RESEARCH

A Novel FPGA-Based Test-Bench Framework for SDI Stream Verification

Giuseppe Conti[1*], Christos Kyrkou[2], Theocharis Theocharides[2,3], Gustavo Hernández-Peñaloza[1] and David Jiménez[1]

Abstract

This paper presents a framework for complete simulation and verification of Serial Digital Interface (SDI) video using a verilog test bench, and geared toward FPGAs. This framework permits simulating the entire process: from test video signal generation to protocol verification in the FPGA which implements the Device Under Test (DUT). The novelty in the design is the combination of a customized test video signal generator with an implementation clone of DUT transceiver for in-depth protocol debugging. Identical input test patterns of the video protocol under test are generated and fed to DUT for verification. Thus, the model not only permits to evaluate the SDI transport layer but also validates the implementation at ultra low pixel level of the video format. This approach provides two advantages; cost saving in terms of additional lab test equipment and delivering all-in-one test solution to verify design and implementation. A practical implementation using a test example of a macro block processing chain using SDI video interface shows the viability of the proposed framework for video protocol testing.

Keywords: Verification; FPGA; Video Test-bench; Real Time System; Simulation

1 Introduction

Nowadays, the video processing growing market requires the development of new devices. Emerging programmable System-on-Chip platforms combining Field Programmable Gate Arrays (FPGAs) with general purpose processors are suitable for running operating systems and to also accelerate algorithms at the hardware level. Hence, such platforms are a promising technology capable of providing the necessary power/performance trade-offs for emerging applications such as AI, video coding, etc. FPGAs can also support verification frameworks for other technologies, for example is possible to design an ASICs (Application Specific Integrated Circuit) device and perform its verification using an FPGA. However, due to the increasing complexity and size of such FPGA-based devices the verification techniques become very important to validate the concreteness of a design and reduce the hardware errors and the time to market. Hence, the verification process represents a big challenge for design and verification engineers. This is further emphasized when considering that in real applications, the design needs to be verified with the operating conditions in mind which must include the communication, and synchronization with external interfaces and design correctness needs to be evaluated across various levels.

Current practices of analyzing designs involve incorporating a test bench into the design to drive certain signal stimuli and compare the outputs of the design with the desired ones. The verification process is usually metric-driven, as for example coverage, that allows to measure the verification progress and determine when the design is ready. A brief metric description is presented in section 2.

These verification metrics have been improved and automatized during the last years by Ray Salemi and Mentor Graphics team, that generated Universal Verification Methodology (UVM) [3, 4]. This method developed an automatic test-bench system able to perform high level DUT verification using System Verilog. UVM goes further by providing mechanisms that allow constraints to be written as part of a test rather then embedded within dedicated verification components. This and other features of UVM facilitate the creating of reusable verification components. During the years these techniques have been supported by Accellera System Initiative [5] and the advantages of its

[*]Correspondence: gic@gatv.ssr.upm.es
[1]Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid, Madrid, Spain
Full list of author information is available at the end of the article
[†]Equal contributor

use are demonstrated in [6] using automotive examples. Verification methods are designed to be general and can be adapted for every testing scenario.

This work focuses on a framework dedicated to systems that use SDI interface to communicate and transmit data, geared on a video application. Video formats are evolving fast and they are increasing in resolution, transmission rates and video stream complexity [1]. SDI family is one of the most extended interfaces for transmission of uncompressed video between professional equipments [2]. As a consequence, the devices for testing purposes have become a concrete need. However, the requirement of new facilities for video analysis usually represents a big cost, and not always available since the most recent video standards are under continuous development. Due to the improvement of video standard resolution and complexity, the need of faster devices has increased the use of FPGA for video applications, as described in [8, 9]. This means that a verification method should be applied to allow testing the communication protocol and the device processing such as encoding, decoding or filtering the signal. Often, the hardware analyzers and generators lack flexibility to modify the patterns for assessment of the video test sequences, they only offer the tests and formats available on the instrumentation, as is possible to check in [23, 22]. As a consequence, stream compatibility issues arise due to different video formats, components or protocols, therefore complete video analysis can not be properly performed.

The main goal of this work is to help designers and verification engineers to perform hardware tests before producing the device, simulating it in a complete test bench environment allowing testing not only the device internal functions but also the data exchange with the external world through its communications protocol.

The main advantages due to using this model are following, it allows to:

- insert video stream testing sequences into the DUT. It has the same behavior than if an external hardware signal generator would be plugged through the HD-SDI physical interface.
- compare the input and output stream of the DUT to verify its processing correctness. Throughout this comparison, it is possible to analyze the SDI transport layer, its protocol functionality and synchronization, and the video data components reaching pixel resolution.
- check the DUT internal blocks delays, therefore FPGA resources or external hardware are not required. Furthermore, it is not dependent on hardware manufacturers, simulators and programming languages.

The verification methodology that is presented in this work can be applied to a numerous practical cases, specially where the data need to pass through a physical interface. In a specific video scenario this method allows to test for example an encoder or a video filter in a professional production television environment, allowing to simulate not only the device but its connections to the rest of other equipments, simulating also the physical connection port and the communication protocol. It will be explain how to apply this framework in order to archive a complete DUT observation, including simulation times, signal observability, chain delay, reduced FPGA resources occupation and same signal captures. It will be also compared where possible the main advantages over the state of art in order to evidence the improvements provided by this framework specially on the observability of internal and external signals.

The remainder of this paper is organized as follows: in section 2, the problem and main conditions are formulated. In section 3, the verification framework is detailed. In 4, the validation experiments to show the model performance are described. In section 5, results to show framework functionality are provided. Finally, conclusions are drawn in section 7.

Table 1 Notation

Acronyms	Meanings
DUT	**Device Under Test**
SDI	**Serial Digital Interface**
HD	**High Definition**
PAL	**Phase Alternating Line**
FPGA	**Field Programmable Gate Array**
SAV	**Start of Active Video**
EAV	**End of Active Video**
IP	**Intellectual Property**
GTX Tranceiver	**Video Tranceiver Xilinx primitive**
Triple-Rate SDI	**Xilinx IP that provides receiver and transmitter interfaces for the SMPTE 259M 292M 425M standards**
SMPTE	**Society of Motion Picture and Television Engineers timecode**
MB	**Macro Block**
CRC	**Cyclic Redundancy Check**
FIFO	**First In, First Out**
Block Ram	**Configurable memory module**

2 Background and Related Work

In this paragraph video analysis systems are described starting from the standard equipments up to practical examples using FPGA test benches.

The fast grow of resolution in video formats such as Ultra High Definition Television (UHDTV) is continuously raising new requirements for information transport. Consequently, demand for techniques and testing/measurement equipments has also increased. A typical deployment for video measurements is shown

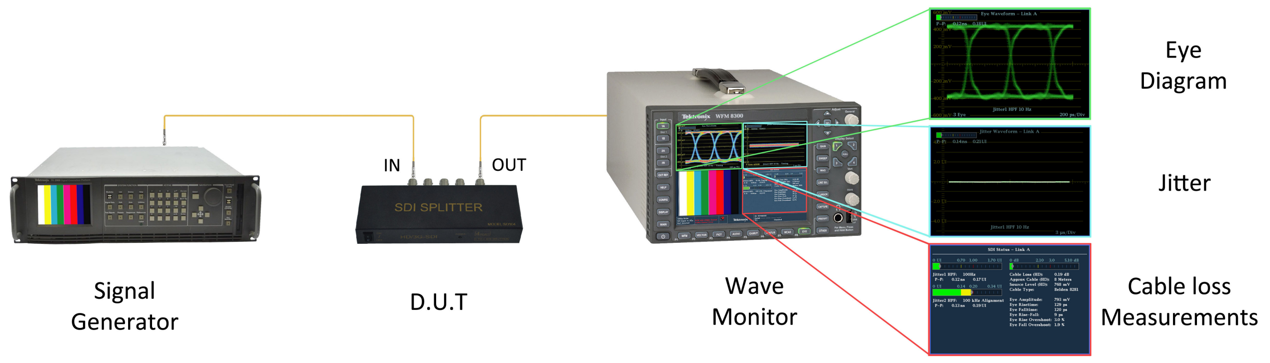

Figure 1 SDI Video equipment connected to the Device Under Test. The right side show the main testing parameters that can be analyzed using a Hardware Wave Monitor

in figure 1. This setting is composed by three main elements: *(a)* a Signal Generator that introduces a pattern signal test into the *(b)* Device Under Test (DUT) and *(c)* a wave monitor employed to visualize the signal characteristics to be analyzed. This common video testing setup allows signal analysis by means of several parameters such as eye diagram, Jitter error and cable losses. Further details can be found in [18, 22]. However, these parameters allow to assess the HD-SDI signal quality of measure only, because the wave monitor can analyze the physical layer, show the transport layer, but cannot debug it. This is an important drawback for more specific tests such as component analysis. The proposed framework is intended to permit SDI transport layer debug, and data verification along the entire DUT components in a detailed manner (up to video components analysis) .

A more specific situation is to consider a development scenario where the DUT is implemented on an FPGA, which limits the clock frequency when compared to the frequency of the design in the real world. Performing a DUT data processing test with a physical wave monitor is not possible due to different clock rates between internal data processing and external data transportation. Additionally wave monitor should be able to extract video data components and compare them with signal generator.

To validate hardware processing correctness, an internal signal analysis is required and it can be performed using FPGA logic embedded analyzers (eg: Vivado Logic Analizer [27], Chipscope [28] or SignalTap [29]). However, video HD-SDI signal testing is not possible using internal FPGA logic analyzers because there is not any interface that allows HD-SDI connection with embedded logic analyzers. A valuable alternative to face this problem is given by performing simulations before FPGA programming, but the standard test-bench can work only at DUT internal frequencies (e.g. 74.25MHz or 148.5MHz for HD video systems). FPGA test-bench cannot operate at SDI frequency (1.485Ghz, 3GHz,...), and is not suitable for SDI signal analysis.

An example of HD-SDI protocol is depicted in Figure 2 and is mainly composed by:

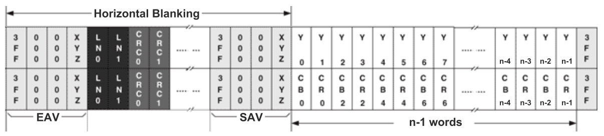

Figure 2 HD-SDI components stream example

A Start of Active Video (SAV) comprises four words that indicate the beginning of video components, an End of Active Video (EAV) comprises four word indicating the end of video components. Both SAV and EAV headers are identified in the data stream by the following words: $3FFh, 000h, 000h$. These words carry the timing information and avoid the use of traditional synchronization signals. The fourth word XYX contains signal information that permits to identify: *(a)* the SAV or the EAV header, *(b)* the field in case the video is interlaced, *(c)* if data represent the vertical blanking or the video active. A digital line blanking ancillary data is sent after the EAV header, it contains a two-word line number (LN0 and LN1), followed by a two-word CRC (CR0 and CR1). The CRC value is used to detect errors in the active video that follows the SAV. The information contained between EAV and SAV is called Horizontal Blanking. The figure 2 depicts the above described protocol, the *n-1 words* in the right part of the figure are called Active Video. It represents the Luma an Chroma pixel information which size depends of the video format. SDI protocol data rate is up to 12 Gbit/s[21]. Further details can be found in [2, 17].

In video processing systems as the one shown in figure 1, DUT internal transceiver matches different data rates. Additionally, it also transforms the signal stream from serial to parallel and vice versa considering transmission or reception stage.

Performing HD-SDI hardware verification requires a generator able to introduce into the DUT a signal according to the video protocol standard. Moreover, checking the output sequence in order to guarantee data integrity in terms of processing and synchronization represents an important issue that test-benches do not solve without additional external hardware components.

A brief description about some verification metrics following:

- *Functional coverage* represents the measure of how much functionality of the design has been exercised by the verification environment.
- *Code coverage* measures how much code is checked by the test bench.
- *Constrained Random Testing* provides a random stimulus as input, to push the DUT into interesting corner cases giving the means to reach coverage goals.

More details are available in [7]. Some test-benches are directly synthesized inside the FPGA. It represents an advantage in terms of test speed and signal data transfer rate between DUT and generator. However, it yields to waste FPGA resources which is critical for large-scale projects development.

It could be used for video testing purposes, however an external video signal transceiver for DUT complete simulation is required.

Alternatively, an FPGA synthesizable test-bench that allows fast execution times and chip observability for debugging is presented in [10]. Nonetheless, in addition to the need of adapting a video signal transceiver, the entire project is executed on the DUT physical memory. It represents a FPGA resources waste.

Another approach to tackle this problem is proposed in [11]. The verification method compiles the test-bench on a computer. It connects internal blocks directly to a PCI-extended bus to introduce and receive signals to/from the DUT. However, HD-SDI implementation is not possible on the PCI-extended bus due to the demands for high data rates which are not possible under this configuration, because the PCI bus of the Altera Stratix II EP2S60F1020C4 used is a 33 MHz bus as detailed in [24], and cannot reach the HD-SDI data transfer rate. It could be possible using new FPGA that use new PCI standard as described in [25].

A hybrid verification platform is presented in [12], where authors combine software simulation and hardware emulation. The experiment synthesizes a 32-bit DLX PCPU (pipelined CPU) on an Altera FPGA, an assembly program with a bug is run and the FPGA internal behavior is recorded. The data is used to reconstruct the relevant segment of a simulation in ModelSIM [38] for debugging. This work allows to reduce the simulation times, it permits to analyze the internal behavior of the FPGA and to debug its processing stage, however in the SDI case an external analysis is needed, because it allows to evaluate the correctness of video components insertion in the SDI protocol to be interchanged with other devices. This work does not consider this case.

The most similar work to the framework presented in this paper is [13], where a Phase Alternating Line (PAL) signal processing hardware and a test-bench are generated to accelerate hardware verification. However, interface debugging is not specified, while in case of video systems it has to be considered not only the internal processing, but also the video data interchange over the interface. This is important because if the video data is not properly inserted, a synchronization issue will make the signal not understandable from the other devices (eg: a television that not receive a correct signal from a decoder, cannot visualize it). Furthermore, this work cannot be used for HD-SDI verification because it uses analogical interfaces (S-Video and VGA) while HD-SDI is a digital interface, they also work with different data rates that are not compatible.

In references [10] to [13] some works about test-benches have been described, a resume and comparison with our work is presented in the table 2. While in references [14] to [16], to emphasize the contribution of our proposed approach, some examples about its application will follow. It will be presented how our approach could help in issues related to the verification of complex video processing systems. A brief description about an existing approaches, will be followed with an explanation on how our proposed alternative can provide a more flexible and economic solution as a verification technique.

In the work [14], a low latency 3D hardware image rectification engine using FPGA and HD-SDI interface is presented. The authors have created a complex structure for verification and a Printed Circuit Board (PCB) for interface integration. In this case, the application of the proposed test-bench represents an advantage for the verification because it allows to compare the two streams during the simulation including HD-SDI interfaces. The simulation can provide important information before the creation of PCB and can prevent its redesign in case of FPGA hardware issues.

In [15], an architecture for exact computation of 2-D Discrete Cosine Transform (DCT) $8X8$ blocks on

Table 2 Main characteristics of literature test-benches and differences with Related work

Work	Type	Connectivity	Data Available	Remarks
[10]	Internal scan chain	N/A	Simulation time vs # of Scan Chain	Overheads in FPGA resources; limited observability to internal signals; the protocol observation is not allowed; its analysis time is fast
[11]	Test bench connected to the board	PCI to interchange data between DUT and PC	Simulation time, FPGA Resources	Overheads in FPGA resources; limited observability to internal signals; the protocol observation is not allowed; its analysis time is fast with less probing nodes; became slow with a high number of probing nodes
[12]	Software simulation and hardware emulation	UART and JTAG for debug	Debugging speed comparison, FPGA Resources	Overheads in FPGA resources; limited observability to internal signals; the protocol observation is not allowed; its simulation time is fast
[13]	Software simulation and hardware on FPGA IP verification	S-Video / VGA signal input and output, USB / RS232 for debug	Unknown - Only structure	Overheads in FPGA resources; limited observability to internal signals; the protocol observation is not allowed; do not provide simulation/analysis time
Our Work	Software simulation	SDI Family, but this method allows to use every interface to carry signals	Simulation Time, simulated internal blocks processing time, FPGA resources	We do not overheads in FPGA resources; unlimited observability to internal and external signals; we allow protocol observation; it's integrable to any simulation tool; commercial or open source; the simulation time is slow

FPGA is presented. The verification is performed using Matlab. The Matlab workspace input data into the FPGA through the JTAG interface, the measured output is returned to the Matlab workspace for verification. In this case, to apply our test-bench avoids to migrate to Matlab workspace using FPGA simulation tools (i.e: [37, 38]) to validate the design. Another advantage is represented from the opportunity to simulate the design in a real environment where the signal is inputted through a video Interface (not JTAG) and data is provided using a generator. Moreover, the complete chain can be verified through direct comparison between input and output video interfaces.

In the work [16], a real-time video stabilization system on FPGA is proposed. In this work, authors use a PAL interface to interchange video data. A measurement environment is generated to evaluate the Peak Signal-to-Noise Ratio (PSNR) between the original video sequence and the stabilized video sequence. By means of the proposed verification test-bench PAL interface can be simulated. Moreover, it permits upgrade to HD formats and the integration of digital video interfaces such as HD-SDI family. Furthermore, applying our test-bench allows to automatize the PSNR calculation and comparison between frames from both input and output for automatic assessment of the results.

In comparison with the other approaches presented in this section, our model allows to test the DUT using the same behavior than if an external hardware signal generator would be plugged through the HD-SDI physical interface. In contrast with the other approaches it is possible to analyze the SDI video stream reaching pixel resolution, and perform DUT verification analyzing internal blocks functionality and delays, without overhead in FPGA resources or external additional hardware. This framework flexibility is not dependent on hardware manufacturers, simulators and programming languages, and allows the complete DUT verification due to its internal connection structure. The idea presented in this work represents a novel approach to the verification state of art because improve the DUT observability, allowing to check DUT internal signals by direct signal routing the the check block, without passing through an interface, a feature that is possible only during simulation.

3 Verification Model

The framework structure is drawn in the right side of figure 3 and it is composed from many blocks. Block 1, Block 2, ... Block N, represent the processing chain of a design to analyze. It could be for example a video filter, an encoder, a decoder, etc. FPGA test benches allow to analyze only these chain structures, where the test signal is introduced at the start of the processing chain and a check block at its end.

The chain is connected to a transceiver that communicate the input signal from physical layer, extract the components that have to be precessed from a protocol and then provide them to the processing chain. After the processing chain has finished its tasks, it provide the data to the transceiver that will encapsulate into a protocol and send it to the physical output layer. The SDI family has a high data rate, it means that the input/output stage of transceiver works with high frequencies (more than 1 GHz for HD signals). A typical FPGA board has an internal clock of around 200MHz,

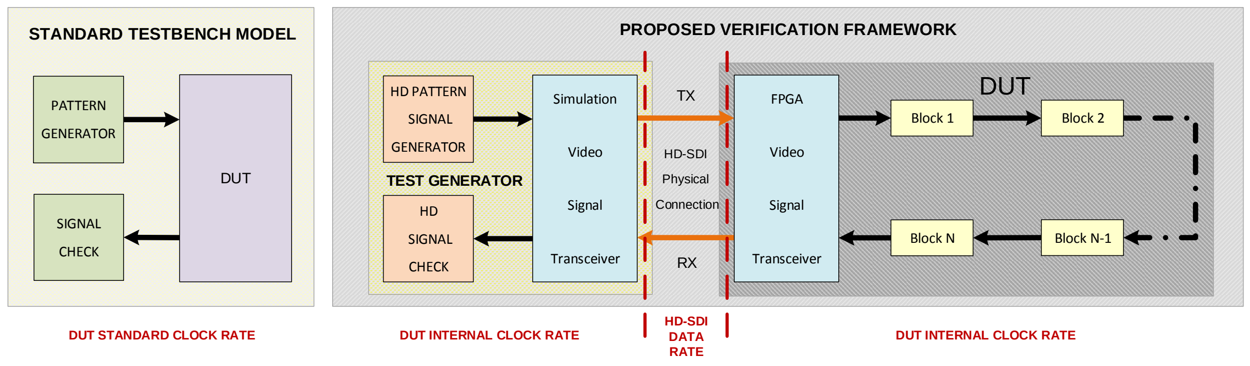

Figure 3 *(Left)* Standard test-bench model. *(Right)* Video test-bench framework with double transceiver system for SDI simulation. Blocks 1 to N represent the video processing chain.

as shown in [32], that cannot reach the SDI clock frequency. The solution to match this different frequencies rates is to transform signal from serial to parallel and work with multiple frequency (eg: external serial signal at 1.485GHz, internal parallel signal of 10 bits and 148.5MHz for HD-SDI).

In case of SDI video application, the transceiver has two functions:

- In the input stage it receives the signal from physical layer, extract the video components from SDI stream, convert the signal from serial to parallel and matches the different signals data rates (eg from 1.485 GHz to 148.5MHz for HD-SDI) to make it suitable for the internal FPGA processing frequency rate.
- In the output stage it receives from the processing chain the video components as parallel signal at 148.5MHz, encapsulate it in the SDI protocol converting the signal from parallel to serial and provide it to the physical output layer (at 1.485 GHz for HD-SDI).

This is not possible to debug with standard FPGA tools (Signal tap by Altera, Chipscope by Xilinx) because they can only work with internal clock rates. To debug the complete signal an analyzer is needed but it only offer some tests as already described in section 2. In a Simulation is possible to analyze the different frequency rates, but is missing a test generator able to communicate with transceiver. At this point the main Idea of this work is to use a double transceiver structure shown in right side of figure 3:

- The first transceiver (called FPGA Video signal Transceiver in right side of figure 3) represent the FPGA transceiver, and allows to communicate with the processing chain using the SDI physical interface and protocol.
- The second transceiver (called Simulation video Transceiver in right side of figure 3) allows the communication between a custom Test Generator block with the rest of the structure, and to receive the processed signal to the signal check block over the SDI channel.

This double transceiver structure allows to simulate the entire DUT and connect a custom generator and a custom check block to test the DUT. It is important to remark that is possible to use every custom routine for signal generating and signal checking that are not available in the physical analyzer and in normal test-benches. This structure can debug the internal DUT over the SDI interface, and allows to test not only the functionalities of processing chain, but also the data integrity of video component and signal sync over the SDI interface. Moreover, conversely to standard hardware testing system shown in figure 1, the proposed framework also can access the response of every internal DUT component and drive its signal directly to the check block without insert them in the rest of processing chain through the transceivers. It results in a significant improvement of the testing observability. An example of a possible measure test is the DUT chain delay D that can be determined according to the following equation:

$$D = \sum_{k=1}^{m} \frac{ccb_k}{clk\_freq_k} \qquad (1)$$

where m is the total number of internal DUT blocks, ccb_k is the number of clock cycles required for k block processing, clk_freq_k is the working clock frequency and D is expressed in seconds.

Thanks to this flexible approach is possible to change the kind of test just changing the code of generator and check blocks, and running an other simulation. It also permits to automatize the testing routines or running

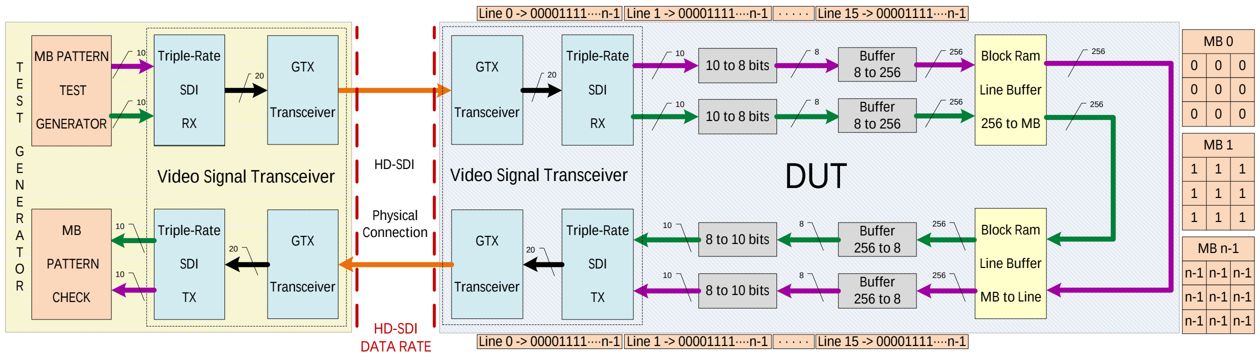

Figure 4 Testbench applied to a real case: The experiment validation model is ported on a Virtex-6 Xilinx FPGA

a simulation with auto checking routines. In next section, blocks employed for experiment validation are described.

4 Experimental Validation Methodology

4.1 Testing Methodology

Video processing techniques such as video compression and video content analysis have been widely used in various applications. Most of them uses a picture subdivision in squared little areas called Macro Blocks. In detail, a Macro Block is a square group of pixel used as elementary unit in video processing techniques, it could have a convenient dimensions depending on the application (eg: 8x8, 16x16, 32x32, etc). A typical application that uses Macro Blocks subdivision is H.264 coding [19]. It represent a widely used standard in video applications, for this reason many systems can interface with such coding schemes so it is necessary to guarantee correct and accurate verification process. The testing methodology developed in the DUT implements a video processing chain oriented to H.264 coding, focusing on Macro Blocks processing. The main idea consists in a system that receives HD-SDI video stream, extracts the video lines and organizes the data into Macro Blocks as shown in figure 4. At this point the Macro Blocks will be ready for coding process. However the experiment was designed to Macro Blocks extraction and line restore, so the data will be sent to the output part of the chain that reorganizes Macro Blocks to create the video line. The proposed method is applied to the process already described, to allow performing its verification. It uses a custom macro blocks generator and a macro blocks check module that permit to compare the two signal outside the DUT through the HD-SDI interface, allowing to evaluate the correctness of DUT internal processing chain.

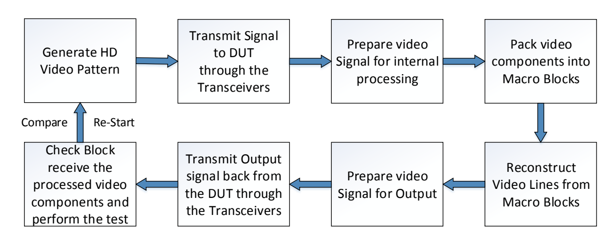
Generate HD Video Pattern
Transmit Signal to DUT through the Transceivers
Prepare video Signal for internal processing
Pack video components into Macro Blocks
Reconstruct Video Lines from Macro Blocks
Prepare video Signal for Output
Transmit Output signal back from the DUT through the Transceivers
Check Block receive the processed video components and perform the test
Compare
Re-Start

Figure 5 Complete chain functionality block diagram

4.2 Global Functional Description

The figure 5 summarize the functionality of the entire verification frame applied to the system, its description is following:

The pattern generator create the data that will be introduced into the transceiver that communicates with the DUT. The DUT Transceiver receives the signal as video line (in figure 4 the components are depicted on the top right part as sequence line 0, line 1, line 15). The video line is processed pixel per pixel until module *Block RAM Buffer 256 to MB* which stores the 16 lines and extracts Macro Blocks (as shown in the extreme right side of figure 4). The Macro Blocks are stored in the module *Block RAM Buffer MB to Line* which reconstruct the line and outputs the components to the rest of chain. The lines pass through the transceivers until *MB pattern check* which permits to perform the comparison and verification with the pattern generation.

A great advantage is represented by the direct connection between each module of the DUT and the *MB PATTERN Check* without entering into the transceiver which is allowed only in the simulation of this framework.

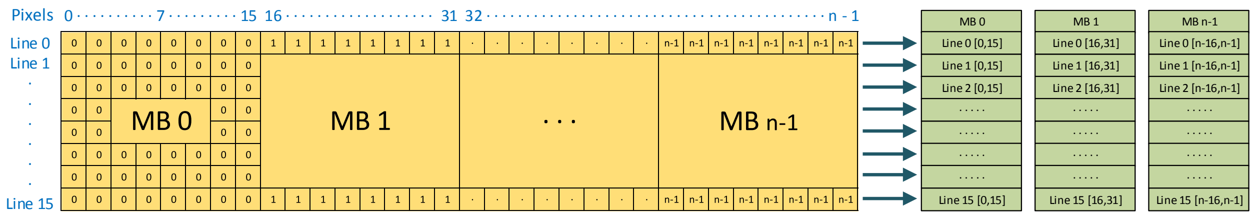

Figure 6 Custom generator for Macro Blocks check. On the left side, the sorted components input line is shown. On the right side, the expected Macro Blocks output is shown.

4.3 Macro Blocks Component Extraction

Before a complete description of the system, it is important to introduce the test method employed to verify the correct extraction of the Macro Blocks components. Following H.264 standard, to encode the video signal, a frame should be divided into Macro Blocks as detailed in references [19, 20]. In this experiment, a 16x16 matrix of pixels will be used as Macro Block dimension. For this reason, to create a Macro Blocks sequence, is necessary to process 16 video lines. To identify each component of the Macro Blocks is useful to have a pattern generator able to insert the data in a known order sequence. For this purpose, a custom frame generator is created to provide an ordered numbers sequence instead of video components. The principal benefit of this approach is the enumeration of each macro block pixel by pixel. It allows to identify an eventual error in the process just checking if the output is respecting the enumeration order provided as input stream. The macro blocks generator output is shown in figure 6 and consist in the generation of an ordered sequence of number repeated for 16 video lines (0 to 15), that allows to identify each Macro Block with its number. An example of pattern is following:

- *line 0*: 0000...1111...2222...3333...$n-1, n-1, n-1, n-1$
- *line 1*: 0000...1111...2222...3333...$n-1, n-1, n-1, n-1$
- . . . .
- *line 15*: 0000...1111...2222...3333...$n-1, n-1, n-1, n-1$

where $n-1$ for 720p resolution (1280x720 pixels) is $1280/16-1 = 80-1 = 79$, 80 corresponds to the number of Macro Blocks every 16 line, for a total of 3600 Macro Blocks per frame. Once this pattern is repeated, the DUT will be able to extract a Macro Block of zeros, a Macro Block of ones, ...etc. If this process is performed n times, the entire frame will be generated, (eg: in 720p will be repeated $n = 720/16 = 45$ times). It is possible to add markers for line pattern labeling. As an example, the line number can be labeled as follows:

- aa0000...1111...2222...3333...aa$n-1, n-1, n-1, n-1$

The aforementioned process is depicted in figure 6. In this figure, an example of pattern generator output and Macro Blocks composition is shown. Horizontal axis corresponds to pixel arrangement, whereas vertical axis represents the line numbers. eg:

- *MB0*: line 0 to 15, pixels 0 to 15.
- *MB1*: line 0 to 15, pixels 16 to 31.
- . . . .
- *MB*$n-1$: line 0 to 15, pixels $n-16$ to $n-1$.

For a 720p format as shown in figure 7, the Macro Block 79 will have line 0 to 15; pixels 1263 to 1279 position. The 3599 Macro Block, the last one in a frame, will have line 703 to 719; pixels 1263 to 1279 position.

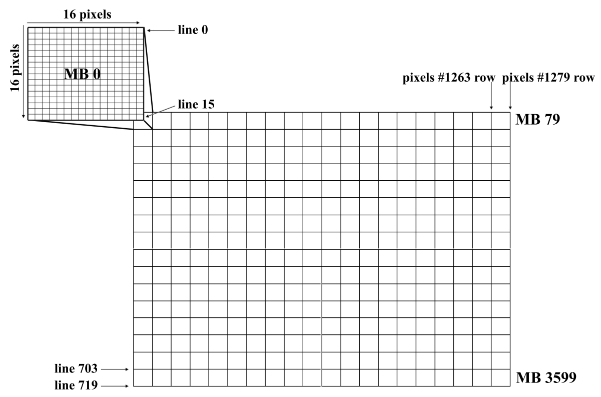

Figure 7 Macro Blocks HD 720p frame subdivision example

4.4 Testbench and detailed Functional Description

The DUT macro blocks processing chain was developed and tested on a Xilinx ML605 evaluation board [33] featuring Virtex-6 XC6VLX240T-1FFG1156 FPGA [34]. Additionally, a Broadcast Connectivity FMC mezzanine card (CTXIL671) [35] was employed as physical interface to connect the HD-SDI signal to the

ML605 Board. As a proof of independence from manufacturer, this framework can also be implemented in other FPGA company products. As example, it could be developed on Altera FPGA using SDI MegaCore Functions IP [31]. Moreover, software for FPGA programming is the ISE 14.7 design suite by Xilinx [26]. The computer used to run the simulations is based on an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz with 4GB of DIMM DDR3 RAM.

The DUT structure was developed according to the proposed verification framework depicted in the right part of figure 4. The figure represents how a real project is entirely simulated in a Computer. In the left part of the figure, the MB pattern test generator shows the generator already described.

The functionality of the DUT blocks is detailed as follows:

- The *video signal transceiver* is composed by two Xilinx Intellectual Properties (IP): GTX transceiver and Triple-Rate SDI. The former allows to receive/transmit the HD-SDI protocol. In reception, the signal is acquired at 1,485 Gbit/s, transformed from serial to a parallel bus at 148,5 Mbit/s, matching the data rate. While, in the transmission stage, the transceiver performs the opposite operations. The latter permits to establish the communication between GTX transceiver and the rest of the chain. In reception it decodes the GTX transceiver data coming from parallel bus into video components (lumas and chromas) and synchronization information (eg SAV, EAV, number of line, etc). Conversely, in transmission it encodes the processed video from chain and permits to add synchronization information. Further details on the Xilinx IPs mentioned can be found in [30].
- The next blocks represent the processing chain. The data they process is the video components in pixels. The information of each pixel is represented using 10 bits in the HD-SDI stream. After is normalized to 8 bit because is more suitable for the internal FPGA memories and communication buses.
- The block *10 to 8 bits* reduces the output of the Triple rate SDI RX from 10 bits to 8, this allows to store luma and chromas into FPGA memories.
- The block *Buffer 8 to 256* of the DUT in figure 4 is a buffer that changes the 8 bits video signal component (Y,Cb,Cr) coming from the previous block into a bus of 256 bits. This size of bus has been chosen to attain a faster transportation of video components to the rest of project blocks.
- The module *Block RAM Buffer 256 to MB* receives the 256 bits data and stores it in a memory with capacity to contain 16 video lines. Afterwards, it reads and reorganizes data in Macro Blocks of 16 x 16 pixels. As a result, every Macro Block is provided at the output of this block under 8 component of 256 bits.
- The module *Block RAM Buffer MB to Line* of figure 4 performs the opposite operation of the previous block. It receives the Macro Blocks, stores them in a FIFO, arranges video data in lines and delivers it to the output.
- The block *Buffer 256 to 8* in figure 4 adjusts the signal from 256 bits to 8 and introduces it into the following Block.
- The block *8 to 10* in figure 4 adjusts the signal from 8 bits to 10 and introduce it into the Triple Rate SDI TX block ready for Output.
- The test generator was previously described in section 2.

The DUT is a real component that could be flashed into the FPGA whereas test generator is not implemented on physical hardware since it is part of the simulation. In next section, results of simulations are provided. The simulations were performed using two different tools: The Xilinx ISim 14.7 [37] and ModelSIM SE 10.1C. [38] which remark flexibility of the proposed framework. Finally, two files with the testbench code employed in this work are provided, and can be downloaded from [41].

5 Experimental Evaluation and Results

In this section, the results of simulations performed according to the structure described in previous sections are presented. Full connectivity verification, video component, internal DUT observability and time delays were verified. These results are shown in figures 8, 9 and 10 where three simulation captures summarize part of video sequences to display the most relevant features of this test. Three main sections to assess the framework performance are distinguished. Each section represents an experiment, in particular the A box of figure 8 refers to the Macro Blocks extraction process from line, the B box of figure 9 represents the opposite operation where the Macro Blocks are reconverted into lines. Finally, the C Box of figure 10 is a test where the pattern generator is replaced by a standard bars generator. This replacement allows the simulation to validate the entire chain functionality. A complete description of every section is following.

5.1 Macro Blocks Extraction

In the experiment **A** of figure 8, the Macro Blocks are extracted from the lines. This experiment, as already depicted in the functional description in section

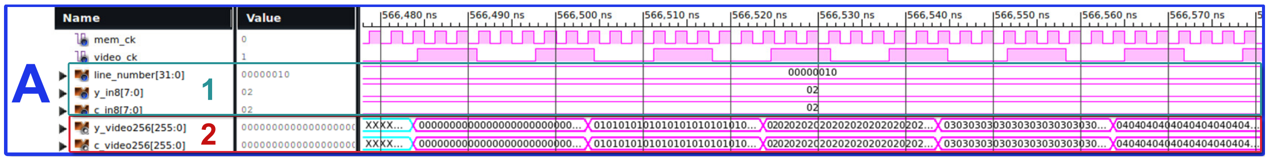

Figure 8 The simulation screenshot represents the Macro Block extraction process.

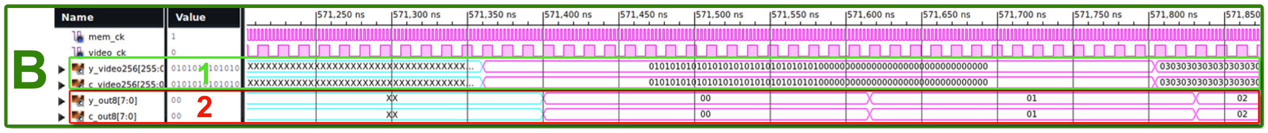

Figure 9 The simulation screenshot corresponds to Line reconstruction experiment.

4, includes the modules *Buffer 8 to 256* and *Block RAM Buffer 256 to MB* of the figure 4. The signals y_in8 and c_in8 in the box **A1** correspond to the input of the module *Buffer 8 to 256*. Furthermore, the signals y_video_256 and c_video_256 in the box **A2**, represent the module *Block RAM Buffer 256 to MB* output. It is very important to remark that the input data is provided by the *MB PATTERN TEST GENERATOR* through the transceivers, but the analysis buses are connected directly to the *MB PATTERN CHECK* block (allowing direct signal check, without passing through the transceivers), this is possible only in the simulation thanks to applying the proposed framework. The video lines arrive into the *Block RAM Buffer 256 to MB* until its buffer is full and therefore ready to extract the Macro Blocks. Once this buffer is full, it starts to output the Macro Blocks. This can be observed in the initial part of picture 8, box **A2** where both signals y_video_256 and c_video_256 are undefined *(XXX...)*. Afterward, it starts to output the Macro Blocks *(0000... 0101...)*. This is clearly shown in the Box **A1** where Macro Blocks are delivered to output after the frame input is in line 16 (signal $line\_number = 10hex$). The pixels in this simulation time are marked 02 according to the test bench structure of section 4.

5.2 Macro Blocks Reverse Conversion

In the experiment **B** of figure 9, the Macro Blocks are received and reconverted to lines. The conditions are the same than experiment **A** of figure 8, where the input signal is provided through the transceivers, but the internal verification buses are connected directly to the *MB PATTERN CHECK*. This experiment analyzes the subsequent stage to the experiment **A** and includes the modules *Block RAM Line Buffer MB to Line* and *Buffer 256 to 8* showed in the figure 4. The signals $y\_video256$ and $c\_video256$ in the box **B1** of the picture 9, represent the input of module *Block RAM Line Buffer MB to Line*, while the signals y_out8 and c_out8 in the box **B2**, represent the output of the module *Buffer 256 to 8*. The capture of this simulation is taken in the instant after the previous module, in the experiment **A** is outputting the Macro Blocks. This is verified by the timing in top part of boxes **A** and **B**. The modules of previous stage start to deliver Macro Blocks around $566,480$ ns of simulation (Box **A2** signals y_video_256 and c_video_256), the data is available at the input of following stage at $571,350$ ns, just 5000 ns later; the time needed by electronics to propagate data. This condition is confirmed by input signals ($y\_video256$ and $c\_video256$) of the module *Block RAM Line Buffer MB to Line* in box **B1**, where before $571,350$ ns signals are undefined *(XXX...)* and after this moment are carrying the Macro Blocks. In box **B2** just after a short delay (33,68ns), it is possible to verify through the signals y_out8 and c_out8 that the line components are distributed. In this case, the delay is minimal because the main component of block *Block RAM Line Buffer MB to Line* is a FIFO, which can output data while the input is arriving [36].

5.3 Bars Generator Test

The complete chain is under test in the experiment **C** of image 10. The input and output signals pass through the transceivers, simulating the same connection than the measurement system shown in figure 1 including transport layer debugging. The block *MB PATTERN TEST GENERATOR* has been replaced by a standard HD bar generator. This replacement allows to test the DUT in simulation in the same manner than using a real equipment. This experiment consists into:

1 generating the HD-SDI signal bars,
2 receiving this signal in the DUT transceiver,

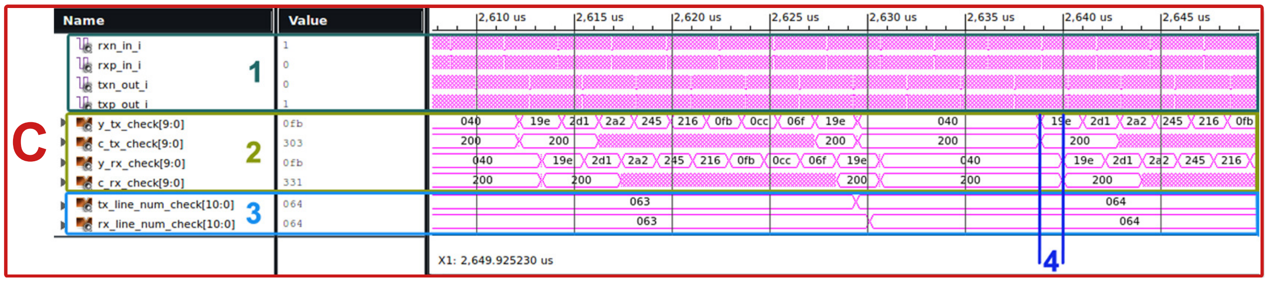

Figure 10 The simulation screenshot shows a complete chain experiment with Bar Generator.

3 extracting Macro Blocks with the modules of experiment **A** of figure 8,
4 recomposing the line through the modules of experiment **B** of figure 9,
5 outputting lines with transceiver and receive the signal in the module *MB PATTERN CHECK*.

If this process is performed without errors by every component of the chain, in *MB PATTERN CHECK* will arrive the same signal created from the bar generator, with a delay accumulated during the chain processing. This experiment simulation is depicted in the box C of picture 10, and detailed as follow:

Box **C1** indicates if DUT and Test Generator are connected and that communication is established. It encloses two HD-SDI serial differential channels that corresponds to the signal stream flow at 1,485Gbit/s between them. These signals represent the TX/RX HD-SDI transport layer connections illustrated in figure 3. No significant differences can be appreciated because of high data rate and channel codification.

Box **C2** is composed by four data buses containing luma and chroma components. *y_tx_check* and *c_tx_check* are sent from signal generator to DUT. *y_rx_check* and *c_rx_check* represent the components that come out after processing stage and are routed to the check block. In this case, comparison between components yields to verify if there is any error in DUT processing modules. Additionally, synchronization data is inserted into these buses and can be also analyzed.

Box **C3** contains a bus depicting the number line of video frame. *tx_line_num_check* was routed from HD Test Generator to signal check block, whereas *rx_line_num_check* was routed from DUT to signal check block.

Marker **C4** highlights the time delay. Simulation process permits to measure the delay evolution in the chain. Model granularity is guaranteed as delay can be observed for every single block. As a result, it is possible to evaluate the processing time for every DUT component, from pattern insertion to processed signal.

Table 3 Simulation time execution and chain delays comparison for several SMPTE standards and video formats in ISIM and ModelSIM.

SMPTE and Video	*(mins per frame)*		*D*: Chain
Standard / Format	ISim	ModelSIM	delay (μsecs)
HD-SDI 720	80	20	360.86
HD-SDI 1080	100	25	838.86
3G-SDI 1080	205	51	838.86
3G-SDI 2K	221	56	871.26
6G-SDI 4K	815	210	3480.86

5.4 Time Measurement and Comparison

The table 3 describes a comparison of simulation time as a function of SMPTE SDI standard rates and video formats. The model was implemented in both mentioned tools (ISim vs ModelSIM). Tests consist in evaluating the time spent by DUT to process a frame of different video resolutions.

The assessment metric for this test was minutes per frame processed. It can be seen that ModelSIM is around four times faster than ISim, probably because ISim does not scale well for larger designs [7]. Time employed for frame processing are long due to high data rate of interfaces ($\approx$ Gbit/s) and large number of simultaneous signals (HD-SDI links increase in 3G and 6G format). It is important to remark that simulations were carried out using a i7-2600 CPU @ 3.40GHz with 4GB of RAM, which is not a powerful architecture for high performance test purposes.

The third column of table 3 shows the chain delay of DUT blocks processing. Simulation results are close to D times calculated using the equation 1.

Table 4 Module time calculation vs time Measure in Simulation @ 1080p

Module	Calculated	Measured
Buffer 8 to 256	$430ns$	$430ns$
Block Ram Line Buffer 256 to MB	$417.8\mu s$	$420.69\mu s$
Block Ram Line Buffer MB to Line	$420.2\mu s$	$423.09\mu s$
Buffer 8 to 256	$430ns$	$430ns$

The table 4 represents an example of time calculation and measurement per component obtained for a 1080p signal simulation test. It allows to verify if every module respects its expected execution times. Additionally,

Table 5 FPGA resource occupation comparison: *DUT with TB:* FPGA occupation resources with Test-bench included. *DUT only:* FPGA occupation resources DUT only

Device Utilization Summary (estimated values)						
Logic Utilization	Used		Available		Utilization	
	DUT With TB	DUT Only	DUT With TB	DUT Only	DUT With TB	DUT Only
Number of Slice Registers	20913	16316	301440	301440	6%	5%
Number of Slice LUTs	18165	13455	150720	150720	12%	8%
Number of fully used LUT-FF pairs	11096	7592	27982	22179	39%	34%
Number of bonded used IOBs	155	102	600	600	25%	17%
Number of Block RAM/FIFO	156	147	416	416	37%	35%
Number of BUFG/BUFGCTRLs	10	12	32	32	31%	37%

there is a little difference between the calculated and measured time of modules *Block Ram Line Buffer 256 to MB* and *Block Ram Line Buffer MB to Line*. It depends on the internal component Block Ram which requires some extra clock cycles due to circuit propagation latency [36].

5.5 FPGA Resources Comparison

The table 5 represents a Xilinx ISE synthesis reports that summarize the FPGA resources utilization. On the one hand, the columns *DUT With TB* show the FPGA resources utilization in case the complete test-bench is synthesized, including signal generator, check block and DUT. On the other hand, the columns *DUT Only* show the FPGA resources utilization when only the DUT is synthesized. It only involves the transceiver, the chain to create Macro Blocks and restore them into lines (right part of figure 4). In the **Logic Utilization** columns, the logic components are specified, in the **Used** columns, the amount of used logic elements is detailed, the **Available** columns show the number of logic elements available in the FPGA. Finally, the **Utilization** columns detail the percentage of logic elements used by the design. Comparing **Utilization** column of both images, it is possible to verify that the design including the test-bench needs more FPGA resources than the design with DUT only. An exception is represented by $BUFG/BUFGCTRLs$, since these are the input/output buffers, the DUT only design requires more buffers to communicate with the output. From this example, it can be observed that proposed framework is less restrictive in terms of FPGA resources, which can be useful those project where FPGA resources utilization is critical.

6 Discussion

To the best of our knowledge, it is not possible to compare this work with other existing models. In most of the referenced papers [10, 11, 13], test signals are not introduced using the double transceiver architecture (to encode/decode video stream) presented here.

Furthermore, other works employ different interfaces (PCI [11], RS232, USB, S-Video[13]) to input/debug data, which means that a previous signal transformation stage should be introduced to enable HD-SDI format analysis. As a result, is not possible to analyze complete HD-SDI stream because information such as synchronization, number of line, etc is lost. Moreover, in [13] an observation window is analyzed, whereas this work allows to debug the complete data flow (all the video frames desired). Referenced works do not allow direct comparison of input and output video data stream, neither are compatible with HD video formats. In addition, they do not provide results about simulation time nor data of interest comparable with our presented work.
Observability improvement of the proposed method is demonstrated since this information is internal and there is not accessibility from outside the DUT. Model flexibility allows to modify the bus observed at the check block by routing the signal/bus desired.

7 Conclusion

In this paper, a novel test-bench framework for SDI video systems has been presented. It permits to perform complete DUT verification in the same manner than a physical flexible hardware in cases where the physical testing hardware is not available. It is especially suitable for deployment in verification environments that exhibit limited resources for DUT occupation, and require flexibility in terms of signal generators and check blocks. In addition the proposed framework is also applicable in cases where the DUT is only accessible through a protocol and/or a transceiver. As shown through the experimental evaluation on an FPGA the verification methodology is applicable to real and practical use-cases and can be used to improve the simulation and verification process for engineers and hardware designers. Finally, this work has demonstrated that the inclusion of transceivers in SDI testing applications is indeed possible contrary to claims provided by manufacturers in design guides [39, 40].

As future work, is interesting to test this framework with other interfaces and protocols (for example HDMI in video systems or others as Ethernet, USB, etc). An other interesting point is the integration with UVM

[3, 4] methodology to improve the automatic testing routines.

Abbreviations
DUT: Device Under Test; SDI: Serial Digital Interface; HD: High Definition; PAL: Phase Alternating Line; FPGA: Field Programmable Gate Array; SAV: Start of Active Video; EAV: End of Active Video; IP: Intellectual Property; GTX Tranceiver: Video Tranceiver Xilinx primitive; Triple-Rate SDI: Xilinx IP that provides receiver and transmitter interfaces for the SMPTE 259M 292M 425M standards; SMPTE: Society of Motion Picture and Television Engineers timecode; MB: Macro Block; CRC: Cyclic Redundancy Check; FIFO: First In, First Out; Block Ram: Configurable memory module;

Availability of data and materials
The test-bench code employed in this work is available in [41], Data sharing not applicable to this article as no datasets were generated during the current study.

Competing interests
The authors declare that they have no competing interests.

Funding
Not applicable

Author's contributions
All authors take part in the discussion of the work described in this paper. The author read and approved the final manuscript.

Acknowledgements
The authors thank editor and reviewers for their helpful comments and suggestions.

Author details
[1]Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid, Madrid, Spain. [2]KIOS Research and Innovation Center of Excellence, University of Cyprus, Nicosia, Cyprus. [3]Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus.

References
1. DigiTAG - The Digital Terrestrial Television Action Group. *ROADMAP FOR THE EVOLUTION OF DTT – A bright future for TV* , DigiTAG and Analysys Mason Limited, 2nd edition, 2014.
2. SMPTE:*SMPTE 292M-1998: Bit-Serial Digital Interface for High Definition Television.*
3. Ray Salemi., *FPGA Simulation: A Complete Step-by-Step Guide* , ISBN: 9780974164908, Boston Light Press, 2009.
4. Ray Salemi., *The Uvm Primer: A Step-By-Step Introduction to the Universal Verification Methodology* , in Design & Test of Computers, IEEE , vol.26, no.6, pp.84-94, Nov.-Dec. 2009 ISBN: 0974164933, 9780974164939 Boston Light Press, 2013.
5. G. Moretti, *Accellera's support for ESL Verification and Stimulus Reuse*, IEEE Design & Test , vol.PP, no.99, pp.1-1
6. M. Barnasconi et al., *UVM-SystemC-AMS Framework for System-Level Verification and Validation of Automotive Use Cases*, in IEEE Design & Test, vol. 32, no. 6, pp. 76-86, Dec. 2015
7. Evgeni Stavinov., *100 Power Tips for FPGA Designers.*, ISBN: 1461186293, 9781461186298 CreateSpace, Paramount, CA 2011.
8. Fowers, Jeremy; Brown, Greg; Cooke, Patrick; Stitt, Greg. (2012). A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA. 47-56. 10.1145/2145694.2145704.
9. P. Sjövall, V. Viitamäki, J. Vanne, T. D. Hämäläinen and A. Kulmala, "FPGA-Powered 4K120p HEVC Intra Encoder," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 2018, pp. 1-5.
10. Mavroidis, I.; Mavroidis, I.; Papaefstathiou, I., *Accelerating Emulation and Providing Full Chip Observability and Controllability* , in Design & Test of Computers, IEEE , vol.26, no.6, pp.84-94, Nov.-Dec. 2009
11. Cheng, X.; Ruan, A.W.; Liao, Y.B.; Li, P.; Huang, H.C., *A run-time RTL debugging methodology for FPGA-based co-simulation* , in Communications, Circuits and Systems (ICCCAS), 2010 International Conference on , vol., no., pp.891-895, 28-30 July 2010
12. C. L. Chuang, W. H. Cheng, D. J. Lu and C. N. J. Liu, *Hybrid Approach to Faster Functional Verification with Full Visibility*, in IEEE Design & Test of Computers, vol. 24, no. 2, pp. 154-162, March-April 2007.
13. Trost, A.; Zemva, A., *Verification structures for design of video processing circuits* , in MIPRO, 2012 Proceedings of the 35th International Convention , vol., no., pp.222-227, 21-25 May 2012.
14. H. Hübert, B. Stabernack and F. Zilly, *Architecture of a Low Latency Image Rectification Engine for Stereoscopic 3-D HDTV Processing* , in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 5, pp. 813-822, May 2013.
15. A. Edirisuriya, A. Madanayake, R. J. Cintra, V. S. Dimitrov and N. Rajapaksha, *A Single-Channel Architecture for Algebraic Integer-Based* $8X8$ *2-D DCT Computation* , in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 12, pp. 2083-2089, Dec. 2013.
16. J. Li; T. Xu; K. Zhang, *Real-Time Feature-Based Video Stabilization on FPGA* , in IEEE Transactions on Circuits and Systems for Video Technology , vol.PP, no.99, pp.1-1
17. Guy Lewis, Michael Waidson.,*A Guide to Standard and High-Definition Digital Video Measurements* , Tektronix, 2009.
18. Werner Klütsch, Winfried Schultz.,*Quality Control and Measurement in HD* , EBU / IRT HDTV Briefing Geneva, 22. November 2005, Tektronix
19. ITU-T Rec. H.264 (10/2016) Advanced video coding for generic audiovisual services `https://www.itu.int/rec/T-REC-H.264-201610-I/en`
20. ITU-T Rec. H.265 (12/2016) High efficiency video coding `https://www.itu.int/rec/T-REC-H.265`
21. *Standard: SMPTE ST 2082-1* 12 GB/S SIGNAL/DATA SERIAL INTERFACE - ELECTRICAL - AMENDMENT 1
22. *WFM8200 and WFM8300 Waveform Monitors User Manual*, Tektronix
23. Signal Generator `https://questtel.com/unit/3g-sdi-pattern-generator/`
24. Stratix II Device Handbook, Volume 1 `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stx2/stratix2_handbook.pdf`
25. Streaming Multichannel Uncompressed Video in the Broadcast Environment `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01121-multichannel-uncompressed.pdf`
26. ISE Design Suite `https://www.xilinx.com/products/design-tools/ise-design-suite.html`
27. UG908 Programming and Debugging `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug908-vivado-programming-debugging.pdf`
28. UG750 - ISE Tutorial: Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Application `https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/ug750.pdf`
29. Using SignalTap II Embedded Logic Analyzers in SOPC Builder Systems `https://www.altera.com/en_US/pdfs/literature/an/an323.pdf`
30. Xilinx Application with Video transceiver IP., `http://www.xilinx.com/support/documentation/application_notes/xapp1075_V6GTX_TripleRateSDI.pdf`
31. Altera Video transceiver IP., `https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-sdi.html`
32. Xilinx Virtex-7 FPGA Board specification `https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html#hardware`
33. Virtex-6 FPGA ML605 Evaluation Kit., `https://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html`
34. Virtex-6 Family Overview., `https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf`
35. CTXIL671 Board., `http://www.cook-tech.com/ctxil671.php?OrderOnline=More+Info&TICOM_AUTHLEVEL=3&TICOM_VALIDATE_INPUT=1`
36. Virtex-6 FPGA Memory Resources., `https://www.xilinx.com/support/documentation/user_guides/ug363.pdf`
37. ISim Xilinx simulator., `http://www.xilinx.com/tools/isim.htm`

38. Mentor Graphics debug and analysis environment., http://www.mentor.com/products/fv/modelsim/
39. SMPTE UHD-SDI v1.0 - LogiCORE IP Product Guide., https://www.xilinx.com/support/documentation/ip_documentation/v_smpte_uhdsdi/v1_0/pg205-v-smpte-uhdsdi.pdf
40. XAP1249 - Implementing SMPTE SDI Interfaces with 7 Series GTX Transceivers, https://www.xilinx.com/support/documentation/application_notes/xapp1249-smpte-sdi-interfaces-7series-gtx-transceivers.pdf
41. Downlodable testbench Code., https://github.com/gic81/verilogtestbench

Figures

Figure 1 Standard Measure test. SDI Video equipment connected to the Device Under Test. The right side show the main testing parameters that can be analyzed using a Hardware Wave Monitor.

Figure 2 HD-SDI protocol. HD-SDI components stream example.

Figure 3 Testbenches application. *(Left)* Standard test-bench model. *(Right)* Video test-bench framework with double transceiver system for SDI simulation. Blocks 1 to N represent the video processing chain.

Figure 4 Testbenches. Testbench applied to a real case: The experiment validation model is ported on a Virtex-6 Xilinx FPGA.

Figure 5 Chain description. Complete chain functionality block diagram.

Figure 6 Custom generator description. Custom generator for Macro Blocks check. On the left side, the sorted components input line is shown. On the right side, the expected Macro Blocks output is shown.

Figure 7 Frame description. Macro Blocks HD 720p frame subdivision example.

Figure 8 Macro Block extraction experiment description. The simulation screenshot represents the Macro Block extraction process.

Figure 9 Line reconstruction experiment description. The simulation screenshot corresponds to Line reconstruction experiment.

Figure 10 Complete chain experiment description. The simulation screenshot shows a complete chain experiment with Bar Generator.