An Efficient Verification Framework for Audio/Video Interface Protocols

Noha Shaarawy[1], Mustafa Khairallah, Khaled Khalifa, Hany Salah, Amr Salah and Maged Ghoneima[2]

[1]Boost Valley, Cairo 11361, Egypt,
noha.shaarawy@boostvalley.com,
[2]Ain Shams University, Egypt
m_ghoneima@ieee.org,

***Abstract*—Audio/video interface protocols, such as High Definition Multimedia Interface (HDMI) and DisplayPort, are major components of today's Entertainment products. Given the shrinking time-to-market and the increasing complexity of System-On-Chips (SoCs), verification teams need to optimize the verification techniques used to test new audio/video interfaces. This paper proposes a unified framework for building robust, configurable, and extendible verification environments suitable for different audio/video interface protocols. It suggests customizable testing mechanisms that can be used to test different profiles. The proposed framework is based on Universal Verification Methodology (UVM) yet can easily fit into other verification methodologies such as Verification Methodology Manual (VMM). The paper shows how the proposed framework is used to ease the development of HDMI transmitter and receiver Verification Intellectual Properties (VIPs), alongside with the testing mechanisms applied to verify the corresponding HDMI transmitter/receiver Design Under Tests (DUTs), highlighting the amount of reuse between the two VIPs.**

***Keywords*—Video, Interface Protocols, Functional Verification, HDMI, DVI, DisplayPort**

I. Introduction

Consumer electronics and entertainment products/devices have evolved over the past years to be complex systems. A typical scenario for entertainment devices nowadays is to operate in an environment that is constantly changing [8]. In addition, the quality of the entertainment content increases with passing days. Recently, 4K ultra-HD videos with multi-channel and multi-stream have become commercial [6].

Audio/video interfaces are major building blocks of these products. HDMI and DisplayPort interfaces are the most commonly used interfaces in consumer electronics. However, designing and verifying such interfaces represent a challenge. For example, video protocols verification requires the ability to deal with huge video frames at different protocol layers; in multiple streams and in complex structures [7]. The verification target is to minimize test run-time and memory consumption, while verifying the design at different levels; block, sub-system and SoC levels.

To express the verification challenge of the state of the art High Definition Television (HDTV) SoCs, the authors of [6] presented a challenging verification problem that was faced during the verification of new versions of Analog and Digital TV chips. Reuse and extension of previously developed verification environment were applied. The first verification project took 18 months from the development of specifications till the tape-out. The verification team consisted of 19 engineers utilizing 20 machines. 30 components within the chip were verified during this project. The problem arose when in a subsequent project, it was requested from a downsized team (team shrank by 47%), only 10 engineers were available, and a downsized number of machines (machines availability shrank by 20%), to verify a new TV chip that included 24 new components within a shorter time frame which was 10 months. The area of the whole chip increased by 50-65% and included 5 new interfaces and a total of 12 interfaces. HDMI and Transport Stream interfaces were among these interfaces.

A unified verification framework for audio/video interface protocols is proposed in this paper. The proposed framework helps build robust, configurable and extendible verification environments. In section II, the related work concerning audio/video interface protocols verification is discussed. The proposed verification framework is discussed thoroughly in section III. An application example is presented in section IV and an experimental results are shown in section V. Finally, the paper is concluded in section VI.

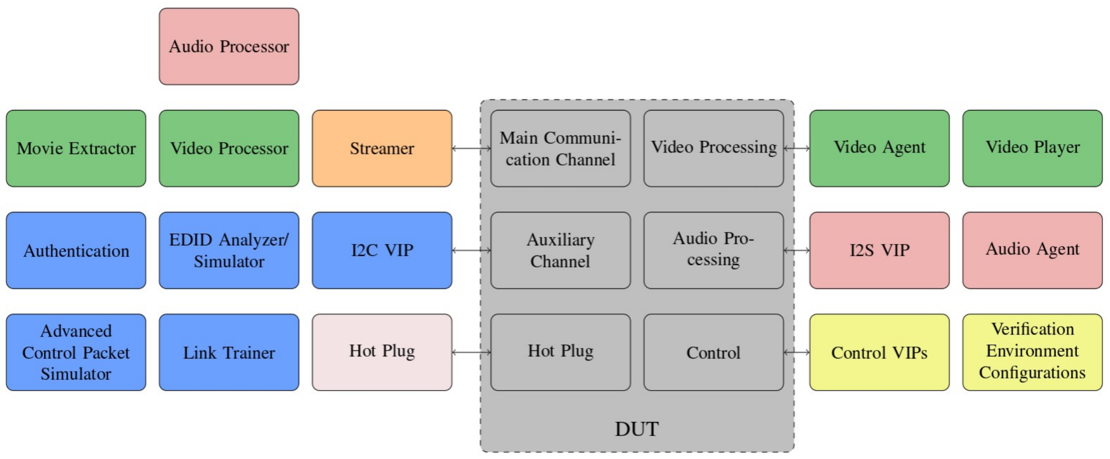


Figure 1: Verification Framework Overview

II. RELATED WORK

To the best of our knowledge, very few attempts have been made towards developing a configurable, and an extendible verification framework applicable on different audio/video interface protocols. In [5], the authors discussed a functional verification environment methodology for audio/video SoC. Usage of Emulation technology to accelerate verification for maximum performance was introduced. HDMI 1.2 protocol, I2S, S/PDIF and 32-bit linear Pulse Code Modulation(PCM) audio protocols as well as several video formats; were covered. However, discussion on any configuration or extension mechanisms to other standards and protocols; wasn't part of [5]. In [9], the authors presented a configurable testbench component that can drive audio/video streams for an HDMI or Mobile High-Definition Link (MHL) receiver. However, these efforts cannot be considered as a general framework as it did not discuss thoroughly many of the verification environment components nor the test generation mechanisms. Besides, the only checking mechanism used was black box end-to-end checking.

III. PROPOSED VERIFICATION FRAMEWORK

A typical DUT that implements a video interface protocol can be divided into six regions; three communication interfaces: 1) Main Communication Channel, 2) Auxiliary Channel and 3) Hot Plug and three protocol-independent blocks: 4) Video Processing, 5) Audio Processing and 6) Control. Thus, the proposed verification framework divides the verification IP components into six similar regions: Streaming, Auxiliary Channel, Hot Plug, Video, Audio and Control regions. The DUT regions and the Verification IP are presented in Figure 1; color coding in the figure represents the regions.

A. *Streaming Region*

Streamers are responsible for preparing the appropriate bit-stream sent over the main communication channel of the interface in question. They are also responsible for decoding the bit-stream and monitoring the interface activity. The function of a streamer is defined in the proposed framework though its implementation depends entirely on the DUT; as different interface protocols use different bus protocols. HDMI and Digital Video Interface (DVI) protocols use Transition-Minimal Differential Signaling (TMDS) protocol, while DisplayPort uses Micro-Packet protocol.

B. *Auxiliary Channel Region*

All modern video interface protocols use an auxiliary communication channel for authentication, control and configuration. The channel functions are:

- To act as the Display Data Channel (DDC) carrying Extended Display Identification Data (EDID) packets.

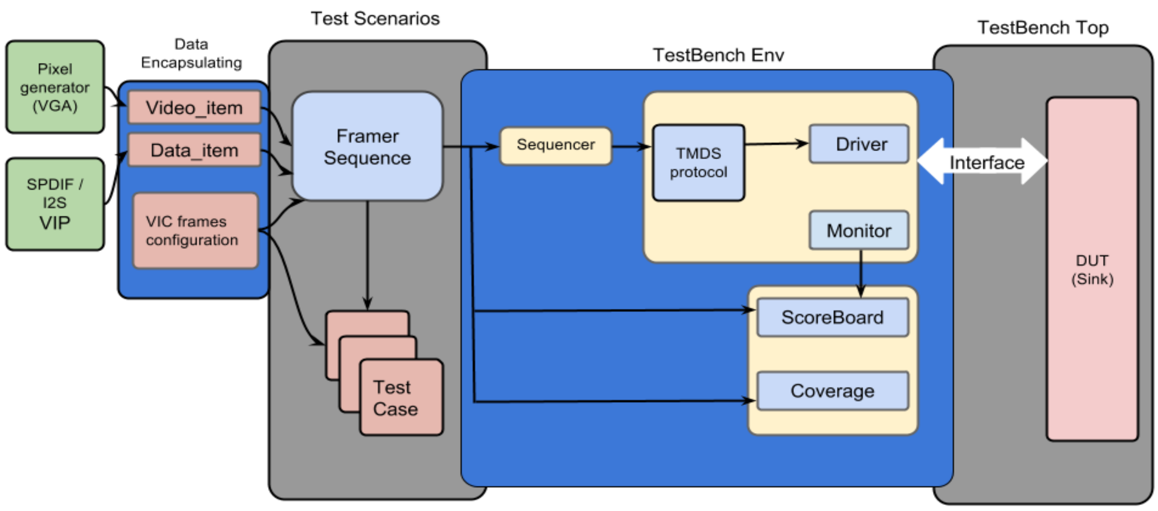


Figure 2: Verification Environment for an HDMI Receiver

- To perform High-bandwidth Digital Content Protection (HDCP) Authentication.
- To carry special control packets defined by different video interface protocols.
- Link training as specified in the DisplayPort specification.

  The Auxiliary Channel is a two wire channel that either uses I2C protocol, as in HDMI and DVI; or uses a special bus protocol defined by the interface protocol specification, such as the auxiliary channel protocol defined by the DisplayPort specification. The channel is represented in the verification IP by the following components:

- I2C Verification IP.
- Auxiliary Channel Verification IP (for DisplayPort only).
- EDID Simulator: responsible for preparing the appropriate EDID packets to be sent to a source interface (DUT).
- EDID Analyzer: responsible for analyzing the EDID packets received from a sink interface (DUT), then verifying their conformance to the standard and sending their content to the rest of the verification IP components to adjust the video stream accordingly.
- Advanced Control Packet Simulator: responsible for sending or receiving the advanced control packets defined by different interface protocols, for example: Status and Control Data Channel packets defined by HDMI 2.0.
- Link Trainer (for DisplayPort only).
- Authentication Agent: responsible for performing the HDCP authentication operation between the interface source and sink, or between the DUT and the verification IP.

C. *Hot Plug Region*

Responsible for the hot plug behavior defined by different interface protocols. It is also responsible for testing scenarios such as the DUT response when the transmission cable is torn or when the sink is powered off. It includes:

- Hot Plug Monitor: responsible for raising the power-on signal upon the user' request and monitoring the hot plug signal. It informs the verification environment when the hot plug signal is asserted.

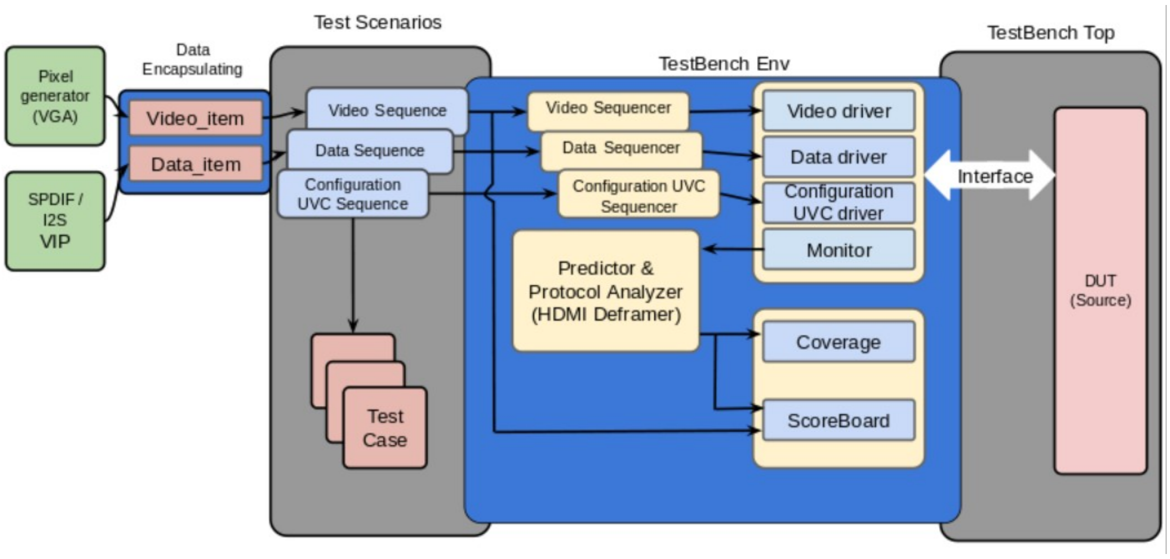


Figure 3: Verification Environment for an HDMI Transmitter

- Hot Plug Driver: responsible for monitoring the power-on signal and responding with the hot plug signal. The user can choose whether the hot plug signal should be asserted normally in response for the RX5V or enforce a de-assertion.

D. Protocol Independent Region

1) Video Agent

This group of components is responsible for all video-related operations. It includes:

a) Application Layer:

- Movie Extractor: It takes a short movie and converts it into binary frames and audio samples that are processed by the video and audio processors. These frames and samples are used by the testbench to generate real-world test-cases.
- Video Player: It is an optional block that compares the input versus output video frames and audio samples.

b) Video Processor: It is responsible for processing the video data stream; by performing a group of video processing functions such as color space conversion or video pixel encoding conversion ...etc.

2) Audio Agent

This group of components is responsible for all audio-related operations. It includes:

- I2S Verification IP.
- S/PDIF Verification IP.
- Direct Stream Digital audio (DSD) Verification IP.
- Audio Processor: responsible for processing the audio data stream; by performing the following functions:

a) Generating specific parameters related to the audio clock regeneration.

b) Saving audio configuration information.

- Custom Audio Verification IPs.

E. Control Region

This group of components is responsible for providing the means to control the DUT and configure its behavior according to the test scenario in-hand. Different DUTs may provide different control interfaces, for example: I2C, JTAG and SPI.

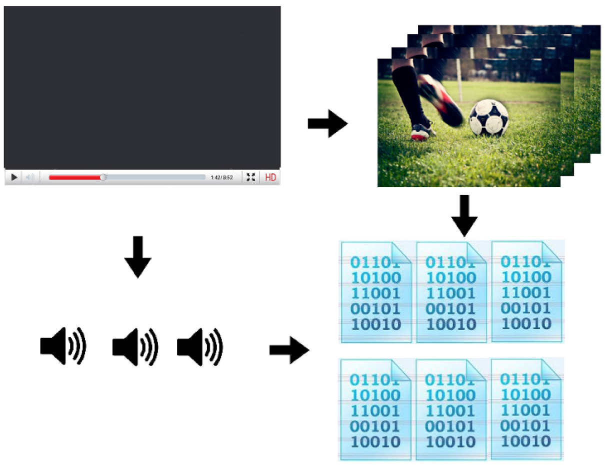


Figure 4: Audio/video extraction tool

F. Verification Environment Configurations

Since high definition video frames have huge sizes, which consumes a lot of resources both in terms of memory space and run-time, the VIP should provide customizable frame formats to help users verify the basic functionality using smaller frames and less simulation cycles. Other possible configurations for the verification environment are video formats, pixel encoding, color depths, audio types, audio sampling frequencies, scrambling enable/disable and HDCP enable/disable.

IV. HDMI 2.0 Verification environment

HDMI is a compact audio/video interface for transferring uncompressed video data and compressed or uncompressed digital audio data from an HDMI-compliant source device, e.g. DVD or Blu-ray, to a compatible sink device, e.g. computer monitor, video projector, digital television, or digital audio device [4].

After the framework base architecture, described in section 3, has been developed, it has been applied to develop two verification environments emulating and testing the HDMI receiver and transmitter functions. Figures 2 and 3 represent the two verification environments. These environments are developed using UVM, where core functionalities are coded in pure System Verilog, so as to be easily used inside any testbench design using any verification methodology. For this paper, the communication between testbench components is handled using UVM.

The testbench incorporating the components described in section 3 is divided into three regions; data encapsulating components, testing scenarios components and testbench environment components. This additional classification helped reuse different components in two different environments. For example, the data encapsulation classes and components are typical in both environments presented this paper and can be directly used in any audio/video protocol verification environment, not necessarily HDMI (e.g. DisplayPort, MHL ...etc). Additionally, the test-cases developed are the same for any audio/video protocol, whether the DUT is a transmitter or a receiver, as the test-cases are simply audio/video streams that are afterwards packed according to the appropriate protocol.

G. Data Encapsulation

The data is encapsulated inside the testbench in protocol independent classes, such as: video_item (which holds active video pixels), data_item (which holds video infoFrames and audio samples), vic_configuration (Video Identification Code that is used to configure the stream into one of the video formats standards). These classes help build customizable and reusable test-benches, independent of the DUT interface or the protocol tested, whether it is a transmitter or a receiver.

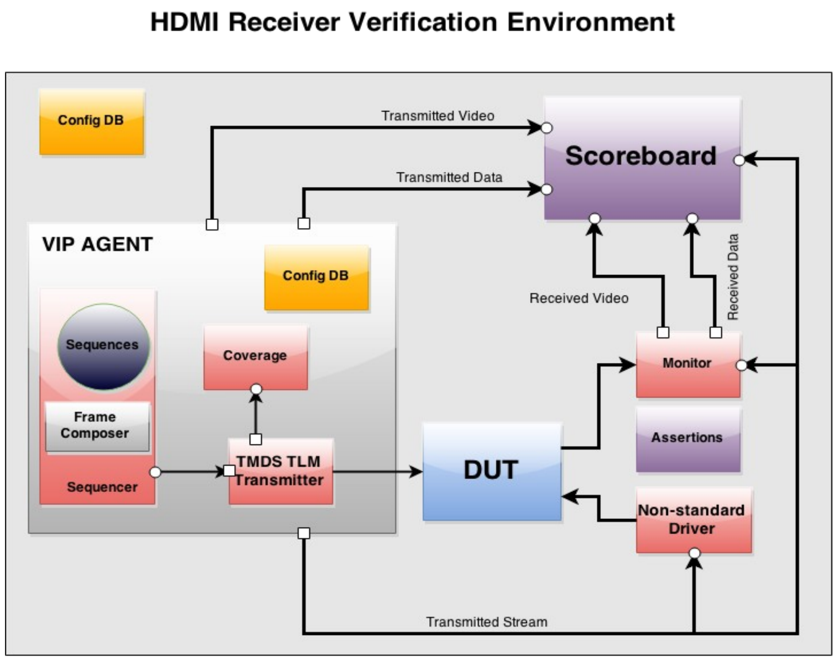


Figure 5: Different TLM connections from different components to the scoreboard

H. Test Scenario

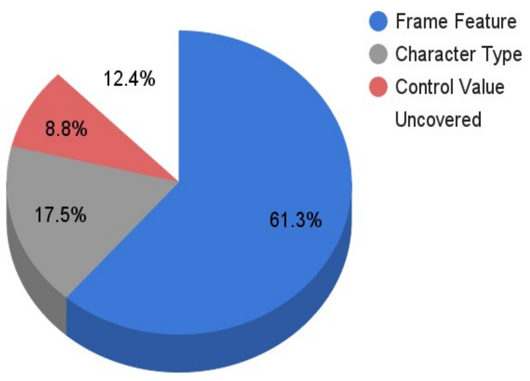
The test scenarios used to test the HDMI DUTs are generated using two different approaches which can be both used for any audio/video protocol. The first approach is to generate protocol compliant video frames, with random video and audio data. This approach allowed discovering and fixing bugs that are protocol specific, such as the video frames formats being sent ...etc. The second approach is to use real-life test cases through a movie extraction software tool that takes as input real videos and coverts them into audio samples and images. The movie extractor, shown in Figure 4, is a simple C++ software program that use open source applications such as:

- Ffmpeg [3]: Converts a real video to a group of frames (pictures) and audio tracks. It is used in our framework to convert the video to JPEG format and audio to either WAV PCM tracks or other compressed formats. It can extract several audio channels within the same video track. The reverse operation also uses Ffmpeg.
- HexDump [1]: Converts the audio tracks and video frames to a hexadecimal representation, and vice versa. The hexadecimal representation is easier to handle in the verification environment.
- ImageMagick [2]: A library of image manipulation APIs. The *"convert"* API is used to convert from the compressed JPEG format to uncompressed RGB format and vice versa.

The way test cases are generated has enabled reusing the test case generators and end-to-end checkers (scoreboards) among different audio video protocols and DUTs. Only protocol analyzers and checkers are protocol specific.

I. Testbench Environment

The testbench environment developed has two types of components: 1) protocol specific components, such as sequencers, drivers and monitors which act as an HDMI transmitter or receiver (deframer), these components are extensions for standard UVM components with protocol specific behavior; and 2) generic reusable components such as end-to-end scoreboards and coverage models. The later is reused between both verification environments for the HDMI transmitter and receiver.



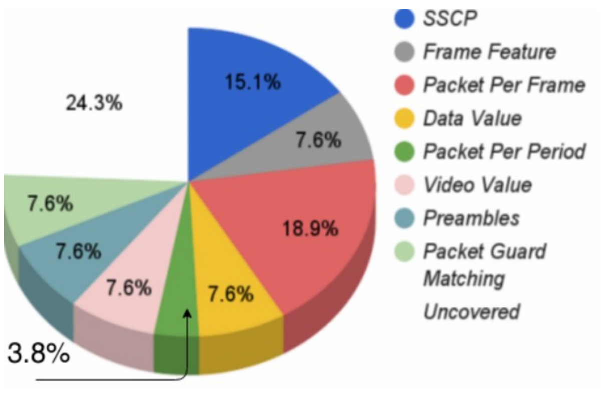


Figure 6: HDMI stimulus coverage result *Figure 7: HDMI receiver coverage result*

a. Protocol Analyzer

The Protocol Analyzer in the proposed environment is protocol specific and could be customized based on the used audio/video protocol. The Protocol Analyzer is designed to be scalable to any number of transmitters. The key to the scalability is the usage of Object Oriented Programming (OOP) concepts built in the System Verilog assertions. In the proposed HDMI verification environment, shown in Figure 3, the Protocol Analyzer is used to verify the HDMI transmitter (DUT) functionality, by checking its compliance with the HDMI protocol rules and specifications. Example for these rules/specs is the HDMI standard restriction on the maximum number of data packets per data island period. The Protocol Analyzer in this case is asserting if the HDMI transmitter (DUT) sends data island period containing data packets more than that allowed maximum number. To verify the HDMI transmitter (DUT) functionality, the Protocol Analyzer performs the functionality of the HDMI Receiver. The protocol analysis occurs at the transaction level. The Protocol Analyzer receives the TMDS outputs of the transmitter DUT through the monitor, then extracts the information needed to translate it into meaningful events and status information. The extracted events and status information from the transmitter DUT are compared with built in protocol rules and specifications of the HDMI.

b. Reusable Scoreboard

As shown in Figure 5, the scoreboard receives different inputs from different testbench components. It can be configured according to the test scenario and testbench architecture to use any of these inputs to verify the DUT and perform end-to-end checking.

c. Reusable Stimulus Coverage Model

A functional coverage model monitors and tracks which parts of the design have been stimulated. For the case of HDMI, two main coverage models were built for transmitter and receiver. The first model, the stimulus coverage model, is connected to the frame composer, shown in Figure 3. Stimulus coverage model is responsible for monitoring the output and confirming that all possible valid frame types were generated for different test configurations. The other coverage model is connected to the protocol analyzer and is responsible for confirming that protocol analyzer can reconstruct all types of frames and all possible video and data values for different test configurations. A generic coding style for coverage components is used for smooth integration of reusable cover groups and cover points.

Cover groups and cover points are reusable across test environments because of the usage of unified enumeration and objects between HDMI transmitter and receiver. Also higher re-usability was achieved by building cover groups based on shared points needed to be covered. For example, a stimulus coverage cover group named "frame feature" that covers frame types and parameters can be reused in any cover model in the test environment because both transmitter and receiver extract frame parameters using same class. Another example is a cover point that covers encryption state transition. It is easily reused as long as the same encryption state enumeration (e.g. enabled, disabled) is used by the transmitter and receiver. Overall coverage is collected using selective tests coverage databases and central database for all tests.

Table 1: Part of the Performance Results of the developed test-cases

Video Format	***CPU Time (in Seconds)***
640*480	248
720*480	270
720*480 (Scrambling Enable)	349

V. Experimental Results

In order to verify the HDMI transmitter and receiver, many test-cases have been developed using the proposed framework. As shown in Figure 6, total stimulus coverage score collected from 107 test cases with 3 weighted cover groups is 88.23%. The cover groups shown in Figure 6 are 1) "frame feature" which covers frame formats parameters crossed with encryption status and scrambling status, 2) "Character type" which covers all valid period type transitions with possible lengths and 3) "Control value" which covers control period values like preambles. The total receiver coverage score collected from 72 test cases with 8 cover groups is 75.52% as shown in Figure 7. The "Frame feature" cover group is reused. Data and video pixel values are covered by "Video Value" and "Data Value" cover groups. Number of data packets per period and per frame including maximum possible number are covered by "Packet Per Period" and "Packet Per Frame" cover groups. Possible unscrambled control period locations are covered by "SSCP" cover group. Number of preambles per frame is covered by "Preambles" cover group. Finally "Data Guard Matching" cover group covers if the receiver can handle the case of matching between data and trailing guard bands. Coverage reports were generated by Synopsys VCS report generator. Moreover, Table 1 shows the performance results for three test-cases, each involves the transmission of 10 frames to an HDMI receiver DUT, running on Synopsys VCS simulator.

VI. Conclusion

In this paper, the verification challenges of audio/video interfaces have been discussed, including protocol complexity, short time-to-market, tight project schedules, huge processing and memory requirements, huge amount of required video and audio formats. The short time to market problem has been addressed through the design of a unified framework for building robust, configurable, extendible verification environments suitable for different audio/video interface protocols. Re-usability of the proposed framework has been discussed, whether among different DUTs or different audio/video protocols, showing how the test-benches developed for both the HDMI transmitter and receiver shared about 50% of the code. This verification environment has been developed in around 2 months, by a team of 4 engineers. It is easily extendible to include other features or be adapted to other protocols.

References

[1] Hexdump, http://www.richpasco.org/utilities/hexdump.html

[2] Imagemagick, http://www.imagemagick.org/.

[3] Zeranoe mpeg, http:// mpeg.zeranoe.com/.

[4] HDMI 2.0 Specications. HDMI Forum, 2013.

[5] S. Gupta. Hybrid functional verification methodology for video/audio soc. In Quality Electronic Design, 2009. ASQED 2009. 1st Asia Symposium on, pages 264-265. IEEE, 2009.

[6] Y. Patil and D. D'Mello. Verification of a next-generation single-chip analog tv and digital tv asic. In Metric-Driven Design Verification, pages 303-324. Springer, 2007.

[7] A. K. Sharma, H. Mittal, M. Singh, R. Mishra, and J. Yadav. Hdmi 2.0 design and verication challenges. 2013. This template has been prepared and adapted for use in DVCon Europe 2015.

[8] P. van der Stok. Dynamic and robust streaming in and between connected consumer-electronic devices, volume 3. Springer, 2005.

[9] H.-Y. Yang. Highly automated and ecient simulation environment with uvm. In VLSI Design, Automation and Test (VLSI-DAT), 2014 International Symposium on, pages 1{3. IEEE, 2014.