Use Case: Verification Automation Improvement Using IP-XACT

Kamlesh Pathak
STMicroelectronics
Agenda

- Typical challenges in verification
- IP-XACT offerings for verification automation
- Applying IP-XACT for verification automation
- Overcoming challenges
- Conclusion
- Q&A
Typical Challenges in Verification

- Developing testbench
  - IP Integration needs knowledge of IP
  - Mechanisms for accommodating IP configuration

- Use of multiple IP suppliers results in inconsistent IP verification views
  - Methodology, testbenches, coverage data, etc. Industry standards late

- Awareness and impact of IP implementation changes & known problems

- Concurrent IP development and SoC integration demands incremental maturity

- Difficult to debug complex interactions between IPs
Typical Challenges in Verification

- **Register implementation**
  - Handling register descriptions at multiple places
  - Cost, productivity, quality
  - Coherency between design teams

- **Writing register test cases**
  - Large number of registers

- **Impact of changes in specification**
  - Additions/changes are problematic and error prone

- **Reuse IP test cases at top level**
  - Huge effort! What subset is needed for integration verification?
IP-XACT Offerings for Verification Automation

- **Single description for all information**
  - All representations generated from the single source

- **Current version of IP-XACT (IEEE 1685-2009) provides**
  - Metadata to describe components, designs
    - Interfaces, ports, registers, bit-fields
    - Component instances, connections between components
    - Configurable attributes
  - Automated configuration of IPs
  - Automated composition, integration and configuration of verification environment
  - Automatic insertion of required transactors based on the abstraction
  - Parameters of design and verification components
  - Design configuration file
  - Easy interface for generators
IP-XACT Offerings for Verification Automation

- Supports verification components
  - Monitor interfaces
  - White-box interfaces
  - Complete API for metadata exchange and database querying
  - Generator plug-ins support to enable automated configuration

- Portability across multiple tools, multiple vendors, EDA

- Command line tools, GUI based tools, EDA

- Provides language and vendor independent description of the testbench configuration and connection to DUT
Applying IP-XACT for Verification Automation

- Based on IP-XACT (1.4/IEEE 1685-2009)
- Automatic IP Packaging in IP-XACT via
  - Functional specification (Framemaker, Word)
  - HDL (Verilog, VHDL)
  - Legacy format (PMAP), custom Excel descriptions
- Automatic generation of verification testbenches
  - From specs, Excel, EDA GUI, etc.
  - In TLM, RTL, mixed TLM-RTL abstraction
- Quickly adaptable to any change in DUT, design, etc.
- Reusable across different design teams and different projects
Applying IP-XACT for Verification Automation

- Function specification
- Excel register description
- Legacy register description/ HDL

Third-party IPs

- IP-XACT packager
- IP-XACT library

- C test generator
- PLT Assembly (Excel/EDA)

- Design XML
- Netlister (EDA)

- C register test
- Verification

C header

Functional test
Applying IP-XACT for Verification Automation

- **Design import**
  - IP-XACT Design XML
  - EDA netlister
  - Verification TB

- **xls tables**

- **IP Database**

- **DUT and other related components and their connectivity**
Applying IP-XACT for Verification Automation

- Quickly adaptable to specification changes
- Single source ensures coherency
Applying IP-XACT for Verification Automation

- Function spec
- Excel register description
- Legacy format for registers (PMAP)
- Register gui
- Others

Single Source (IP-XACT XML)

```xml
<spirit:name>MCR</spirit:name>
<spirit:addressOffset>0x0000</spirit:addressOffset>
<spirit:size>32</spirit:size>
<spirit:access>read-write</spirit:access>
<spirit:value>0x00004001</spirit:value>

<spirit:name>MSTR</spirit:name>
<spirit:bitOffset>31</spirit:bitOffset>
<spirit:access>read-write</spirit:access>
```

C header

```c
#define MCR_SIZE     (32)
#define MCR_OFFSET (0x4)
#define MCR_RESET_VALUE (0x6)
#define MCR_BITFIELD_MASK (0xFFFFFFFF)

#define DATAREADY_OFFSET  (0x0)
#define DATAREADY_WIDTH     (1)
#define DATAREADY_MASK    (0x1)
```

ASM

```asm
/* MCR - Module Configuration Register */
equ DSPI_A_MCR, (DSPI_A_REGS_BASE+0x0)
```

SV

```sv
/* MCR - Module Configuration Register */
define DSPI_A_MCR (`REG_BASE + 32’h0000_0000)
```

C register test

```c
errorNbr += VALregister_test_32(address, data)
pattern = 0x55555555;
errorNbr += RWregister_L_test_32(address, pattern, uart_data_RWMASK);
pattern = 0xAAAAAAAA;
errorNbr += RWregister_L_test_32(address, pattern, uart_data_RWMASK);
```

Mnemonic Map

- `errorNbr += VALregister_test_32(address, data)`
- `pattern = 0x55555555;`
- `errorNbr += RWregister_L_test_32(address, pattern, uart_data_RWMASK);`
- `pattern = 0xAAAAAAAA;`
- `errorNbr += RWregister_L_test_32(address, pattern, uart_data_RWMASK);`
Overcoming Flow Challenges

- **Bus-definition misalignment**
  - Integration issues due to misalignment in bus definitions
  - Use of own copy of bus definition/abstraction definition of same protocol
  - Inconsistency and misalignment between different teams

- **Solutions**
  - Standardize generic bus definitions
  - Centralized bus definitions used by different teams
Overcoming Flow Challenges

- One-to-many connections for bus interfaces are not allowed in IP-XACT
  - « IP-XACT SCR 2.3 : A particular component/bus interface combination shall appear in only one interconnection element in a design »

- Solutions
  - EDA tools allow one to many connections (with a warning)
    - violates IP-XACT compliance
  - Auto-insertion of a virtual component to manage the one-to-many connection
Overcoming Flow Challenges

- Integration issues due to incomplete/incorrect IP-XACT descriptions
  - Solution
    - Built a set of utilities to create and complete the IP-XACT descriptions
    - Built/use checker utility to ensure
      - IP-XACT description compliance w.r.t. IP-XACT schema, semantic rules
      - IP-XACT description compliance w.r.t. to custom requirements for specific flows

- Integration issue due to different schema versions
  - Solution
    - Built convertors to align on schema version (1.4 => 1685-2009)
Overcoming Flow Challenges

- Multiple IP-XACT view addressing different needs
  - Different teams responsible for assembly, verification, etc. results multiple IP-XACT files
  - Registers from spec/Excel
  - Interfaces from HDL
  - Additional information (fileSets, etc.)

- Solution
  - Built a utility to merge the several IP-XACT descriptions with different information
Overcoming Flow Challenges

- Configurable IP-XACT descriptions and TGI limitations
  - TGI APIs are not capable enough to handle generic IP-XACT component descriptions
  - Accellera Systems Initiative IP-XACT TC requirement 42, SWG
  - Not easy to handle configurable IPs

- Solution
  - Defined a set of vendor extensions to specify configurability (plan to be standardize later in Accellera/IEEE)
  - No of ports, registers
  - Presence, absence of interfaces/registers/ports
  - Many more
  - Built a generic generator based on the predefined vendor extensions to generate configured IP-XACT description
Overcoming Flow Challenges

- **Register configurability**
  - Added specific vendor extensions to specify the configurability and standalone generator based on these vendor extensions to create configured IP-XACT

- **Register side effects**
  - Added specific vendor extensions to handle register side effects

- **Iterated register descriptions**
  - Compact notation to describe iterated registers in spec
  - Added specific vendor extensions to describe iterated registers

- **Special registers behaviors**
  - Added specific vendor extensions to describe special registers and their behaviors

- **Custom flow to address specific needs and legacy**
  - Through specific vendor extensions
  - Through command line, GUI options

- **UVM specific needs**
  - Some specific vendor extensions has been added to address specific needs w.r.t. UVM (to be standardized in Accellera Systems Initiative)
Conclusion

- IP-XACT simplified integration, verification
- Automatic flow to avoid manual repetitive jobs
- Maximum reuse, no duplication
- Quickly adaptable to any changes
- Ensure coherency with other design teams
- Standard allows multi-vendor IPs/EDA tools use