2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Metadata Based Testbench Generation

Daeseo Cha[1], Soonoh Kwon[2], Ahhyung Shin[3]

Samsung Electronics Co., Ltd.

accellera SYSTEMS INITIATIVE

Agenda

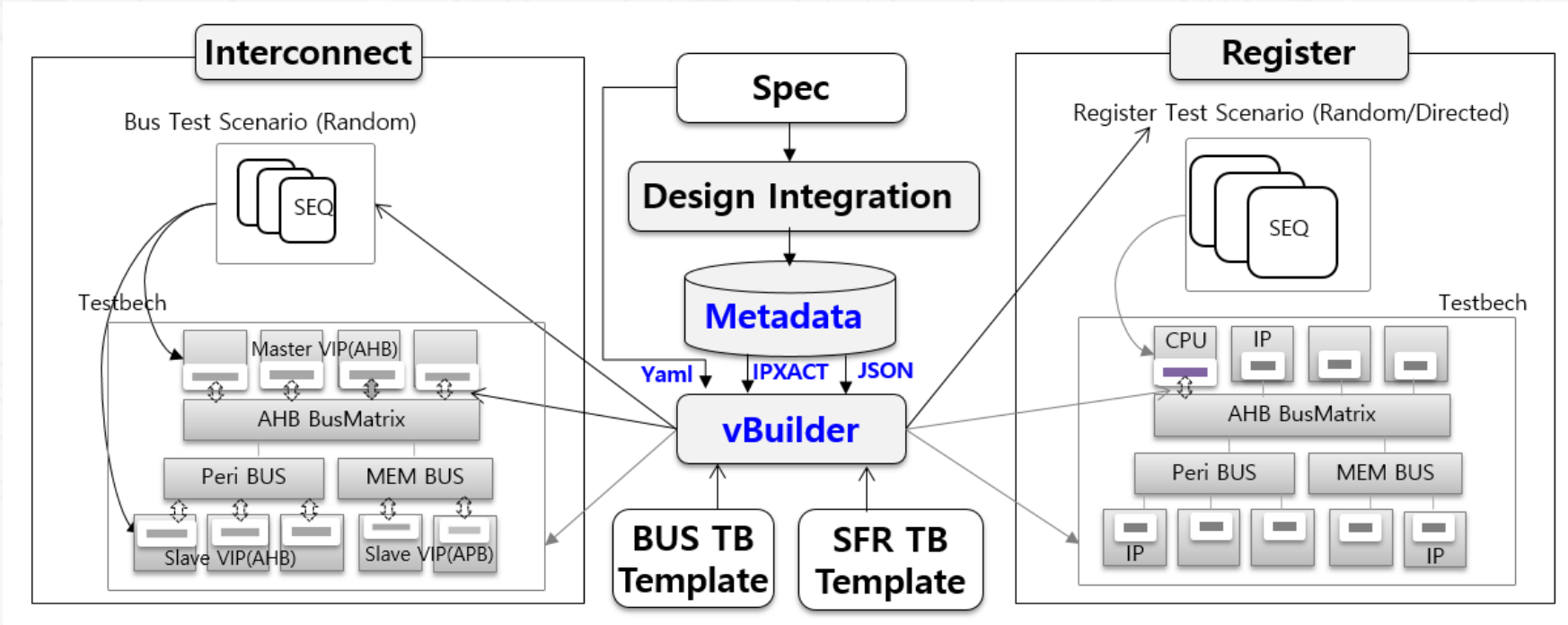
- Problem Statement
- Proposed Approach
- Metadata
- Case Study 1. Full chip Register Testbech Generation
- Case Study 2. Full chip Interconnect Testbech Generation
- Summary

Problem Statement

- Verification Challenge
  - Hard to develop verification environment as design size is growing
  - Hard to maintain the consistency between design and verification environment because design changes frequently during development
  - Basic structural verification such as register and interconnect, needs to completed within short time in early design cycle but, it is not …
- Design
  - RTL integration process is automated using IP-XACT database
  - But, it is not used well in verification

Proposed Approach

- Metadata
  - Traditionally, IP-XACT(IEEE 1685) is used for RTL integration
  - Verification needs more information with Metadata
- Proposed flow



Metadata (1)

- Connectivity Information
  - Jason format: It holds hierarchical instance name, interface name for masters and slaves, interface parameters

```
"/BLK_CPU/AHB2APB_CPU_0": {
            "interfaces": {
                "AHB00_S": {
                    "mode": "slave",
                    "connection":
"/BLK_CPU/AHB_BRIDGE_CPU,AHB02_MS",
                    "portmap": {
                        "HADDR": "HADDR",
                        "HCLK": "HCLK",
                        "HRDATA": "HRDATA",
                        "HREADY": "HREADY",
                        "HREADYOUT": "HREADYOUT",
                        "HRESETn": "HRESETn",
                        "HRESP": "HRESP",
                        "HSELx": "HSEL",
                        "HSIZE": "HSIZE",
                        "HTRANS": "HTRANS",
                        "HWDATA": "HWDATA",
                        "HWRITE": "HWRITE"
                    }
                },
```

```
"PRDATAS0": {
                    "width": "32",
                    "direction": "in",
                    "slices": {
                        "31,16": {
                            "connection": [
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS0,15,0",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS0,31,16",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS1,15,0",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS1,31,16",
                                "/BLK_CPU/TAP,p_rdata,15,0"
                            ]
                        },
                        "15,0": {
                            "connec
                            "connection": [
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS0,15,0",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS0,31,16",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS1,15,0",
                                "/BLK_CPU/AHB2APB_CPU_0,PRDATAS1,31,16",
                                "/BLK_CPU/TAP,p_rdata,15,0"
                            ]
```

Metadata (2)

- Bus topology information
  - Master/slaves, address space, interface protocol

```
{
    "Master_list": [
        "/BLK_CPU/bbio_ahb_32_bit_filter, AHBLite_M",
        "/BLK_CPU/cortex_m4, AHBLite_M_Code",
        "/BLK_CPU/cortex_m4, AHBLite_M_Data",
        "/BLK_CPU/cortex_m4, AHBLite_M_System",
        "/BLK_CPU/udma_top, AHBLite_M"
    ],
    "Slave_list": [
        {
            "Size": 256,
            "Base_addr": "0x40006600",
            "Interface_info":        "/BLK_CPU/SYS_ALIVE/ccd_top,
APB_S_CCD_TOP",
            "vlnv": {
                "vendor": "samsung.com",
                "library": "SENSOR",
                "name": "ccd_top",
                "version": "1.0. META"
            },
            "AMBA_type": "APB"
        },
```

```
"AHB_BUSMATRIX": {
        "Instance_hierarchy": "/BLK_CPU/AHB_BUSMATRIX_MAIN_GNX",
        "Sparse_connection": {
            "s00_ahb_32": [
                "m00_ahb_32",
                "m01_ahb_32",
                "m02_ahb_32",
                "m03_ahb_32",
                "m04_ahb_32",
                "m05_ahb_32",
                "m06_ahb_32"
            ],
            "s01_ahb_32": [
                "m00_ahb_32",
                "m01_ahb_32",
                "m02_ahb_32",
                "m03_ahb_32",
                "m04_ahb_32",
                "m05_ahb_32",
                "m06_ahb_32"
            ],
```

Metadata(3)

- Project specific information
  - Yaml format: Transaction details, restrictions

```
iwb_special_addr_map : "'{
        //---------------USER CUSTOMIZATION+head
        //MST_NAME | SLV_NAME       | ST_ADDR       | END_ADDR    |  GRANULARITY | WR_MSK | PATTERN       | USER DEFINE
        '{,       ,               32'h200C_0000, 32'h200C_0FFF,   32'h1000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h200C_1000, 32'h200F_FFFF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4000_8100, 32'h4000_89FF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4000_8A00, 32'h4000_8BFF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4000_8E00, 32'h4000_9FFF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4000_A300, 32'h4000_FFFF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4001_0000, 32'h4001_0FFF,   32'h10_0000,     0,      NO_ACCESS,     }, // for unmapped test
        '{,       ,               32'h4001_1900, 32'hFFFF_FFFF,   32'h1000_0000,     0,      NO_ACCESS,     }, // for unmapped test

        '{,       MSPI,           0,             0,               32'h1000,     0,      USER_DEFINED,  },
        '{,       BBIO_RSGN,      0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CNT_CPU_GPIO,   0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       AON_ALIVE0,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       AON_ALIVE1,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CNT_ISPEND,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CMU_ISPEND,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CNT_SENSOR,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CMU_SENSOR,     0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CNT_FE,         0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CNT_ISP,        0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CMU_ISP1,       0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       CLKGEN,         0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       MEM_SENSOR,     0,             0,               32'h1000,     0,      READ_ONLY,     },

        '{,       ROM,            0,             0,               32'h1000,     0,      READ_ONLY,     },
        '{,       ,               32'h2010_0400, 32'h2010_09FF,   32'h0400,     1,      READ_WRITE,    },
        '{,       ,               32'h2010_B000, 32'h2010_B7FF,   32'h0400,     2,      READ_WRITE,    },
```

Testbench Generation using Metadata

- vBuilder (Verification Builder)
  - Renders testbench for IP, full-chip and register/interconnect using Metadata and testbench template code

```
class busitb_base_test_vseq_c extends itb_sys_env_tsg_base_vseq_test;

    typedef struct {
        string master_name;
        string slave_name;
        bit [31:0] st_addr;
        bit [31:0] end_addr;
        bit [31:0] gran;
        int wdata_type;
        rw_pattern_e rw_pattern;
        string user_define;
    } access_pattern_e;

    access_pattern_e special_target_list [] = {{itb_special_addr_map}}
```

Jinja template

Rendering

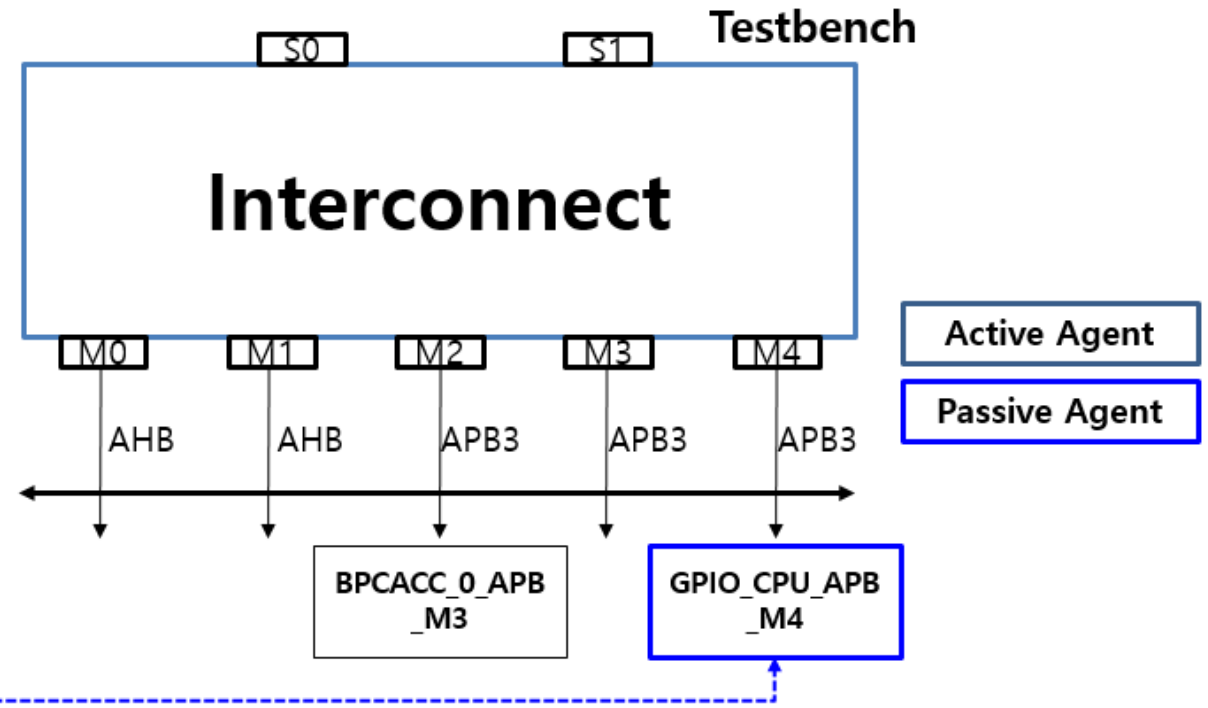
```
class busitb_base_test_vseq_c extends itb_sys_env_tsg_base_vseq_test;

    typedef struct {
        string master_name;
        string slave_name;
        bit [31:0] st_addr;
        bit [31:0] end_addr;
        bit [31:0] gran;
        int wdata_type;
        rw_pattern_e rw_pattern;
        string user_define;
    } access_pattern_e;

    access_pattern_e special_target_list [] = '{
        //----------------USER CUSTOMIZATION+head
        //MST_NAME | SLV_NAME       | ST_ADDR        | END_ADDR       |  GRANULARITY | WR_MSK | PATTERN       | USER DEFINE
        '{"",       "",             32'h200C_0000,  32'h200C_0FFF,    32'h1000,      0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h200C_1000,  32'h200F_FFFF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4000_8100,  32'h4000_89FF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4000_8A00,  32'h4000_8BFF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4000_8E00,  32'h4000_9FFF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4000_A300,  32'h4000_FFFF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4001_0000,  32'h4001_0FFF,    32'h10_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
        '{"",       "",             32'h4001_1900,  32'hFFFF_FFFF,    32'h1000_0000,   0,       NO_ACCESS,      "" }, // for unmapped test
```

Testbench Generation using Metadata

- Configuration change for target testbench
  - By changing attributes for metadata, target testbench will be easily changed.

SYS.CSV

	Attach Active Agent	Attach Passive Agent
BPCACC_0_APB_M2	Y	Y
GPIO_CPU_APB_M4	N	Y



Automated testbench generation for Sign-off

- Sign-off for Register Access

Metadata (IP-XACT)	Testbench (UVM_REG)
<spirit:field> <spirit:name>PA_TxHsG1SyncLength_MSB</spirit:name> <spirit:**writeValueConstraint**> <spirit:minimum>0</spirit:minimum> <spirit:maximum>1</spirit:maximum> </spirit:writeValueConstraint> </spirit:field>	class hsi_unipro16_pa_std_region_pa_txhsg2synclength_pa_txhsg2synclen_msb_c extends uvm_reg_field; **constraint valid { value inside { ['h0:'h1] }; }** endclass : hsi_unipro16_pa_std_region_pa_txhsg2synclength_pa_txhsg2synclen_msb_c

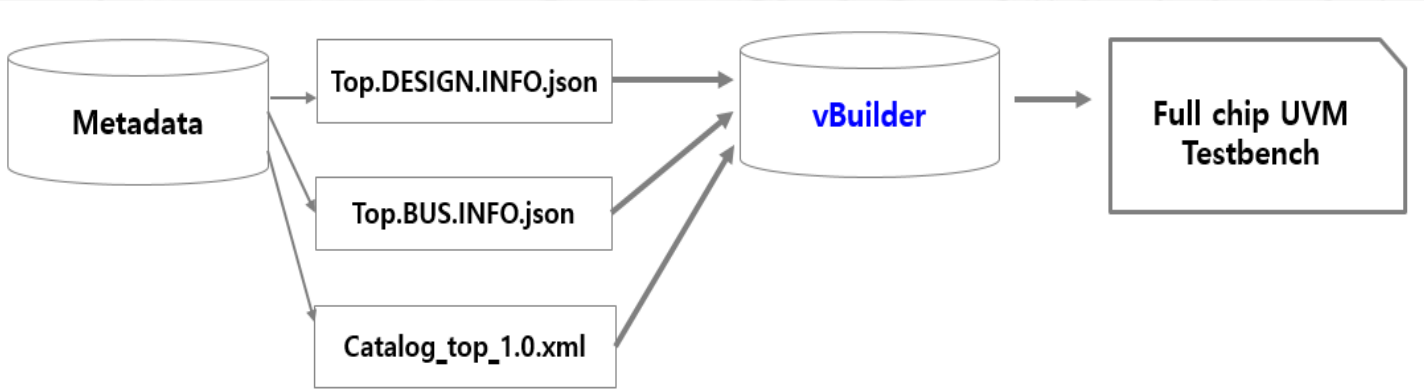
<spirit:register>
<spirit:name>COMP_OPTION_SUITE</spirit:name>
<spirit:addressOffset>0x20</spirit:addressOffset>
<spirit:size spirit:resolve="immediate">32</spirit:size>
<spirit:field>
<spirit:name>Reserved_FF0</spirit:name>
<spirit:bitOffset>2</spirit:bitOffset>
<spirit:bitWidth
spirit:resolve="immediate">30</spirit:bitWidth>
</spirit:field>
<spirit:field>
<spirit:name>rx_symbol_clk1_reset_type</spirit:name>
<spirit:bitOffset>1</spirit:bitOffset>
<spirit:bitWidth
spirit:resolve="immediate">1</spirit:bitWidth>
</spirit:field>
<spirit:field>
<spirit:name>rx_symbol_clk0_reset_type</spirit:name>
<spirit:bitOffset>0</spirit:bitOffset>
<spirit:bitWidth
spirit:resolve="immediate">1</spirit:bitWidth>
</spirit:field>
</spirit:register>
	class hsi_unipro16_component_region_comp_option_suite_c extends uvm_reg;
	rand uvm_reg_field reserved_ff0;
	rand uvm_reg_field rx_symbol_clk1_reset_type;
	rand uvm_reg_field rx_symbol_clk0_reset_type;
	covergroup cg_vals;
	option.per_instance = 1;
	reserved_ff0: coverpoint reserved_ff0.value[29:0]
	{
	`AUTO_COV_MIN_MAX(30)
	}
	rx_symbol_clk1_reset_type: coverpoint
	rx_symbol_clk1_reset_type.value[0:0];
	rx_symbol_clk0_reset_type: coverpoint
	rx_symbol_clk0_reset_type.value[0:0];
	endgroup
	define AUTO_COV_MIN_MAX(VAL) ₩
	bins min = {0}; ₩
	bins mid = {[1:({VAL{1'b1}})-1]}; ₩
	bins max = {{VAL{1'b1}}};

Use testable, constraint and coverage

<spirit:register> <spirit:name>PA_AvailTxDataLanes</spirit:name> <spirit:field> <spirit:name>PA_AvailTxDataLanes</spirit:name> **<spirit:testable>false</spirit:testable>** </spirit:field>	class hsi_unipro16_pa_std_region_pa_availtxdatalanes_c extends uvm_reg; **uvm_resource_db#(bit)::set({"REG::",get_full_name()},** **"NO_REG_HW_RESET_TESTS", 1, this);** uvm_resource_db#(bit)::set({"REG::",get_full_name()}, "NO_REG_ACCESS_TESTS", 1, this);
<spirit:register> <spirit:name>COMP_RESET</spirit:name> <spirit:field> <spirit:name>comp_reset</spirit:name> <spirit:testable spirit:**testConstraint="readOnly"**>true</spirit:testable> </spirit:field>	class hsi_unipro16_component_region_comp_reset_c extends uvm_reg; **uvm_resource_db#(bit)::set({"REG::",get_full_name()},** **"NO_REG_ACCESS_TESTS", 1, this);**

Case Study 1: Full-chip Register Testbench

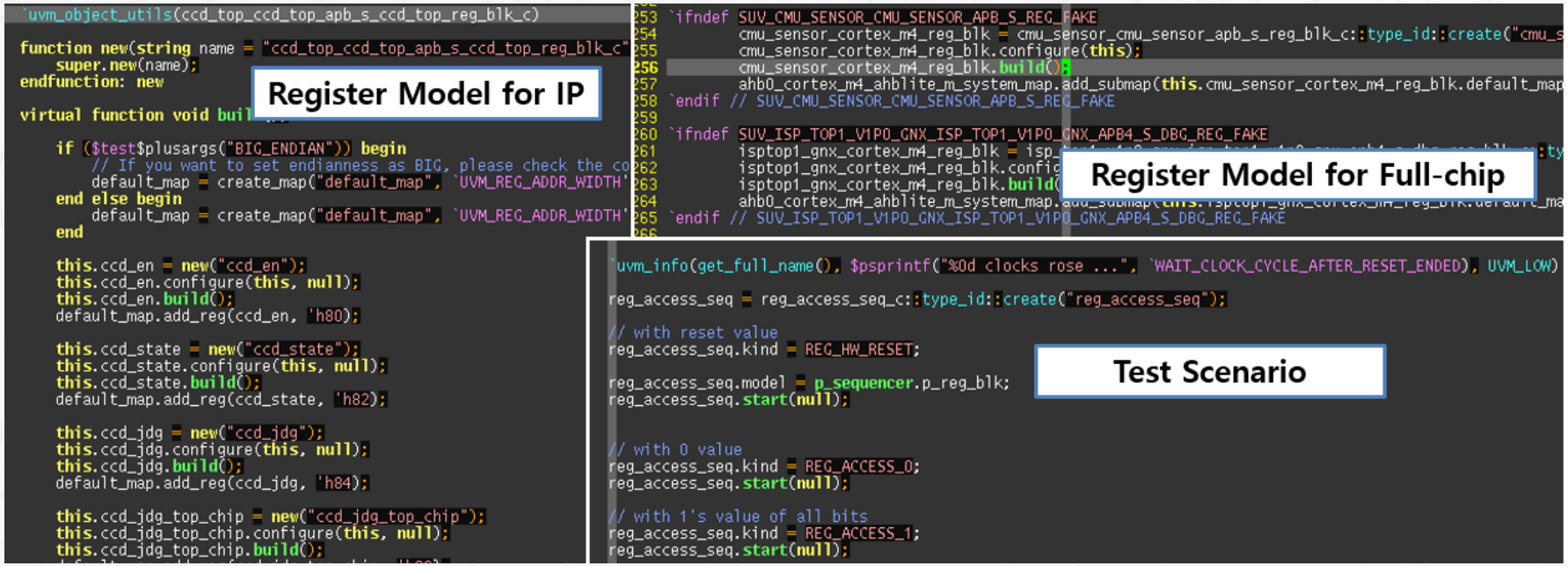
- Inputs
  - Top.DESIGN.INFO.json: AMBA bus interface details for all masters and slaves
  - Top.BUS.INFO.json: Back-bone bus details
  - Catalog_top_1.0.xml: IP/TOP (Registers, Fileset, Clock, Reset)
- Flow




```
% vbuilder -ipjson top.DESIGN.INFO.json -sbjson Top.BUS.INFO.json ₩
          -xmlcat Catalog_top_1.0.xml -filelist vcode.f ₩
          -corename cortex_m4_AHBLite_M_System
```

Case Study 1: Full-chip Register Testbench

- Code



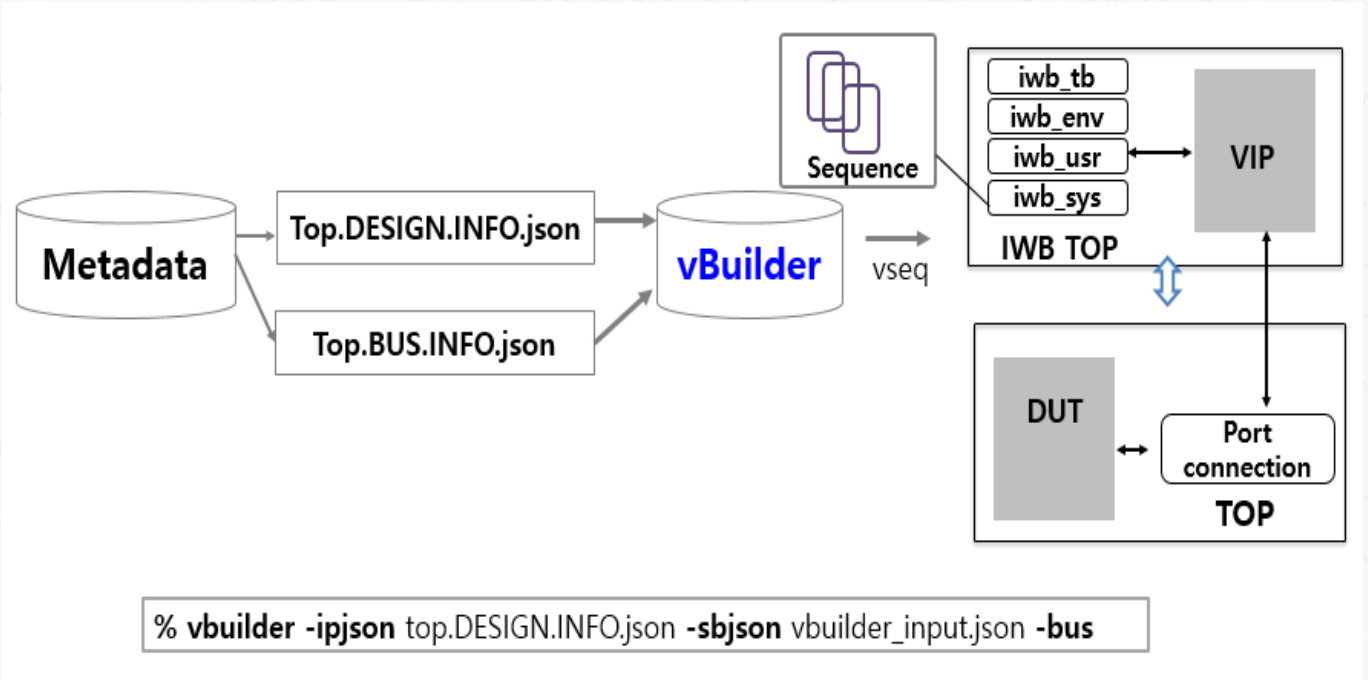
- Result

Step	Time	Iteration	Total Time
1. Testebench Generation Time	00:02:49	1	**00:38:26**
2. RTL compile	00:00:30	1	
3. Compile (multi-snapshot)	00:00:30	49 (IP)	
4. Run Tests	00:00:13	49 (IP)	

Table 2. Full-chip Register Testbench Runtime

Case Study 2: Full-chip Interconnect Testbench

- Inputs
  - Image Sensor (50Mp)
  - Bus Master: 5 with AHB interfaces
  - Bus Slave: 12 AHB slaves, 105 APB slaves
- Flow



Case Study 2: Full-chip Interconnect Testbench

- Code snippet

```
//master0
force top.BLK_CPU.bbio_ahb_32_bit_filter.HADDRM = HIN_00_haddr;
force top.BLK_CPU.bbio_ahb_32_bit_filter.HBURSTM = HIN_00_hburst;
force top.BLK_CPU.bbio_ahb_32_bit_filter.HLOCKM = HIN_00_hmastlock
force top.BLK_CPU.bbio_ahb_32_bit_filter.HPROTM = HIN_00_hprot;
                                          filter.HSIZEM = HIN_00_hsize;
                                          filter.HTRANSM = HIN_00_htrans;
```

Connect VIP to DUT

```
cdn_ahb_active_master_sv #(
    .interface_soma(interface_soma),
    .ADDR_WIDTH(ADDR_WIDTH),
    .DATA_WIDTH(DATA_WIDTH),
    .RESP_WIDTH(RESP_WIDTH)
) i_shell_cis_tb_top_top_BLK_CPU_bbio_ahb_32_bit_filter_AHBLite_M(
    .AHB_HCLK(aclk0__clk),
    .AHB_HRESET(nReset0__resetN),
    .AHB_HADDR_out(AHB_HADDR_out__HIN_00_haddr),
    .AHB_HBURST_out(AHB_HBURST_out__HIN_00_hburst),
    .AHB_HLOCK(AHB_HLOCK__HIN_00_hmastlock),
    .AHB_HPROT_out(AHB_HPROT_out__HIN_00_hprot),
    .AHB_HSIZE_out(AHB_HSIZE_out__HIN_00_hsize),
    .AHB_HTRANS_out(AHB_HTRANS_out__HIN_00_htrans),
    .AHB_HWDATA_out(AHB_HWDATA_out__HIN_00_hwdata),
    .AHB_HWRITE_out(AHB_HWRITE_out__HIN_00_hwrite),
    .AHB_HRDATA(HIN_00_hrdata__AHB_HRDATA),
    .AHB_HREADY(out1__AHB_HREADY),
    .AHB_HRESP(HIN_00_hresp__AHB_HRESP),
    .AHB_HGRANT(tie_i_shell_cis_tb_top_top_BLK_CPU_bbio_ahb_32_bit_fi
    .AHB_HMASTLOCK(tie_i_shell_cis_tb_top_top_BLK_CPU_bbio_ahb_32_bit
    .AHB_HSEL(sink_i_shell_cis_tb_top_top_BLK_CPU_bbio_ahb_32_bit_fil
                                          b_top_top_BLK_CPU_bbio_ahb_32_bit
                                          b_top_top_BLK_CPU_bbio_ahb_32_bit
```

Instantiate VIP

```
class busiwb_all_to_all_test_vseq_c extends busiwb_base_test_vseq_c;

   virtual task do_stimulus ();

      foreach(tgt_slave_list[s_idx]) begin
      int ss = s_idx;
      access_pattern_e tmp_access_route_list[$];
      access_pattern_e tmp_route_consider_except[];

      // Make Route list
      foreach(src_master_list[ss][m_idx]) begin
          int mm = m_idx;
          access_pattern_e access_region;
          access_region.master_name = src_master_list[ss][mm];
          access_region.st_addr = 0;
          access_region.end_addr = 0;
          access_region.gran = 32'h1000;
          access_region.wdata_type = 0;
          access_region.rw_pattern = READ_WRITE;
          access_region.user_define = "";
          access_region.slave_name = tgt_slave_list[ss];
          tmp_access_route_list.push_back(access_region);
      end

      make_pattern_list_with_exception( .ref_list(tmp_access_route_list),
                                        .exception_list(special_target_list),
              .result_list(tmp_route_consider_except) );

      foreach(tmp_route_consider_except[t_idx]) begin
          int ii = t_idx;
```

all masters to all slaves test

```
ame = tgt_slave_list[ss];
```

- Result

Step	Time	Iteration	Total Time
1. Testebench Generation Time	00:04:39	1	**02:07:57**
2. Build simulation snapshot	00:03:18	1	
4. Run Sanity Test	00:10:00	12	

Summary

- Result
  - Image Sensor (50MP)
    - Interconnect Testbench Generation Time: 1 weeks → 2 Hour
    - Reduced human errors completely
  - Mobile AP
    - Interconnect Testbench Generation Time: 2~3 weeks → 1 day

- Conclusion
  - Automated register and interconnect testbench generation using metadata
  - Easy to generate testbench variance using metadata for power/performance

Questions