TrueConnect 2017
Enriching Collaboration for Innovative Excellence
BENGALURU

Dynamic configurability in static blocks

- Giridhar Chelluri
- Sukhjeet Singh

7th Dec 2017

Agenda

1. Why Configurability
2. Static block - overview
3. Existing ways of configuration
4. Difficulties in configuration
5. Our Idea
6. Advantages over existing methods
7. Block diagram
8. Sample code
9. Q & A

Why Configurability ?

- For various functional realization
- Effectively reduce the verification cycles
- Attain maximum functional coverage
- Configuration knobs for specific functional requirements
- User specific state flow implementations

Static Block

HDL Synthesizable Block

Compile time flow-

- Memory allocation
- Binds instances to modules
- Resolves library references
- Propagates Defparams
- Unrolls generate statements
- Checks all hierarchical names and function/task calls

Existing Ways of Configuration

Error injection techniques a necessity to validate dependability of a system. Most of the developed techniques fall into below mentioned categories:

- Compiler directives
- Parameterize blocks
- Configurable Bit

Difficulties in Configuration

- Risk of damage for the configured system
- Drastic change in Memory usage
- Increases complexity of block
- Limited scope of configuration
- Large number of configuration knobs
- Limited reusability
- Adaptability of test environment for testing

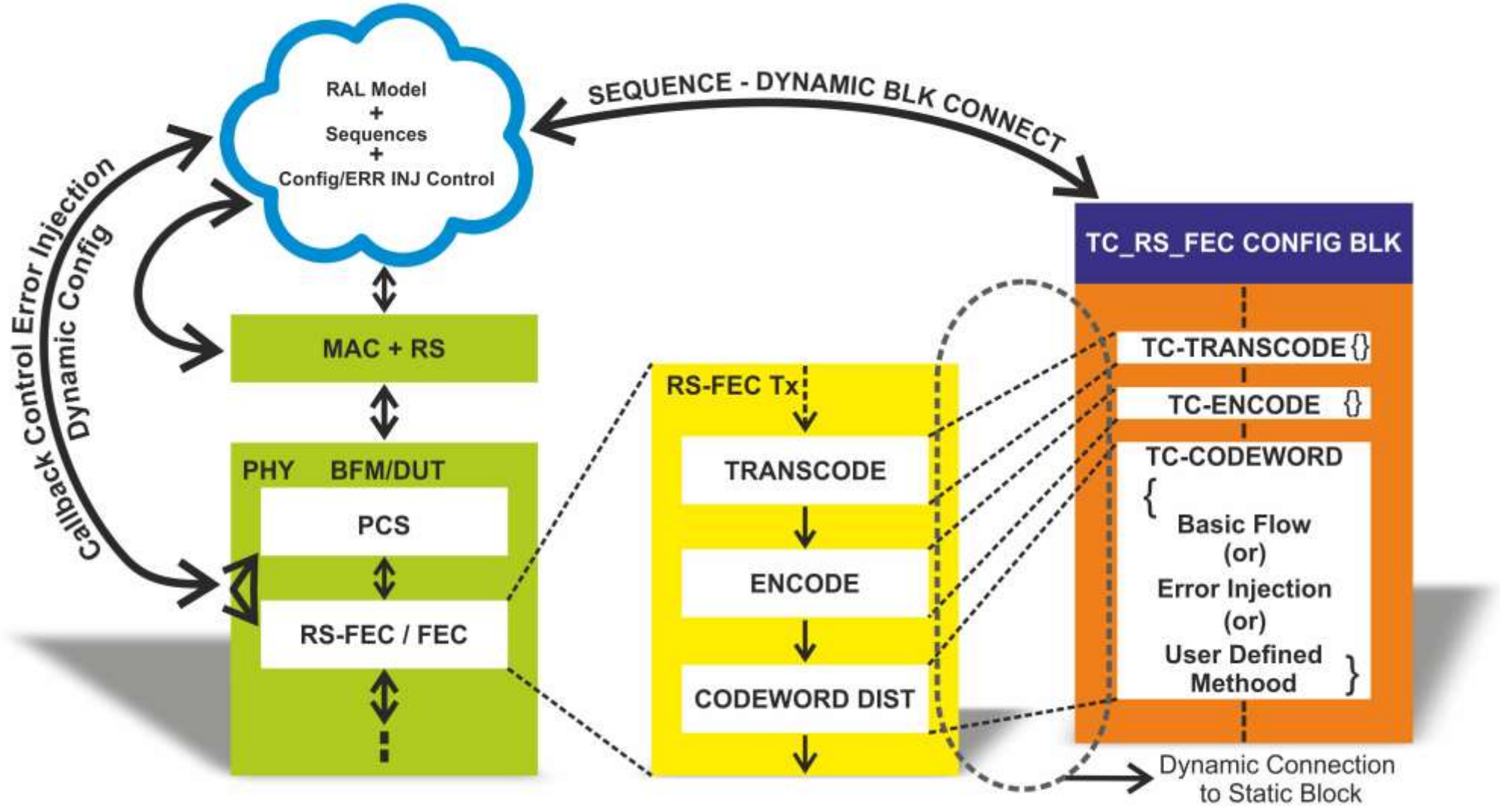
Our Idea

- BFM independent from configuration/error block
- Dynamic reconfigurability from sequences
- Adaptable to future implementations
- Intelligent monitors with self masking
- Easy to use with limited protocol knowledge
- Easy data flow monitoring

Advantages over Existing Methods

- Easy to change VIP BFM's behaviour
- Add on for future implementations
- Abstract process flow monitoring
- Reduce the overhead of unnecessary data structure
- Reduces sequence complexity
- Reusability of HDL block for emulation and for VIP BFM's

Block Diagram



Sample Code- Part 1


```
class tc_ethernet_100G_RS_FEC_Err_Inj_base_Pkt extends uvm_object;

event Err_Injected;
event Non_Crpt_AM;
event Codeword_Being_Sent;
event Valid_RAM;

bit Enable_Symbol_Corruption;
bit [31:0]Num_Corrupt_Symbols;//Also Used for Nibble Corruption
bit Corrupt_Pad_Field;
bit [31:0]SYMBOL_CNT;
bit [31:0]FRAME_CNT;
bit [1:0]FRAME_1285_CNT;
int Err_Position_Q[$];
int Nibble_Error_Injection[$];
int LANE_NUM;
bit Dont_Flip_Pad_Field;
```

Sample Code- Part 2


```
virtual function [5:0] Replace_AM(input in0,input in1,input in2,input in3,input in16,input in17,input in18,input in19,input [6:0]index_count);
 begin

  if((index_count>=26 && index_count<=33)||(index_count>=58 && index_count<=65))
   begin
    Replace_AM = {in1,in2,in3,in17,in18,in19};
   end
  else
   begin
    Replace_AM = {in0,in0,in0,in16,in16,in16};
   end

 end
endfunction

virtual function [9:0] TC_CODEWORD(input [9:0] Symbol_In, input [31:0] Symbol_In_Count, input [31:0]Frame_Count);
 begin
  Corrupt_Symbol = Symbol_In;
  SYMBOL_CNT     = Symbol_In_Count;
  FRAME_CNT      = Frame_Count;
 end
endfunction

virtual function [255:0] Corrupt_AM_Payload(input [63:0]am3,input [63:0]am2,input [63:0]am1,input [63:0]am0,input [4:0]Lane);
 begin
  Corrupt_AM_Payload = {am3,am2,am1,am0};
 end
endfunction

virtual function [256:0] TC_TRANSCODE(input [65:0]blk3,input [65:0]blk2,input [65:0]blk1,input [65:0]blk0,input [4:0]Lane,input [1:0] Blk_Count_1285);
 begin

  bit [255:0]tx_payload;
```

Sample Code- Part 3


```
module tc_ethernet_RS_FEC(
                          .
                          .
                         );

  tc_ethernet_100G_RS_FEC_Err_Inj_base_Pkt rs_fec_clbk; //Creation of dynamic object handle
                                                .
                                                .
                                                .
                                               -//Callback method for controlling replacement of AM received from PCS
            tx_coded_buff <= rs_fec_clbk.Replace_AM(unitdata_in_0,unitdata_in_1,unitdata_in_2,unitdata_in_3,unitdata_in_16,unitdata_in_17,unitdata_in_18,
                                                    unitdata_in_19,tx_coded_in_bit_index);
                                                .
                                                .
                                               -//Callback method for the corruption of the tanscoded block
            tx_xcoded[lane_no/4][256:0] <= rs_fec_clbk.TC_TRANSCODE(tx_coded[lane_no+3],tx_coded[lane_no+2],
                                                                    tx_coded[lane_no+1],tx_coded[lane_no],(lane_no/4),Block_Count_1285);
                                                .
                                                .
                                                .
                                               -//Callback method for controlling AM nibble corruption during AM Mapping
            tx_xcoded[lane_no/4][255:0] <= rs_fec_clbk.Corrupt_AM_Payload(tx_coded[lane_no+3][63:0],tx_coded[lane_no+2][63:0],tx_coded[lane_no+1][63:0],
                                                                          tx_coded[lane_no][63:0],lane_no);
                                                .
                                                .
                                                .
                                               -//Callback method for controlling Corruption at symbol level
            symbol_out <= rs_fec_clbk.TC_CODEWORD(data_mem[R_ptr],symbol_count_tx,FEC_frame_count);
                                                .
                                                .
                                                .
endmodule:tc_ethernet_RS_FEC
```

Sample Code- Part 4


```
virtual task body();
 begin

   TC_RS_FEC_100G_Err_Inj_Pkt = new(); //Creating dynamic Error Injection Object

   Create_Err_Inj_Interconnect(0,100,3,TC_RS_FEC_100G_Err_Inj_Pkt);//Creating Static-Dynamic Blk Interconnect

  `uvm_do(RS_FEC_rx_am_lock)

  `uvm_do(PCS_Block_Lock)

  `uvm_do(PCS_am_Lock)

  `uvm_do(PCS_align)

   repeat(2)
   begin
   TC_RS_FEC_100G_Err_Inj_Pkt.Enable_Symbol_Corruption = 1'b1;//Selection for Symbol Corruption
                                                              //(NOTE:This bit is self clearing)

   TC_RS_FEC_100G_Err_Inj_Pkt.Num_Corrupt_Symbols = $urandom_range(min,max);//Selecting Number of Symbols (Random Selection)
                                                                            //to be corrupted

   end
```

Sample Code- Part 5


```
corrupt_count=$urandom_range(3000,3500);

repeat(`TC_RANDOM_RUN_COUNT)
 begin

  `uvm_do_with(MAC_pkt_wt,{MAC_pkt_wt.len_typ == 1'b0;
                           MAC_pkt_wt.DES_ADD==48'b0;
                           MAC_pkt_wt.SRC_ADD==48'b0;})//Write
  end

@(TC_PCS_100_40G_Error_inj_pkt.PCS_TX_State==3'b011)//Terminate state

@(TC_PCS_100_40G_Error_inj_pkt.PCS_TX_State==3'b001)//control state

TC_PCS_100_40G_Error_inj_pkt.sync_header_corruption=1;

TC_PCS_100_40G_Error_inj_pkt.invalid_sync_header=1;

TC_PCS_100_40G_Error_inj_pkt.corruption_count=corrupt_count;

-> TC_PCS_100_40G_Error_inj_pkt.pcs_call_back_event;

uvm_report_info(get_name(),$psprintf("Sending %d Invalid Sync headers",TC_PCS_100_40G_Error_inj_pkt.corruption_count));

@(TC_PCS_100_40G_Error_inj_pkt.invalid_sync_header==0);
```

Q & A