# Test IP: Bringing the Tools and Methodology from Pre-Silicon Verification to Post-Silicon Validation

Al Czamara
Richard Proto
Paul Tomashevskyi

Test Evolution
Hopkinton, MA, USA

www.testevolution.com

**ABSTRACT**

*Post silicon testing of 1st silicon tends to be an ad hoc process, using protocol testers from various manufacturers, and trying to stitch them together to create test cases and debug issues. While IC's are getting to market, the process is far from ideal. Leveraging the tools and methodology of pre-silicon verification into post silicon validation is a key enabler to higher productivity. This paper will discuss the challenges of targeting HDMI 1.4b Transmit and Receive IIP and PHY technology from Synopsys to the High-performance ASIC Prototyping Systems (HAPS) including our solutions to timing closure at 4K speeds, and signaling to and from the PHYs. The use of these HDMI 1.4b "Test IP" instruments in a post-silicon test application, and the results obtained, will show the power of this new post silicon test methodology to find issues faster, characterize problems quicker, and debug more efficiently than with existing test solutions.*

# Table of Contents



# Table of Figures



# Table of Tables

# 1. Introduction

This paper will introduce a post-silicon test solution based on Test IP (TIP), with is analogous to Verification IP (VIP) in the pre-silicon verification domain. Why TIP-based post-silicon validation is useful, how we used Synopsys IP and tools to implement the first article for HDMI 1.4b testing, and how it was used in a real application along with results, will be discussed.

### *What is Test IP-Based Post Silicon Validation*

Test IP-based PSV is a paradigm shift from the way post silicon validation is currently done. Current PSV solutions are ad-hoc[1]. They cobble together individual protocol testers, scopes, misc equipment from various vendors, and on-chip logic, and try to create meaningful stimulus, response checking, and debugging. The problems with these ad hoc solutions:

- Protocol boxes from multiple vendors generally do not work together well. They all have their own driver software and programming model, and all have their own debugging tools.
- Tool flows from pre-silicon to post-silicon to production and back are nonexistent. This makes writing tests challenging and thus generally limited in robustness. Also, debugging suspected problems is difficult across more than one interface.

The fact is that no single integrated platform is available that allows single to multiple protocol testing and debugging. The Test IP based PSV system is targeted primarily at functional testing, as this is where the biggest holes in PSV are to be found. See Table 1 below. Functional testing is important in post silicon[2], and is getting even more important as feature lengths get smaller.

| Post Silicon Validation | Production Testing |
|---|---|
| Can't simulate everything. | SCAN is not covering all defects. |
| Power distribution network and power domain testing. | 50% Non-Detected Faults. |
| Fringe functionality. | No correlation to bugs in post silicon. |
| Performance validation. | Has always been done in some form, but is difficult and limited on ATE. |
| Use case testing | |

**Table 1: Functional Testing Problems & Requirements in Post Silicon & Production Test**

Figure 1 below shows a high level Test IP block diagram. The Protocol Engine is responsible for meeting the interface protocol lower level layer specifications. The Protocol Engine also contains any protocol-specific PHY. The Transaction Processor is responsible for most of the functionality of the TIP, like transaction flows and error checking, keeping the scoreboard running, and synchronization with the rest of the TIP instances in the system, for example. The Event and Waveform Log is responsible for transaction and debug information flows from the Transaction Processor in a time-accurate fashion. This architecture allows the tools and

methodologies from pre-silicon verification to be used in post-silicon and eventually production testing. The green block is a TIP carrier, which is FPGA based.

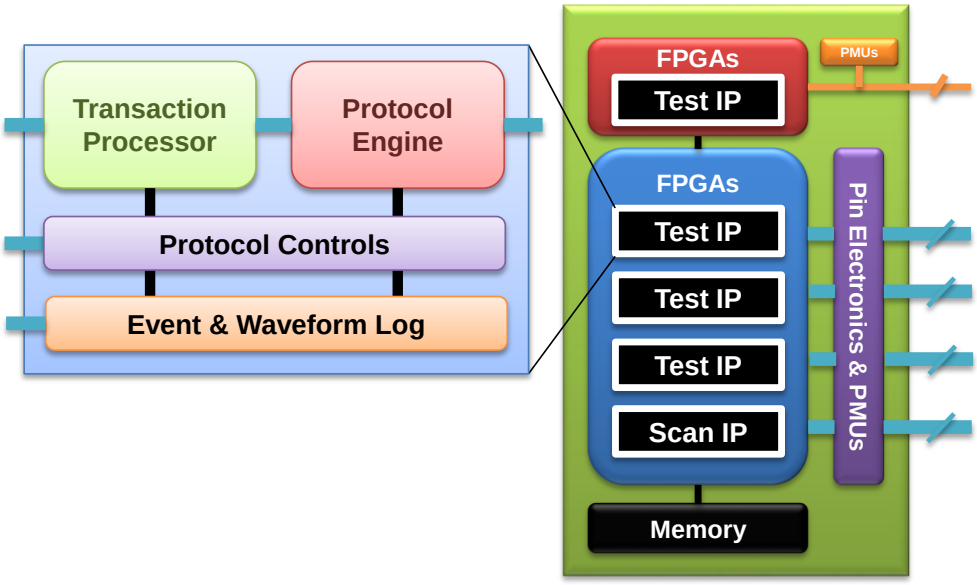


**Figure 1: Test IP High Level Block Diagram**

Figure 2 below shows how Test IP would be setup in a post-silicon testbench that excersizes multiple SoC interfaces. Just like in pre-silicon using VIP, Test IP protocols will connect to the matching protocols on the SoC. For example, HDMI RX TIP will connect to an HDMI Transmitter interface, USB 2.0 device TIP will connect to a USB 2.0 host interface on the SoC, which is the Figure is just a picture of a TI OMAP device.

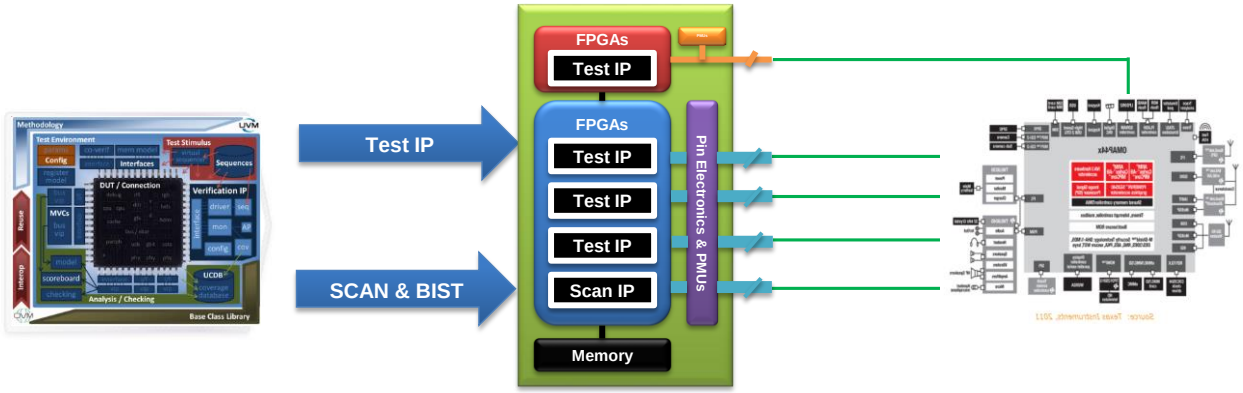


**Figure 2: Test IP Connection to an SoC**

The TIP-based PSV system has fully integrated stimulus generation, response checking, and debug features available across interface protocols, which are found only in the pre-silicon world. Table 2 below summarizes the features available in this platform.

| PSV Testing Requirement | TIP-Based PSV System Feature |
|---|---|
| Pre-to Post-Silicon Leverage | • Same methodology & tools as pre-silicon<br>• Use models allow Pre-Si <->Post-Si leverage |
| Deterministic Operation & Repeatable Test Cases | • TIP synchronization methods analogous to pre-silicon<br>• Seed-based algorithmic stimulus |
| Back-door Access | • Pre-load stimulus and response<br>• Pre-load memory TIP (e.g. DDR, Flash) |
| Self-checking Tests | • Scoreboard methodology<br>• Algorithmic response checking<br>• Pre-loaded expect data with self-checking |
| Debug Hooks | • View stimulus and response from DUT via Waveform/Protocol Debug tools across all interfaces<br>• Time-stamped scoreboard data<br>• Recreate in pre-silicon DV environment for internal DUT debug visibility |
| Clock & Voltage Flexibility | • Part of the TIP Carrier HW, controlled by TIP API |
| Error Injection | • Scenario-based callbacks in transaction processing flow for each TIP type<br>• Pre-defined error injection mechanisms |
| Full Protocol Functionality & Coverage | • Yes, for each TIP type<br>• Each TIP type has a configuration object |

**Table 2: TIP-Based PSV System Feature Summary**

## 2. HDMI Test IP Implementation

This section details the implementation flow for the HDMI TIP, both hardware and embedded coding. Most of the issues were with the hardware implementation, but some lessons learned with embedded software are discussed.

### *Hardware Development*

Test Evolution developed HDMI TX Test IP and HDMI RX Test IP on a Synopsys HAPS-62 High-Performance Prototyping System. The HAPS-62 contains two Xilinx Virtex-6 XC6VLX760 FPGA's. A block diagram showing one FPGA and HAPS-6 resources and interfaces is shown in the block diagram in Figure 3 below.

With two FPGA's, Test Evolution was able to implement HDMI TX and HDMI RX Test IP on a single HAPS-62 platform. Synopsys HDMI TX and HDMI RX PHY cards connected to the

HAPS via HapsTrak II connectors and PHY signals were routed on board to the FPGA's. The Test PC used to configure and control the Test IP connected to the HAPS CDE connectors and communicated with Test IP over a UMR bus. An HDMI cable connected from the TX PHY to the RX PHY provided for end to end HDMI protocol testing.

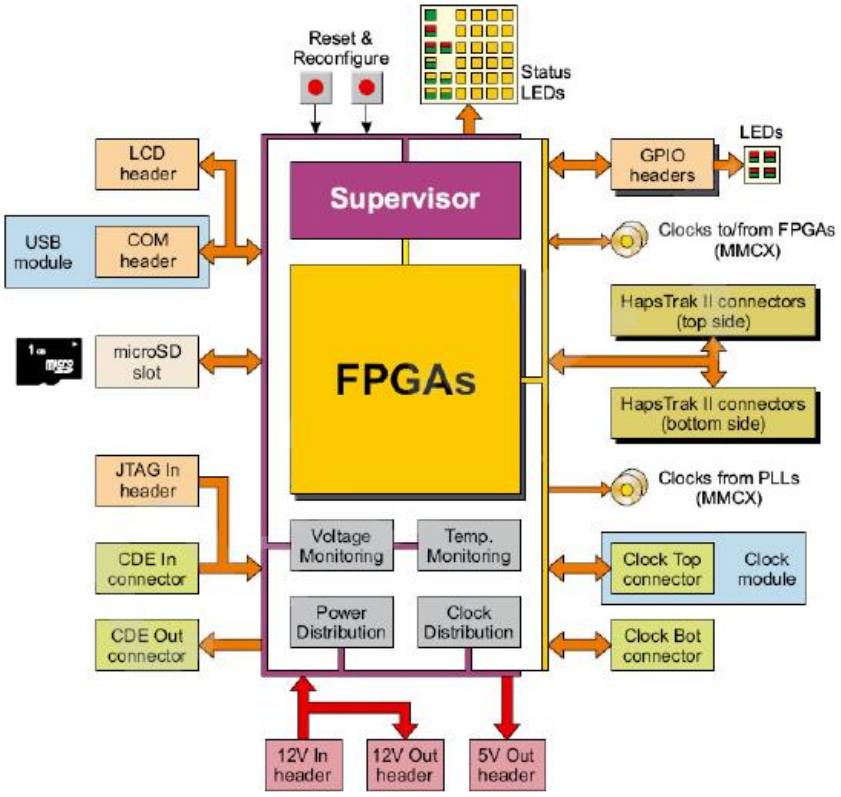


**Figure 3: Synopsys HAPS-62 Development Platform**

A block diagram of the HDMI Test IP is shown in Figure 4 below. A 27MHz clock supplied by the HAPS board provided the reference clock used to generate the pixel clock frequencies needed for the supported video formats. Pixel clock frequencies ranged from 27 MHz to 297 MHz with multiple frequencies in between.

A Xilinx Digital Clock Manager (DCM) was used to generate all pixel clock frquencies from the 27 MHz reference clock. Through the DCM's dyanmic reconfiguration port, embedded software running on the ARC 625 processor had access to the DCM's frequency synthesis elements and could change the frequency as needed when cycling through video formats.

A number of Synopsys tools were used to generate IP for the Test IP design. The HDMI 1.4b Controllers, both TX and RX, were generated using CoreConsultant. CoreConsultant allowed Test Evolution to customized features and parameters of the IP and output Verilog RTL. Core Assembler was used to generate DesignWare components for all the AXI4 structures and bus

interconnect and easily stitch the DesignWare components together. ARChitect2 was used to generate the ARC 625 embedded processor and configure processor resources including data and instruction memory.

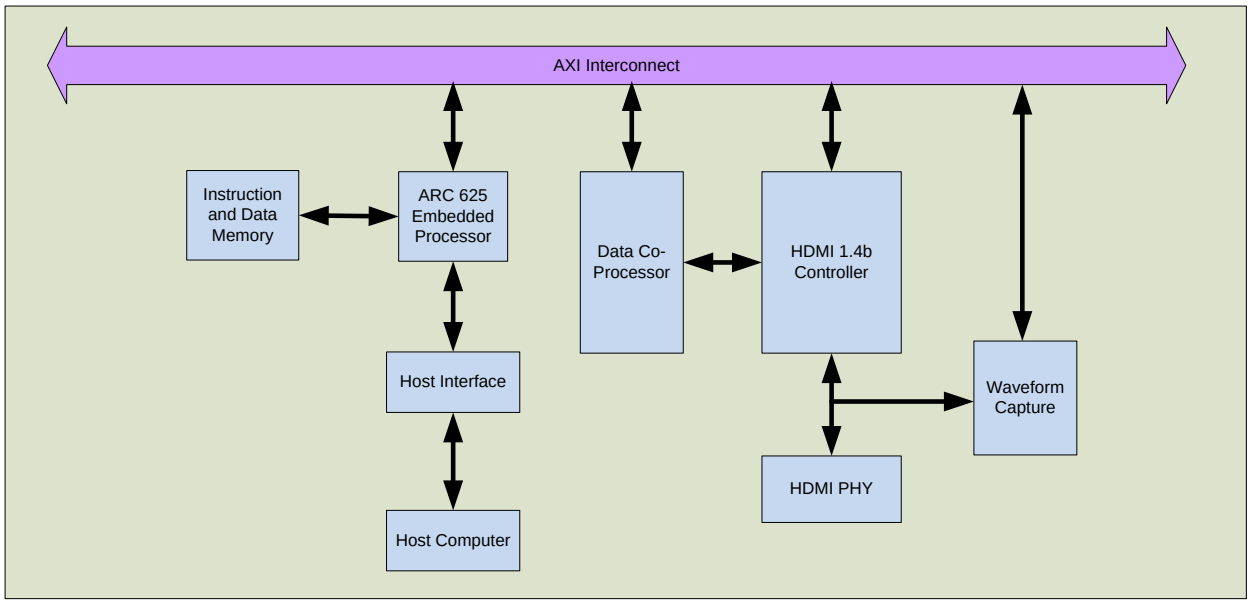


**Figure 4: Test Evolution HDMI Test IP**

Synopsys Synplify Premier was used for design synthesis and Xilinx ISE for design impementation and bitfile generation. The Test IP Data Co-Processor and Test PC interfaces were developed by Test Evolution using Verilog RTL.

The Virtex-6 FPGA's provided more than enough logic, memory, and IO resources for this project, although the design utilized almost 70% of the Virtex-6 block RAMs, mostly for ARC processor memory and Data Co-Processor Tile Memory. The challenge was meeting the timing requirements to support 4K Ultra HD video format (3840 pixels wide by 2160 pixels tall) which required the pixel clock to run at 297 MHz.

The major timing issues in the design involved the HDMI 1.4b IIP as well as the Test Evolution custom Data Co-processor module. The Data Co-Processor is responsible for providing source video and audio data to the HDMI 1.4b IIP. The following description of the steps taken to achieve timing closure refers to the HDMI TX TIP, however, the same approach was used on the HDMI RX TIP.

At the synthesis stage, Synplify Premier provides several implementation options that effect design synthesis and ultimately design performance. Best results were achieved by turning off resource sharing since area was not a major concern, and turning on retiming to enable register balancing, as well as flattening the design hierarchy to allow optimization across hierarchical boundaries. The syn_maxfan and syn_dspstyle attributes were also used to limit the fanout on

troublesome nets and control the use of dsp stuctures on some critical paths. The FPGA synthesized with these options timed in the 150-160 MHz range after implementation, far short of the 297 MHz target.

The Synopsys HDMI 1.4b IIP had six different asynchronous clock domains as configured for the HDMI Test IP project, and many multi-cycle clock paths. Once the timing and clock domain crossing constraints, and multi-cycle clock paths were added to the synthesis constraints file, initial implementations timed in the 190-200 MHz range.

Timing in the Data Co-Processor also limited overall performance of the design. A fairly large 64K x 48 dual port memory operating in the pixel clock domain was the critical path. The memory required 192 16Kb Virtex-6 block RAMs to implement. The large number of block RAMs needed to implement the memory were physically located over a large area of the FPGA, resulting in long routing delays.

The Xilinx CORE Generator tool was used to generate the block RAMs. Significant timing improvement was acheived by enabling the register at the output of the memory primitives as well as the register at the output of the memory core (which gets implemented in the FPGA fabric). After implementing the pipeline registers on the block RAM outputs, the Data Co-Processor met the 297 MHz timing target.

Initially, the Synopsys HDMI 1.4b IIP was fully configured with all features enable. Since RGB to/from YCC color conversions and pixel repetition features were not needed for this project, the IIP was reconfigured to exclude these features with the expectation that the reduced complexity, especially in the color conversion logic alone would offer an improvement in performance. Without color conversion and pixel repetition in the IIP, performance improved to 260-265 MHz.

At this point, Test Evolution turned to the Xilinx SmartExplorer feature. SmartExplorer automatically explores multiple pre-defined design strategies in order to achieve design goals, in this case, to achieve timing closure. The built-in TimingPerformance strategy produced the best results when run 100 times overnight with different placer cost tables. SmartExplorer provides the ability to run multiple iterations iterations in parallel. The run time was 13-14 hours for 100 iterations when running 4 iterations in parallel.

Although SmartExplorer offered significant timing improvements, the 297 MHz target operating frequency was not achieved in the Virtex-6 technology (295 MHz was reached on the HDMI TX TIP and 289 MHz was reached on the HDMI RX TIP), the experience provided Test Evolution with the confidence that timing closure could be achieved on the target platform, which used Virtex-7 technology. Figure 5 below illustrates the timing progression from 150 MHz out of the box to 295 MHz on the HDMI TX TIP, and the strategies used to get there.

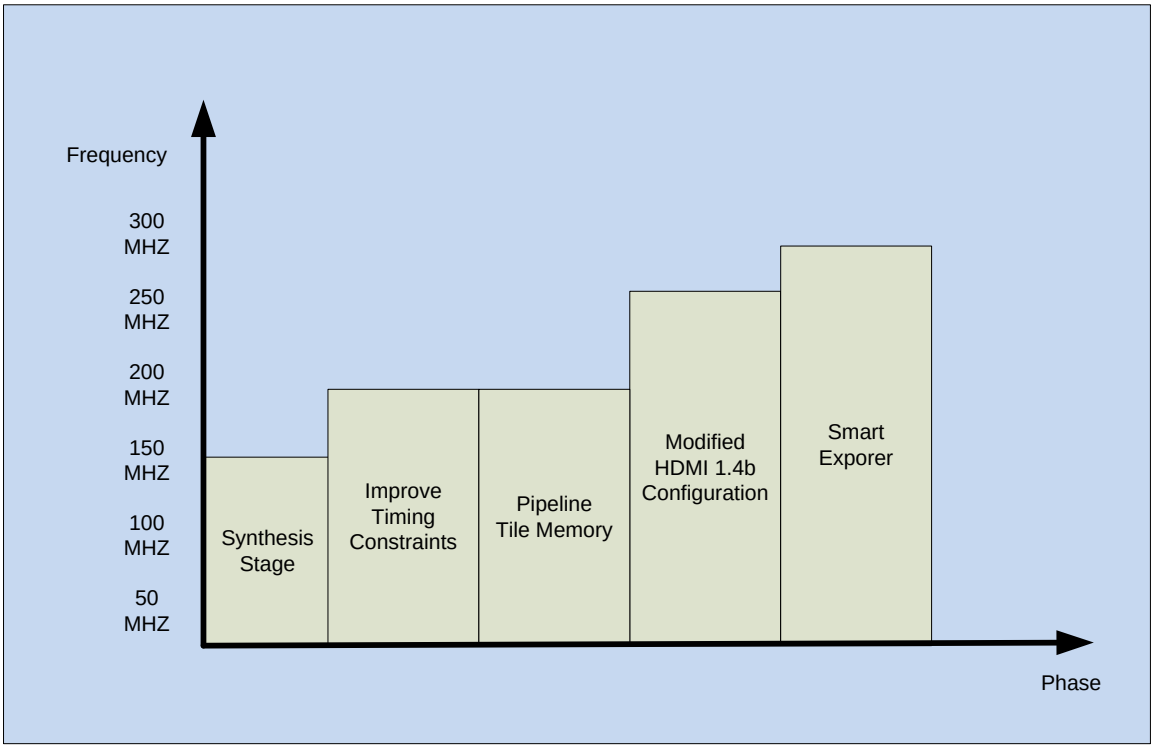


**Figure 5: HDMI TIP Timing Closure**

### *Embedded Software Development*

The main reason to instantiate an embedded ARC625 processor in the design was to minimize development time and cost. Empirical data suggests the following rule: for complex systems RTL implementation takes 5x more time/resources than embedded SW implementation, and embedded software takes 5x more time/resources than non-embedded SW.

So, the design goal was to shift as much as possible (traffic generation, pre- and postprocessing) to the host SW and embedded SW, and to perform the rest in the RTL. The main challenges for this approach were:

1. ARC processor speed was limited to 50 MHz.
2. Absence of external DDR meant that the amount of memory available to the processor is significantly limited (256Kb in our design)
3. UMR bus that was used to communicate between the host PC and embedded system is tricky to implement right in the RTL

For example, due to the processor speed limitations, embedded SW was unable to receive and process per-line interrupts even in 1080p@60 mode, and memory limitations didn't allow us to store a full frame in the memory. To resolve this, real-time compare logic was implemented in the RTL and only mismatched data is saved in the memory and later used by embedded and host SW to reconstruct received frame.

The communication between host SW and embedded SW using UMR bus ended up being more complex task than originally thought due to RTL complexity and insufficient amount of documentation, so this sould be taken into account during planning and estimation.

The software was developed with MetaWare Development toolkit, and the MQX RTOS was utilized to provide RTOS services. Even though the application didn't use multithreading, other services provided by MQX ended up being very useful, like interrupt handling that was matched to HDMI events.

## 3. HDMI Test IP Application

This section will cover the application of Test Evolution's Test IP (TIP) based Post Silicon Validation (PSV) platform to the validation of a leading edge set top box & smart TV platform IC. The validation tasks included pre-tape out characterization of a 28nm HDMI receiver PHY to be used as part of the Video Capture (VCAP) logic of the Multi-Media Subsystem (MMS). Once 1st silicon was available, the TIP-based PSV platform was used for functional validation of the VCAP and surrounding logic.

Figure 6 below illustrates the 28nm PHY test setup. HDMI Source (or TX) TIP drives the HDMI PHY. HDMI Sink (or RX) TIP receives PHY interface signals from the test chip. Note that the HDMI TIP can be configured with or without the PHY, and in this case the RX TIP receives the IC-level interface. The TIP and PHY modules all reside on the HAPS62 board. The Test PC connects to both HDMI TX and RX TIP and controls configuration, and real-time traffic generation and pixel compare via a Visual Studio test program. The goals of the deployment were to validate the TIP-based PSV platform, and the power of this new testing paradigm.

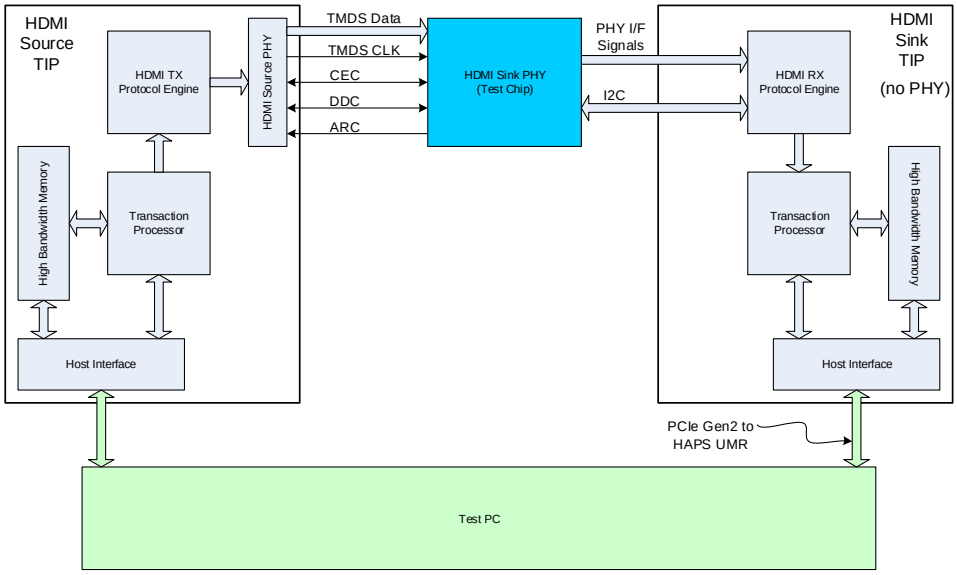


**Figure 6: 28nm PHY Characterization with TIP**

Based on PHY test requirements, the TIP-based PSV system looped through all video modes while sending a constant audio tone, and constantly compared the PHY output frame by frame, pixel by pixel, capturing and reporting error history. Repeatable tests with random video data

were quickly created, and overnight runs done across error cases with revi-1 silicon, showing no errors in rev2 silicon. Week-long regression runs were also done with error tracking, showing PHY stability prior to tape-out. The testing done with the TIP-based PSV system would have been difficult to impossible to do with existing protocol box instruments, allowing quicker test plan closure with more coverage. In addition, the new 4K/60Hz/4:2:0 mode was used, which was not available with other test equipment.

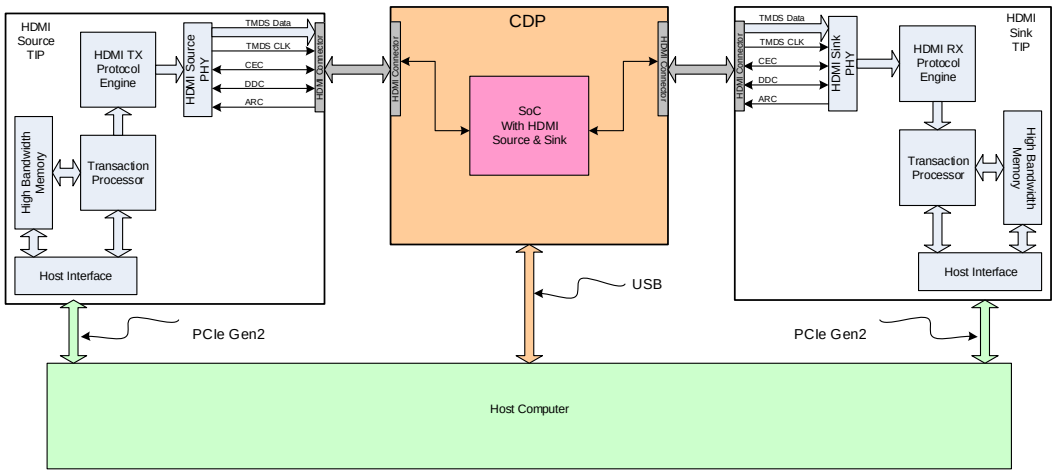


**Figure 7: CDP with TIP-Based PSV Platform**

Once 1st silicon was available, and a Characterization Debug Platform (CDP) board was allocated for TIP-based testing of the VCAP, a test setup was created as shown in Figure 7 above.

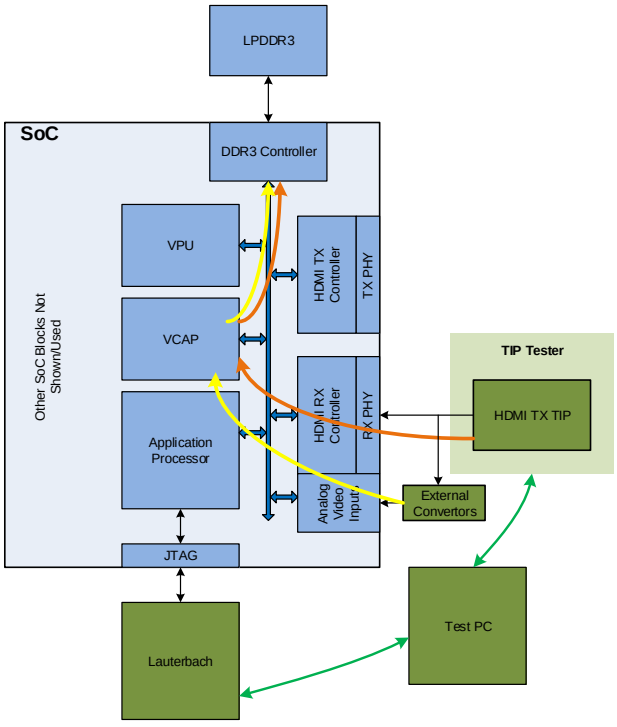


**Figure 8: SoC PSV Setup**

Here, the HDMI TX TIP was setup to drive video and audio into the CDP via an HDMI cable, and the HDMI RX TIP was setup to capture output video via an HDMI cable. As with the PHY testing, the Test PC controlled configuration, and real-time traffic generation and pixel compare via a Visual Studio test program. In addition, the TIP API was developed to match the pre-silicon verification environment, allowing portability between pre- and post-silicon validation.

In addition, looking at Figure 8 to the left, the Test PC was also connected to the Lauterbach debugger, which was fully integrated into the Visual Studio program, and was used to interface into the On-chip DSS (Debug Sub-System). Finally, external hardware was used to convert the HDMI stream into either Composite,

Component, or VGA video and driven into the Analog Front End (AFE) inputs. This allowed testing all VCAP inputs.

HDMI TX TIP was used to drive both a color wheel and random traffic into the HDMI input and the AFE across all modes. The DDR memory was used to capture the data, and compare pixel by pixel with expected results, for a push-button end to end test. Both VCAP channels were used, in some cases with HDMI and AFE video, in other cases with HDMI video on both channels. Regression runs were done overnight across HDMI, CEA, and VESA modes, along with focused stress cases like 4K video on both channels, and with 25' and 50' HDMI cables.

The portion of the program used to walk though all HDMI video modes with random frame generation is shown below:

```
for (UINT formatIndex = 0 ; allTestFormats[formatIndex] != HDMI::FMT_NOT_VALID ;
          formatIndex++)
{
 HDMI::VideoFormat format = allTestFormats[formatIndex];
 RandomizeFrameData(tileVsize, tileHsize, tileMemory);
 SetupFormatInDebugger(hsize, vsize, bytesPerPixel);
 VideoTiledFormatTest(env, hdmiTx, format, HDMI::RGB_4_4_4, HDMI::COLOR_DEPTH_8,
                      summaryFileName, true, tileVsize, tileHsize, tileMemory);
 ReadMemoryInDebugger(address, buffer, readSize);
 badPixels = CheckFrame(format, hsize, vsize, bytesPerPixel, buffer, bufferSize,
                        tileVsize, tileHsize, tileMemory, summaryFileName);
}
```

The bulk of the test code is in the `VideoTiledFormatTest()` function, shown below:

```
HDMIFrameSettings frameSettings;
frameSettings.SetEncoding(encoding);
frameSettings.SetColorDepth(HDMI::COLOR_DEPTH_8);
frameSettings.SetVideoFormat(format);

HDMITileFrameFactory frameFactory(frameSettings);
frameFactory.SetupTile(hsizeTile, vsizeTile);
hdmiTx.SendFrames(displayTime, frameFactory);
```

Like in the pre-silicon world, the test code is object oriented, and at a transaction level. There are constraints and stimulus generators, like with Verification IP (VIP). This makes generating complex tests much easier in PSV that with traditional protocol boxes.

The work was done over 4 days, with some cleanup and miscellaneous runs done over a 3 day period. There were numerous issues found, shown in Table 3 to the right. The first 9 issues shown in the table were deemed driver issues. Since the DSS and scripts were being used instead of real bare-metal SW drivers, some corner cases issues were either found and fixed, or were looked at by designers and "blessed."

The CDP Board issue was found before the TIP-based PSV was deployed, and was discovered again and characterized. The CDP board that was used for TIP-based testing was not updated.

| Issues Found | Type of Issue | Status |
|---|---|---|
| 7 | Potential SW Driver problem | Some are definitely SW driver issues. Others need further investigation. |
| 2 | Test site setup | AFE related. Impacts automatic testing. |
| 2 | PHY | Two runs were done. Both indicating silicon issues at 297MHz pixel rates. |
| 1 | CDP Board | This is a known issue found independently. TIP system was used for more characterization of noise, showing ~25%. |
| 4 | Unknown | These are related to mode switching and dual-channel throughput into and out of DDR. |

**Table 3: PSV Issues Found**

There were also 4 unknown errors that were and could be repeated, but when the CDP was completely reset, they disappeared. These were likely either script corner cases that caused the MMS to get into a strange state, or were corner case issues. Either way, more investigation was needed to close them. Finally, there were 2 PHY issues found using 50' cables and overnight regressions that showed a statistically significant number of pixel errors across various video modes that required a 297MHz pixel clock rate into the HDMI receiver PHY on the SoC. The results are graphically shown in Figure 9 below.

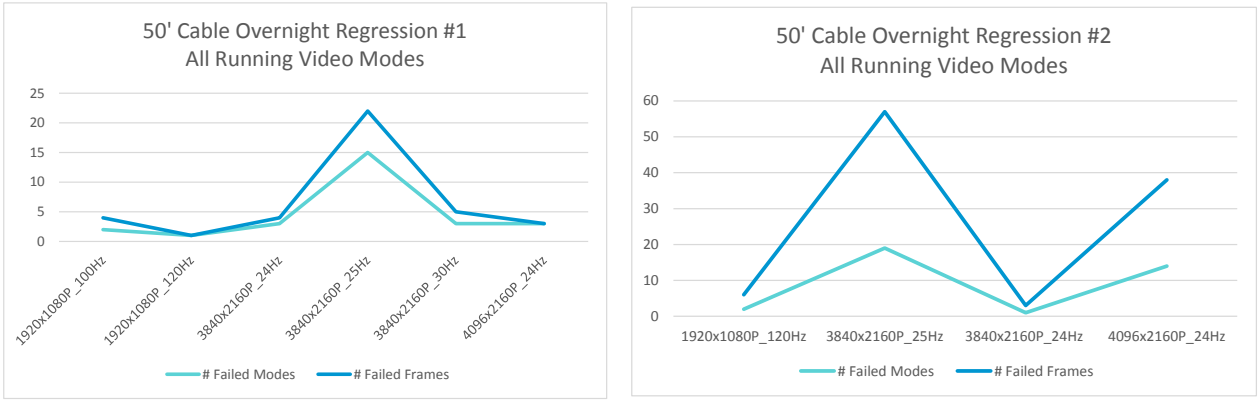


**Figure 9: 50' Cable Regression Results Showing Pixel Errors**

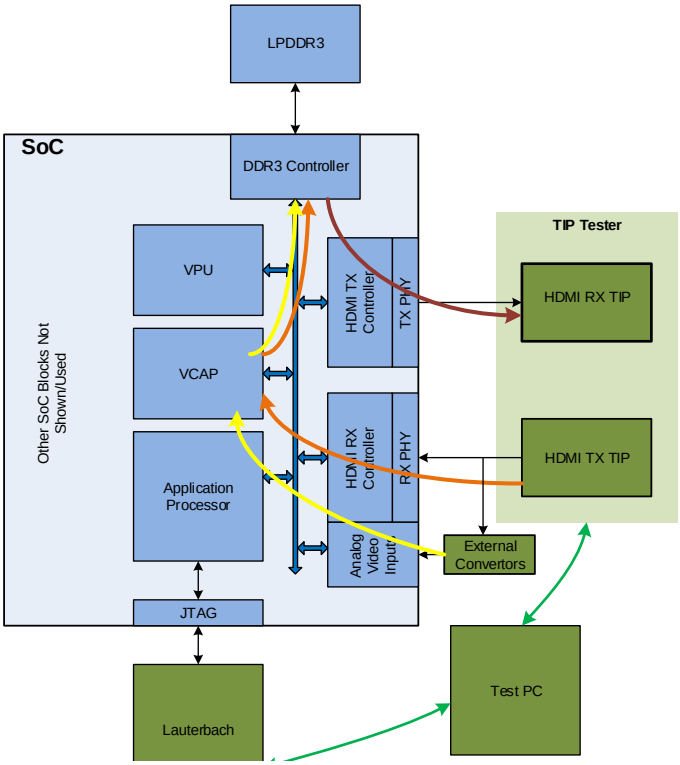


**Figure 10: TIP PSV End to End Testing**

The failures were randomly distributed over 14 hours of testing. These were deemed silicon issues. Regression #1 was run with typical silicon. Regression #2 was run with slow silicon, and the graph shows more errors with slow silicon. It's important to note that this is an issue that would be difficult to find via a production ATE tester, as it requires full functional testing using the PHY at speed.

After the general PSV work on the SoC, a proof of concept was done, using the end to end configuration shown in Figure 10 at left. Here, HDMI traffic is sent via HDMI TX TIP, goes thru VCAP and into DDR. The Multi-media Display Sub-System (MDSS) then takes the video and outputs it to the HDMI RX TIP. A full end to end test case was run at 1080p resolution, with AFE video also running on a separate channel into DDR. The HDMI TX TIP ran with random data, and the HDMI RX TIP was setup to continuously compare the received data to the random expected data over a number of hours, checking for pixel errors over a large number of frames.

To add the end to end compare, only the addition of the HDMI RX setup needs to be added to the `VideoTiledFormatTest()` function, shown below:

```
HDMIFrameSettings frameSettings;
frameSettings.SetEncoding(encoding);
frameSettings.SetColorDepth(HDMI::COLOR_DEPTH_8);
frameSettings.SetVideoFormat(format);

HDMITileFrameFactory frameFactory(frameSettings);
frameFactory.SetupTile(hsizeTile, vsizeTile);
hdmiTx.SendFrames(displayTime, frameFactory);

HDMITileFrameFactory frameFactory(frameSettings);
frameFactory.SetupTile(tileData, hSize, vSize);
hdmiRx.StartCompare(compareStrategy, frameFactory);

totalMiscompares = compareStrategy.GetResult(hdmiRx, compareHandle);
```

1080p was the only mode supported at the time, but the proof of concept showed that with end to end testing, the time to run thru the whole test suite – estimated at 2-3 minutes – would be 20-30x faster than with the DDR-based compare method – estimated at about 40-60 minutes. In

addition, the functional coverage and closed-loop testing advantage was much greater than the DDR method. This would make limited volume early silicon testing much more efficient, on top of the benefits of a push-button, fully self-checking test suite.

Table 4 below compares the original goals for the 4 day period, versus what was actually accomplished.

| Goals | Accomplished |
|---|---|
| Configure a CDP board to use the HDMI, Video AFE inputs in standalone mode. | Integrated TIP HW, SW, and Lauterbach in a Visual Studio test program with CDP board for push-button testing. |
| Configure TIP system to drive an HDMI splitter and converter in standalone mode. | Configured TIP system to drive HDMI into SoC and convertors for the AFE, and wrote self-checking test code for pixel-level comparisons. |
| Understand the work and effort involved to incorporate TIP with CDP into an effective limited volume early production test system | Scoped the work and effort involved to incorporate TIP with SoC CDP into an effective early production test system based on work done the past week. |
| | Found 15 overall issues, from low to high priority, that require investigation. Some unexplained errors with concurrent video streams into/out of DDR, and silicon PHY issues. |
| | Ran focused nightly regressions that were difficult to do with existing test equipment, and found issues. |
| | Quickly confirmed existing issue with VGA noise, and characterized. |

**Table 4: Original Goals vs Accomplished**

## 4. Conclusions

In summary, applying the TIP-based PSV platform to the SoC, both for pre-silicon PHY characterization and PSV of the VCAP and MMS, demonstrated the following:

- A platform that runs real-time across multiple protocols.
- The ability to easily create robust tests that are repeatable and self-checking.
- The ability to craft tests to find issues quickly.
- The ability to run tests overnight or over a week, with end to end self-checking and error history.
- Quick discovery & characterization of silicon issues that would be difficult to find with existing test equipment, like protocol analyzers.
- Extending the PSV work to concurrent testing would be generally straightforward.

Moving forward, the TIP-based PSV test platform:

- Will scale across multiple TIP instances – MIPI DSI, MIPI CSI-2, USB 2/3x, UFS – with native support for synchronizing TIP instances, and for debugging the DUT across protocol interfaces.
- Can be used across PSV groups, from interface to subsystem to full chip, leveraging tests and methodology across groups.

- Will reduce the cost of PSV testing by both integrating functionality that is limited in existing test equipment, and providing in one platform functionality that would require multiple pieces of existing test equipment integrated in an ad-hoc way.
- Will reduce the time it takes to both find bugs in silicon, and to debug the silicon, with both repeatable tests and integrated graphical debugging across all protocols.

# 5. References


[1] *Post-Silicon Validation in the Era of Multibillion Transistor Chips and the Compute Continuum From Phones to High Reliability Servers*, DAC 2014 Tutorial. Kevin Reick - IBM Corp., Round Rock, TX, William Lindsay - Intel Corp., Folsom, CA.

[2] *How Much Testing is Enough?* Semiconductor Engineering, May 2014. Mark LaPedus.