Solving Next Generation IP Configurability

David Murray, CTO,
Duolog Technologies

Simon Rance, VP of Product Marketing,
Duolog Technologies
INTRODUCTION
IP Configurability is Evolving

• IP configurability is evolving due to today’s highly complex systems and competitive IP market

• IP providers are having to deliver reasonable cost, high quality and highly configurable IP to meet various end applications
This Poses Unique Challenges …

• The IP provider needs 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
IP Configurability Challenges

• Yesterday’s challenges comprised of offering simple RTL configurability for hardware design using concepts like ifdef parameters

• Today’s challenges comprise of a high level of intricate configurability on both the hardware and software sides
IP Configuration Challenges

• Industry standards do not cover the full scope
• Configurability is becoming IP specific
• Detailed documentation is required to describe the IP configurability
• Poor adoption of standards and methodologies for IP configurability is making efficient and reusable integration more difficult
What is Needed as a Result?

• 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
  – Predictable IP integration
WHAT IS CONFIGURABLE IP?
Static or Fixed IP

• An IP that has a static hardware interface and design
• No configurability options
• IP interface and logic are fixed and not parameterizable
• Not dependent on any other part of the system
• Can be used off-the-shelf
Configurable IP

• An IP that has different configuration options
• Options can be a range of different complexity
• Simple configurability example
  – Configurable port widths
• Complex configurability example
  – Configurations of different internal logic, hardware interface and HW/SW interface
Example of Configurable IP

CoreLink™ DMC-400 Dynamic Memory Controller

Highly Configurable
DMC-400: What is Configurable?

• Each of the ACE-Lite interfaces are configurable
  – 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)
  – …

• Each of the memory interfaces are configurable
IP-XACT

THE IP & IP INTERFACE STANDARD
What is IP-XACT?

Interoperable standard format between IP providers, consumers and EDA vendors that represents a design.
IP-XACT

Representation of a Design

An IP-XACT design element defines a hierarchical structure and associated connectivity.

- Interface on the component boundary
- Hierarchical Connection
- Monitor Interconnection
- Monitor Interface
- Interconnection
- Component Instance
- adHoc connection

Design

i_A1

Interface Z

Interface X

Port A

i_A2

Interface Y

Port B

Monitor Interface

Interconnection
IP-XACT Vendor Extension

An extension point that can be used on IP-XACT elements:

– component
– ports
– registers
– fields
– instances
– designs
– ...
HOW IS CONFIGURABILITY CURRENTLY MODELED IN RTL AND IP-XACT?
Types of Configurability

IP Configuration approach depends on the complexity of the IP
IP Configurability - Fixed

**Characteristics**
- No configurability

**Flows**
- No configuration flows
- Fully supported in VHDL, Verilog and IP-XACT
IP Configurability – RTL Parameters

Characteristics

– Configuration through component instantiation and parameter passing (configurable elements)

Flows

– Fully supported in
  • VHDL (Generic),
  • Verilog (Parameter)
  • IP-XACT DE (ModelParameter)
IP Configurability – General Parameters

Characteristics
– Non RTL parameters

Flows
– IP-XACT – Parameters
IP Configurability - Conditionality

Characteristics
- Conditionality

Flows
- Verilog: using `define
- VHDL: `generate` (Limited)

Native support in IEEE-1685-2014 called ‘Conditionality’
IP Configurability - Generation

Characteristics

- Multiple layers of configurability
- E.g. a parameterizable number of parameterizable interfaces
  - E.g. 2 AXI interfaces – 1 has read data size of 32 and the other has read data size of 64

Flows

- VHDL, Verilog and IP-XACT is GENERATED
LIMITATIONS AND ISSUES WITH CURRENT MODELING
IP Configuration Challenges

With VendorExtensions (Limited)

- Industry standards do not cover full scope
- Parameterization complexity goes well beyond IP-XACT
- Configurability becomes IP specific

### IP Configuration Challenges

- **Generation**
- **Conditionality**
- **General Parameters**
- **RTL Parameters**
- **Fixed**

**Languages**

- VHDL
- Verilog
- IP-XACT

**Notes**

- Industry standards do not cover full scope
- Parameterization complexity goes well beyond IP-XACT
- Configurability becomes IP specific
SOLUTION #1
MODELING CONDITIONALITY IN
IP-XACT – IP-XACT++
IP-XACT++ Solution

- Many customers use IP-XACT Vendor Extensions to model the concept of Conditionality
- Conditionality attributes (e.g. isPresent) are assigned to different elements
- Conditionality attributes can have expressions dependent on parameters
- Many parameters have now have dependencies
- An overloaded IP-XACT definition is used
- This will always be a non-standard flow
- Not scalable as complexity grows
## IP-XACT++ Solution

### Advantages
- Still using IP-XACT
- Flow can be used across different IP
- All the data (and flows) are in one file

### Disadvantages
- IP-XACT becomes very complex very quickly
- A lot of duplication
- Difficult to manage parameter dependency and validation
- Logic/Behaviour is not easy to define in IP-XACT structure
- This will always be a non-standard flow
- Not scalable as complexity grows
SOLUTION #2
MODELING CONFIGURABILITY
Key Requirements

• Configuration options
• Configuration definition needs to be validated
• Configuration spec needs to be rendered into RTL & IP-XACT
• Simple user interface with a range of scripting languages to manage configuration parameters, validation and processing.
• Method should be standardized
IP Providers Solutions

• Configuration parameters stored in files and processes with different scripting languages

• Configuration utilities need to be provided to end users/customers
  – e.g. CoreLink™ AMBA Designer by ARM to rapidly configure ARM AMBA components and run automatic checks to ensure valid configurations
EDA Solutions

• EDA tools need to provide graphical or command-based configuration models

• Tool environment is ideally based on IP-XACT as well as the mechanism to configure IP using the IP-XACT++ approach described previously
SOLUTION #3
IDEAL SOLUTION
Generic Ideal Solution

• **Provides:**
  – Full IP-XACT design environment
  – Mechanism to configure IP using the IP-XACT++ approach

• **Has the ability to:**
  – Define any structured data model
  – Render command APIs to the model automatically
  – 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
Example Configuration Model

XML Tree Structure of a Configuration Model

- GIC (Generic Interrupt Controller)
  - 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 to aid in rapid & correct configurability.
Automatically Rendered into a GUI to Guide Configurability