DVCon 2013
Design & Verification Conference & Exhibition

February 25-28, 2013
DoubleTree, San Jose

accellera SYSTEMS INITIATIVE

SYNOPSYS Accelerating Innovation

QUALCOMM

C through UVM: Effectively using C based models with UVM based Verification IP

by
Adiel Khan
Staff CAE
Synopsys

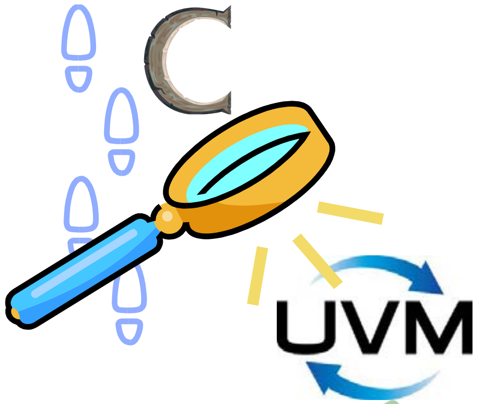
Chris Spear,
Kevork Dikramanjian,
Abhisek Verma
Synopsys

Senay Haile
Qualcomm

Sponsored By:

Agenda:

- Introduction
- Project Milestones
- The Testbench
- The Test Flow
- Stimulus Generation
- Reconfiguration
- Audio/Video Score-boarding
- Take Away!



Sponsored By:

Introduction

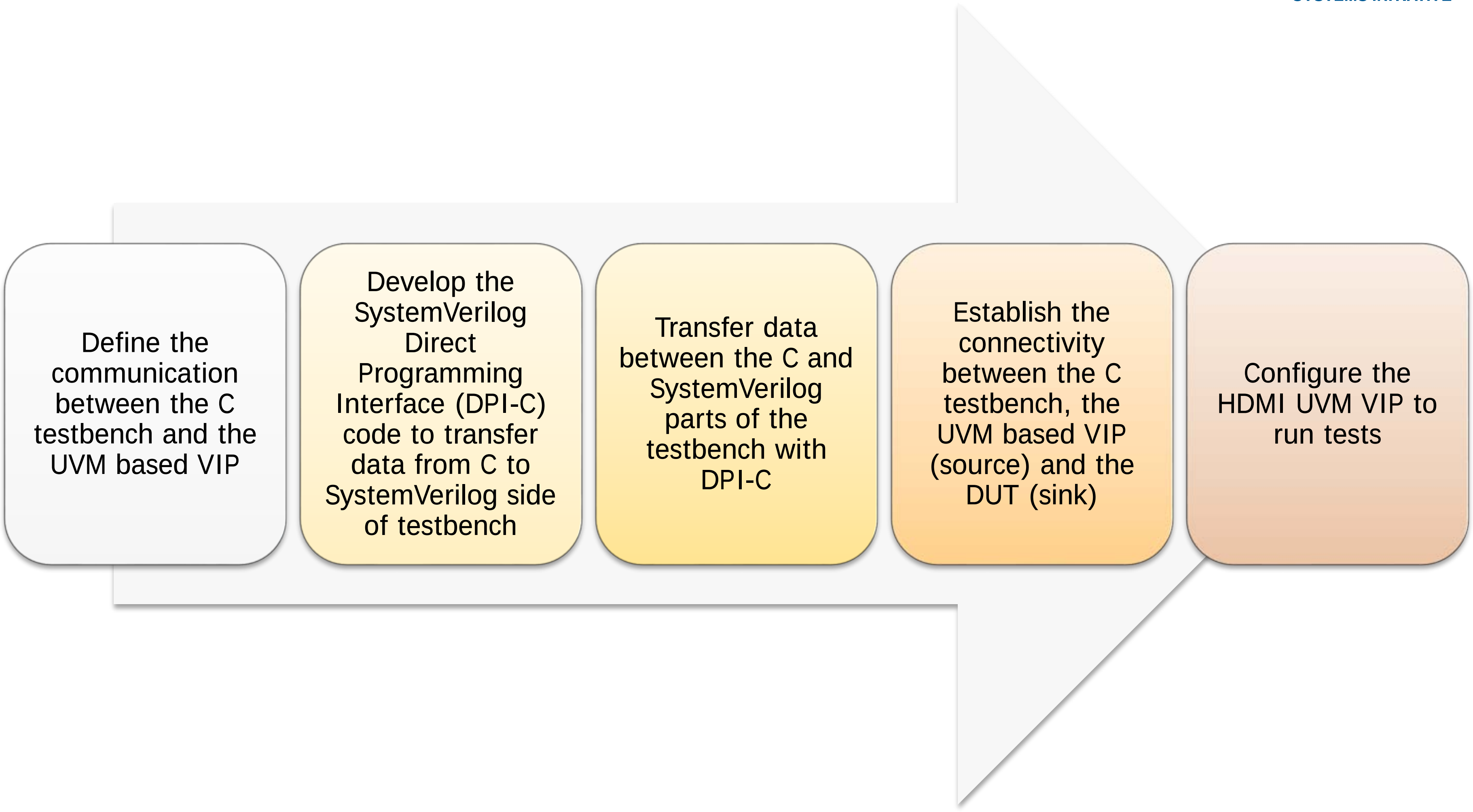
The challenge for this project was to integrate the legacy C testbench used for HDMI stimulus generation with the HDMI UVM based VIP

Ensure complete reuse of the C-based testbench and efficient usage of the HDMI UVM VIP

The HDMI VIP uses UVM-compliant classes to represent protocol activity and the characteristics of that activity.

The C testbench acts as a stimulus generator and creates the frame as per the HDMI protocol. The frame data is passed to the UVM based VIP, which drives them on signal interface as per the protocol

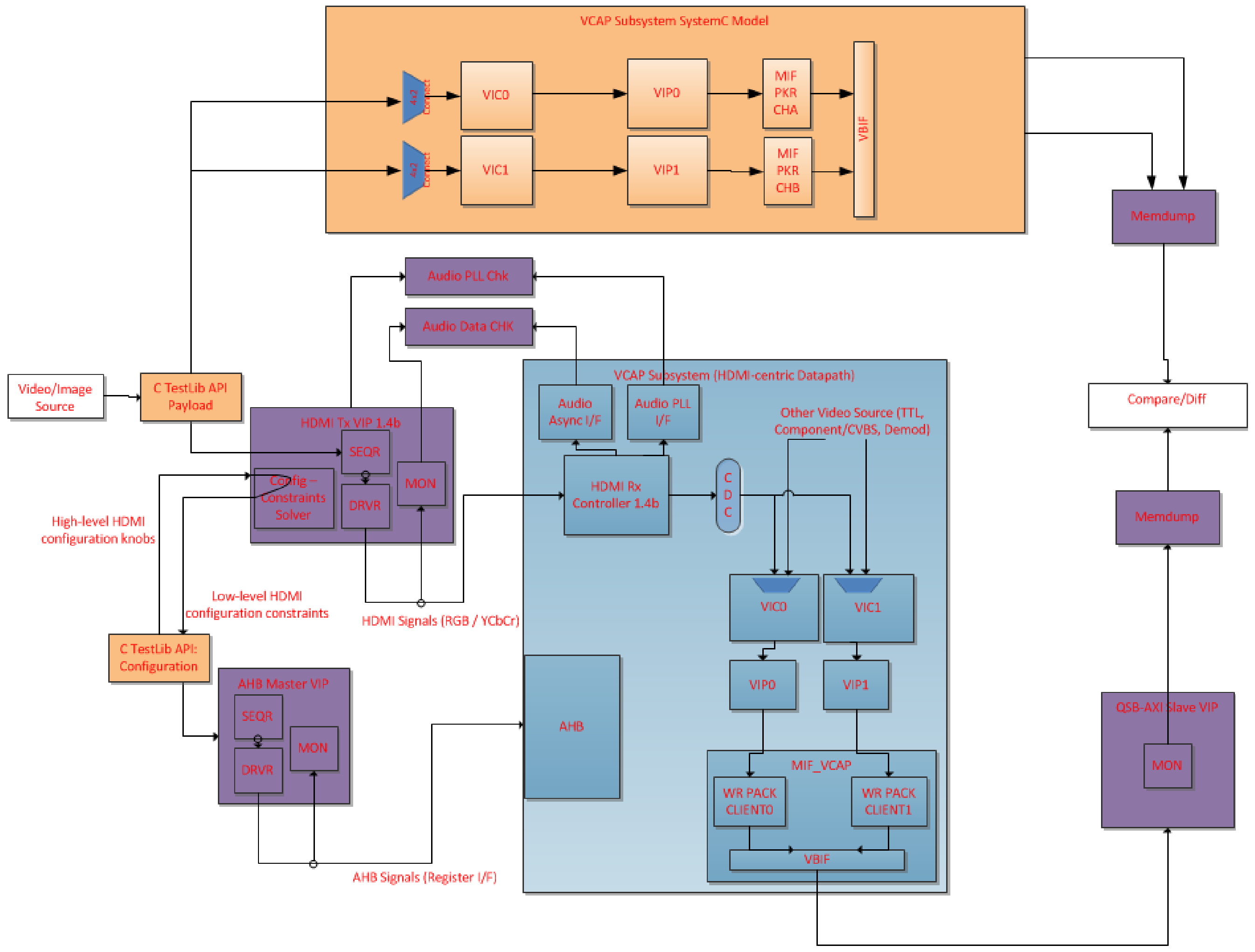
Project Milestones


Define the communication between the C testbench and the UVM based VIP
Develop the SystemVerilog Direct Programming Interface (DPI-C) code to transfer data from C to SystemVerilog side of testbench
Transfer data between the C and SystemVerilog parts of the testbench with DPI-C
Establish the connectivity between the C testbench, the UVM based VIP (source) and the DUT (sink)
Configure the HDMI UVM VIP to run tests

DVCon 2013 Design & Verification Conference & Exhibition

Sponsored By: accellera SYSTEMS INITIATIVE

The Testbench



🟧 C COMPONENTS

🟪 SNPS HDMI VIP

🟦 QCOMM DUT

Sponsored By:

SYSTEMS INITIATIVE

The Test Flow – 1/2

- The UVM VIP enters build phase, deliberately skips the randomization of the configuration class and then enters the connect and run phases where it polls the *test_done* flag, set by the C model.

- The C model initializes and then calls test library functions to set high-level knobs. DPI-C functions are used to set fields which eventually go into SystemVerilog constraint blocks within the configuration class. Constraints include video, audio and packet mode and traffic profile.

Sponsored By:

SYSTEMS INITIATIVE

The Test Flow – 2/2

- The C model calls a SystemVerilog function which builds and randomizes the configuration class with the applied constraints
- The C model reads back low-level constraints which were solved by HDMI VIP and then configures the DUT with same constraints.
- C model starts HDMI traffic sequence in the VIP. In the UVM VIP, the sequence generates N transactions and sends it to driver and then to the DUT

Sponsored By:

Stimulus Generation – 1/2

- The video frame stimuli needed to be generated from C test library and sent to the HDMI VIP via DPI-C tasks.

- The DPI-C export task *sv_hdmi_send_frame_line()* is invoked from C test library, and within the task a DPI-C import task *c_hdmi_get_frame_line_pixels()* is invoked from HDMI VIP to retrieve the frame line from C test library and pack it to the HDMI VIP sequence item for transmission.

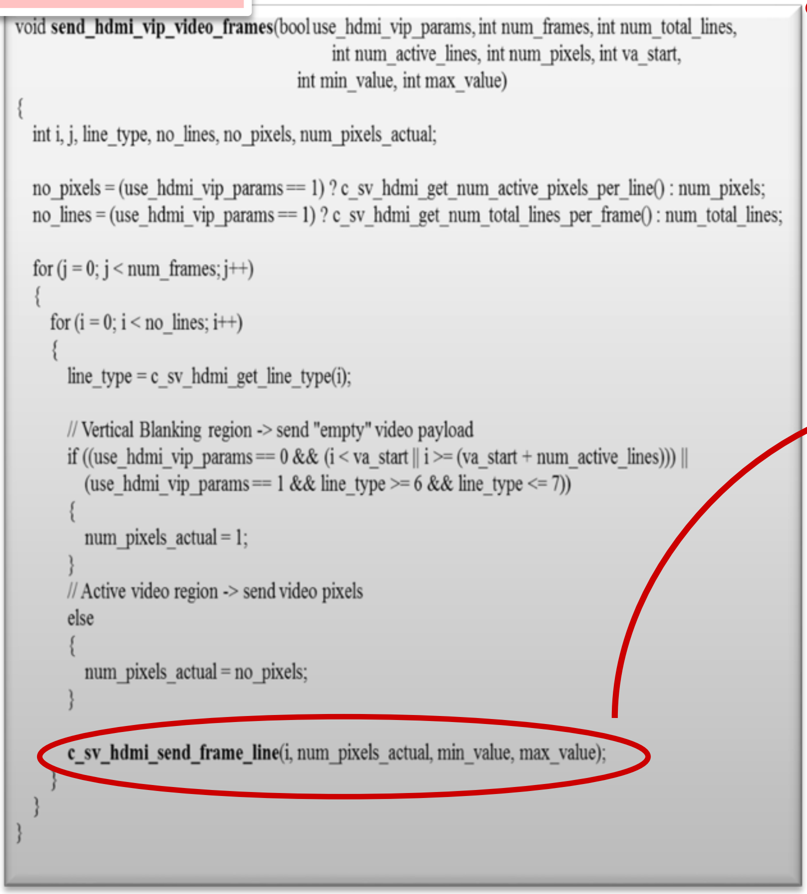
DVCon 2013 Design & Verification Conference & Exhibition

Sponsored By:

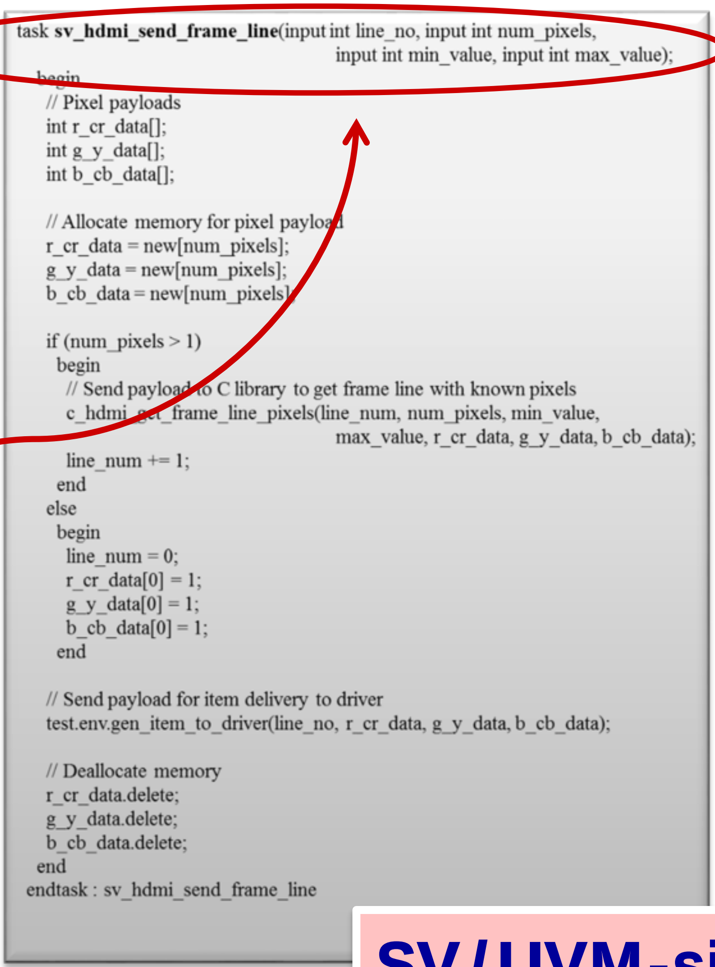
accellera SYSTEMS INITIATIVE

Stimulus Generation – 2/2

C-side



```
void send_hdmi_vip_video_frames(bool use_hdmi_vip_params, int num_frames, int num_total_lines,
                                int num_active_lines, int num_pixels, int va_start,
                                int min_value, int max_value)
{
  int i, j, line_type, no_lines, no_pixels, num_pixels_actual;

  no_pixels = (use_hdmi_vip_params == 1) ? c_sv_hdmi_get_num_active_pixels_per_line() : num_pixels;
  no_lines = (use_hdmi_vip_params == 1) ? c_sv_hdmi_get_num_total_lines_per_frame() : num_total_lines;

  for (j = 0; j < num_frames; j++)
  {
    for (i = 0; i < no_lines; i++)
    {
      line_type = c_sv_hdmi_get_line_type(i);

      // Vertical Blanking region -> send "empty" video payload
      if ((use_hdmi_vip_params == 0 && (i < va_start || i >= (va_start + num_active_lines))) ||
        (use_hdmi_vip_params == 1 && line_type >= 6 && line_type <= 7))
      {
        num_pixels_actual = 1;
      }
      // Active video region -> send video pixels
      else
      {
        num_pixels_actual = no_pixels;
      }

      c_sv_hdmi_send_frame_line(i, num_pixels_actual, min_value, max_value);
    }
  }
}
```



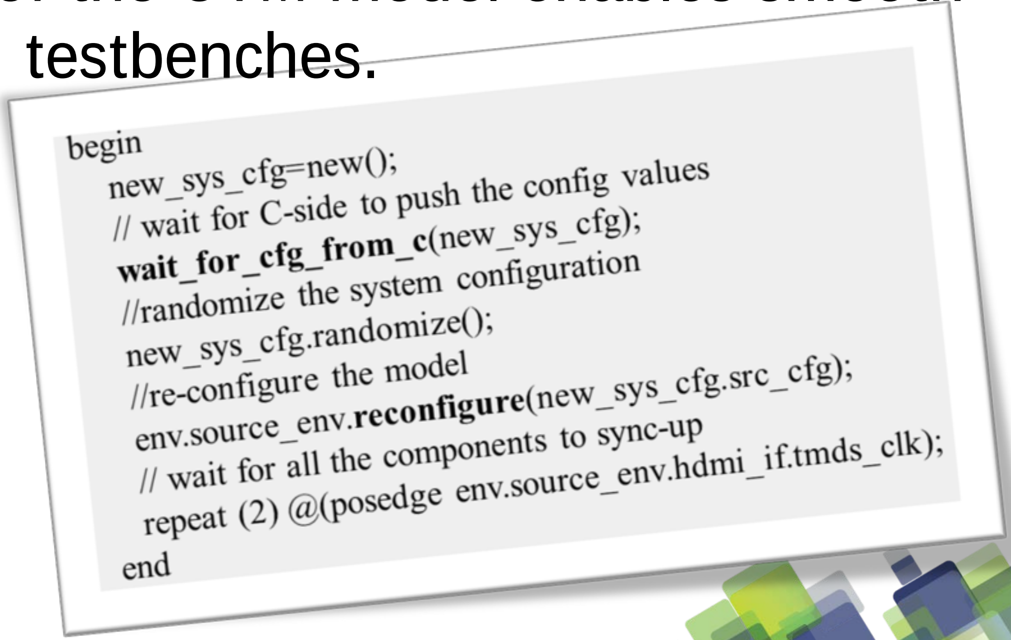
```
task sv_hdmi_send_frame_line(input int line_no, input int num_pixels,
                             input int min_value, input int max_value);
  begin
   // Pixel payloads
   int r_cr_data[];
   int g_y_data[];
   int b_cb_data[];

   // Allocate memory for pixel payload
   r_cr_data = new[num_pixels];
   g_y_data = new[num_pixels];
   b_cb_data = new[num_pixels];

   if (num_pixels > 1)
    begin
     // Send payload to C library to get frame line with known pixels
     c_hdmi_get_frame_line_pixels(line_num, num_pixels, min_value,
                                  max_value, r_cr_data, g_y_data, b_cb_data);
     line_num += 1;
    end
   else
    begin
     line_num = 0;
     r_cr_data[0] = 1;
     g_y_data[0] = 1;
     b_cb_data[0] = 1;
    end

   // Send payload for item delivery to driver
   test.env.gen_item_to_driver(line_no, r_cr_data, g_y_data, b_cb_data);

   // Deallocate memory
   r_cr_data.delete;
   g_y_data.delete;
   b_cb_data.delete;
  end
endtask : sv_hdmi_send_frame_line
```

SV/UVM-side

Sponsored By:

Reconfiguration

- UVM models generally configure during build phase.
- Need to reconfigure the UVM model, once C testbench has determined its configuration
- Reconfigure architecture of the UVM model enables smooth integration with non-UVM testbenches.



```
begin
    new_sys_cfg=new();
    // wait for C-side to push the config values
    wait_for_cfg_from_c(new_sys_cfg);
    //randomize the system configuration
    new_sys_cfg.randomize();
    //re-configure the model
    env.source_env.reconfigure(new_sys_cfg.src_cfg);
    // wait for all the components to sync-up
    repeat (2) @(posedge env.source_env.hdmi_if.tmds_clk);
end
```

Sponsored By:

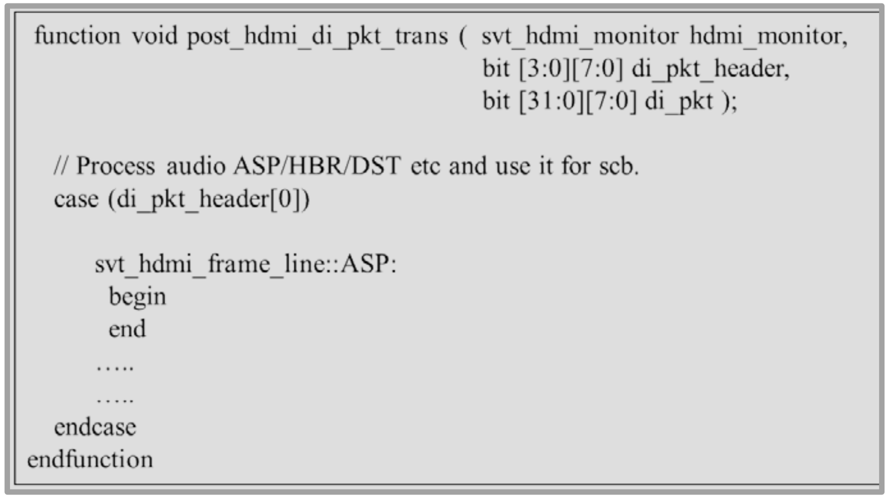
Video Score-boarding

- By allowing the HDMI VIP to retrieve video frames from the C test library, the same video frame stimuli can be applied to both testbenches for producing golden memory dumps
- The video data scoreboard relied on the C reference testbench for reference video frames.
- The C reference testbench contains models which emulate the video processing functions found in the DUT.
- Once a C simulation completes the memory output file is used to validate the RTL simulation results. In this way the video data scoreboard acts as a post-process checker.

Audio Score-boarding

- Audio data was entirely generated and checked within the context of the HDMI Source VIP.

- A callback function within the UVM based HDMI Source monitor was used to extract reference audio samples and queue them up in the scoreboard for future comparison.



```
function void post_hdmi_di_pkt_trans ( svt_hdmi_monitor hdmi_monitor,
                                        bit [3:0][7:0] di_pkt_header,
                                        bit [31:0][7:0] di_pkt );

  // Process audio ASP/HBR/DST etc and use it for scb.
  case (di_pkt_header[0])

      svt_hdmi_frame_line::ASP:
        begin
        end
      …..
      …..
  endcase
endfunction
```

Sponsored By:

SYSTEMS INITIATIVE

Take Away!

Integrating Qualcomm legacy C testbench with the Synopsys HDMI UVM VIP helped pull in project verification schedule by at least 3 man months.

The simulation performance was measured as a tradeoff between the relaxed System Verilog constraint solver efforts and overhead for DPI-C calls.

Though the UVM based HDMI VIP was used to demonstrate this flow, the approach can be well leveraged with other VIPs and methodologies across various constrained random verification environments to increase the verification productivity

Sponsored By:

Thanks for your time!!