Design and Implementation of Test Harness for Device Drivers in SOC on Mobile Platforms

Harsh Arora
School of Computing Science And Engineering
Vellore Institute Of Technology
Vellore, India
Harsh_Arora@ieee.org

Lokesh N. Jaliminche
Wireless Research and Development,
Intel Mobile Communications,
Bangalore, India
lokesh.jaliminche@gmail.com

***Abstract*—Modern SOCs are comprised of a wide range of modules, such as microprocessor cores, memories, peripherals, and customized components, relevant to the targeted application. Testing external peripherals is easy, but testing Embedded peripherals in SOC is challenging task. In order to efficiently carry out design verification of peripheral cores, it is necessary to evaluate device under test with functional coverage metrics. The overall process can be divided in to two closely correlated phases mainly module configuration and module operation. The first one configures the peripheral on the different operation modes, the second one is responsible for exciting the whole device and observing its behavior.**

In the current work, we propose to develop a test harness for DMA controller and High Speed Synchronous Serial Interface. This proposed test harness can be used for stimulating the distinct functionalities of device under test as well as used for behavioral analysis. We have developed test drivers which can be exploited for testing by adding suitable observability features. Experimental results are provided with suitable functional test coverage to evaluate effectiveness of this test harness .

Keywords—Test driver , Stimuli generation, DMA Controller , High Speed Synchronous Serial Interface (HSI). Test Harness, Device Under Test (DUT).

I. INTRODUCTION

A. Background

The modern SOC families incorporate a wide variety of peripheral devices relevant to the targeted application. This has simplified the SOC design phase but complexity of testing process has increased. As a result, verification for completeness check still remain a daunting task for researcher community.

Many Testing techniques have been proposed to test peripherals on SOC. These consist of both Hardware and Software based techniques. Hardware based techniques consist of SCAN chains and BIST insertions [2].These methods takes less effort but sometimes degrade overall performance. We cannot achieve speed testing using scan chains. Sometimes it requires changes to be made in core structures, which is often an impossible task to do.

Software based techniques are more promising and less intrusive as compared to Hardware based techniques. One of the software based technique is Software Based Self Test (SBST) [2] . In these, we use a special program which is able to test the processor and peripherals connected to it. It is relatively flexible and fast as compared to traditional hardware based techniques. SBST techniques has already been explored deeply. Normally these techniques consist of excitation of different functions and resources of the processors [3]. Sometimes these support generating the test program starting from high-level functional descriptions [4].

The proposed work focusses on functional testing of Peripherals. I/O peripherals are relatively easy to test as compared to peripherals embedded in SOC. Many techniques have been proposed for peripheral testing by research community in recent times..

A. Apostolakis et al [5] have proposed a solution for test of communication peripherals . This solution is divided in to two phases. In first phase, specific peripheral is programmed according to its specifications . In later phase this program is executed and its behavior is observed and analyzed. If peripheral is supporting communication between SOC and other devices, then observation of external interfaces is required. This can be done by using Automatic Test Equipment. Hwang et. Al [8] have proposed a novel approach which uses system or peripheral bus to feed test stimuli and observe test responses . Huang et.al [11] proposed a test methodology for testing IP cores in SOC wherein test programs runs on processor core, which generates test patterns in SOC and analyzes test responses. A.Apostolakiset. al [9] proposed a functional self test approach and applied this approach on to two popular peripherals UART and Ethernet. Jayaramanet.al [12] proposed a functional self-test approach for SoCs which exploits the processor's instruction set architecture to test itself as well as the peripheral components of the SOC.

In this paper, we plan to create a test harness for peripheral testing based on functional description of peripherals. The proposed prototyped test harness has been verified for Direct Memory Access Controller and HSI.

`

B. Test Generation and Functional Test Coverage

In order to precisely excite functionalities of these peripherals, aset of stimuli are devised and relevant test drivers are created. These test drivers are used to stimulate respective functionality. Test coverage measures the amount of testing performed by a set of test. Coverage metrics were firstly defined in software testing. This is the measure to exercise a given piece of code by a test set[14]. Similarly, as specified in software testing [13], it is possible to measure capability of set of stimuli using coverage metrics.

Functional coveragemetrics is used to check functionalities of Device Under Test(DUT). There are four phases of creating functional coverage metrics: 1.Finalizing functionalities to be covered 2.Test sets to be written for corresponding functionalities 3.Collecting data on functionality 4. Evaluating results. This metric ensures that the device under test fulfills the design functionalities as described in the specifications.

The organization of the paper is as follows. The paper provides some background information regarding SOC and Peripheral testing techniques in initial section. Section II provides some general information about DMA controller and MIPI HSI interfaces. Section III provides detailed insight for proposed methodology. Section IV presents a Case Study for DMA controller . Further section V contains Experimental results with functional coverage metric to show effectiveness of this test harness. Finally, Section VI draws some conclusions and summary of the research .

II. WALK-THROUGH DMAC AND HSI

A. DMA

Fig. 2.1 shows Direct memory access (DMA) system. DMA allows specific hardware within the computer to access main memory without intervention of the central processing unit (CPU). Without DMA, all the memory related operation has to be handled by CPU . So, most of its time CPU remain unavailable to perform other work, because for entire duration it remains fully occupied to perform read and write operation. Insertion of a (DMA Controller) DMAC into a SOC lowers the CPU workload as different devices are able to do transferof data without requiring the CPU 's involvement. Throughout data transfers, CPU is allowed to carry out other jobs ; furthermore in some cases, the transfer of data is faster when executed by a DMAC.

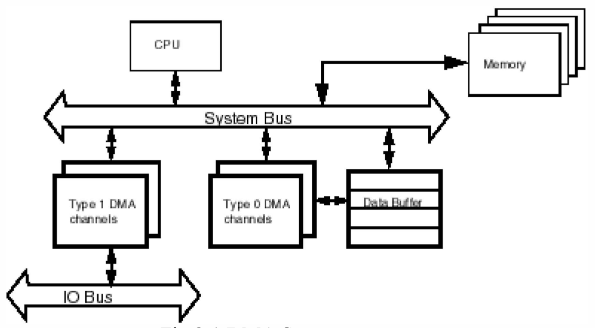


Fig.2.1.DMA System

Many hardware systems uses DMA such as disk drive controllers, graphics cards, network cards and sound cards. Multichannel DMAC is comprised of a set of configuration registers, a set of transfer channels, a priority arbiter, and a DMA engine. To perform all DMA operation properly, necessary configuration is provided by configuration registers. Each channel performs independent data transfers, it uses individual set of registers for this purpose. Each peripheral connected to the DMAC can perform burst DMA request or single DMA request. Each DMA channel has a specific hardware priority. DMA channel 7 has the lowest priority and channel 0 has the highest priority. If two channels requests at the same time, the channel with the highest hardware priority is serviced first. It supports 4 types of transfer Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers.

B. High Speed Synchronous Serial Interface (HSI):

The High Speed Synchronous Serial Interface (HSI) is used to provide high bandwidth, point-to-point, serial communication between two peers, like the cellular processor and application processor on a mobile platform, Fig. 2.2 shows System Integration of High Speed Synchronous Serial Interface. The basic architecture has mainly two parts, Cellular chip and Application chip. Cellular chip is connected to memory and CPU and Application chip is connected to the peripherals. Each of these chips has an internal master and slave components which are configuration components of the chips.

C. Basic Operation of HSI :

HSI is used for fast communication between the peripherals and Cellular chip. A transmitter and receiver are available on both the chips for the communication. The communication protocol used by HSI is based on some fixed signals.

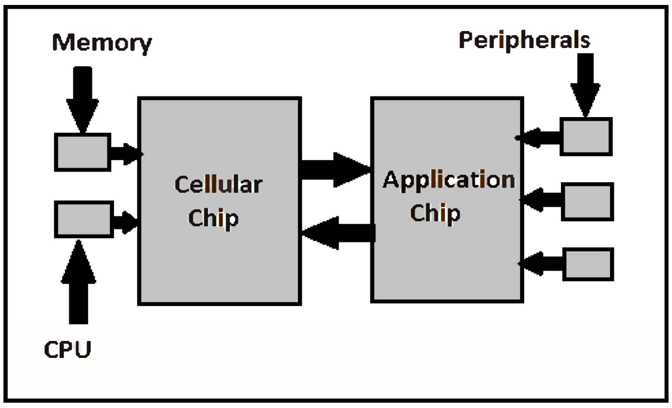


Fig.2.2. System Integration of High Speed Synchronous Serial Interface

The four signals used are :

- WAKE
- READY
- DATA
- FLAG

When data needs to be sent by a chip, it first creates a frame as per the predefined format. The content of the frame comprises of information such as: data to be sent, channel number through which data is to be sent etc. Once the frame is ready, WAKE signal is send by the transmitter to the receiver.

This signal power's up the Cellular chip. READY signal is sent back to the transmitterindicating that it is ready to receive the data. Transmitter sends DATA and FLAG to the receiver as soon as it receives the READY signal. Upon receiving a frame, it is unpacked at the receiver and the data and channel number are determined. The data is then forwarded to the channel management block which consists of an arbiter. Arbiter redirects the data to the specified channel. Thereafter, data is sent to master interface which handles the transfer of data to/from memory.

III. Proposed Approach of Test Harness

Below mentioned Fig. 3.1 shows depicts the design of test harness. Design of test harness is divided into three phases.

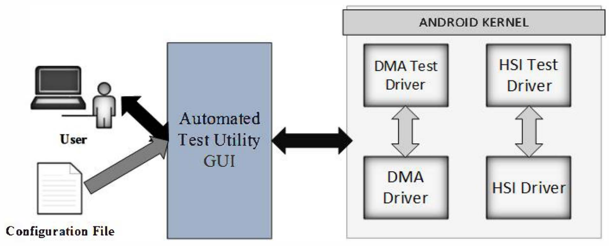


Fig 3.1. Design Of Test Harness

First phase is devoted for creation of test drivers. In peripheral testing, test data is generated by running test programs. Processor captures data by executing corresponding test drivers. On being a functional one, our methodology resembles the functional verification process for the peripheral but it applies an enriched set of test drivers to detect manufacturing faults. The overall creation of test drivers is outlined in figure 3.2.

While creating test driver, it is assumed that a test engineer is in charge of manually developing the test driver. As shown in first step, initial analysis of peripheral must be done. Main goal behind this step is, test engineer must identify main components required for configuration of peripheral such as different registers required for configurations and identifying all functionalities which can be derived from available description. Second step consist of identifying shared and unshared resources as stated in [15].In case of DMAC,DMA module is shared one and channels are unshared resources. Third step is dedicated for creating state machines for each identified functionalities of peripherals . This gives clear idea of timing and signal for each functionalities . Finally, test drivers are created utilizing the peripheral analysis results obtained and corresponding state machine initiation.

Second phase is devoted for creation of configuration file for test drivers . In this phase, tester can create infinite number of configurations for device under test. These configurations can also be specified as stimuli for device under test. In case of DMA controller, these configurations consist combinations of channel number for transfer of data, size of data to be transferred, size of source buffer , size of Destination buffer, Type of transfer etc. Generally all configurations are not useful.

In order to reduce number of configurations, test engineer can make use of shared and unshared resources. Dependingon shared and unshared resources configuration graphs can be created. Exploiting this graph, configuration paths can be derived. Figure 3.3 shows sample configuration graph.This shows configuration graph for both shared and unshared resources. Control register comes under shared resource and channel comes under unshared one

This graph consists of nodes at each level. Each node has a specific configuration. Configuration path should be derived in such a way that it should touch only one node from each level as each node is configured exclusively. In this way, there would be 2n configuration for n nodes. All these configurations are not useful. In order to select useful configurations, pseudo-random algorithm is used [12].

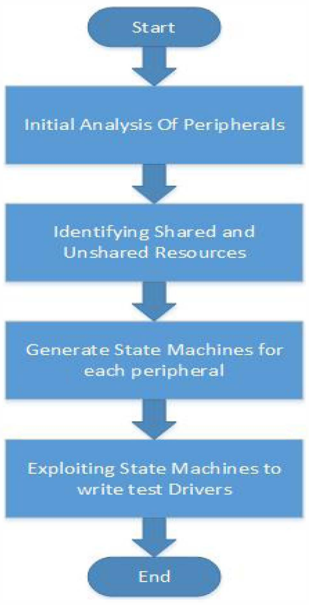
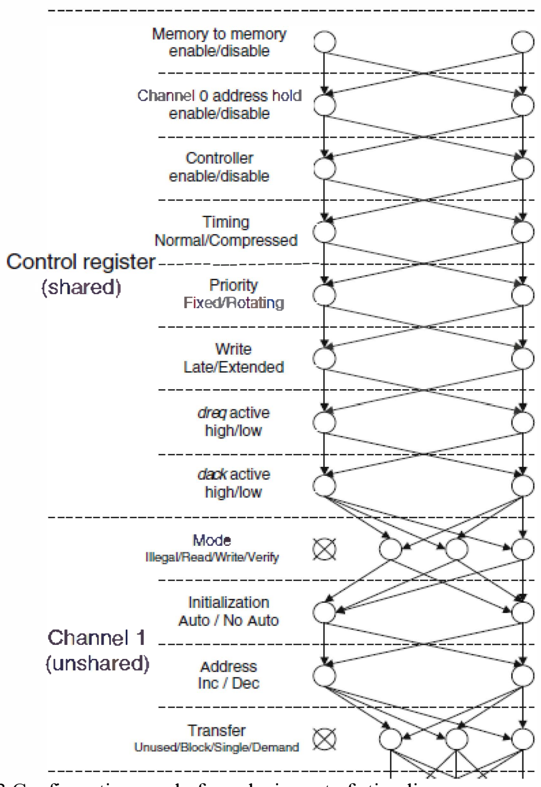

Start
Initial Analysis Of Peripherals
Identifying Shared and Unshared Resources
Generate State Machines for each peripheral
Exploiting State Machines to write test Drivers
End

Figure 3.2.Flow chart For creation of Test Drivers

Third phase consist of all the logic which is used to collect kernel messages while executing tests on peripherals. While executing specific test driver on DUT, these kernel messages signifiesthe behavior of kernel and corresponding peripheral,. This can be specified as test log. This log can be useful specially for identifying the manufacturing faults of the device under test.

IV. Case Study

In order to evaluate the effectiveness of the test harness we considered benchmark SOC Containing ARM Cortex-A9 IP Core , 2 GB Data RAM , 2GB ROM, PL080 compliant DMA Controller and some peripherals with DMA transfer capability as shown in Fig 4.1.


Memory to memory enable/disable
Channel 0 address hold enable/disable
Controller enable/disable
Timing Normal/Compressed
Control register (shared)
Priority Fixed/Rotating
Write Late/Extended
dreq active high/low
dack active high/low
Mode Illegal/Read/Write/Verify
Initialization Auto / No Auto
Channel 1 (unshared)
Address Inc / Dec
Transfer Unused/Block/Single/Demand
Fig 3.3 Configuration graph for reducing set of stimuli

PL080 DMA Controller consist of Eight DMA channels, 16 peripheral DMA request lines , Single DMA and burst DMA request signals[10]. Each peripheral connected to the DMAC can perform burst DMA request or single DMA request. Each DMA channel has a specific hardware priority. DMA channel 7 has the lowest priority and channel 0 has the highest priority. If two channels requests at the same time, the channel with the highest hardware priority is serviced first. It supports 4 types of transfer namely

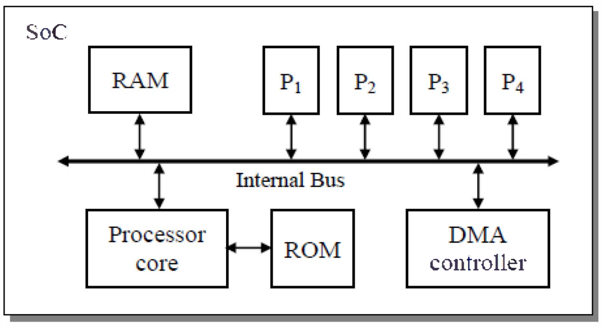


Fig. 4.1 Benchmark SOC Diagram

Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers.

V. Implementation and Experimental Results

According to approach described above we have developed test harness for both DMA controller and Highspeed serial synchronous interface. First peripherals analysis is done . Depending on peripheral analysis, shared and unshared resources have been identified as shown in Table 1 .

Shared Resources	Unshared Resources
DMA Engine, General Registers , Internal Buses , Priority Arbiter	Transfer Channels, Address registers, Word count registers, Channel mode registers

Table I. Shared and Unshared Resources

Thereafter,functionalities are identified specific to DMAC and HSI. Relevant test sets are developed for identified functionalities. These test sets are used to show functional coverage metric to show effectiveness of test harness as shown in Table 2.

All these test sets are used to stimulate the Device under test (in this case DMA controller and HSI) and its behavior analysis is done. We have also created and developedGUI-based utility for this test harness. This gives many features such as: user friendly interface for driver testing, GUI-based forms to create new test stimuli, Options for customization of test cases, readable reports to do behavioral analysis of DUT.

Setup Environment is fully automated for critical manual works which results in saving lots of time for facilitating test harness. .

Functionalities	**DESCRIPTION**	**Completeness**
Memory to Memory transaction Using DMA	**Test driver should invoke driver for memory to memory transfer**	☑
Memory to Peripheral transaction using DMA	**Test driver should invoke driver for memory to peripheral transfer**	☑
Peripheral to Memory transaction using DMA	**Test driver should invoke driver for peripheral to memory transfer**	☑
Peripheral to Peripheral transaction using DMA	**Test driver should invoke driver for peripheral to peripheral transfer**	☑
Selection of particular channel to perform DMA transfer	**DMA transfer should happen using particular channel**	☑

Enter no of threads for particular channels	**DMA transfer should be performed by specified no of threads.**	☑
Enter Buffer size , number of iteration for DMA transfer	**DMA transfer should be performed according to specified parameters**	☑
Interrupt transfer from cellular chip to application chip	**Test driver should invoke driver to pass interrupt from cellular chip to application chip**	☑
transparent read/write using HSI	**Test driver should invoke driver for transparent read/write**	☑
Scatter/Gather read/write using HSI	**Test driver should invoke driver for Scatter-Gather read/write**	☑
Interrupt transfer from cellular chip to application chip	**Test driver should invoke driver to pass interrupt from cellular chip to application chip**	☑

Table II. Functional Coverage Achieved

A. Experimental Setup:-

In order to evaluate test harness, our required experimental setup consists of two operating systems namely, Windows 7 on the host side and Android operating system on the target one. On the hardware side, it consists of SOC containing PL080 compliant DMA Controller and MIPI HSI interface.

Using GUI utility (Snapshots : Fig.4.2, Fig 4.3, Fig 4.4), on the host side, test stimuli is dispatched over target. Test driver uses this test stimuli to stimulate the peripheral. Simultaneously kernel logs are captured and sent to GUI utility for generation of test report. A detailed behavioral Analysis follows the report(Snapshot: Fig.4.5)

B. Snapshots of the Developed Test Harness:-

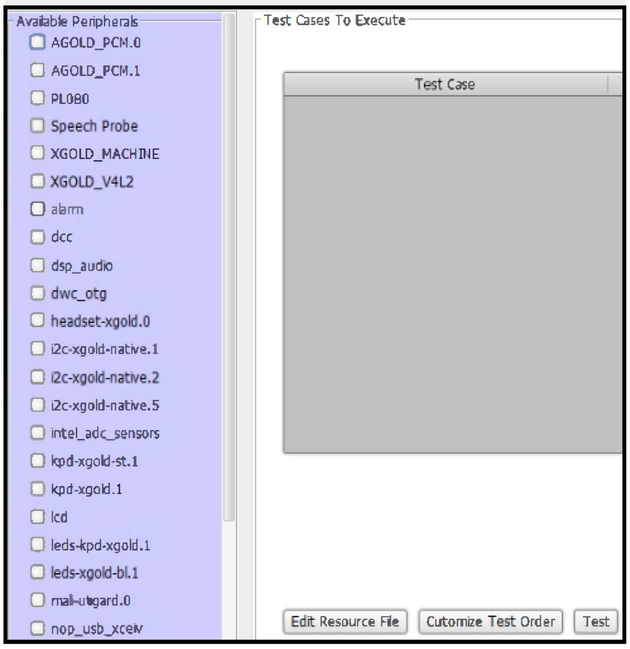


Fig 4.2.:Available Peripheral list

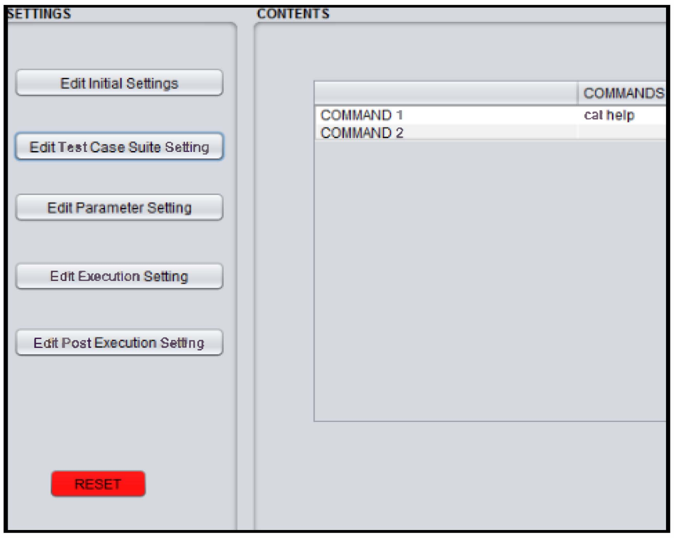


Fig 4.3.:GUI For Configuration File

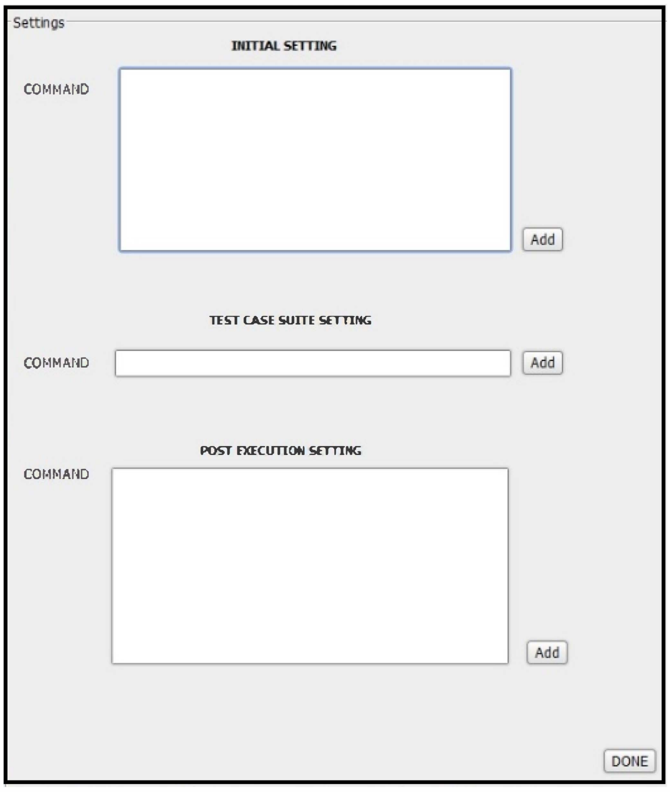
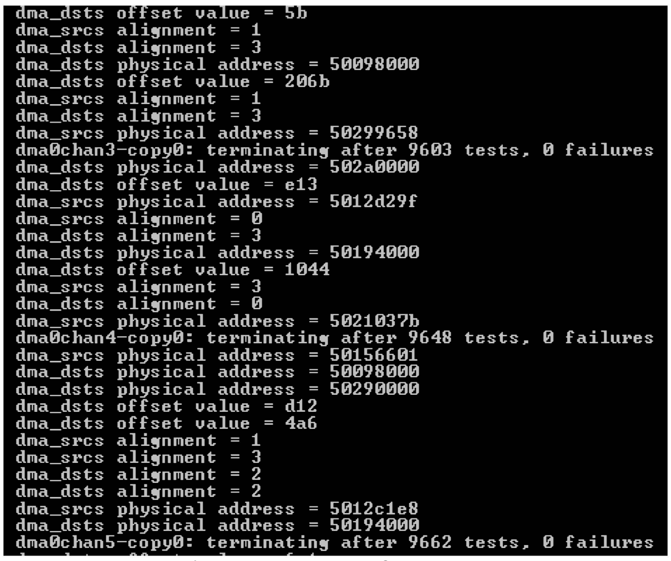


Fig 4.4 :GUI For Editing Parameters For Test Driver



```
dma_dsts offset value = 5b
dma_srcs alignment = 1
dma_dsts alignment = 3
dma_dsts physical address = 50098000
dma_dsts offset value = 206b
dma_srcs alignment = 1
dma_dsts alignment = 3
dma_srcs physical address = 50299658
dma0chan3-copy0: terminating after 9603 tests, 0 failures
dma_dsts physical address = 502a0000
dma_dsts offset value = e13
dma_srcs physical address = 5012d29f
dma_srcs alignment = 0
dma_dsts alignment = 3
dma_dsts physical address = 50194000
dma_dsts offset value = 1044
dma_srcs alignment = 3
dma_dsts alignment = 0
dma_srcs physical address = 5021037b
dma0chan4-copy0: terminating after 9648 tests, 0 failures
dma_srcs physical address = 50156601
dma_dsts physical address = 50098000
dma_dsts physical address = 50290000
dma_dsts offset value = d12
dma_dsts offset value = 4a6
dma_srcs alignment = 1
dma_srcs alignment = 3
dma_dsts alignment = 2
dma_dsts alignment = 2
dma_srcs physical address = 5012c1e8
dma_dsts physical address = 50194000
dma0chan5-copy0: terminating after 9662 tests, 0 failures
```

Fig. 4.5 :Test Report After DMA test

VI. CONCLUSIONS

In this paper, we have proposed and implemented a test harness with complete flow of testing peripherals on SOC. Our proposed test harness allows tester to perform behavioral analysis of peripherals embedded in SOC . On the basis of this analysis , tester could be able to identify manufacturing faults of these peripherals.

Currently , we have successfully prototyped this test harness for DMA Controller and HSI interface. To evaluateeffectiveness of this test harness, experimental results are provided with functional test coverage.

VII. ACKNOWLEDGEMENT

The research work had been carried out jointly by CoE-VLSI-CAD & Intel under supervision of Prof. Harsh. Researchers would like to thank University for facilitating the same.

REFERENCES

[1] N. Kranitis, A. Paschalis, D. Gizopoulos, G. Xenoulis, “Software-based self-testing of embedded processors”, IEEE Transactions on Computers, Vol 54, issue 4, pp 461 – 475, April 2005.

[2] R. Chandramouli and S. Pateras, “Testing Systems on a Chip,” IEEE Spectrum, Nov. 1996, pp. 1081-1093

[3] S.Thatte and J.Abraham, “Test Generation for Microprocessors”, IEEE Transactions on Computers, vol. 29, n. 6, June 1980, pp. 429-441

[4] K. Jayaraman, V.M. Vedula, J.A. Abraham, “Native Mode Functional Self-Test Generation for Systems-on-Chip”, IEEE International Symposium on Quality Electronic Design, 2002, pp. 280-285

[5] A. Apostolakis, D. Gizopoulos, M. Psarakis, D. Ravotto, M. SonzaReorda, “Test Program Generation for Communication Peripherals in Processor-Based SoC Devices," IEEE Design & Test of Computers, vol.26, n.2, March-April 2009, pp.52-63

[6] Linux Device Drivers, Third Edition By Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman

[7] Essential Linux Device Drivers by Sreekrishnan Venkateswaran

[8] S. Hwang, J.A. Abraham, “Reuse of Addressable System Bus for SoC Testing”, IEEE Int’l ASIC/SoC Conference, 2001, pp. 215-219

[9] A. Apostolakis, D. Gizopoulos, M. Psarakis, D. Ravotto, M. SonzaReorda "Functional Processor-Based Testing of Communication Peripherals in System on Chip"IEEE Transaction on Very Large Scale Integration(VLSI) Systems DOI: 10.1109/TVLS1.2007.9000750

[10] PrimeCell® DMA Controller (PL080) Revision: r1p3 Technical Reference Manual

[11] J-R. Huang, M.K. Iyer, K-T. Cheng, “A Self-Test Methodology for IP Cores in Bus-b ased Programmable SoCs”, IEEE VLSI Test Symposium, 2001, pp. 198-203

[12] K. Jayaraman, V. M. Vedula, J. A. Abraham, “Native Mode Functional Self-Test Generation for Systems-on-Chip”, Proc. Intl. Symp. on Quality Electronic Design, pp. 280–285, 2002.

[13] Tasiran S and Keutzer K (2001) “Coverage metrics for functional validation of hardware designs”, IEEE Design & Test of Computers, vol.18: no.4 pp. 36–45

[14] Goodenough JB and Gerhart SL (1977) “Toward a theory of testing: data selection criteria, current trends in programming methodology”, vol. 2. In Yeh RT (ed.) Prentice-Hall, Englewood Cliffs, pp. 44–79

[15] Grosso, M., Pérez, W.J.H.,Ravotto, D., Reorda, M.S. Medina, J.V“Functional Test Generation for DMA Controllers”, Test Workshop (LATW),2010 11th Latin American.