Multimedia IP DMA verification platform

Suhyung Kim, Sangkyu Park, Myungwoo Seo, Sangjin Lee, Jiyeon Park
{soohyk.kim, sangq.park, mw.seo, sj103.lee, jyoni.park}@samsung.com
Samsung Electronics Co., Ltd. 1-1, Samsungjeonja-ro, Hwaseong-si, Gyeonggi-do, 18448, Korea

***Abstract*- DMA verification is usually done by senior verification engineer because even a single bus protocol violation could hang up the whole SoC. In this paper, we introduce a DMA verification platform for multimedia IP that could be easily used by verification beginners and could accumulate DMA verification knowledge how of experienced engineers without much effort. Through this platform, even a novice verification engineer could develop a new DMA testbench in 30 minutes and we could verify several DMA instances at the same time with sufficient verification quality.**

I. Introduction

In recent Exynos SoC, most multimedia IPs require direct external memory access to process large amount image or voice data. For this reason, the multimedia IPs have several DMA instances to read or write various types of data. One example can be an image scaler which directly reads images from and writes to DRAM through the bus. If the DMA instance has a bug, bus system could be hanged and this could affect whole SoC. Therefore, DMA should be verified very carefully to obtain a satisfactory verification quality. So one senior verification engineer could normally only verify up to two or three DMA instances at the same time. However, in recent years, the number of DMAs in multimedia IPs for Exynos SoC are increasing to meet the market's needs for the processing power. And it became increasingly difficult to meet the verification schedule where the number of senior verification engineers are limited. In order to overcome this situation, we created Multimedia IP DMA Verification Platform that could be easily used by verification beginners and could accumulate DMA verification knowledge how of experienced engineers to this platform. So far, this verification platform has undergone several refactoring processes for more convenient use. Currently the platform is well established in our team and make it possible to perform several DMA verification projects successfully with less manpower. In this paper, we will describe what the Multimedia IP DMA Verification Platform (hereafter we call it MDVP) is, and its development process of MDVP including why, and how it was developed.

II. Background

Unlike general purpose DMA for CPU, multimedia IP DMA of Exynos SoC has a simple structure and is included in multimedia IP as shown in Fig. 1.

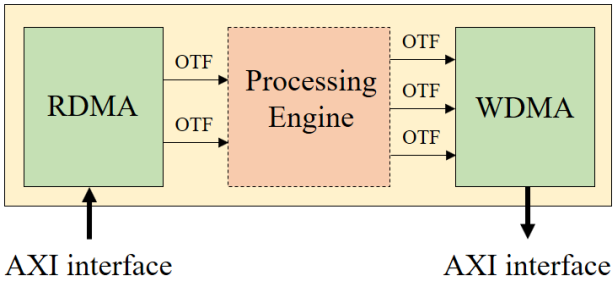

Figure 1. The architecture of general multimedia IP that has bus interface.

However, multimedia IP DMAs have various types as they have to support different types of data format and interface protocol between DMA and processing engine. As a result, many DMA instances for multimedia IP are verified in one SoC project and every DMA instance has some common items that should be checked:

1. Check if the RTL output is matched with the expected value.
2. Check if the protocol of on-the-fly interface is correct.
3. Check if there is all bus request are expected.
4. Check AXI protocol

Among the four items above, item 3 and 4 are related to the bus and therefore to thoroughly check those items verification engineer should be an expert in entire system. This makes novice engineer hard to conduct DMA verification without any help. Hence, in the past, mainly senior verification engineer verified multimedia IP DMA

and the senior engineer made a new testbench by utilizing the testbench of the previous project as much as possible to perform new DMA verification because given time was not enough.

The senior verification engineer had to perform the following tasks in the process of creating a new DMA testbench:

1. Check differences between the specification of new DMA and previous testbench.
2. Modification or implementation of driver and monitor according to the specification of on-the-fly interface.
3. Maintenance of DMA reference model if necessary.
4. Maintenance of scoreboard to match output with golden value if necessary.
5. Generation of compressed data if DMA supports data compressor.
6. Maintenance of VIPs such as valid bus transaction checker and AXI slave memory model.
7. Maintenance of regression environment.

In addition, the senior verification engineer had to continue to help the junior or novice verification engineers directly or indirectly by conducting code review on the existing testbench or sharing the developed VIP to get the satisfactory verification quality of DMA verification. For this reason, even though the senior verification engineer could verify more, senior verification engineer in one SoC project generally verified only two or three DMA instances.

Recently, as the function of multimedia IP has greatly expanded, the development of DMA instance has also increased dramatically. However, the number of senior verification engineer are limited and therefore the DMA development has become a critical path in the whole multimedia IP development.

To overcome this situation, we devised MDVP for four reasons: 1) the testbench was required to get enough verification quality even if novice verification engineer do the job. 2) initial testbench, that can work well, was necessary to be released in a short time. 3) we wanted to maximize the amount of codes being reused. 4) and we also wanted to provide an environment that can rapidly ramp up novice verification engineer. We will describe a detail of MDVP at the next chapter.

III. WHAT IS MDVP

MDVP is UVM based verification platform specialized for Multimedia IP DMA. MDVP consists of a DMA reference model for multimedia IP DMA, a reusable testbench that can be shared among different DMA instances and a configurable UVM testbench according to the specification of the DMA instance. [1]

A. Input Data for multimedia IP DMA

Generally, the behavior of the multimedia IP DMA does not vary depending on the patterns of data to write or read. So random data are normally used as input data. However, in case of DMA including data compressor to reduce bandwidth or memory footprint, random data input may not be sufficient. Therefore, MDVP provides random data pattern and non-random data pattern (e.g. image) as input of DMA. In particular, in the case of DMA for image processing, we use ImageMagick software to generate various natural data patterns that desired image resolution, pixel bit width and image format such as bayer, RGB and YUV, based on the tens of thousands predefined image DB [2].

B. Reference model for MDVP

Reference model of multimedia IP had its own DMA model but it was not suitable for use as the DMA reference model in MDVP because the IP DMA model was implemented as various form and it took a lot of time to convert the IP DMA model to the DMA reference model for MDVP. So we developed newly DMA reference model for MDVP that supports all kinds of memory format for multimedia IP DMA in Exynos SoC through analyzing every IP DMA model.

The main roles of DMA reference model are pixel formatting and converting memory map using input data pattern and configuration of DMA such as base address, stride, pixel format and memory format. The outputs of DMA reference model of MDVP consist of two type's information as below:

1. Valid address information: they are used for valid bus transaction checker.
2. Expected value: they are used as input for RDMA and reference data for WDMA.

In MDVP, valid address and expected value information is stored in the form in Fig. 2. We call it as info_queue and data_queue. And since info_queue is created only once in dma sequences, there is an advantage that UVCs and VIPs do not need to recalculate valid addresses.

```
bit [31:0] dma_info_queue[$];
bit [31:0] dma_data_queue[$];

// 1st valid address range and expected data

// Start address of 1st valid address range
dma_info_queue[2*0+0] = 32'h0000_1000;
// Number of bytes that are valid continuously from the address (=dma_info_queue[2*0+0])
dma_info_queue[2*0+1] = 7;

dma_data_queue[0] = 32'h8765_4321;
dma_data_queue[1] = 32'h00ed_cba9;

// 2nd valid address range and expected data

// Start address of 2nd valid address range
dma_info_queue[2*1+0] = 32'h0000_2000;
// Number of bytes that are valid continuously from the address (=dma_info_queue[2*1+0])
dma_info_queue[2*1+1] = 4;

dma_data_queue[2] = 32'h89ab_cdef;
```

Address	Data
0x1000 + 0	0x87 0x65 0x43 0x21
0x1000 + 4	0xed 0xcb 0xa9
⋮	⋮
0x2000 + 0	0x89 0xab 0xcd 0xef
0x2000 + 4	

Stored data in external memory

Figure 2. The example of info_queue and data_queue

C. Reusable testbench

Most of the multimedia IP DMA have a similar structure. In this case, the majority of testbench can be reused by using a DUT Wrapper that instantiates DMA as shown in Fig. 3. Therefore, verification engineer does not need to spend time debugging and they can proceed with faster and more effective verification.

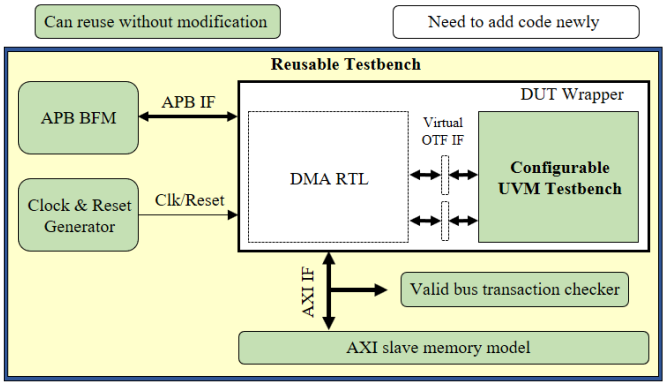

Figure 3. The architecture of reusable testbench of MDVP

Reusable testbench consists of a bus functional model for APB, clock and reset generator, valid bus transaction checker and AXI slave memory model. Among them, valid bus transaction checker and AXI slave model will be described in detail below:

1. Valid bus transaction checker: In MDVP, we defined the role of scoreboard for WDMA verification as checking whether there is expected value in the expected address after finishing DMA operation. However, if DMA issues an unintended write bus transaction to access the invalid address region, the scoreboard cannot detect this. And in case of RDMA, scoreboard for output of on-the-fly interface cannot detect unintended read bus transactions of external memory if unintended read data is not used as output. So this unintended bus request should be detected in somewhere because the request can affect performance degradation of DMA. In the MDVP, valid bus transaction checker uses valid address information of DMA, which is made by DMA reference model, as input and the checker can detect unintended bus request by checking an address whenever a bus request occurs.

2. AXI slave memory model: Most EDA venders provide AXI slave memory model in VIP form and many verification engineers are using the VIP to conduct DMA or bus verification. In the case of multimedia IP DMA verification, however, we often use only a part of the VIP function provided by EDA vender. So we have developed in-house AXI slave memory model that is optimized for multimedia IP DMA verification. By integrating in-house AXI slave model, we were able to increase simulation speed by 100% compared to the testbench using commercial VIP and it had a great influence on regression turn-around-time reduction. Moreover, we were able to reduce VIP license cost as well.

D. Configurable UVM Testbench

The main differences among multimedia IP DMA instances are the types of on-the-fly interfaces connected to the processing engine and the formats of data written to or read from the external memory. In order to verify DMA instances that have such differences with one testbench, driver, monitor, checker and scoreboard are required for each on-the-fly interface and a memory format converter, which can express various memory format, is also needed.

Therefore, the configurable UVM testbench of MDVP provides various UVCs corresponding the on-the-fly interface and provides also a UVM testbench skeleton consisting of various pre-implemented sequences and methods such as input data generator, DMA reference model, UVC linking method and common DMA operation sequence so that new DMA testbench can be released quickly with MDVP even if novice verification engineer do.

Fig. 4 shows the architecture of configurable UVM testbench in MDVP.

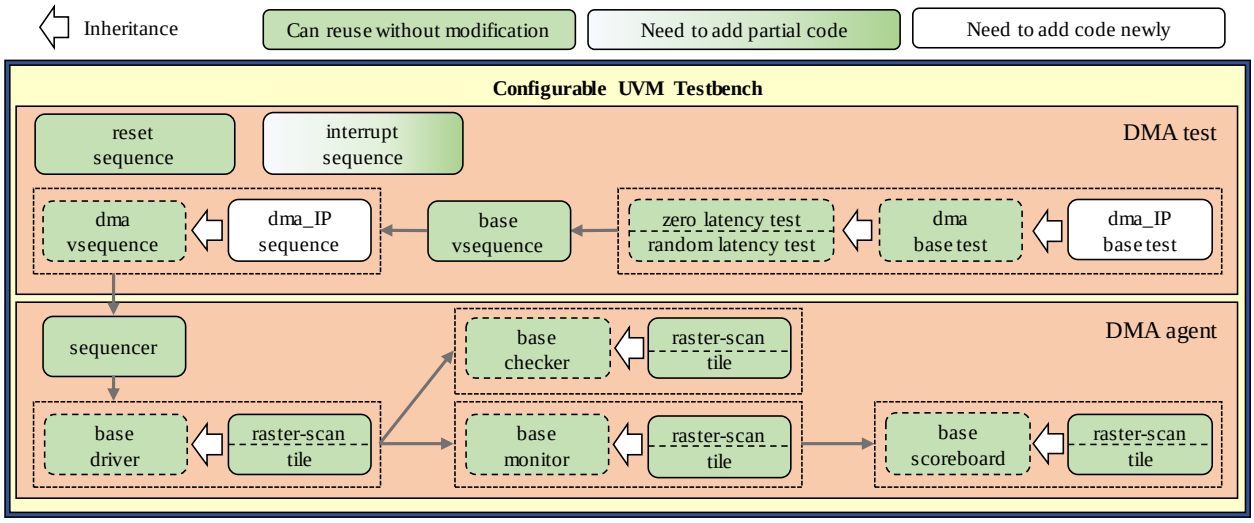

Figure 4. The architecture of configurable UVM testbench of MDVP

This configurable UVM testbench has undergone several refactoring processes over several projects. The earliest configurable UVM testbench was developed focusing on the reusability of the implemented UVCs. Below describes how to create a reusable UVC and how to conveniently use it in multimedia IP DMA verification:

1. DMA agent: MDVP provides DMA agent which driver, monitor, checker and scoreboard were connected with uvm_analysis_port. When the development of UVC for the new on-the-fly interface was required, the DMA agent can help engineers to focus only on component implementation regardless of the testbench architecture.

2. DMA base test: MDVP provides the guidance to the way factory overriding for UVCs. In multimedia IP DMA verification, the configuration of UVCs does not need to be changed dynamically on the simulation run-time. So each DMA verification engineer can configure the desired DMA verification environment simply through factory overriding for UVCs required for DMA verification in the dma_IP_base_test class. And all test scenario classes inherited DMA base tests, so redundant code for UVC configuration could be reduced.

3. Parameterized component and interface: Multimedia IP DMA uses various types of on-the-fly interfaces and these interfaces may have the same type but can differ only in the bit width of signal. Therefore, we implemented interfaces and UVCs using parameter of bit width of signal as shown in Fig. 5. [3]

```
interface tile_if#(DATA_WIDTH=10, POS_X=12, POS_Y=12) (input CLK, input RESETn);
  logic [POS_X-1:0] pos_x;
  logic [POS_Y-1:0] pos_y;
  logic [DATA_WIDTH-1:0] data;
    ⋮
endinterface : tile_if

class tile_driver_c#(DATA_WIDTH=10, POS_X=12, POS_Y=12) extends driver_c;
  `uvm_component_param_utils(tile_driver_c#(DATA_WIDTH, POS_X, POS_Y))

  virtual tile_if    #(DATA_WIDTH, POS_X, POS_Y)     i_if;
  tile_seq_item_c#(DATA_WIDTH, POS_X, POS_Y)     i_seq_item;
    ⋮
endclass : tile_driver_c
```

Figure 5. The example code for parameterized interface and component

By using the above technique, the DMA verification team consisting of one senior engineer, two junior engineers and two novice verification engineers carried out the first project of verifying 25 DMA instances at the same time. At the beginning of verification, it took a lot of time to develop UVC libraries for the interfaces, but once UVCs were ready, the time for testbench implementation gradually decreased. However, despite the reuse of UVCs, testbench bugs occurred frequently because each verification engineer still had to implement its own dma_vsequence in accordance with DMA instance. By this reason, we considered MDVP with reusable sequences and test codes for all DMA verifications. As a result, we built a newly defined base_vsequence and redefined dma_vsequence in MDVP to minimize code development:

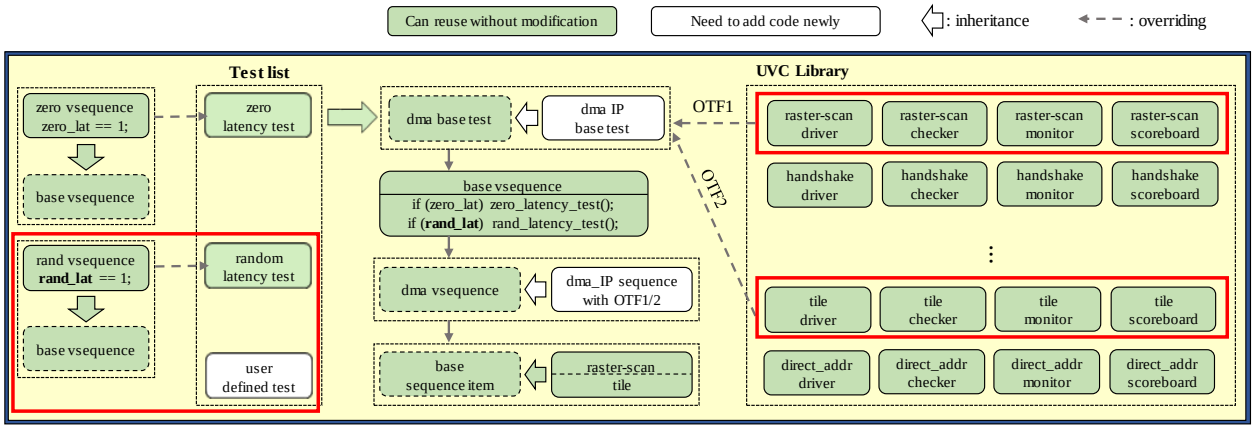

Figure 6. The architecture of common test and sequence of MDVP

1. base_vsequence: We found common test scenarios used for each DMA verification while performing the first project with MDVP and created additional sequence layer named base_vsequence. The common test scenarios were built into base_vsequence with control knobs as random variables. The control knobs were determined in each common test scenarios. The common test scenarios could activate control knobs by factory overriding base_vsequence with a scenario sequence having desired control knob in common test scenarios as Fig. 6. As a result, each verification engineer can minimize code development about test scenario.

2. dma_vsequence: In earliest MDVP, verification engineer had to developed their own dma_vsequence and had lot of time to build the sequence. Therefore, we redefined and improved this sequence for more efficient testbench development. In the revised MDVP, the dma_vsequence has a sequence of function/task that can be used for every DMA verification. This sequence consists of 1) predefined methods that commonly used for all DMA verification and 2) empty methods that need to be developed differently for each DMA verification. Thus, a verification engineer can complete dma_IP_sequence quickly by inheriting dma_vsequence and just developing empty methods according to the DMA specification.

3. Parameterized sequence: For parameterized UVCs, both interface and sequence item had to be implemented using parameters. In the early stages of MDVP, however, the parameters were not well defined in the dma_IP_sequence so the verification engineer had to develop a large number of code for connection between configuration sequence and sequence item for UVC. In addition, a number of testbench bugs occurred in the process and novice verification engineer was difficult to link configuration sequence to sequence item for each UVC.

Therefore, we created the parameterized dma_vsequence according to the on-the-fly interface parameter. And we also built in methods for linking configuration sequence and sequence item of UVC in MDVP. These methods are developed by parameterized sub-routines because virtual task and function cannot support method overloading with other arguments. [4]

IV. Conclusion

MDVP is designed to verify multimedia IP's DMA instances efficiently with less manpower and it has used on several DMA verification project for Exynos SoC. In the first project using MDVP, it took a day to release initial testbench that worked well as before. However, as the newly developed UVCs and sequences accumulated in MDVP after undergoing several projects, it now takes only 30 minutes for a novice engineer to build a new testbench for new DMA verification. And also DMA verification team consisting 5 engineers can handle more than 20 DMA verifications successfully in a given schedule. Table 1 summarizes the effect of applying MDVP to the DMA verification.

TABLE I
THE EFFECT OF APPLYING MDVP

	Without MDVP	With first MDVP	With current MDVP
Line count required for development per each DMA instances	N/A	1298 lines	556 lines
Required time to make an initial testbench	1~2 day	Average 1 day	Average 30 minutes
The number of DMA that can be performed simultaneously per verification engineer	Senior : 2~3 EA Junior : 1~2 EA	Average 4 EA	Average 5 EA

Acknowledgment

We would like to thank Hosung Han and Yongmi Lee's excellent debugging support of our verification environment. Especially, we thank Sanggyu Park for encouraging the publication of our results for MDVP.

References

[1] Accellera, "Universal Verification Methodology (UVM) 1.2 User's Guide", October 2015

[2] ImageMagick 6.7.2-7 [Computer software]. (2017). Retrieved from http://www.imagemagick.org.

[3] Geng Zhong, Jian Zhou, Bei Xia, "Parameter and UVM, Making a Layered Testbench Powerful", October 2013 [2013 IEEE 10th International Conference on ASIC].

[4] IEEE Standards association, "IEEE Standard for SystemVerilog – Unified Hardware Design, Specification, and Verification Language", Chapter 13.8 (2017)