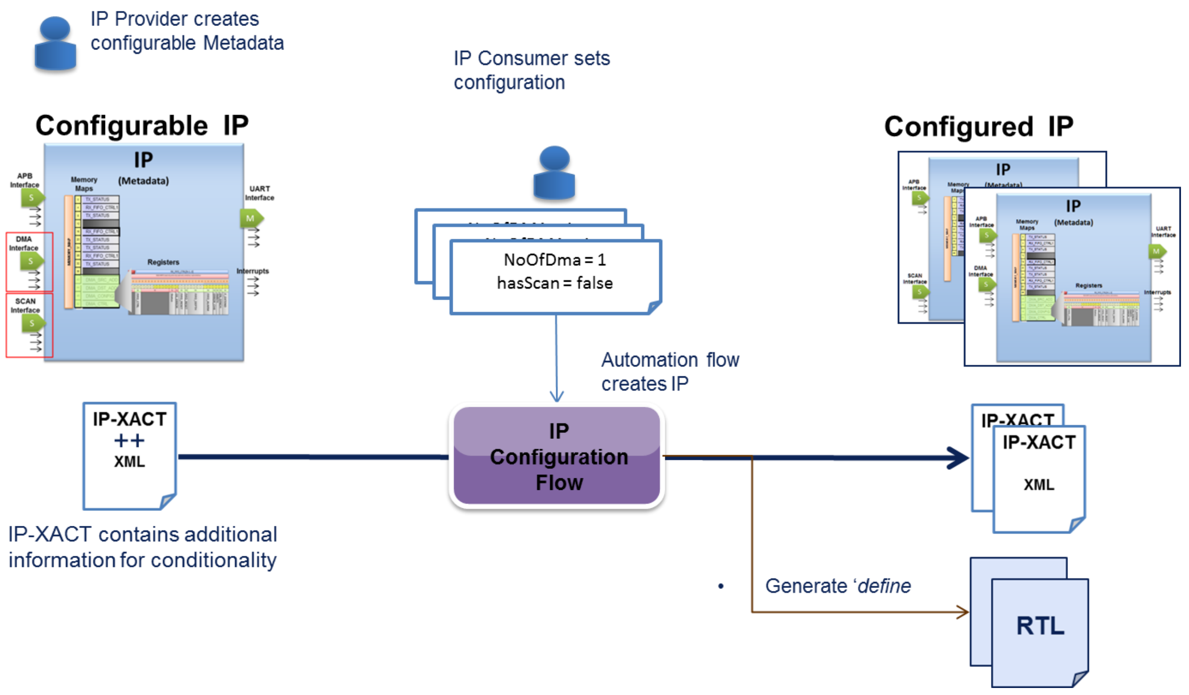

IP Provider creates configurable Metadata
IP Consumer sets configuration
Configurable IP
IP (Metadata)
NoOfDma = 1
hasScan = false
Configured IP
Automation flow creates IP
IP-XACT ++ XML
IP Configuration Flow
IP-XACT XML
IP-XACT contains additional information for conditionality
Generate 'define
RTL

Fig. 7. Example IP with fixed interace with parameterizable port width

Methods similar to this seem to be becoming more common place and requirements for such a mechanism have been defined for the next version of the IP-XACT standard. This will provide the ability to define the following table:

		VHDL	VERILOG	IP-XACT (++)
Fixed Interface		✓	✓	✓
Configurable IP Interface	Port Widths	✓	✓	✓
	Port Existence	✗	✓	✓
	Registers Widths	N/A	N/A	✓
	Registers Existence	N/A	N/A	✓
Configurable IP design	Configurable Instances	✗	✓	✓
	Configurable Connections	✗	✓	✓

Fig. 8. Example IP with fixed interace with parameterizable port width

This solution can provide a standard mechanism to define basic conditionality that can be leveraged across IP providers, IP consumers and EDA solution providers. This solution however has some limitations and nuances and as configurability becomes more complex it isn't very scalable.

B. isPresent limitations

For IP-XACT conditionality to work properly, the resulting IP-XACT after generation is required to be semantically correct. It is relatively straightforward to add an isPresent attribute to a port via a vendorExtension but what happens when that port is mapped to a busDefinition/ AbstractionDefinition port using a businterface portMap. It would mean that the portMap mapping should also be removed. An IP-XACT portMap however does not contain a VendorExtension element, so more processing would be required. The same constraints could apply to adHocConnections and other port dependencies, so as design complexity grows, managing this conditionality also grows.

C. Parameter Dependencies

While these are some issues with the isPresent, there are other concept issues that start to emerge when configurability becomes very complex. In the case of the DMC-400 discussed earlier, we note that there are two levels of parameterizations. There is a parameter to define the number of ACE-Lite interfaces and depending on the value of the parameter then other parameters need to be configured so some parameters themselves may need an isPresent attribute. Another nuance is that depending on one parameter there may be different configuration parameters required. For example, if on a certain IP there were options to have an AHB, AXI or ACE-Lite interface then depending on this selection there could be different options to configure the required interface. An example of the PCI Express IP configuration dependency is shown in Fig. 9:

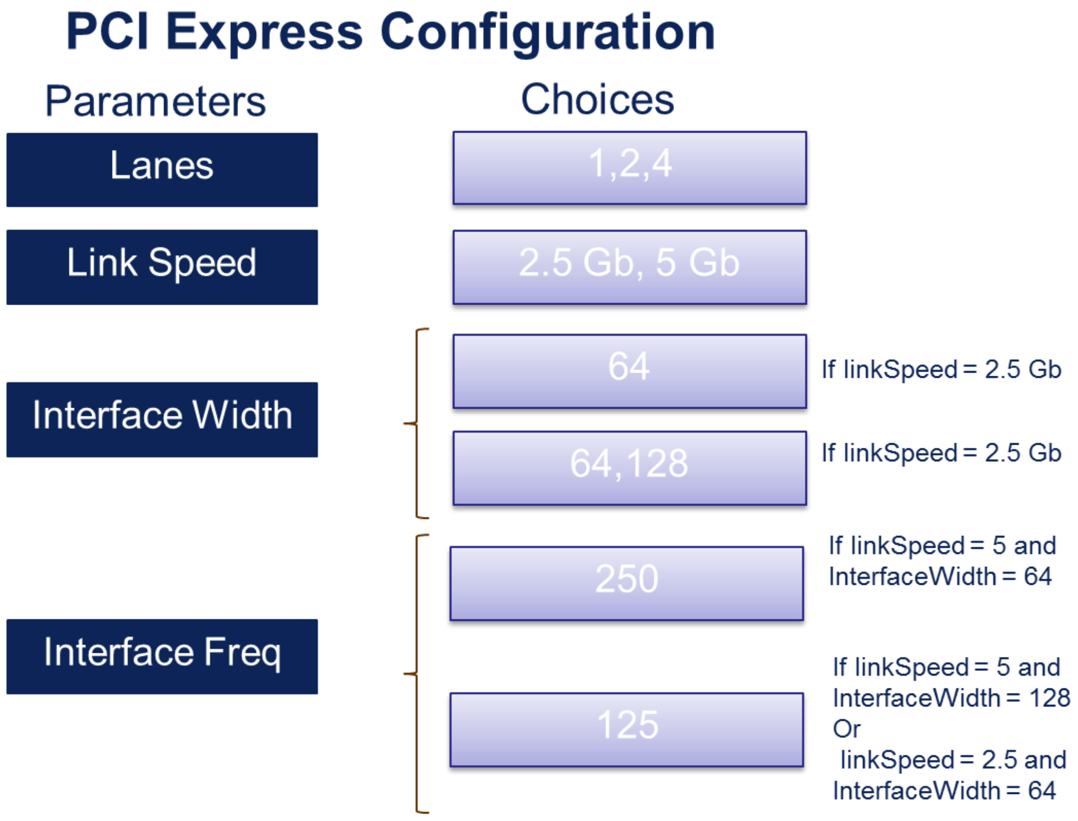

PCI Express Configuration
Parameters
Choices
Lanes
1,2,4
Link Speed
2.5 Gb, 5 Gb
Interface Width
64
If linkSpeed = 2.5 Gb
64,128
If linkSpeed = 2.5 Gb
Interface Freq
250
If linkSpeed = 5 and InterfaceWidth = 64
125
If linkSpeed = 5 and InterfaceWidth = 128 Or linkSpeed = 2.5 and InterfaceWidth = 64

Fig. 9. Example IP with fixed interace with parameterizable port width

In this example we see that the choices for setting interface frequency is dependent on linkspeed and interface width. As parameter dependency increases this conditionality mechanism becomes very unwieldy, and isPresent attributes are used to manipulate IP-XACT metadata at a low-level.

SOLUTION #2 – MODELING CONDITIONALITY

From the analysis so far we see that an extended IP-XACT solution can solve basic configurability and give a standardized IP configuration flow across IP providers, consumers etc. For complex configurations we need a different solution. The following are the key requirements that need to be addressed.

- Configuration options need to be defined in easy to use format and configuration dependencies need to be managed. The use should be able to enter these in a GUI or text format.
- A configuration definition need to be checked to make sure that it's coherent and any errors need to be reported to the user.
- A configuration specification needs to be able to be easily rendered into corresponding IP-XACT and RTL definitions.
- As the configuration parameters, validation and processing are dependent on the IP itself, there should be a very easy user interface available in a range of scripting languages.
- This method should be standardized.

D. Internal IP Providers Solutions

As IP configurability is now a necessity for rapid integration, there are many internal IP teams that adopt their own adhoc methods for IP integration. Configuration parameters can be stored in files and processes with different scripting languages e.g. Perl, TCL, Ruby or Python. IP-XACT, and RTL. These scripts can be very difficult to maintain.

When IP providers deliver third party IP that is highly configurable, it usually comes with configuration utilities. An example of this is ARM, who provide CoreLink™ AMBA Designer that contains a configuration engine that allows rapid configuration of ARM AMBA components and automatic checks to ensure valid configurations.

E. EDA solutions

Synopsys provides CoreBuilder that provides graphical or command based configuration menus for Synopsys provided IP as well as allowing IP providers' IP. ARM provides Socrates that also provides a graphical and command based environment based on IP-XACT as well as the mechanism to configure IP using the IP-XACT++ approach defined previously.

There are also several IP-XACT design environments available although some of them are rigid to the current IP-XACT version and are limited for describing and managing complex configurability. These platforms however can model less complex configurable designs and future IP-XACT releases (IP-XACT++) will address modeling the more complex configurable designs.

F. Other Solutions

Xilinx provides its Vivado platform to assist with its IP customization for FPGA designs. Although not specific to IP-XACT, it provides an ideal solution for FPGA designers to handle configurability for complex FPGA designs.

SOLUTION #3 – IDEAL SOLUTION

G. Modeling Configurability

A generic ideal solution is needed that provides a full IP-XACT design environment, as well as the mechanism to configure IP using the IP-XACT++ approach defined previously. An ideal solution should have the ability to:

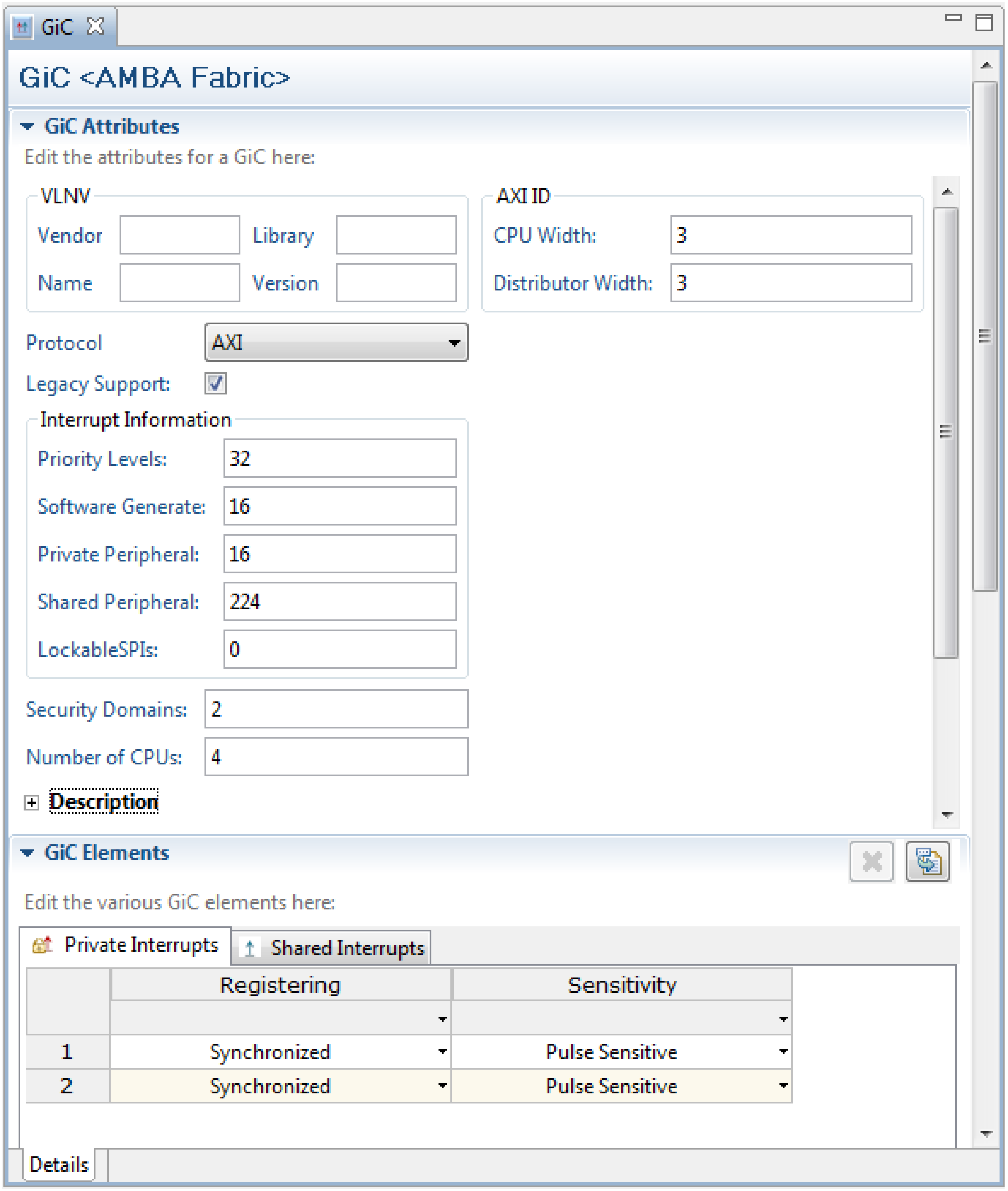
- Define any structured data model
- Render command APIs to the model
- Render it easily into a GUI for visualization
- Run checks using a scriptable API
- Generate IP-XACT
- Generate other formats for HW, SW, DV & documentation

For example, a GIC Generic Interrupt Controller specified in an XML format could provide a tree-like structure of a configuration model as follows:

- GiC
  - CPU_AXI_ID_Width
  - Description
  - Distributor_AXI_ID_Width
  - Legacy Interrupts Support
  - Library
  - LockableSPIs
  - Name
  - NumberOfCPUs
  - PriorityLevels

  - PrivateInterrupts
    - PrivateInterrupt
      - Registering
      - Sensitivity
  - Private_Peripheral_Interrupts
  - Protocol
  - SecurityDomains
  - SharedInterrupts
  - SharedPeripheralInterrupts
  - SoftwareGenerateInterrupts
  - Vendor
  - Version

An ideal solution would be able to take this defined XML configuration model and render it automatically into a GUI as shown in Fig. 10.



GiC <AMBA Fabric>

GiC Attributes

Edit the attributes for a GiC here:

VLNV

Vendor		Library	
Name		Version	

AXI ID

CPU Width:	3
Distributor Width:	3

Protocol	AXI
Legacy Support:	✓

Interrupt Information

Priority Levels:	32
Software Generate:	16
Private Peripheral:	16
Shared Peripheral:	224
LockableSPIs:	0

Security Domains:	2
Number of CPUs:	4

Description

GiC Elements

Edit the various GiC elements here:

Private Interrupts | Shared Interrupts

	Registering	Sensitivity
1	Synchronized	Pulse Sensitive
2	Synchronized	Pulse Sensitive

Fig. 10. Example GUI rendoring of a GiC configurable model rendered using Eclipse

A graphical representation of the configuration model of the GiC would provide the user with visualization of what can be configured and the types of inputs that are potentially available and acceptable to the model. Although an ideal solution should also allow the user to configure the IP model via commands, a GUI rendering enables the user to visualize the model and configurations options quickly while potentially minimizing human error for configurable data entry.

An ideal solution would also have the capability to develop and manage scripted flows to manage data dependencies and configuration validation. Complex design configurations typically will require several data model configurations and may have dependencies between the data model configurations themselves. For example, defining a particular configuration for the example GiC described above may have a dependency impact on other IP in the design. The ability to script flows to manage the inter-model dependencies are needed to ensure configuration coherency across the overall design. These scripted flows could also be leveraged to provide configuration warnings and errors that can be directly fed back to the GUI or command line for instant feedback. This will allow design configuration errors to be flagged and managed accordingly earlier in the design flow potentially reducing the burden and effort on verification downstream.

The ideal solution should leverage and build upon the current IP-XACT standard (IEEE1685-2009) to handle multiple configuration models and inter-dependencies. By leveraging the IP-XACT standard as the underpin base model for the design, a full IP-XACT API can be utilized to allow fast creation of IP-XACT metadata. The API is generic as well as model agnostic and comes with helper methods to find elements and attributes within the configuration model. A full IP-XACT SCR checker is also needed to ensure that it adheres to the standard (IEEE1685-2009), as well as other utilities for register management, connectivity and RTL creation.

SUMMARY

In summary, the growth of third party IP usage and the lack of hardware and hardware/software interface standardization. In order to truly leverage reusable IP, the three communities (IP providers, IP consumers and EDA industry) and their internal teams need an interoperable solution for configurability. There is a way to model conditionality that will result in a next generation IP configurability solution.

The key is to model conditionality in a standard like that proposed for IP-XACT++ in this paper as well as modeling configurability. The ideal solution combines both the conditionality proposed for IP-XACT++ and the ability to model configurability. The result is next generation IP configurability that covers hardware and software design, their interfaces, inter-dependencies and associated documentation detailing the configurability of the IP. Such a solution should satisfy both IP providers and IP consumers.

REFERENCES

[1] UART Design: http://www.asic-world.com/code/hdl_models/uart.v

[2] DMC-400 TRM, ARM: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0466a/CACDGAJE.html

[3] PCIE Design, Xilinx: http://www.xilinx.com/support/documentation/ip_documentation/pcie_7x/v2_2/pg054-7series-pcie.pdf

[4] AMBA Designer: http://www.arm.com/products/system-ip/amba-design-tools/amba-designer.php

[5] Synopsys Core builder: http://www.synopsys.com/dw/ipdir.php?ds=core_builder

[6] 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

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