A RANDOM CONSTRAINED MOVIE VERSUS A RANDOM UNCONSTRAINED MOVIE APPLIED TO THE FUNCTIONAL VERIFICATION OF AN MPEG-4 DECODER DESIGN

George S. Silveira, Karina R. G. da Silva and Elmar U. K. Melcher
Universidade Federal de Campina Grande
Aprigio Veloso Avenue, 882, Bodocongo, Campina Grande - PB, Brazil

Keywords: Randmovie, movie, stimuli,VeriSC, SystemC, functional coverage, verification.

Abstract: The advent of the new VLSI technology and SoC design methodologies has brought about an explosive growth to the complexity of modern electronic circuits. One big problem in the hardware design verification is to find good stimuli to make functional verification. A MPEG-4 decoder design require movies in order to make the functional verification. A real movie applied alone is not enough to test all functionalities, a random movie is used as stimuli to implement functional verification and reach coverage. This paper presents a comparison between a random constrained movie generator called RandMovie versus the use of a Random Unconstrained Movie (RUM). It shows the benefits of using a random constrained movie in order to reach the specified functional coverage. With such a movie generator one is capable of generating good random constrained movies, increasing coverage and simulating all specified functionalities. A case study for an MPEG-4 decoder design has been used to demonstrate the effectiveness of this approach.

1 INTRODUCTION

Hardware architectures are becoming very complex nowadays. These designs are composed of microprocessors, micro-controllers, digital signal processing units and Intellectual Properties cores (IP cores) designed to perform a specific task as a part of a larger system. In order to implement these complex designs and the chips, the implementation should be composed of many project phases such as: specification, RTL implementation, functional verification, synthesis and prototyping.

Functional verification represents the most difficult phase of all. Literature shows that 70% of all project resources are involved in this process (Bergeron, 2003).

Functional verification can be used to verify if the design has been implemented in agreement with specification. It uses simulation to verify the DUV - Design Under Verification. During simulation all results coming from the DUV are matched to the results coming from a Reference Model (Golden Model). Verification can only achieve the required objective if all specified functionalities are exercised and verified.

One of the challenges posed by functional verification is that of applying good stimuli in order to exercise the specified functionalities. Some techniques attempt to solve this problem by using randomly generated stimuli (James Monaco and Raina, 2003) and (ahi). Random stimulus can emulate automation: Left to its own, a properly designed random source may eventually generate the desired stimulus. Random stimulus will also create conditions that may not have been foreseen as significant. When random stimuli fail to produce the required stimulus, or when the required stimulus is unlikely to be produced by an unbiased random stimuli source, constraints can be added to the random stimulus to increase the probability of generating the required stimulus. Although randomly generated stimuli are often better than direct stimuli (Bergeron, 2003), one can not be certain whether all specified functionalities have been exercised, being, as it is, impossible to show the existence of any functionality that has not been exercised. To get around this problem one must use coverage measurements together with random stimuli. Coverage, in its broadest sense, is responsible for measuring verification progress across a plethora of metrics, helping

S. Silveira G., R. G. da Silva K. and U. K. Melcher E. (2008).
A RANDOM CONSTRAINED MOVIE VERSUS A RANDOM UNCONSTRAINED MOVIE APPLIED TO THE FUNCTIONAL VERIFICATION OF AN MPEG-4 DECODER DESIGN.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 271-278
DOI: 10.5220/0001938602710278

the engineer to assess the rating of these metrics relative to the design specification (Piziali, 2004).

Random generation can automate test vector generation but is worthy only with constraints applied along with coverage measurement. Constraints are used to avoid the generation of illegal stimuli as well as to steer toward interesting scenarios and the coverage approach is used to measure the verification performance of the system. Some works have been produced aiming at producing good random movies, as (G.Miyashita, 2003) and (Seong-Min Kim and Kim, 2003).

The purpose of this work is to use an MPEG-4 decoder design to compare approaches of stimuli: Random-constrained and Random-Unconstrained Movie. For that matter Random-Unconstrained Movie has been used in the MPEG-4 simulation, but the Bitstream processor (BS) - part of MPEG-4 decoder - has not achieved the desired coverage by means of these stimuli. Consequently, in order to verify the BS, a synthetic movie generator to create pseudo-random images was implemented. The synthetic movie generates pseudo-random images in MPEG-4 Simple Profile Level 0.

The remaining of the paper is organized as follows: Section 2 shows the MPEG-4 decoder design; Section 3 explains the architecture approach; Section 4 deals with implementation of the movies; Section 5 presents the results, and Section 6 draws some conclusions.

2 MPEG-4 DECODER DESIGN

MPEG-4 is an open standard created by the Motion Picture Experts Group (MPEG) to replace the MPEG-2 standard. MPEG-4 coding can be carried out in various profiles and levels. The reason for such a division is to define subsets of the syntax and semantics. This is why MPEG-4 fits into a variety of applications, some of which can take advantage of VLSI implementation optimizing and power dissipation.

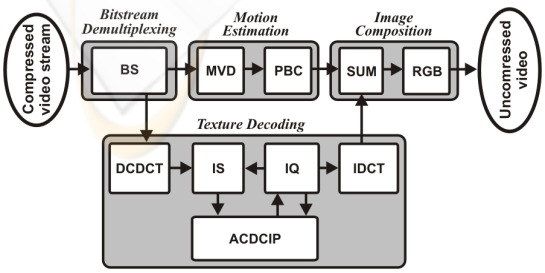

Figure 1: MPEG-4 decoder schema.

The used MPEG-4 movie decoder IP core under consideration is a Simple Profile Level 0 movie decoder. The block schematic of the MPEG-4 decoder is described in Figure 1. The Reference Model is the XVID software (Team, 2003).

Today this MPEG-4 IP core is implemented in a silicon chip. It has $22.7mm^2$ at a 0.35μm CMOS 4 ML technology with a 25 MHz working frequency (Ana Karina Rocha and Araujo, 2006).

The MPEG-4 decoder IP-Core verification was implemented using VeriSC methodology and the BVE-COVER coverage library (K. R. G. da Silva and do N. Cunha, 2007). A hierarchical approach was employed for verification. The MPEG-4 was divided in modules and each module was verified separately (K. R. G. da Silva and Pimenta, 2004).

Verification was carried out for each module of the MPEG-4 decoder, MVD (Motion Vector Decoding), MVD_VOP (Motion Vector Decoding Video Object Plane), PBC (Prediction Block Copy), DCDCT (Decoding Coefficient for DCT), IS (Inverse Scan), ACDCPI (AC and DC Prediction for Intra), IDCT (Inverse DTC), IQ (Inverse Quantization) and BS (Bitstream Processor). In the verification phase, it was used as test vector, specific random stimuli for each block of MPEG-4, always measuring the specified coverage. Most of the blocks reached 100% coverage during verification, except the Bitstream processor (BS).

The Bitstream processor is a special module. It can be simulated only using movies as input, because it is the first module of the MPEG-4 decoder. The BS is a module that receives the compressed movie stream in the MPEG-4 format and feeds the other blocks with the proper data and/or configuration parameters so that, each one is able to execute its function. Figure 2 shows the block schematic of the BS.

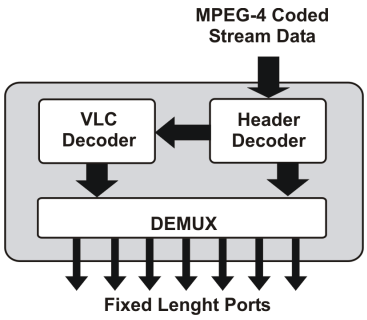

Figure 2: Bitstream schema.

In the BS verification module was first used a Random-Unconstrained Movie as stimuli, as shown in the next subsection.

2.1 Random-Unconstrained Movie Architecture

The architecture of the Random-Unconstrained Movie generator has been designed to be simple, flexible and reusable. The Random-Unconstrained Movie is a movie that uses pure randomization, where a range of values are generated and there are no constraints applied to create scenarios and reach specified coverage aspects. It could be used in the testbench of the MPEG-4 IP core or any other video-based systems as an input movie to stimulate all the functionalities. In order to implement the Random-Unconstrained Movie a frame generator has been developed. Is created a frame QCIF (177x144), each pixel is selected randomly between a range [0,255]. Produced frames are submitted in sequence to an encoder that compresses the video in the desired format. Figure 3 shows the Random-Unconstrained Movie architecture.

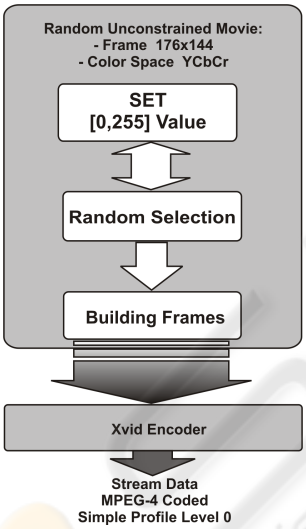

Figure 3: Random-Unconstrained Movie architecture adopted.

2.2 MPEG-4 Coverage

The MPEG-4 verification first attempt was accomplished using Random-Unconstrained Movie together with random stimuli. The Bitstream module (BS) did not achieve 100% coverage, but all the other modules achieved 100% coverage, as shown in Table 1.

BS block is a dedicated processor implemented to demultiplex multimedia streams. The BS features are presented in Table 2.

The BS coverage was measured in its output ports, because it was necessary to verify if the BS was generating demultiplex stream data correctly. The Bitstream has 7 output ports (2 with image data and 5 with configuration data).

Table 1: MPEG-4 coverage accumulated blocks.

Modules	RUM Coverage %
MVD	100
MVD_VOP	100
PBC	100
DCDCT	100
IS	100
ACDCIP	100
IDCT	100
IQ	100
BS	67

Table 2: BS Features.

RTL SystemC	1667 lines of source code
SystemC Testbench	2607 lines of source code
ROM Size	16K
Logic Elements in Altera Stratix II FPGA	6000

2.3 The Bitstream Low Coverage Causes

The BS low coverage, lead to an analysis of the simulation data, and it was discovered that the problem was caused due to the movie stream used as simulation input. The problem was caused by the following reasons: The AC coefficients vary from each other by a small threshold, due to the characteristics of the quantization method used by video encoders, which (depending on quantization parameters) may cause a significant loss to the high frequency coefficients. This loss may result in a significant distortion from the original image and, consequently to the low variation of the AC coefficients. The Figure 4 show a coefficients block example passing by the quantization process.

Another problem was the low motion vector variation. This occurs because the search of a reference pixel of a previous frame yields the most similar pixel to the current pixel. Due to the excess of information from the previous frame, during the search of pixels block to serve as reference, the encoder will have a lot of similar blocks option to be used as block of pixels reference when generating the vector. These have very low differences amongst their coordinates which result in very small motion vectors, as shown in Figure 5.

As shown in Table 3, the DCDCT and MVD output ports of the BS module have a low coverage rating because of the characteristics of the encoding process.

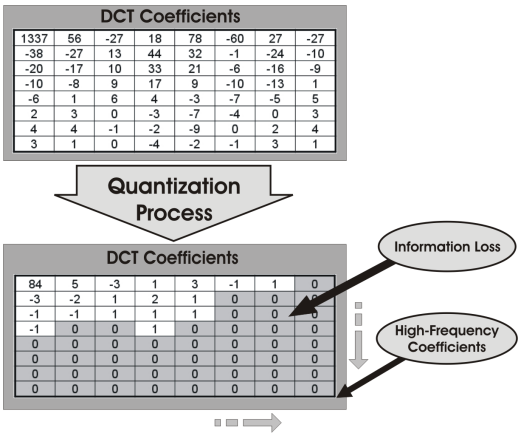

Figure 4: Quantization process.

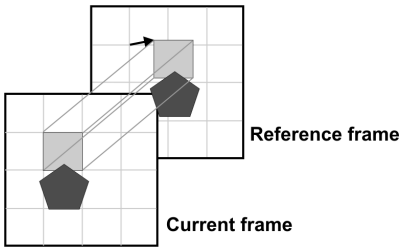

Figure 5: Low variation vectors.

Due to the coverage problems, as explained in this section, was not possible to achieve the coverage in the functional verification, than was necessary to look for other solutions to improve the required coverage. Then, was chosen a Random-constraint process to be applied to a movie generator. This movie generator was implemented and is called RandMovie. It is capable of putting into operation a set of functionalities to guarantee that the MPEG-4 will be exercised in order to achieve its specified coverage, as explained in next section.

Table 3: BS output port coverage.

BS output port	RUM Coverage%
MVD	**47**
MVD_VOP	55
PBC	54
DCDCT	**43**
IS	61
ACDCIP	64
IQ	57
Total Accumulated Coverage %	**67**

3 RANDMOVIE ARCHITECTURE

The RandMovie has a similar architecture from the Random-Unconstrained Movie. But, the differences are very important to create the necessary scenarios to reach coverage parameters.

In the RandMovie the generated videos are created intending to hit high level of coverage in the functional verification process. With this proposal many different scenarios were created. Figure 6 shows the RandMovie architecture.

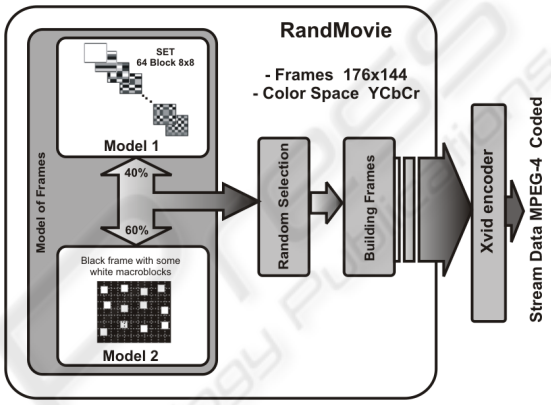

Figure 6: RandMovie architecture adopted.

The differences from RandMovie and Random-Unconstrained Movie are mainly in the frames construction, before submitting it to the encoder. To achieve this, the randomicity has to be constrained to achieve the coverage. It is constrained in two different moments:

- When it is necessary to reach the DCDCT coverage, which can not be reached by the Random-Unconstrained Movie.
- In order to constrain the MVD, the frames are implemented with less texture information between the frames. They are implemented with extremes colors, black frames with white 16x16 macroblocks inserted randomly in 32x32 pixels space.

The DCT produces an energy concentration in the DC coefficient, generating AC coefficients close to zero. Because of this, the frame generator has to stimulate the DCT to generate the AC coefficient for medium and high frequencies above a minimum value. This has to be done to guarantee that medium and high frequencies coefficients will not be suppressed by quantization process.

Any 8x8 pixels block can be represented as a sum of 64 base patterns (Rao and Yip, 1990) as shown in Figure 7. The DCT output is the set of weights for

these bases (DCT coefficients). It is necessary to multiply each base pattern by its weight and sum all of them, producing as a result a decoded image blocks.

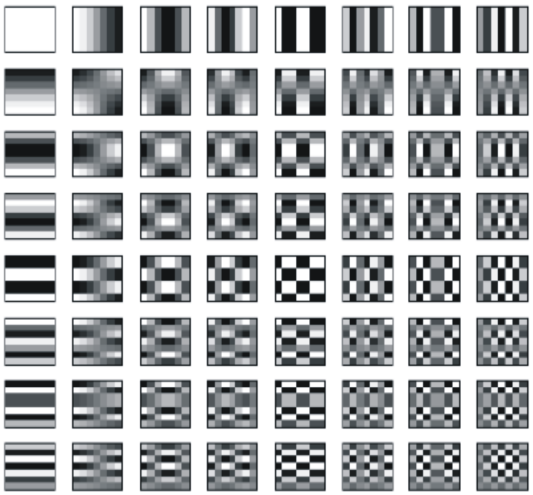

Figure 7: 8x8 DCT base patterns.

Using the base patterns, random frames can be generated in each 8x8 pixels block of the frame. It will have the visual pixels to some of the 64 blocks of the base pattern. Each 8x8 frame blocks are random selected from 64 blocks pattern. Then, each block have visual characteristics have the same base patterns as any block. This way, DCT will generate the coefficients DC and AC with values that after the quantization process will obtain AC of averages and high frequencies with significant values. Figure 6 show an example these frames.

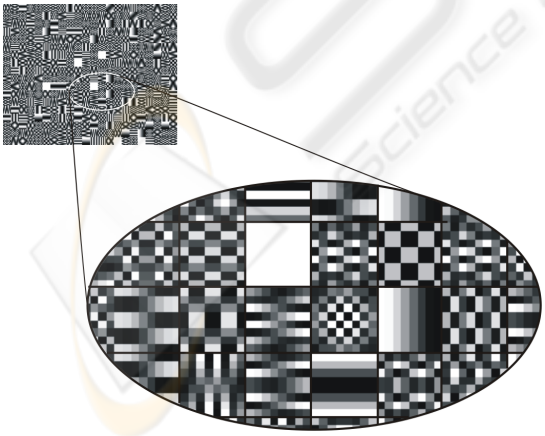

Figure 8: Frame with feature to DCT.

Motion estimation is block based and involves searching the VOP reference from the closest match to the current block to be coded. Once the block is founded, a motion vector is used to describe its location.

Motion estimation is performed on each of the four luminance blocks in a macroblock. Depending on the macroblock type up to four motion vectors are coded. Inside the current frame, the motion estimation generates a motion vector making a reference to the previous frame, searching for a pixel in the previous frame that are in the same place of the pixel in the search window in the current frame. It makes a scan inside a search window to implement the matching criteria between the pixels. This seek is not based on a perfect pixel match, but based on the group of pixel that are closest to the target pixels.

Due to the characteristics of the process of motion estimation for the encoder, it should reduce the amount of similar images inside of the search window. Thus, they can generate frames with extreme colors to maximize the difference among the pixels blocks and to add the minimum of similar pixels inside of the search window, as shown in the Figure 9.

Thus, during the search for similar blocks inside of the delimitation of the search window, the encoder won't have options of close images to the current block to use as reference. In this case, it will have to look for the similar blocks in the neighborhood of that search window, causing the generation of larger vectors.

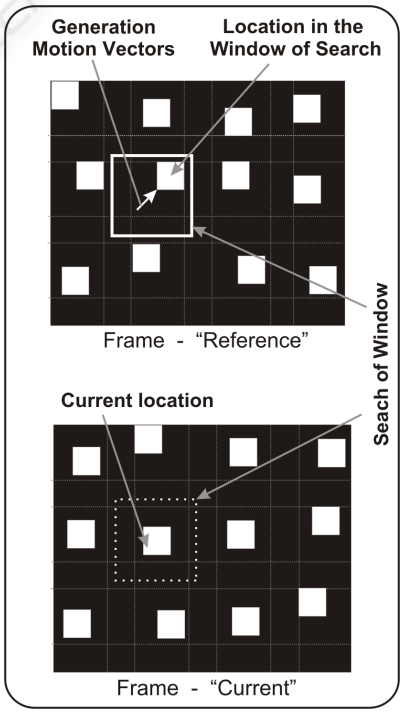

Figure 9: Motion Estimation.

4 IMPLEMENTATION

This section shows the implementation of the Random-Unconstrained Movie and the implementation of the RandMovie in order to understand the insertion of constraints in the generated movie.

4.1 Random-Unconstrained Movie Implementation

The Random-Unconstrained Movie generates random images made up of texture information, which are properly coded by means of Variable Length Coding (VLC) and Fixed Length Coding (FLC) for the header information.

The encoder features are: open source C++ to be used together with SystemC and coupled to the testbench. Xvid v0.9 was selected as an encoder because it is capable of implementing all functionalities of an MPEG-4 Simple Profile Level 0 (ISO/IEC, 2004).

The variables and value ranges were specified, in order to generate a random-constrained movie with texture and motion based on MPEG-4 standard. The main variables in this process came from the DCT e MVD modules.

The DCT parameters should reach the values for the following variables:

- Level: [-2048 to 2047], indicates the pixel value;
- Run: [0 to 63], shows the quantity of zero value before a level value;
- Last: [0 and 1], indicates the last matrix coefficient.

The MVD parameters should reach the values for the following variables:

- Horizontal_mv_data: [-32 to 32], horizontal coordinate of the motion vector;
- Horizontal_mv_residual: [0 to 32], difference between frame current and reference;
- vertical_mv_data: [-32 to 32], vertical coordinate of the motion vector;
- Vertical_mv_residual: [0 to 32], difference between frame current and reference;

The generated frames are built with dimensions QCIF (177x144), each value of the pixel is selected among a value range of [0,255].

4.2 RandMovie Implementation

The adopted strategies provides 2 ways of creating frames to the videos production: a form based on the base patterns, where each macroblock of the frame visually matches a base patterns and the other way is based on the creation of black frames with some white macroblocks of pixels inserted in random places in the frame. To guarantee that RandMovie can generate the necessary stimuli to exercise the movement estimation way necessary to constrain the randomness in which the selection of a frame format is made, because it is necessary to guarantee some sequences of black/white-doted frames inside the video. This is needed because of the strategy that the encoder uses to generate the motion vectors, as shown in the Figure 10.

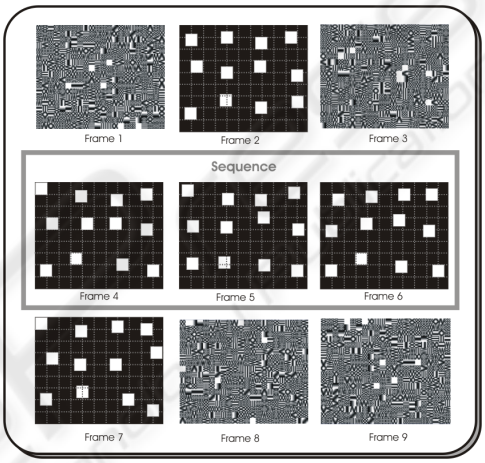

Figure 10: Sequence frames.

The modification of the randomness of RandMovie to generate efficient stimuli to texture encoding and motion estimation was the redistribution of the two ways of creating frames probability. This way, the frames using the base patterns have the occurrence probability of 60%, while the black/white-doted frames have 40% chance of being selected.

5 RESULTS

The Random-Unconstrained Movie and RandMovie were applied in the Functional Verification of MPEG-4. Then, was necessary to make new simulations in the MPEG-4 and their sub-modules in order to reach the specified coverage.

The BS module coverage was measured in 7 output ports, using the same parameter as specified for the Random-Unconstrained Movie. During the simulation, was possible to verify which values were covered in the port variables. With the results was possible to compare the final results of both movie as input in the verification. The comparison is presented in the Table 4. In this table is possible to see that the Rand-

Movie was better for almost all the variables specified. The total coverage was 97% compared to 67% coverage from the Random-Unconstrained Movie.

Table 4: Coverage between Random-Unconstrained Movie and RandMovie.

BS output port	RUM Coverage %	RandMovie Coverage %
MVD	47	100
MVD_VOP	55	100
PBC	54	100
DCDCT	43	75
IS	61	100
ACDCIP	64	100
IQ	57	100
Total Accumulated Coverage %	**67**	**97**

Another result can be seen in Figure 11. It shows improvement of the coverage in function of the time by using Random-Unconstrained Movie versus RandMovie. It is possible to see that the RandMovie reached better coverage first then Random-Unconstrained Movie. The RandMovie generator reached the specified coverage earlier and was considered satisfactory to the Bitstream module.

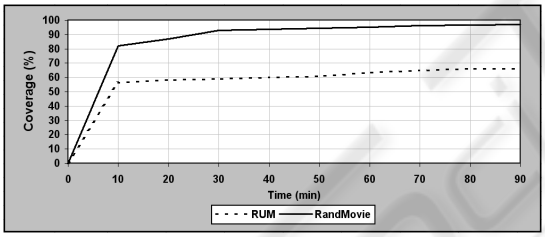

Figure 11: Comparison between Random-Unconstrained Movie and RandMovie.

Another very important result was obtained in the verification using the RandMovie. The coverage analysis revealed a coverage hole in the BS simulation results, i.e. a functionality that has not been exercised before with the Random-Unconstrained Movie. The analysis of this coverage hole revealed an error in the BS implementation. The discovered error was leading to a wrong communication between Bitstream and MVD modules. This could cause an error in the composition of the image in the MPEG-4 IP-Core. The error occurred because a register with 6 bits was used when 7 bits should be used. Due to functional coverage and the RandMovie stimuli generator, the error was eliminated in the MPEG-4 IP-Core.

RandMovie was limited by the encoder implementation, mainly the DCT coefficient generator in the Xvid encoder: it implemented saturation in the 8x8 blocks after DCT transformation. This saturation keeps values in the range [-1024, 1024]. Due to the Xvid encoder limitations, in spite of the fact that frames did present visual characteristics in the base pattern, it was not possible to stimulate the Xvid encoder sufficiently to make it generate coefficients to cover the whole range of values [-2048, 2048].

RandMovie has some advantages if we consider the related works (G.Miyashita, 2003) and (Seong-Min Kim and Kim, 2003), like the utilization of randomness applied to the video generation, assuring the high level of coverage rating, simple implementation, simple attachment to the MPEG-4 IP core testbench and flexibility to be reused directly in other kinds of testbenchs for video decoding systems. One could, for example, reconfigure Xvid for a higher resolution, or change the encoder to build a random video in H.264 format.

6 CONCLUSIONS

This paper proposes to compare two approaches of synthetic movie generation, Random-Unconstrained generation and random-constrained generation. With the constraints applied in the movie generator it was possible to generate good random-constrained movies, increasing coverage and simulating all the specified functionality with respect to a Random-Unconstrained Movie. With the RandMovie it was also possible to find a real error in the implementation of the MPEG-4 design.

The approach has the disadvantage that it depends on the capabilities of the encoder used, but analyzing the presented results, it is possible to conclude that the directed stimuli used in the random-constrained movie generation are more efficient that the presented in the Random-Unconstrained Movie.

REFERENCES

Ana Karina Rocha, Patricia Lira, Y. Y. J. E. B. E. M. and Araujo, G. (2006). Silicon validated ip cores designed by the brazil-ip network. In *IP/SOC 2006*.

Bergeron, J. (2003). *Writing Testbenches: Functional Verification of HDL Models*. Kluwer Academic Publishers, MA, USA, 2nd edition.

G.Miyashita (2003). High-level synthesis of a MPEG-4 decoder using systemc. In *Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark*.

ISO/IEC, J. T. C. (December 2004). *Information technology-coding of audio-visual objects - part 2: Visual*.

James Monaco, D. H. and Raina, R. (2003). Functional verification methodology for the powerpc 604 microprocessor. In *DAC '96: Proceedings of the 33rd annual conference on Design automation*, New York, NY, USA. ACM Press.

K. R. G. da Silva, E. U. K. Melcher, G. A. and Pimenta, V. A. (2004). An automatic testbench generation tool for a systemc functional verification methodology. In *SBCCI '04: Proceedings of the 17th symposium on Integrated circuits and system design*. ACM Press.

K. R. G. da Silva, E. U. K. Melcher, I. M. and do N. Cunha, H. (2007). A methodology aimed at better integration of functional verification and rtl design. In *Design Automation for Embedded Systems*.

Piziali, A. (2004). *Functional Verification Coverage Measurement and Analysis*. Kluwer Academic, USA, 1nd edition.

Rao, K. R. and Yip, P. (1990). *Discrete cosine transform: algorithms, advantages, applications*. Academic Press Professional, San Diego, CA, USA.

Seong-Min Kim, Ju-Hyun Park, S.-M. P. B.-T. K. K.-S. S. K.-B. S. I.-K. K. N.-W. E. and Kim, K.-S. (2003). *Hardware-software implementation of mpeg-4 video codec*. ETRI Journal, London.

Team, X. (2003). Xvid api 2.1 reference (for 0.9.x series).