Step-up your Register Access Verification

Nisha Kadhirvelu, Sundar Krishnakumar

Cypress Semiconductor Technology India Pvt. Ltd.,

Wesley Park, Mentor, A Siemen Business, Inc.,

accellera SYSTEMS INITIATIVE™

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Agenda

- Introduction
- Register Verification
- Prior Methodology
- Challenges
- Formal Solution
- Results
- Future work
- Conclusion

Introduction

- Working of a DUT

- Defined by Control inputs and status outputs
- Stored in Memory-mapped registers

- **“START POINT OF VERIFICATION”**  **“VERIFY MEMORY-MAPPED REGISTERS”**

Memory-Mapped Registers

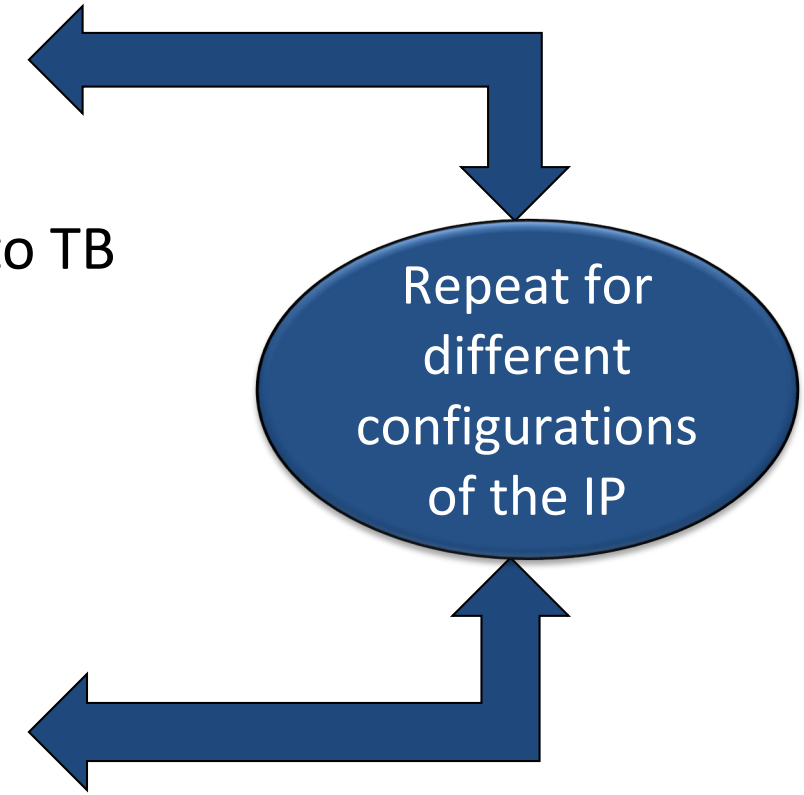
- Memory-mapped register:
  - Resides in a address space
  - Accessible through an AMBA-based or custom interfaces
  - Contains different fields of different bit sizes
  - Has varying access policies (W/RW/RO/W1C etc.)
  - Scalable based on compile-time parameters

Register Verification

- Checks for register accessibility
- Address correctness
- Correct default values on reset
- Basic Write-to/Read-from checks
- Tests for the access policy implementation
- Correctness of all the fields
- Stability of the register
- Front-door and Back-door access

Prior Methodology

- Simulation Approach:
  - Register model generation
  - TB development
  - Integrating register model to TB
  - Creating N test cases
  - Coverage generation
  - Simulation
  - Failure debug
  - Coverage closure



Challenges

- **Necessity of TB & sequences**
- **Building of Register model and its integration**
- Understanding of register access APIs
- Additional test cases for customized register behavior
- Time-Consuming & Susceptible to manual errors
- **More machine resources**
- Missed corner cases & **negative checking**
- Higher Simulation effort
- GUI Debug & Regression closure
- **Less reusability for different configurations of IP**

Challenges

- Iterate for different interfaces
  - AHB, APB, custom, etc.,
- In-spite of automation:

Challenges with our IPs

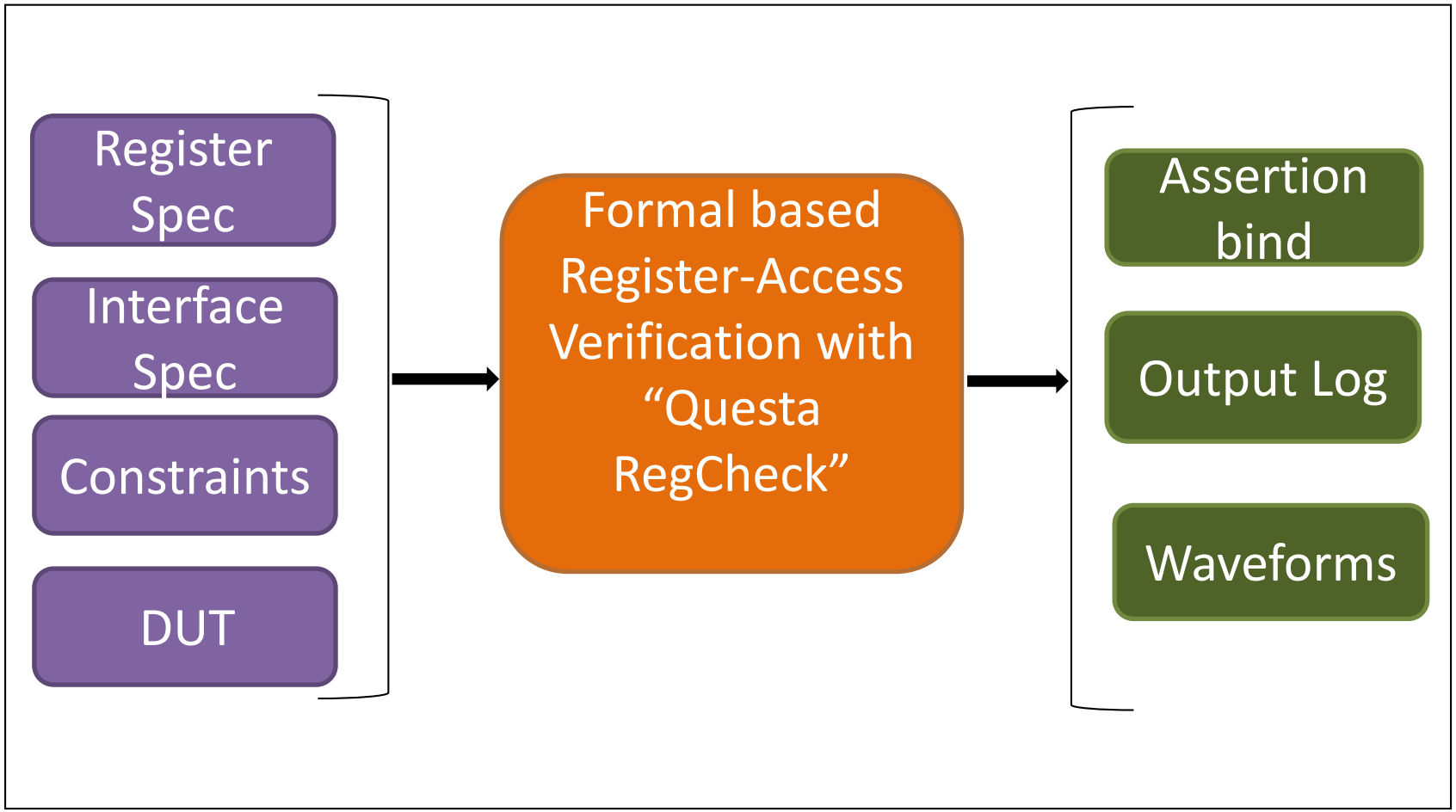
- Need of separate tests for Retention checks
- Interdependency between registers
- HW latency in updating registers

- Dedicated Reset agent
- Order of register access
- Calculation of expected value
- Proper handshaking

Formal Solution

- Questa RegCheck:
  - Formal solution to exhaustively verify registers
  - No requirement of TB/register-model/testcases
  - Supports front-door & backdoor accesses
  - Fully verifies the functionality with “counter” examples
  - Uses Pre-defined assertions specific to an interface
  - Automatic binding of assertions to DUT
  - Comparably lesser user inputs

Flow



Inputs – 1. Register Spec

- Register specification in **CSV** format
- **IPXACT**-based access policy description
- Partial Snippet of CSV format

```
Offset,Position,Title,Identifier,Array,Access,readAction,modifiedWriteValue,volatile,Type,Reset Value,Reset Mask,Description,.memmap_write_internal,.memmap_resetn_global,.memmap_resetn_local_as
ync,.memmap_write,.memmap_write_address,.memmap_write_data,.memmap_write_mask

0x004,,comp0_status,comp0_status,,read-write,read,write,TRUE,register,0x00000000,0x0000ffff,"comparator_structures_comparator_0_status"
,[15:0],comp0_out,comp0_out,,read-only,read,write,TRUE,configuration,0x00000000,0x0000ffff,"Active_comparator__comp0_out___outputs_"

0x700,,intr,intr,,read-write,read,write,TRUE,register,0x00000000,0x0000ffff,"interrupt"
,[15:0],comp0,comp0,,read-write,read,oneToClear,TRUE,configuration,0x00000000,0x0000ffff,"This_interrupt_cause_field_is_activated__HW_sets_the_field_to__1__when_a_comparator_0_event_is_generat
ed",(|mxevtgen_top.act_0.mmio_0.mmio_int_set_comp0_sw_set)
```

Inputs – 1. Register Spec

- Easier than register model & related TB component generation
- Integration to DUT default taken care by the tool
- **EFFORT IN DEFINING CSV FOR DIFFERENT IP CONFIGURATIONS**
  - Increases with Increase in number of registers
- **AUTOMATE** the creation from your register specification format
  - **ONE-TIME EFFORT** (Automated using in-house script)

Inputs – 2. Interface Spec

- Configuration file in .txt format:
  - Register module path in design
  - IP interface (AMBA, custom)
  - Base address of IP
  - Format of register spec (IPXACT/UVM)
  - Map IP interface to assertion module ports
  - Backdoor access (default)/front-door access
- **Very Minimal user effort**

```
-register        act_0.mmio_0.mmio_regs_0.$register_$field
-interface       amba_ahb
-base_addr       0x403f0000
-spec_type       ipxact
-signal_match    nocase,prefix,postfix
-interface_port hselx = mmio_hsel
-interface_port haddr = mmio_haddr[31:0]
-interface_port htrans = mmio_htrans[1:0]
-interface_port hwrite = mmio_hwrite
```

Inputs – 3. Constraints

- Simple .do file to list the constraints of the IP:
  - Reset signal values
  - Clock frequencies
  - IP-specific inputs & assumptions
  - Black-box modules
  - Formal compile & verify commands

```
onerror {exit}
netlist clock clk_ip -period 10
netlist constraint rst_ip_n -value 1'b1 -after_init
netlist property -name haddr_used -assume {mmio_haddr[11:0] < 12'h800 && mmio_haddr[11:0] > 12'h000}
formal compile -d ip_top -cuname BIND_QFL_MMREG_ip
formal verify -timeout 60m -sanity_waveforms -init qft_files/init.seq
exit
```

Inputs – 4. DUT

- List of top-level parameters
- Design-Specific defines
- DUT Compilation List

Formal Run

- Performs formal register check based on inputs
- Generate + Compile properties + formal run
- Major functionalities checked:
  - Global/local reset behaviors
  - Bus read/write operations
  - Any conflict behaviors
  - Volatility
  - No operation (where the register value should not be modified by any other transactions on the bus)

Outputs – 1. Assertion bind

- Generates an Assertion Bind module output
  - Instantiates corresponding assertion module for each register field
  - Concatenates assertion module name, REG base + offset address, REG/FIELD name, access policy for instance name
- Binds top assertion bind module to DUT top based on .txt file and register specification

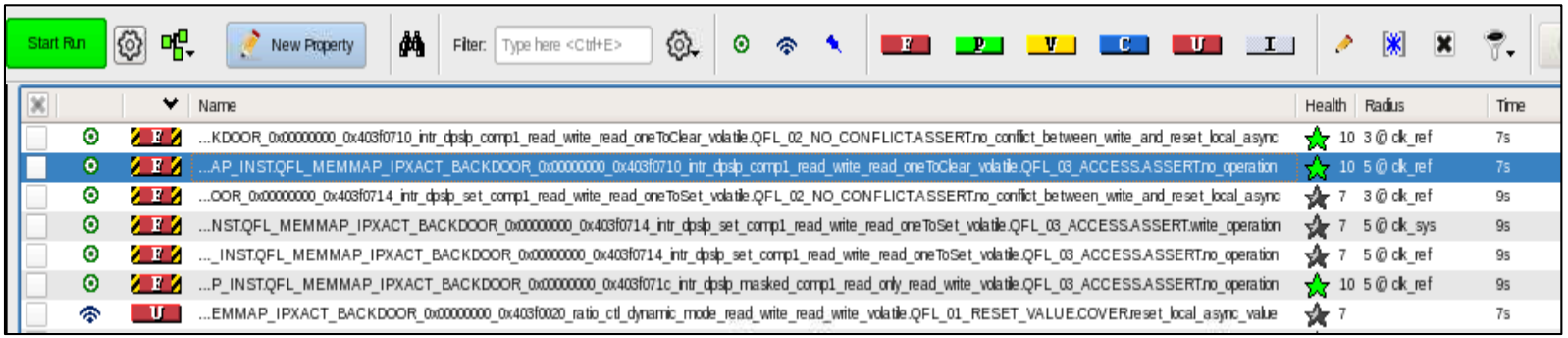
```
qfl_memmap_reg_ipxact_backdoor
#(
  .ADDR_WIDTH          ( ADDR_WIDTH ),
  .ADDR_MASK           ( 0 ),
  .ADDR_VALUE          ( baseAddr0 + 'h0004 ),
  .DATA_WIDTH          ( DATA_WIDTH ),
  .DATA_LEFT           ( 15 ),
  .DATA_RIGHT          ( 0 ),
  .DATA_RESET_GLOBAL_MASK  ( 'h0000ffff ),
  .DATA_RESET_GLOBAL_VALUE ( 'h00000000 ),
  .REG_WIDTH           ( 16 ),
  .REG_OFFSET          ( 0 ),
  .ACCESS_POLICY           ( "read-only" ),
  .ACCESS_READ             ( "read" ),
  .ACCESS_WRITE            ( "write" ),
  .ACCESS_VOLATILE     ( "true" )    )
QFL_MEMMAP_IPXACT_BACKDOOR_0x403f0000_0x0004_comp0_status_comp0_out_read_only_read_write_volatile
(
  .memmap_register     ( ip_top.act_0.mmio_0.mmio_regs_0.comp0_status_comp0_out ) ,
  .*
);
```

Outputs – 2. Output log

- 6 types of status:
  - Assumed
  - Proven
  - Covered
  - Inconclusive
  - Fired
  - Uncoverable

Outputs – 3. Waveforms

- Visualizer GUI for simpler failure debug
- Waveforms of all assertions available
- Logs clear transaction and firing timestamp
- Dumps all related signals



Results

- Deployed this in 3 IP’s
- Used both AMBA & Custom interface specs
- Drastic Improvement in Register Verification
- Faster IP-level verification

Results – IP#1

- APB interface – Declared custom due to its simpler implementation

S. No	Number of registers: 8	
	Property	**Count**
1	Assumed	2
2	Proven	220
3	Covered	127
4	Inconclusive	0
5	Fired	0
6	Uncoverable	33
7	**TOTAL**	**382**
TIME TAKEN: 1 minute 14 seconds!!!		

Results – IP#2

- APB interface – Declared custom due to its simpler implementation

S. No	Number of registers: 5	
	Property	**Count**
1	Assumed	1
2	Proven	112
3	Covered	64
4	Inconclusive	0
5	Fired	0
6	Uncoverable	16
7	**TOTAL**	**193**
TIME TAKEN: 1 minute 8 seconds!!!		

Results – IP#3

- AHB-Lite interface – In-built

S. No	Number of registers: 64	
	Property	**Count**
1	Assumed	39
2	Proven	1596
3	Covered	873
4	Inconclusive	0
5	Fired	0
6	Uncoverable	267
7	**TOTAL**	**2775**
TIME TAKEN: 18 minutes!!!		

Results – Retention Checks

- Retention testing
  - 2 kinds of registers based on retention & non-retention reset
  - Add a column to specify retention reset name in CSV
  - Retention reset in de-asserted state
  - Toggle non-retention reset during formal run
  - Check if data retains for retention registers
- Enables early retention checking before starting Power-Aware sims

Number of registers: 56
TOTAL properties: 2332
TIME TAKEN: 14.7 minutes!!!

Results – Complex registers

- Register volatility – dependent on other registers, variable delays, etc.,
- Cannot be described in IPXACT policy
- CSV then updated with tool provided debug signals
- Complete **PUSH-BUTTON solution**

Results – Issues Caught

- Address decoding
- Overlap of HW & SW updates to register
- Definition of No. of. Cycles delay in register update

Future Work

- Test on AHB5 IP
- Registers in different modules in single formal run
- Automation of creation of the full setup

Conclusion

- Noticeable user effort reduction
- Enables exhaustive verification of registers
- Significantly lesser development time
- Easily portable/reusable setup
- Minimal resource consumption
- Simpler failure analysis
- Automatic Negative checking capability
- **Altogether “POTENTIAL TIME-SAVING”**

Questions