A Video Decoder Components Verification Scheme based on Transaction-level Information to Randomized Behavioral-level Operations Transformation

Jiayi Zhu, and Shinji Kimura
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 807-0135, Japan
E-mail: jzhu@aoni.waseda.jp

Abstract: Verification of video decoder is a difficult work. Even each component of the decoder is so complex that takes a lot of time to verify.

In this article, we propose a video decoder components verification scheme based on the collaborations between reference software and VLSI test-bench. In the scheme, verilog models of DUT's neighbors translate the transaction-level information generated by the reference software to behavioral-level operations which interact with the DUT component. In addition, timing of the behavioral-level operations of these verilog models are randomized to improve the verification efficiency.

Our experience shows that once the DUT components pass the verifications under this scheme, the possiblitiy they fail in the verification of the whole decoder is low.

Keywords—**randomized behavioral-level operations, video decoder verification, transaction-level information**

1. Introduction

Video compression technology keeps development in last decades. Various standards like MPEG-2, MPEG-4, H.263, H.264, and HEVC have been developed. The newer standards have better performance in video compression than previous standards. Once one standard is developed, many researchers and engineers devote into the high performance VLSI video decoder implementations.

The design of VLSI video decoder is full of challenges because of its large scale. Especially, with the people's quick rising demand for higher resolution and frame rates, the, the scale of VLSI video decoder becomes larger and larger. Even the components of video decoder like motion compensation, de-blocking filter et al. are quite complex digital logic.

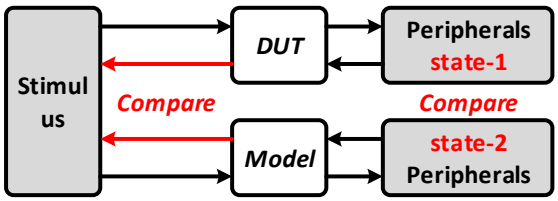

Figure 1. An example of Verification for digital VLSI design

For such large scale of logic, verification is always the critical bottleneck and occupies most of its development time. People invent many verification methods for large scale digital logic verification. One example is shown in Figure 1. The basic feature of these kind of verification scheme includes 1. It generates randomizing stimulus; 2. It compares the output of DUT (design under test) and the model which has the same function; 3. It also compares the states stored in the peripherals connected to the model and DUT to confirm that the function of DUT is correct.

Although the methods work well for many designs, they are not completely suitable for video decoder due to the following reasons:

- The effective stimulus of the video decoder are bit-streams conforming to the video compression standards. As a consequence, large amount of bit-streams designed particularly for the purpose to verify large amount of DUT corner cases are needed. However, these kind of bit-streams are uneasy to design and hence they are expensive to obtain. It is difficult for people to prepare enough bit-streams covering as many as possible corner cases to various DUT.
- The second reason is that the reference software models of video decoder components are quite complex. Therefore, only the officially released software models are regarded as authoritative models. For example, when we design the VLSI decoder of HEVE, all the output of our design shall be compared with the output of HM, which is the HEVC official software model. Only if output of our design is same to that of HM, then we can regard that the design maybe correct. All other models, especially designed by the design owner or the verification owner himself, are not regarded as authoritative.
- The third feature is that, not only each component itself is complicated. The system overall is even complicated.

Therefore, we propose a scheme for decoder verification in this article. Our work is based on the basic model as shown in Figure 1, with some particular improvements for the decoder verification. To illustrate the proposed scheme, we use the HEVC In-Loop Filter shown in Figure 2, which contains deblocking filter and SAO (Sample Adaptive Offset), as the DUT example. As shown in Figure 2, the input of deblocking filter is reconstructed picture and deblocking filter parameters. The output of deblocking filter is the deblocked picture. The input of SAO is deblocked picture and SAO parameters. The output of SAO the the picture after SAO, i.e. the decoded pictures. The deblocking filter and SAO together form the in-loop filter of HEVC decoder.

The feature of our proposed verification scheme is as following:

- Software-hardware cooperated verification
- Intermediate transaction-level information
- Randomizing behavioral-level operation timing

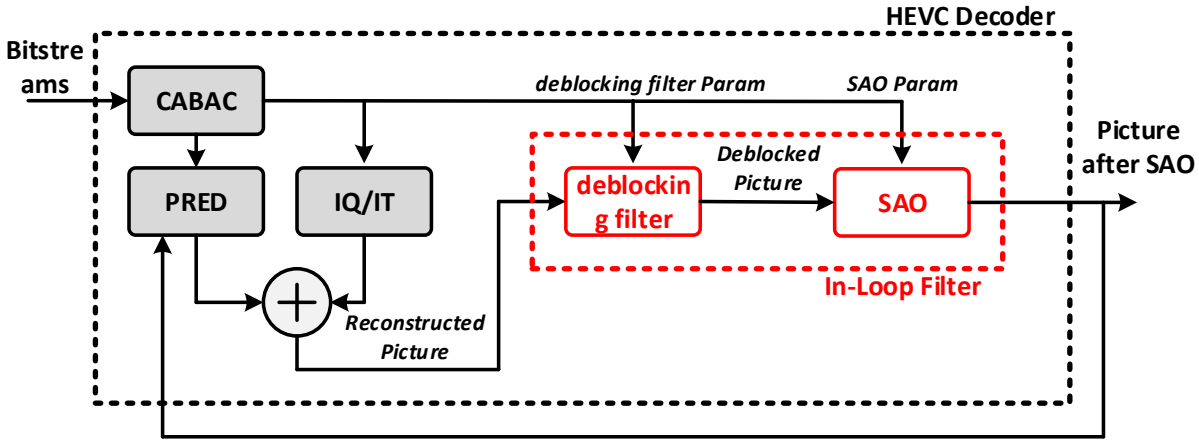

Figure 2. In-Loop Filter in HEVC Decoder

- Peripherals can be replaced by other DUT to form the whole video decoder.

The rest of the article is organized as following: Section 2 talks about our proposed scheme. Section 3 talks about the trial performance of our proposal. Section 4 concludes this article.

2. Proposed Scheme

2. 1 Overview and verification flow

As mentioned above, we take the deblocking filter and SAO of HEVC decoder as example, which is shown in Figure 2.

To verify the two components (deblocking filter and SAO), three factors are needed: the stimulus, peripherals, and output compare mechanism. The overview of the proposed scheme is shown in Figure 3. The upper half is the reference software part. The lower half is the DUT part.

The stimulus are "Reconstructor model", "DBF param model", and "SAO param model" in the lower half of Figure 3. The three models read files recording transacation-level information, which are generated by the software when it decodes bit-streams.

The periphrals in the example of Figure 3 is in the "Receiver model & compare". The receiver model interact with the SAO DUT and accept the output data of SAO.

The output compare mechanism is the "Monitor & compare" and "Receiver model & compare". The former one compares the output of deblocking filter DUT with the output of deblocking filter in official reference software model. The latter one compares the output of SAO DUT with the output of SAO in official reference software model. Any unconsistence found, then the simulation stalls, which means it is possible that there exist bugs in the design.

2. 2 The basic verification flow

- Input a bit-stream "Bit-stream" to the official reference software model and run the software, as shown in the upper half of Figure 3.
- Obtain five files which are printed out by the official reference software model, as shown in the upper half of Figure 3. The five files are "Reconstructed Picture", "deblocking filter param", "Deblocked Picture", "SAO param", and "Picture of SAO". In these five files, the transaction-level information are recoded. For example, in "SAO param", for each CTU, the SAO parameters like sao_merge_left, sao_merge_upper, sao_type_luma, sao_type_chroma, sao_start_band_position_luma, ... are recorded. These data are read out by the "SAO param model" in the lower half in Figure 3.
- Three models (Reconstructor, DBF param, SAO param) shown in the lower half of Figure 3 read the corresponding files obtained in last step and turn the transaction-level information to behaviral operations which directly interact with the DUT (deblocking filter and SAO).
- When the DUT works, the “Monitor & Compare” module in Figure 2 monitors the output of deblocking filter and compare it to the data read from the file “Deblocked Picture”. It stops and warns once unconsistency found in comparsion.
- When the DUT works, the “Receiver model & Compare” module in Figure 2 receives the output of SAO and compare it to the data read from the file “Picture after SAO”. It stops and warns once unconsistency found in comparsion.
- If reference software finish the decoding of a whole bit-stream and all the generated files are read to the bottom and no warns are reported, then we judge that this bit-stream pass the test.

2. 2 Randomizing behavioral-level operations timing

The behavioral-level operations of the models of DUT’s neighbors shall be randomized. It means that the behavioral of the model is not fixed. All the signals of the model interface can be any possibility as long as it is allowed according to the protocol.

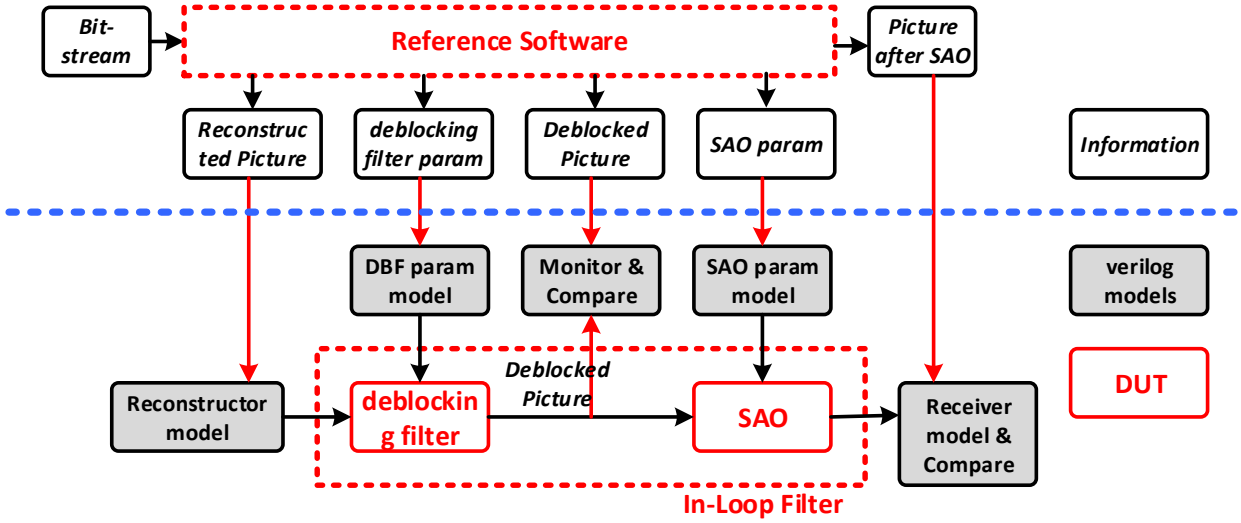

Figure 3. Framework of HEVC In-Loop Filter Verification Scheme

One example to illustrate it is shown in Figure 4, suppose model X is one neighbor of DUT Y and their connections are three signals (WE_N, WACK, and WDATA) as shown.

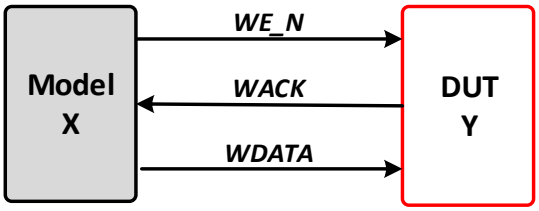

Figure 4. DUT Component and one of its neighbor

The model X output two signals: "WE_N" and "WDATA". Its behavioral operations shall be randomized means that the timing of model X output can be any legal case. For example, as shown in Figure 5 and Figure 6, the operations of "WE_N" in the two figures are different but both are legal. In Figure 5 "WE_N" is ineffective in clk0 and effective in clk1, clk2, and clk3. "WACK" is effective in clk0, clk1, and clk3. It is ineffective in clk2. Hence, in clk1 and clk3, both "WE_N" and "WACK" are effective and "WDATA" are successfully transmitted. In Figure 6, the operation of "WE_N" is different with that in Figure 5, it is effective in clk0, clk1, and clk2. It is ineffective in clk3. "WACK" is effective in clk0 and clk2. It is ineffective in clk1 and clk3. Therefore, "WDATA" are successfully transmitted in clk0 and clk2, in which both "WE_N" and "WACK" are effective.

It is meaningful to enable all the model output operations be possible. The following Verilog is use to enable the function:

'while({$random}%(A)>=B) @(posedge clk);

The parameters "A" and "B" are two parameters deciding the possibility that the condition is satisfied. B should be less than A, otherwise the condition is always false. For example, if A is set to be 10 and B is set to be 5, then {$random}%(A) generate a number between 0-10 with equal possibility for each value. If {$random}%(A) is 5, 6, 7, 8, 9, 10, then the condition is true. If {$random}%(A) is 0, 1, 2, 3, 4, then the condition is false. The possibility of condition true is 6/11. If set A to be 9 and B to be 5, then the possibility of condition true is 50%. When the condition is false, then one clock cycle is ticked. Otherwise, the statement is passed and the following code will be executed.

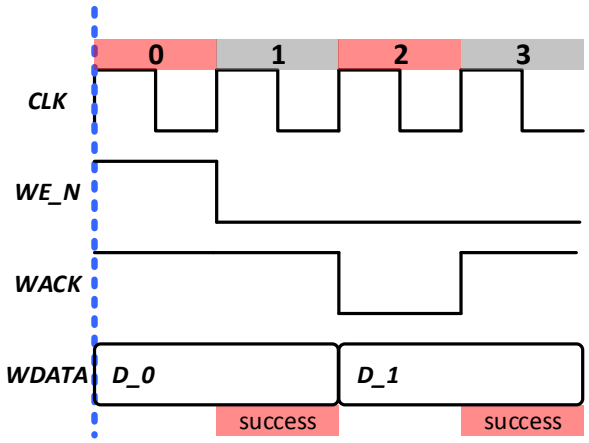

Figure 5. WE_N operation case 1

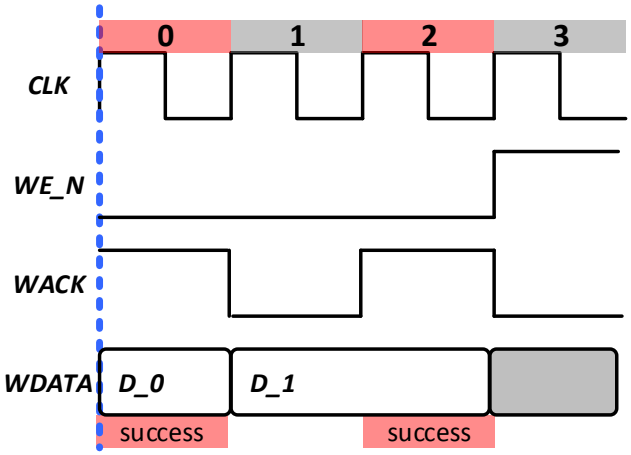

Figure 6 WE_N operation case 2

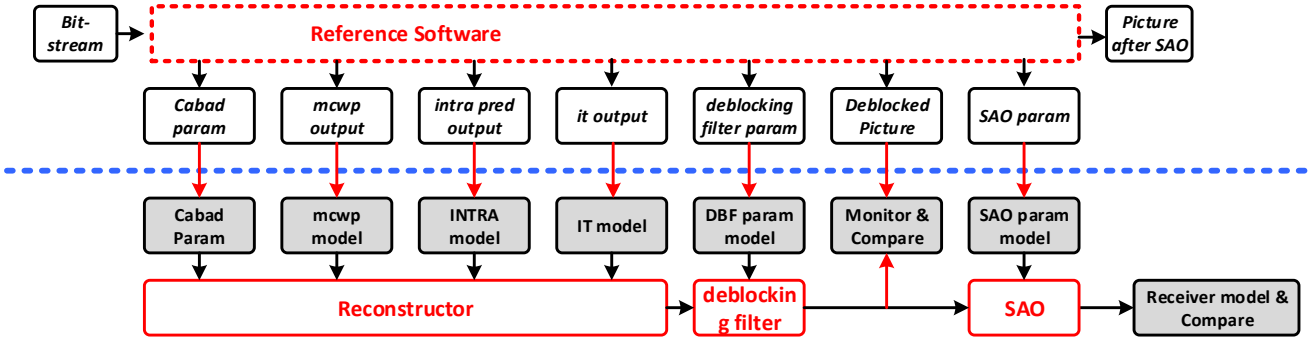

Figure 7 Expanded framework to verify deblocking filter, SAO and reconstructor

2. 3 System-level expand in verification

This verification scheme can be used to verify all other components in our HEVC decoder chip [2]. It is expanded to include multiple components in the scheme. As shown in Figure 7, the reconstructor is also verified together with deblocking filter and SAO.

The reconstructor model in Figure 3 is replaced by the reconstructor DUT shown in Figure 7. In order to make the reconstructor DUT work, four models serve as the stimulus are created. They are "Cabad Param", "mcwp model", "INTRA model", and "IT model" shown in Figure 7. The data of these models are from the files generated by the reference software, which records the transaction-level information of corresponding data.

The scheme can be extended to more other components and finally the whole decoder can be included. This scheme allows the components to be verified in the same time respectively. After the verification of each components are verified, multiple components are connected and then verified. The files recording the transaction-level information can be used not only for verification of individual components, but also for the verification of the whole decoder.

3. Experimental Result

This verification scheme is used to verify all the components in our HEVC decoder chip [2]. We tested 404 bit-streams. Under the condition that the interface between DUT and its neighbors' model is same to the interface between DUT and its real neighbors, the proposed scheme does not miss a bug. If all the components pass all of these bit-streams, then after they are connected, the possibility that the whole decoder pass the test is high.

4. Conclusion

In this article, we propose a verification scheme for the verification of video decoder. The idea is to use models to translate transaction-level of information generated by reference software to behavioral-level operations which interact with DUT. Another important point is to randomize the behavioral-level operations timing, which allows various corner case to happen and increase the possibilities to find bugs. In addition, our scheme can verify all the components within the decoder and it eases the verification of the whole video decoder.

References

[1] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, "High Ef-ficiency Video Coding (HEVC) text specification draft 8", document JCTVC-K1003, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 11th meeting, Shanghai, CN, 10-19 September 2012.

[2] Dajiang Zhou, Shihao Wang, Heming Sun, Jianbin Zhou, Jiayi Zhu, Yijin Zhao, Jinjia Zhou, Shuping Zhang, Shinji Kimura, Takeshi Yoshimura, and Satoshi Goto, "A 4Gpixel/s 8/10-bit H.265/HEVC Video Decoder Chip for 8K Ultra HD Applications", International Solid-State Circuits Conference (ISSCC), pp. 266-268, 31 January - 4 February 2016.

[3] Jiayi Zhu, Dajiang Zhou, Gang He, and Satoshi Goto, "A combined SAO and de-blocking filter architecture for HEVC video decoder", IEEE International Conference on Image Processing (ICIP), pp. 1967-1971, 15-18 September 2013.

[4] Jiayi Zhu, Dajiang Zhou, and Satoshi Goto, "A High Performance HEVC De-Blocking Filter and SAO Architecture for UHDTV Decoder", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E96-A, no. 12, pp. 2612-2622, December 2013.