Using an Enhanced Verification Methodology for Back-to-Back RTL/TLM Simulation

Frank Poppen and Ralph Görgen, OFFIS Institute for Information Technology
Kai Schulz, Andreas Maurerer and Jan-Hendrik Oetjens, Robert Bosch GmbH
Joachim Gerlach, Hochschule Albstadt-Sigmaringen
The Challenge

- electronics in heterogeneous systems
- ambient and safety relevant
- increasing complexity
- design and verification
- lining up for the task
  - tailored solutions
  - standards
  - languages
  - tools
No „One Size Fits All“

- verification engineers choose and combine what ...
  - fits best for the company
  - the design-team
  - the application domain
  - the abstraction level
  - (budget, roadmap, ...)
- deep roots in the design process
- changes endanger productivity
- change carefully and incrementally
Outline

- Motivation
- What is the Integrated Functional Verification Script Environment (IFS) and why use it?
- What was missing and what did we add to IFS?
- Making use of it for Back-to-Back comparison between RTL and TLM
- Conclusions
IFS long before SystemC/-Verilog

- enhanced from VHDL with ...
  - VHDL-AMS
  - SystemC
  - Matlab/Simulink
  - SystemVerilog and UVM
- SystemC based library simulates with any simulator (IEEE 1666)
- tailored to relevant use scenarios in special contexts
- simple IFS command language for (self-checking) test cases
  - digital designer
  - analog designer
  - verification engineer
  - system engineer
  - software engineer
IFS Simulation Environment

• **design under verification**
• **testbench modules**
  – cmd loop
  – predef. cmd
  – user def. cmd
• IFS-controller
• IFS-script

```
TBM_1 PRINT „Executing Test“  -- predef. module cmd
#LOOP 100                    -- predef. controller cmd
  IFS SYNC ALL               -- predef. controller cmd
TBM_1 write $(100+#i)        -- user def. module cmd
TBM_2 read $(100+#i)         -- user def. module cmd
#EOL                         -- predef. controller cmd
IFS QUIT                      -- predef. controller cmd
```
Outline

• Motivation
• What is the Integrated Functional Verification Script Environment (IFS) and why use it?
• What was missing and what did we add to IFS?
• Making use of it for Back-to-Back comparison between RTL and TLM
• Conclusions
Communication Abstraction

IFS-Script
IFS-Controller

RTL  TLM

TBM
TBM
TBM

DMA  I²C
AMBA  CAN

DUV

separating communication and behavior

TBM
Mixing Communication Abstraction
Interface Definition

struct master_rt_if : public master_if_base
{
    // Ports
    sc_in< bool > clk; // clock
    sc_in< bool > rst; // reset
    sc_out< bool > req; // master request
    sc_in< bool > gnt; // grant from arbiter
    sc_out< bool > rreq; // read request
    sc_out< bool > wreq; // write request
    sc_out< sc_dt::sc_bv<16> > addr; // address
    sc_out< sc_dt::sc_bv<32> > wdata; // write data
    sc_in< sc_dt::sc_bv<32> > rdata; // read data
    sc_in< bool > ack; // acknowledge

    void if_write( unsigned int, unsigned int );
    unsigned int if_read( unsigned int );

    master_rt_if();
};

struct master_tl_if : public master_if_base
{
    // Ports
    sc_in< bool > clk; // clock
    // TLM-2 socket, defaults to 32-bits wide, base protocol
    tlm_utils::simple_initiator_socket<master_tl_if> socket;

    void if_write( unsigned int, unsigned int );
    unsigned int if_read( unsigned int );

    master_tl_if();
};
Communication Behavior

**Master TL If: if_read( unsigned int address )**

```c
unsigned int
master_tl_if::if_read( unsigned int address )
{
    // transaction pointer
    tlm::tlm_generic_payload* trans =
        new tlm::tlm_generic_payload;
    sc_time delay = sc_time(30, SC_NS);
    unsigned int data = 0;
    // Initialize 8 out of the 10 attributes
    trans—>set_command( tlm::TLM_READ_COMMAND );
    trans->set_address(address);
    trans->set_data_ptr( reinterpret_cast <unsigned char*>( &data ) );
    // Blocking transport call
    socket->b_transport( *trans, delay );
    // obliged to check response status
    if ( trans—>is_response_error() )
        ifs_error("TLM—2.0: Response error");
    // ...
    return data
}
```

**Master RT If: if_read( unsigned int address )**

```c
unsigned int
master_rt_if::if_read( unsigned int address )
{
    // set request
    req.write(true);
    // wait for grant
    wait(gnt.posedge_event());
    wait(clk.posedge_event());
    // set read request and address
    rreq.write(true);
    addr.write(address);
    // wait for acknowledge
    wait(ack.posedge_event());
    wait(clk.posedge_event());
    // deassert request and read request
    req.write(false);
    rreq.write(false);
    return rdata.read().to_int();
}
```
Template:

```cpp
template < typename T >
IFS_MODULE(master) , public TLM_READ_COMMAND
{
    typedef T if_type;

    /* Commands */
    void Write(std::list<std::string> parameters);
    void Read(std::list<std::string> parameters);
    // IF Module method impl
    void ack_write_msg(int, unsigned int);
    void ifs_error(const char *);

    //Constructor master(sc_module_name);
}
```

Instantiating TBM:

```cpp
master<master_tl_if> *m1;
    m1 = new master<master_tl_if>("MS1");

master<master_rt_if> *m2;
    m2 = new master<master_rt_if>("MS2");
```
Outline

• Motivation
• What is the Integrated Functional Verification Script Environment (IFS) and why use it?
• What was missing and what did we add to IFS?
• Making use of it for Back-to-Back comparison between RTL and TLM
• Conclusions
Back-to-Back Simulation

- methodology for V-model verification
- DSP from RTL to ISS + TLM
- abstraction implies changed timing behavior.
Abstraction and Time

- **simulated time (SIT)**
- **model execution time (MET)**
- SIT and MIT differ across abstractions

A. Specification
B. Component-assembly
C. Bus-arbitration
D. Bus-functional
E. Cycle-accurate computation
F. Implementation

System Modeling Graph by L. Cai and D. Gajski.
Synchronization of BtB

- high abstraction is expected to execute faster
- sequence of results not guaranteed to be identical
- BtB requires synchronization
  - generally not a trivial task to accomplish
  - this scenario allows sync to external IRQ
BtB Verification Setup for DSP at RTL VHDL and IIS SystemC

- IRQ, CLK and stimuli connected parallel
- Script(s) define test(s)
- IRQ used for sync of both DUV + script
- same TBM master in two flavor
- automatic compare
- manual analysis in one waveform viewer

© Accellera Systems Initiative
Motivation

What is the Integrated Functional Verification Script Environment (IFS) and why use it?

What was missing and what did we add to IFS?

Making use of it for Back-to-Back comparison between RTL and TLM

Conclusions
Conclusion

- seamless flow combining several languages and abstraction levels
- comfortable adaption of test environment to DUT variants and abstraction level
- comfortably analyzing deviations between models
  - automatically generated assertions through BtB
  - human readable tests specified in IFS scripts
  - easier to understand than generated test vectors
- BtB across abstractions requires synchronization
Acknowledgements: This work has been funded by the German Federal Ministry for Education and Research (Bundesministerium für Bildung und Forschung, BMBF) under the grant 01IS13022 (project Effektiv). The content of this publication lies within the responsibility of the authors.