Lessons from the field
IP/SoC integration techniques that work

David Murray, CTO,
Duolog Technologies
Acknowledgements

• Sean Boylan, Duolog
• Sujatha Sriram, ARM
4 Pictures – 1 Word

CONNECT
Overview

- Challenges for IP-based SoC Integration
- Integration methodology (Rules-based)
- Case-Study/Results
- Future Directions
IP Integration
IP-Based SoC Integration

- ARM Based system
  - 4 Processor clusters
  - 3 bus Interconnect systems (CCI/NICx2)
  - 4 sub-systems, (DMC, Peripheral, LCD, Interconnect)
  - 35 independent IP’s to be integrated
  - The AMBA® protocols within the system included APB™, AHB™, AHB-Lite™, AXI™, ATP™, LPI™, AXI4™, APB4™, ACE™ and ACE-Lite™
Challenge - Complexity

• Complexity
• Configurability
• Standardization
• Quality
• Collaboration
IP-Based SoC Integration Solution

- IP Interface standardization

- Efficient SoC Assembly/Connectivity
  - Standard Connectivity
  - Hierarchy Management
  - Configurability
  - Reusability
  - Interoperability
  - Usability!
Managing Hierarchy

- Hierarchy
- Managing Boundaries
- Collaboration
- Fluidity
- Refactoring
IP Interface Standardization

- Executable IP Interface Specification
  - HW Interface
    - Bus Interfaces
    - Ports (Mapping to bus definition signals)
  - SW Interface
    - Memory Maps
    - Registers
    - Bitfields
  - Consumption
    - IP Automation Flows
    - Integration Flows
  - Industry Standardization
    - IP-XACT
    - Common Bus definitions
Rules-Based Integration Methodology

- Select & configure integration-ready IP from libraries
- Rules-based system integration
  - Text-based Integration instructions
- Define & refine system hierarchy
Integration Instructions

- Powerful Integration Instructions operate on IP Interface metadata

Component Instances

- Connect
- Export
- Reflect
- Insert
- '0'
- Tieoff

Component Periphery

* = Created
In the sample SubSystem, bring all instance ports of type [Interrupt] up to the next level of hierarchy and prefix the interface ports with the name of the source instance.

```plaintext
# Export ports of type Interrupt
rule("Export Interrupts") {
  export instances.ports{definition "Interrupt"}, :port_name => "$(instance)_${port}"
}
```
Connection Example

ACE Interface has 60+ signals

Connect instances("uP1").interface("M1"),
instances("uP1").interface("S1")
Concurrent Integration

- Connectivity Ownership
- Merging or Include
- Autogenerated IP interfaces

More users can be added to focus on specific types of connectivity.
Full System Creation

- Synthesis of hierarchical system connectivity
- Netlist generation
- Rapid Integration timeframes
- Integrated with other flows
  - IO, Bus fabric, power, verification
Case-Study

ARM IP-Based SoC Integration
ARM IP- Based System

- ARM Based system
  - 4 Processor clusters
  - 3 bus Interconnect systems (CCI/NICx2)
  - 4 sub-systems, (DMC, Peripheral, LCD, Interconnect)
- 12,000+ lines of verilog code describing connectivity
  - 6-7 Weeks to develop normally
  - Many connectivity errors
• 35 IP leaf components
• Most IP had IP-XACT descriptions
• Additional IP packaging
  • The packaging of each IP, with creation of relevant bus interfaces took 1-2 hours per IP.
• Utilized 109 IP-XACT bus definitions
3 engineers worked within 3 levels of hierarchy
Results

- The integration of 3 sub-systems, 1 major sub-system and a top-level integration was completed within 4 schedule days.
- This activity could have taken 35+ working days in the past.

<table>
<thead>
<tr>
<th>Integration Task</th>
<th>Duration (Days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMC Subsystem</td>
<td>1</td>
</tr>
<tr>
<td>SMC Subsystem</td>
<td>1</td>
</tr>
<tr>
<td>LCD Subsystem</td>
<td>1</td>
</tr>
<tr>
<td>NIC Subsystem</td>
<td>2</td>
</tr>
<tr>
<td>Top-level</td>
<td>2</td>
</tr>
</tbody>
</table>

An 8x schedule improvement over previous methods
Metrics - % Connectivity/Day

Device Top Connectivity

- Bus Interconnect
- CPU Packaging
- Export Connectivity
- Sub Systems
  - Interconnect
  - LCD
  - Interrupt controller
- Utilities Library
- CLK/RESET
- CPU Tieoff
- Interrupts

Merge

Team working concurrently

TUESDAY

WEDNESDAY

THU
The average ratio of instructions to lines of Verilog code was 31:1.

- The 3 levels of hierarchy were put together using 41 rules, with 372 instructions.
- These were synthesized and netlisted to 12,045 lines of Verilog code in total.

- Rules
  - Higher level of abstraction
  - High level of reuse
  - Easier to maintain
  - Higher quality
Considerations

• Some members of the team were not familiar with the target architecture and needed to understand the connectivity by walking through the legacy Verilog code
• Some members were not familiar with Weaver and rules-based approach and needed a quick ramp up
• The rules layout had to be developed throughout this task
• Some packaging had to be done within the integration activity
• Rules optimizations were performed within the integration. This included the creation of macros for commonly repeated tasks.
Connect_AMBA function

- Raising the level of user abstraction for AMBA connectivity
- ACE can connect to ACELite, AXI4, AXI3
- Different tieoffs needed

<table>
<thead>
<tr>
<th></th>
<th>ACE</th>
<th>ACE-lite</th>
<th>AMBA</th>
<th>AXI4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLK</td>
<td>ACLK</td>
<td>ACLK</td>
<td>ACLK</td>
<td>ACLK</td>
</tr>
<tr>
<td>ACLKEN</td>
<td>ACLKEN</td>
<td>ACLKEN</td>
<td>ACLKEN</td>
<td>ACLKEN</td>
</tr>
<tr>
<td>ACADDRESS</td>
<td>ACADDR</td>
<td>ACADDR</td>
<td>ACADDR</td>
<td>ACADDR</td>
</tr>
<tr>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
</tr>
<tr>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
</tr>
<tr>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
</tr>
<tr>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
</tr>
<tr>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
</tr>
<tr>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
</tr>
<tr>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
</tr>
<tr>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
</tr>
<tr>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
</tr>
<tr>
<td>AR</td>
<td>AR</td>
<td>AR</td>
<td>AR</td>
<td>AR</td>
</tr>
<tr>
<td>ARID</td>
<td>ARID</td>
<td>ARID</td>
<td>ARID</td>
<td>ARID</td>
</tr>
<tr>
<td>ARLEN</td>
<td>ARLEN</td>
<td>ARLEN</td>
<td>ARLEN</td>
<td>ARLEN</td>
</tr>
<tr>
<td>ARLOCK</td>
<td>ARLOCK</td>
<td>ARLOCK</td>
<td>ARLOCK</td>
<td>ARLOCK</td>
</tr>
<tr>
<td>ARQOS</td>
<td>ARQOS</td>
<td>ARQOS</td>
<td>ARQOS</td>
<td>ARQOS</td>
</tr>
<tr>
<td>ARREADY</td>
<td>ARREADY</td>
<td>ARREADY</td>
<td>ARREADY</td>
<td>ARREADY</td>
</tr>
<tr>
<td>ARREGION</td>
<td>ARREGION</td>
<td>ARREGION</td>
<td>ARREGION</td>
<td>ARREGION</td>
</tr>
<tr>
<td>ARUSER</td>
<td>ARUSER</td>
<td>ARUSER</td>
<td>ARUSER</td>
<td>ARUSER</td>
</tr>
<tr>
<td>ARVALID</td>
<td>ARVALID</td>
<td>ARVALID</td>
<td>ARVALID</td>
<td>ARVALID</td>
</tr>
<tr>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
<td>AWR</td>
</tr>
<tr>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
<td>AWRID</td>
</tr>
<tr>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
<td>AWRLEN</td>
</tr>
<tr>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
<td>AWRLOCK</td>
</tr>
<tr>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
<td>AWRQOS</td>
</tr>
<tr>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
<td>AWRREADY</td>
</tr>
<tr>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
<td>AWRREGION</td>
</tr>
<tr>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
</tr>
<tr>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
</tr>
<tr>
<td>AW</td>
<td>AW</td>
<td>AW</td>
<td>AW</td>
<td>AW</td>
</tr>
<tr>
<td>AWID</td>
<td>AWID</td>
<td>AWID</td>
<td>AWID</td>
<td>AWID</td>
</tr>
<tr>
<td>AWLEN</td>
<td>AWLEN</td>
<td>AWLEN</td>
<td>AWLEN</td>
<td>AWLEN</td>
</tr>
<tr>
<td>AWLOCK</td>
<td>AWLOCK</td>
<td>AWLOCK</td>
<td>AWLOCK</td>
<td>AWLOCK</td>
</tr>
<tr>
<td>AWQOS</td>
<td>AWQOS</td>
<td>AWQOS</td>
<td>AWQOS</td>
<td>AWQOS</td>
</tr>
<tr>
<td>AWREADY</td>
<td>AWREADY</td>
<td>AWREADY</td>
<td>AWREADY</td>
<td>AWREADY</td>
</tr>
<tr>
<td>AWREGION</td>
<td>AWREGION</td>
<td>AWREGION</td>
<td>AWREGION</td>
<td>AWREGION</td>
</tr>
<tr>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
<td>AWUSER</td>
</tr>
<tr>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
<td>AWRVALID</td>
</tr>
</tbody>
</table>
System Configurability

- User sets specific configuration parameters
- Rules are designed to handle different configurations
- Weaver creates a specific configuration of the system.

Any one of thousands of possible configurations can be generated in less than 7 minutes.
30x Schedule Improvement??

- Build 1 of 1000s of systems in 7 minutes
- Interface standardization – Interrupt controller upgrade in 15 minutes
Benefits

• **Schedule:**
  - It is estimated that once the methodology is deployed, it is reasonable to expect a 10x-15x improvement in schedule for new projects and 20x-30x for derivative projects.

• **Quality**
  - This netlist was right-first-time as it passed a previous regression test. IP standardization with formal rules eliminates a lot of tedious errors.

• **Productivity**
  - Massive improvements in productivity (10x) is expected with this methodology
Future Considerations

• Interface Standardization
• Chip-in-a-day
• Verification Enablement
Conclusion

• A rules-base methodology with powerful integration instructions and selection mechanism can reduce the overall SoC integration task by a factor of 15x-30x.
• IP Standardization is a key enabler for this flow and IP-XACT provides an interoperable method of formalizing IP interfaces.
• The rules themselves are specified using a very small instruction set (DSL) that can be instantly used by anyone familiar with the integration domain.
• Highly configurable systems can be rendered very quickly with this methodology.
Conclusion - 4 Pictures – 1 Word

- To Make ..
- To Interlace ..
- To construct ..
Questions??
Thank You

• Please visit us in Booth 1002