2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

HISIG- An Efficient Gate Level Simulation Flow for a Hard IP Inside a Soft IP

Ishwar Ganiger, Senior Engineer, Infineon Technologies India Pvt. Ltd., Bangalore
(Ishwar.Ganiger@infineon.com)
Vishal Dalal, Principal Engineer, Infineon Technologies India Pvt. Ltd., Bangalore
(Vishal.Dalal@infineon.com)
Johannes Grinschgl, Principal Engineer, Infineon Technologies AG, Munich, Germany
(Johannes.Grinschgl@infineon.com)

***Abstract*- Gate Level Simulation (GLS) is an important aspect of the verification cycle in SoC (System-on-Chip) design [3]. It can identify issues that may have been missed by other flows, such as Register Transfer Level (RTL) simulations, synthesis, Static Timing Analysis (STA), and logic equivalence checks [4]. The individual IP's for which timing closure is not done (called as Soft-IP) are integrated together to form a SoC. Gate level netlist of these Soft IP's are generated at SoC to leverage area and timing optimizations. Therefore, GLS is a natural choice to do at SoC level. In some cases, where timing closure is done at IP level itself (called as Hard IP) GLS need to be done at Hard IP level to catch timing issues using IP level Testbench (TB) and Design Verification (DV) setup. Timing closed Hard IP netlist is used as it is in SoC GLS.**

There are IP's where a particular IP sub-block (e.g. IO pins) is hardened (timing closed) at the IP level itself while rest of the IP sub-blocks are retained as RTL (timing not closed hence still Soft IP). This is called Hybrid-IP where Soft and Hard IP components co-exist. In this case, GLS need to be performed only for hardened IP sub-block to check its timings. Timing checks for soft part will be done using SoC level netlist. This poses several challenges for sub-block GLS which cannot be done at Hybrid IP level with signals from soft IP part crossing to hard sub-block and vice versa which create timing issues.

In this paper we share details of a Hard IP inside Soft IP GLS (HISIG) flow used to verify timings for a hard sub-block instanced inside a soft IP. HISIG was used only for hard IP sub-block by creating an innovative TB architecture as existing full Hybrid-IP TB can't be reused due to timing issues. This paper elaborates how HISIG was used for timings check sign-off, its benefits, overheads, and challenges. HISIG has already been proven in one of the live designs covering all functional and timings critical scenario. It's easy to use and saved overall ~90% of IP GLS efforts.

Keywords- GLS, Soft IP, Hard IP, Hybrid IP, SDF, Automation, Timing checks, RTL

I. INTRODUCTION

GLS is usually executed at the full-chip level by annotating timing information from SDF (Standard Delay Format) to the gate level netlist [2,4]. Typically, the gate level netlist for the individual IPs is not available because synthesis is performed at the complete chip level to leverage various timing and area optimizations. GLS at IP level can still be performed if timing is already closed at the IP level (Hardened IP). For this Hard IP GLS, existing TB infrastructure can be reused after including relevant netlist and SDF. Standard GLS procedures like removing timing checks on sync flops, migration of TB infrastructure as per netlist hierarchy, register initializations etc need to be done which is similar to any SoC GLS. This methodology is well known and adopted across design executions.

There are IP's where a particular sub-block is hardened with timing closure while rest of the IP sub-blocks are retained as RTL (Soft-IP). This is called as Hybrid IP [1]. The netlist for soft part is generated at SoC level while hardened part netlist is timing closed at IP sub-block level. GLS need to be performed only for hard IP sub-block to check timings as the timing checks for rest of the soft IP part will be done at SoC level. However, running a GLS at an IP level for Soft-Hard Hybrid IP requires the instantiation of the Hardened netlist inside the RTL of the Soft IP as shown in Fig. 1. Hardened block has timing arcs while rest of the Hybrid IP logic is still in RTL.

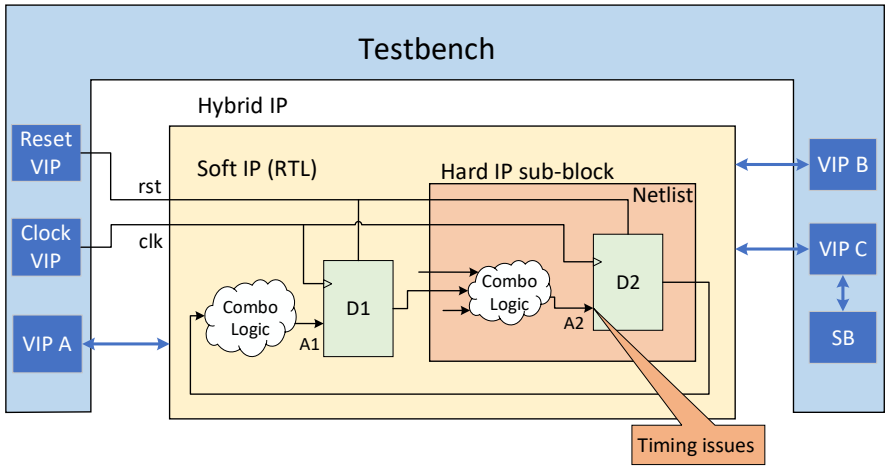

Figure 1. Signals crossing soft and hard logic domains create timing issues.

Fig. 1 shows a typical TB architecture consisting of various standards Verification IP (VIP) driving/sampling with Hybrid Design Under Test (DUT) IP IO ports, Scoreboard (SB), Clock VIP and Reset VIP. This was used to verify when full DUT was in RTL but can't be reused if hard IP sub-block is replaced with timing closed netlist.

This combination of Hard and Soft components together presents several challenges to check timing at Hybrid IP level and are summarized below.

- As shown in Fig. 1 inputs to hard block comes from soft IP without any delay. Outputs from hard block are with timing delays. These outputs can go as primary outputs of the Hybrid IP or can feedback to soft IP logic (RTL). This makes them incompatible to work together in a single TB.
- Timing issues will arise in flops inside hardened block since only the hardened part is timing annotated while signals that affect the flops inside hard block may originate from both the soft and hard blocks as shown in Fig. 1.1.

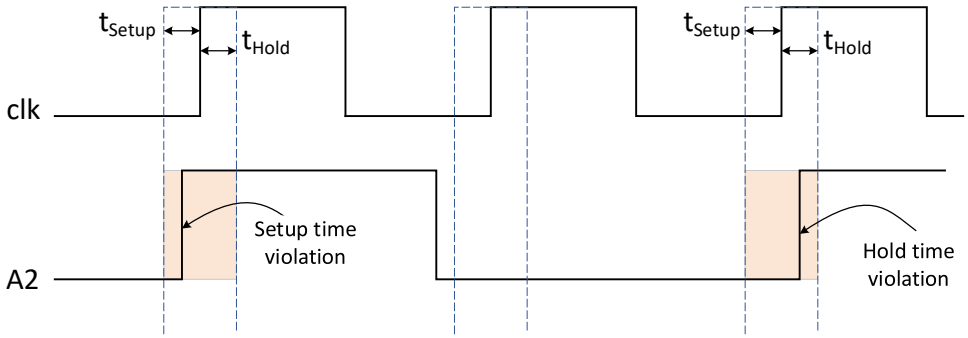

Figure 1.1. Timing violation due to signal crossing

- Existing IP verification flow could not be reused as signals crossing soft and hard domains need to be delayed to meet timing inside hard block for all test cases (TC) which is practically not possible.

- Behavioural model of hard IP sub-block that include timing information was not available as it was humongous effort to create an accurate model considering its complexity and given schedule.
- RTL of hard sub-block was developed along with soft IP part. Hard IP sub-block RTL was not verified standalone. Instead, it was verified along with full IP RTL leveraging soft IP TB components of Fig. 1. Efforts required to create standalone TB for hard IP sub-block was approximately 5 Man Months (MM) which was too high for the given DV schedule.
- Another option was to transfer hard IP sub-block GLS to SoC. This was ruled out as it will be too late to find timing issues inside hardened block with potential to risk SoC tape out due to late IP bugs.

We were left with no option but to think of an innovative flow or methodology to effectively check timings for the hardened sub-block that should do essential GLS checks in a given schedule. Hard IP inside Soft IP GLS (HISIG) flow provided an efficient DV flow to resolve these issues and complete GLS for hard IP sub-block.

II. HISIG FLOW

To address the challenges of implementing GLS at the IP level for soft-hard combination Hybrid IP, we deployed HISIG flow by replaying the stimulus from the RTL test cases at Hybrid IP level to drive inputs of hard block netlist in a standalone TB environment. This is similar to what is used in flows that calculate IP power numbers using design switching in commercial tools like PowerReplay [5].

The steps involved to complete GLS using HISIG flow are summarized as follows (see also Fig. 2):

- Run the existing testcase (TC) for the full IP TB that include RTL of soft and hard IP sub-block.
- Capture the hard IP sub-block IO ports activities in standard VCD (Value Change Dump) format.
- Convert the VCD file to a SystemVerilog (SV) stimulus file such that all the input signals of hardened block are available as nets along with complete input toggle information as shown later in Fig. 4. This can be easily hooked up with the netlist later.
- Create standalone TB, instantiate the hardened block netlist, annotate it with SDF and include the SV stimulus file generated above to drive the toggle at input ports.
- Run GLS using standard procedures like removing timing checks on Sync flops, register initializations, enable timing checks of behavioural models etc.
- Replay the input stimulus that include complete toggle information from RTL simulations and generated above.

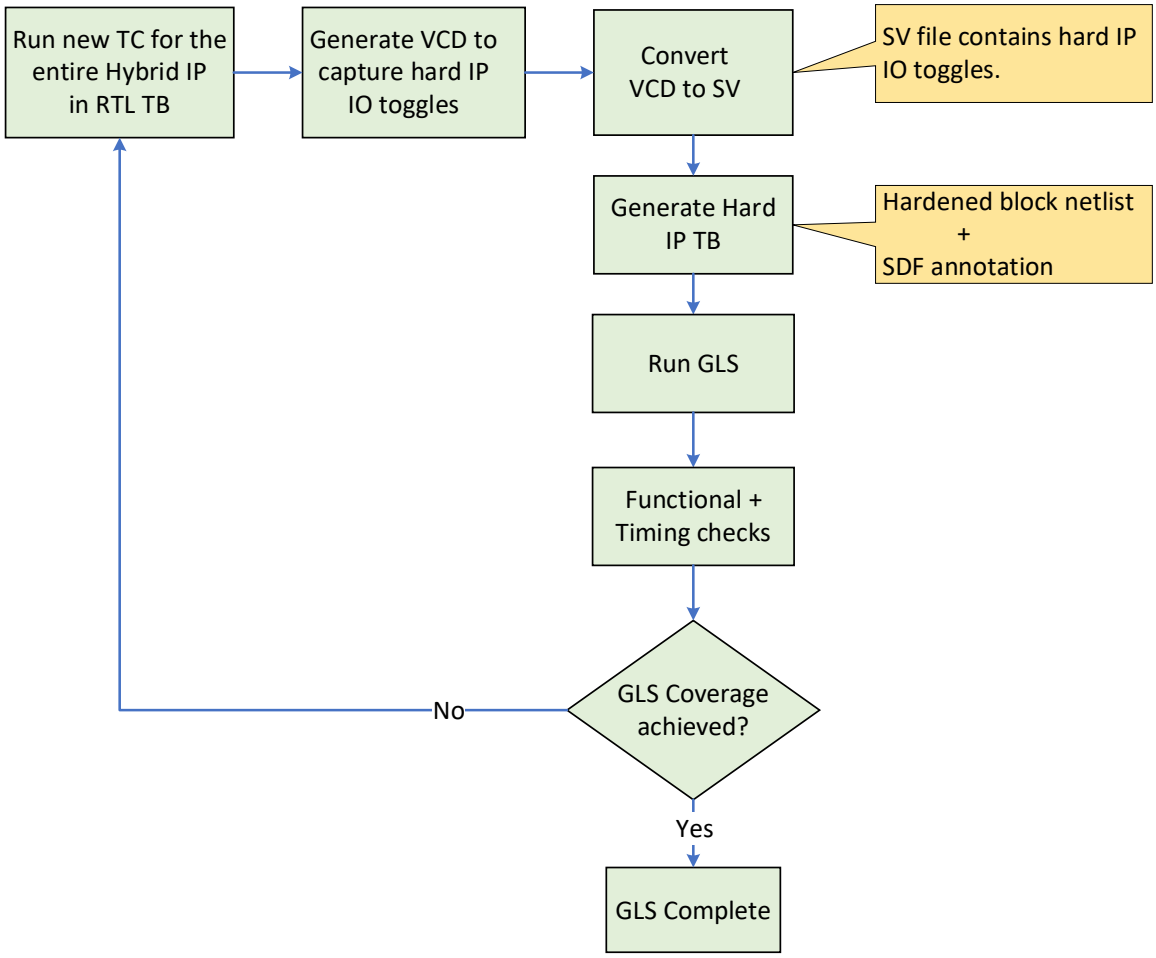

Figure 2. Hard IP inside Soft IP GLS (HISIG) Flow

- Perform functional and timing checks of hard IP sub-block. Add automated checking using assertions. Fig. 5 later illustrates an example assertion for checking the toggle behaviour of an output signal. In this case, the observed output signal is compared with the signal captured from RTL simulation.
- Run additional testcases to cover all timing critical paths and desired functional coverage like Single Data Rate (SDR) and Double Data Rate (DDR).

III. TB Setup for HISIG Flow

The TB setup for the HISIG flow is illustrated in Fig. 3. It is designed to replay the stimulus generated from Hybrid IP RTL runs and apply to inputs of Hard IP. This involves the following steps:

- "VCD Processor" block reads the VCD generated from RTL simulation using existing full IP (Hybrid) TB infrastructure.
- "Functional pin toggle capture" block extracts the input port toggle information from VCD and convert them to SystemVerilog nets.

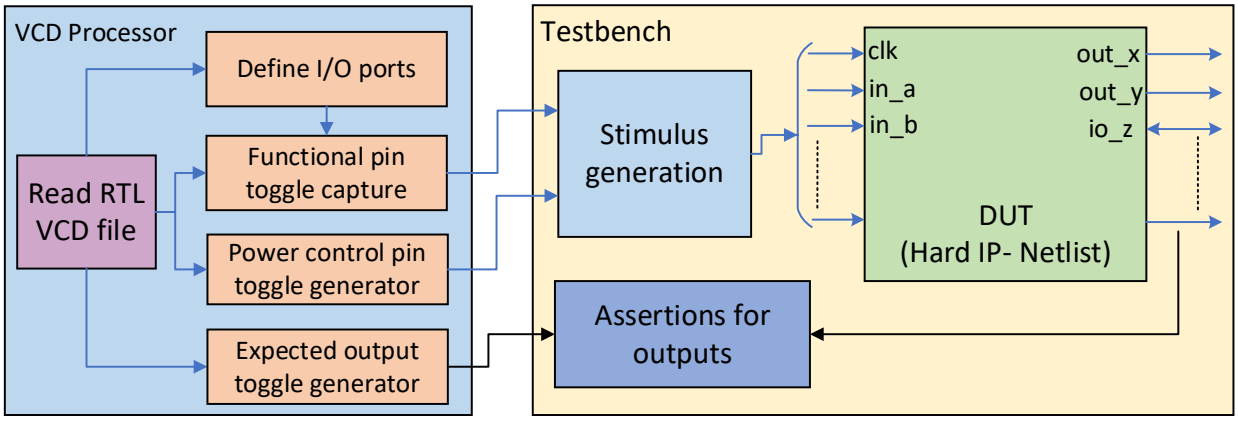

Figure 3. HISIG automation & TB Setup

- Fig. 4 shows sample input toggle information captured from the VCD and converted to SV nets. The delays mentioned in picoseconds here are extracted from the VCD.

```
logic in_a;

initial
begin
  #1000ps;
  assign in_a = 1'b0;
  #3000ps;
  assign in_a = 1'b1;
  #1500ps;
  assign in_a = 1'b0;
  #5000ps;
  ……
  ……
end
```

```
logic[7:0] in_b;

initial
begin
  #2000ps;
  assign in_b = 'h8;
  #7000ps;
  assign in_b = 'h3;
  #5500ps;
  assign in_b = 'h1a;
  #8000ps;
  ……
  ……
end
```

Figure 4. VCD to SV converted stimulus file

- The timing relationships for input signals, such as clock-to-data timing, are manually incorporated to maintain the right skew. This is required to enable correct data sampling.
- In order to establish a stable initial state for the simulation, the power control inputs, and test mode inputs are driven to inactive values by “Power control pin toggle generator” block in Fig.3. This ensures that any unintended behaviours caused by transition of these signals are avoided.

- "Expected output toggle generator" block generates expected toggle values of hard IP sub-block outputs using information captured from RTL VCD. This expected output (rtl_out_x) generated by HISIG flow is compared with the actual hard IP DUT output toggle (gls_out_x) observed during GLS. This is done using assertions as depicted in Fig. 5.

```
property output_check;
   logic sig_rtl;
   @(posedge clk)
   ($changed(rtl_out_x), sig_rtl = rtl_out_x) |-> ($changed(gls_out_x) && gls_out_x == sig_rtl);
endproperty

a_out_x: assert property(output_check);
```

Figure 5. Output assertion check

Fig. 6 depicts a high-level TB pseudocode representation of the automation done by "VCD Processor" block used inside HISIG flow.

```
READ VCD; //Generated from the Hybrid IP RTL
signal_list = get_signal_list(); //Get all the IO signals of hard IP

FOR i=0 TO len(signal_list)
BEGIN
  define_sv_net(signal_list[i]); // Define SV net for each signal
  collect_ip_io_toggle(signal_list[i]); //Collect toggling information for each signal

  IF signal_list[i] is INPUT
  BEGIN
     gen_input_toggles(signal_list[i]); // Drive SV net using collected IO toggle information
  END
  ELSE
  BEGIN
     gen_expected_output_toggles(signal_list[i]); // Generate expected toggle behaviour for outputs
     gen_assertion_for_outputs(signal_list[i]); //Generate automated checks for output toggle behaviour
  END
END
```

Figure 6. VCD Processor Pseudocode

IV. RESULTS

HISIG flow is applied to an IP that uses Serial Peripheral Interface (SPI) master protocol for IO communications and instanced hardened sub-block of ~50K inside the Hybrid IP. The hardened sub-block has 162 input ports, 68 output ports and 21 Inout ports. 7 test cases were judiciously used to cover timing critical aspect like SDR/DDR and covering all DUT functional IO ports. Average run time for the tests was ~15 minutes. GLS was done for both MAX and MIN timing corners. Logs were reviewed for timing violation with design team. Although there were no functional bugs or timing violations caught by HISIG flow, its usage greatly improved the overall confidence on timing closure of hardened sub-block. Later during SoC GLS also no IP issues were found, and design was successfully taped out with high confidence on Hybrid IP timing closure.

TABLE I
RESULTS

IP gate count	~50K
# Testcases	7
Timing corners checked	MIN, MAX
Average TC run time	~15 minutes
HISIG GLS setup time	0.5 MM
HISIG usage efforts saving	~90%
Reusability across designs	YES

Key advantages of HISIG flow are summarized below.

- Only 0.5 Man Months (MM) of efforts to create GLS setup. Standalone TB setup for hard IP sub-block with all its components like VIP, Scoreboard, Drivers, monitors etc would have taken ~5 MM efforts. Therefore, HISIG flow offers ~90% efforts saving as shown in Table 1.
- Flow is simple and easy to use with plug and play TB. Reference workflow needed for HISIG is quite straight forward.
- Improved DV quality with timing checks done earlier in the design cycle at IP level itself. This minimized the risk of finding timing or potential functional issues at SoC level GLS. This was critical to avoid any impact on SoC design due to IP issues.
- As the flow is reading the VCD and generating TB infrastructure from the VCD, it is easily portable and can be reused across different design.

The overheads and limitations of HISIG flow are summarized below.

- GLS snapshot need to be regenerated with each test case as the input toggle stimulus file will change for respective TCs. This is a bigger issue at SoC GLS due to huge netlist size and time needed to compile entire design. This is not significant at IP level as corresponding netlist size is not big compared to SOC. HISIG flow was also modified to compile design/netlist and TB files separately. This removed the repetitive netlist compilation for each TC reducing the compile time.
- Hard block IO toggle information is derived from RTL simulations and not driven directly from actual netlist. This can lead to minor inaccuracy like delay in input toggles by few picoseconds only and therefore does not impact functional data sampling.
- Data integrity checks are done using assertions comparing RTL and GLS signals only as VIP's were not integrated in HISIG TB. As the stimulus is derived from Hybrid IP RTL simulation run, it is not possible to incorporate randomness into the stimulus during GLS.

- Timing checks between last bit flop of soft IP part and first bit flop of hard IP part (shown as D1 and D2 in Fig. 1), and vice versa, can't be fully proven with HISIG as this timing is manually set in TB. This problem would have manifested even if GLS was done at standalone hard IP netlist level. Checks for these cross over paths need to be done at SoC level GLS.

V. CONCLUSION

This paper discussed the challenges and benefits of GLS at the IP level for soft-hard Hybrid IP. HISIG flow is simple, effective and can easily be reused across designs. It optimizes IP GLS DV cycle by ~90% and completes timing check earlier in the design cycle. HISIG implementation is demonstrated through a case study and desired timing as well as functional coverage was achieved improving the overall quality of the IP. Advantages of using HISIG in a given schedule outperforms its overheads.

REFERENCES

[1] Gagandeep Singh "Addressing Renewed Gate Level Simulation Needs for 10nm-28nm and Below", DVCON Europe, 2016
[2] Ateet Mishra, Deepak Mahajan and Shiva Belwal "Absolute GLS Verification- An Early Simulation of Design Timing Constraints", DVCON India, 2015
[3] Ashok Chandran, Roy Vincent "Gate level Simulations: Continuing Value in Functional Simulation", DVCON India, 2014
[4] Vishal Dalal, "Relevance of Gate Level Simulations in Today's SoC Verification" short tutorial at 13th VLSI Design and Test Symposium, VDAT India, 2009
[5] Synopsys PowerReplay flow, https://www.synopsys.com/verification/simulation/powerreplay.html