DVCon India 2015

TITLE OF PAPER	**Methodology for Hardware Software Co-verification of Video Systems on Pre and Post Silicon**
AUTHOR	**Name : Vinesh Peringat** **Organization: Xilinx India Technology pvt Ltd** **Job Title: Sr. Systems engineer** **Email ID: vineshp@xilinx.com** **Mobile no: 9912692048** **Alternate contact no: - +91 40 67214619**

ABSTRACT

Video Systems HW-SW Co-Verification is highly critical on emulation platforms to make sure SoC designed would meet its intended end application.

Challenges in traditional Verification/Validation approach for Video Systems:

- Main challenge in Functional Verification is time taken to run Video tests on Simulation platform and lack of capability of producing real world application scenarios.(This brings in pressing need to validate Video systems on Emulation platforms)
- Problems with traditional Display validation:
    - Test bench need to save Video frames with lot of memory to do offline frame checking. Creating pixel accurate IP model results in increased complexity of test bench.
    - Frame checksum option just helps to identify if there is a problem or not, but doesn’t help to root cause the failure
    - Creating custom loop back interface to memory (for Display interface) option violates the use case validation if usage is just to display frames.
    - Visual checking by connecting a Display monitor is less effective as only low resolution can be checked, and then data (pixel) integrity is not tested.

Summary of Proposed Validation methodology for Video Systems:

- The proposed Verification and Validation Methodology for Video systems would use simple and efficient Test bench on Emulation Platform (and Silicon) to find Display Pipeline RTL bugs, repeatable failures and quick root cause of defects.
- In current world display systems the frame rates are high and resolutions are more than High Definition data rates, so the proposed solutions also makes sure that the test bench doesn't need huge memory to store Video Frames (not even Video lines) to do Pixel accurate checking.
- Display systems, being just more than connectivity interface has inherent DMA modules and processing features like CSC, Alpha Blending and Scaling. Traditional Verification methods use reusing IP C-models for pixel accuracy checking, whereas this method uses "approximate" pixel checking to make sure data integrity is intact without much loss of coverage. (Based on fact that the human eye is less sensitive to colour changes in LSB, and this proves to be very effective for YCbCr frames too).
- Solid platform for Use – case validation or Application verification where in systems are exercised with concurrent chains (several CPUs and DMAs) operating in parallel.
- Frame changes (process of switching from one frame buffer to another), while effectively checking the rate of switching(Frame rate change performance benchmarking)
- Synchronizing system traffic with Video timing event(s) like (Pixel count, Line count or V/HSYNC) to regenerate a specific traffic condition to replicate failure.
- The register file generated/used by Verification team is reused and "HAL" (Hardware Abstraction layer is built on top of this, which would be helpful for drivers (software) team to reuse the tested APIs.
- Testing the "funny Video timings" and resolutions for easy reproduce of issues on simulation with custom Video timings.

The main focus of this Validation methodology is for Validating Video Use cases(with driver APIs) on emulation platform(FPGA/Veloce/Palladium/Zebu) or Silicon , and provides way to run several application threads in system, and root cause the failures quickly using features on Test bench using repeatable system traffic.

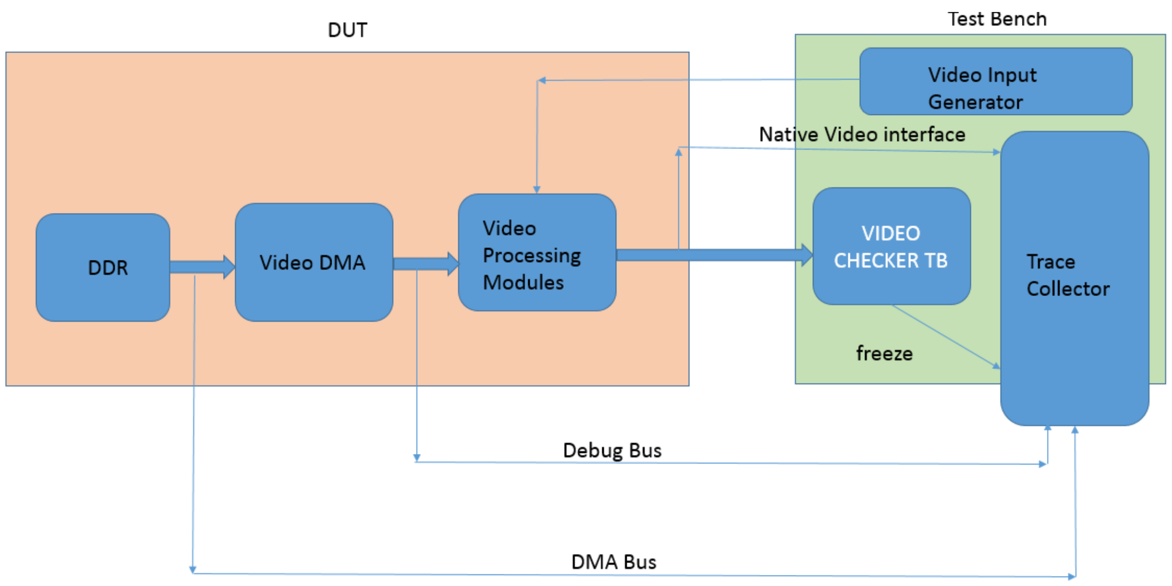
And most of bugs in Video system reproduce needs in designers words " ***This issue needs the events to be happening in a specific manner , and can be seen after running long thin several skinny frames with minute blanking***" – Hence, Idea is to vigorously test systems with varied configurations for long hours on FPGA prototyping board or Palladium – To alleviate the issues like " ***My Digital Video Recorder box hangs after running for several days on Field***" – which are otherwise very tough to reproduce scenarios.

Architecture and/or flow diagram

The proposed validation system comprises of SoC emulated on FPGA or Veloce platform, with Video Test bench shown below. Test bench components include a simple Pixel checker and Video timing and pattern generator, with a simple register interface for configuration, and a trace collector (to store snapshot of interesting signals when pixel errors are detected). Idea is same Test bench components can be reused for silicon bring up/debug.

The register libraries used by functional verification is reused after developing "Hardware Abstraction Layers" to ensure early development/co-verification of drivers.

Finally, the concurrent chains (building blocks for Use cases) are developed and run for targeted system stress testing to mimic the real Video application scenarios, while effectively using the environment for failure repeatability and quick root cause of defects.



DDR: Stores Video Frames in memory typically prepared by CPU/GPU.

Video DMA: Engine reads from DRAM and transfers data to Processing modules.

Video Processing: Performs Video arithmetic functions like CSC/Blending/Scaling/ Color Keying.

Video checker TB: Performs pixel comparison operation against a standard/ramp pattern.

Trace Collector: Collects few pixels before failure happened along several points on Video pipeline. (For post silicon useful buses could be tapped similarly, if needed encoded)

Video input generator: Generates Video timing & data as per programmed resolution.

Summary of Features:

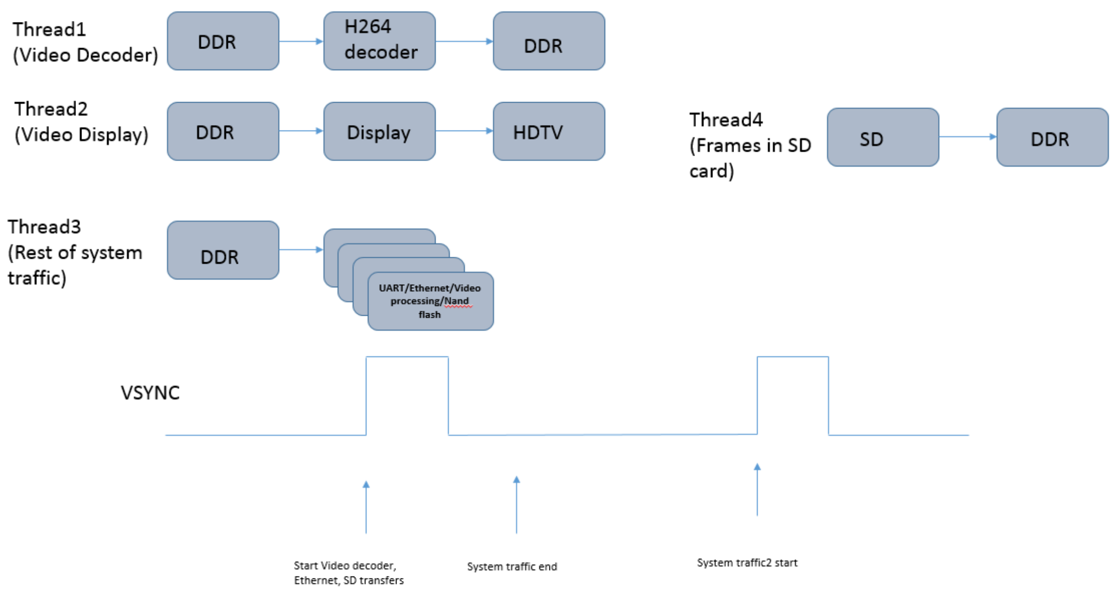
Below are the illustrations of proposed Validation method:

1. Simple RGB frame checking(Data path testing)
   a. Constant varying frame buffer created in memory with pixel toggle coverage as desired.
   b. Simple pattern registers implemented in Test bench to recreate the pattern and checked against incoming video pixels, after the VSYNC.
   c. Mismatches are detected and trace collector records few pixels (say 100-200) pixels before the failure to toot cause failure with DMA BUS and Video out signals.
2. YCbCr/ Blending checks (Video Arithmetic testing)
   a. Simple conversion from RGB to YCbCr implemented in Checker based on ITU-T standard or blending function like ***alpha.Stream1 + (1-alpha) Stream2***, and regenerated in Test Bench. Loss in conversion process could be comprehended as the tolerance in LSBs. So this is "approximate pixel checking" without much loss of data integrity.
3. Frame Buffer switching (Video frame changes/rate tests)
   a. Command from GPU to switch frame buffers is given from CPU / GPU, at the same time the patterns are updated in the Video Checker TB.
   b. By design, Video systems frame updating happen on next VSYNC to avoid tearing. We take advantage of this and by next frame both DUT and Pixel checker's Test bench will have switched to new frames (patterns).
   c. There will be cases where DUT's frame will not switch if the command is close to VSYNC etc., benchmarking this is useful to make sure performance of new frame switching. This Test bench efficiently can be used to measure frame rate changes benchmarks.
4. Repeatability with System traffic(Effective way to reproduce failures)

   Video timing events (say VSYNC) are used as synchronizing point to align video with system traffic to make any traffic for use case validation (where several Devices work in tandem in a system).

5. **Use case Validation and pre-verification of driver APIs**.

Sample use case Validation (identified from architecture and application perspective) shown below where encoded frames are stored in SD card and H.264 Decode engine will decode the frames and passed to Display engine.



Key Results

- Avoidance Validation of Video systems as a duplicate effort for Verification, where the IP features are checked against hardware specification.
- Software (HAL APIs) are co-verified with hardware on Pre-silicon environment.
- Find system level bugs like Display underruns, overflows, hangs etc. in presence of heavy backpressure and system traffic.
- Measure Quality of Service for Video traffic on DDR/ High speed memory and identify potential system bottlenecks.
- Validate Video arithmetic functions from a system level perspective with less loss of coverage on arithmetic error.
- Simple Test bench for Emulation and Silicon testing, providing ease of debug with focus on system level issues.
- 10-15 unique system level bugs on Display Validation, which is tough in simulation.

Reference

US 8020126 (Links and chains verification and validation methodology for digital devices).