Test bench Design for validating Inter Processor Communication (IPC) in a multi-core SoC

P. Nagabhushan Reddy[1], Dr. T. Bhaskara Reddy[2]
[1]P. Nagabhushan Reddy, Research Student, RU, Kurnool – 518002
[2]Professor, Dept of Computer Science. S.K. University. Anantapur-515003

Abstract- Multi-core devices are used in a wide range of applications including medical, automotive, industrial, communication infrastructure, graphics and mobile handsets. Multi-core devices have more than one programmable processor on the same silicon die.

Applications designed for multi-core device typically require Inter Processor Communication (IPC) f. IPC is a software module used for messaging, data transfer and synchronization among the cores. The number of cores and varying operating systems across these cores increase the complexity of the IPC. Testing an IPC module involves creating a multi-core test framework to execute test scenarios. A test scenario is a set of execution contexts, each being a thread/process running on a specific processor executing a sequence of operations. The purpose of this paper is to explain the design of the multi-core test framework, its features and challenges during implementation.

Keywords- Inter Processor Communication (IPC), Test Frame work, PAL, Shared Memory

I. INTRODUCTION

A. Multi-core device

A multi-core device has more than one programmable processor (core) on the same silicon die. If all the cores in the silicon die are same, then the device is called Homogeneous and if the cores are different it is called Heterogeneous device. Typically in an application on SoC, ARM is used for overall system control, user interface etc and DSP is used for computational intensive tasks like multimedia processing.

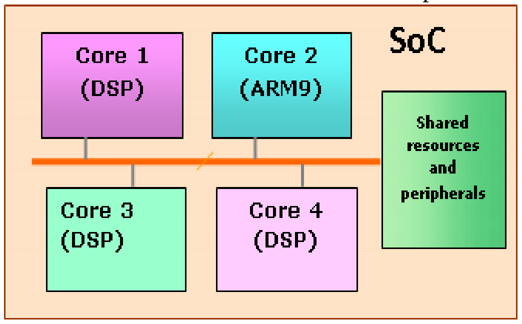

Fig.1: Block diagram of a Multi-core SoC (example)

B. Inter processor communication (IPC)

The application programmers require an easier and robust way to control and communicate with different cores present in the SoC. At hardware level, the processor cores communicate among them using different mechanisms like inter-processor interrupts, shared memory, shared peripherals etc. Usage of Inter processor interrupts and other mechanisms directly by the applications running on the main processor is not the right way to work with multiple cores, as it increases the complexity of the application manifolds and makes the system and software prone to bugs.

IPC software module provides a standard way for all the processors in a SoC to communicate. It provides peer to peer protocols (for control, streaming, notification etc) for the communication. Refer Fig.2 for various services offered by IPC. IPC module abstracts the hardware and provides different interfaces and features to the application. IPC module internally uses the hardware mechanisms the device offers.

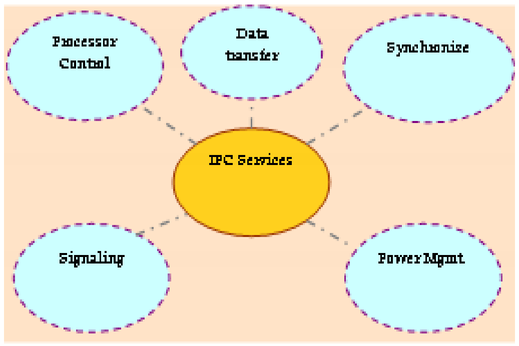

Fig.2: Services offered by IPC

With IPC module in place, application writers can focus on developing the applications rather than spending effort for developing custom ways to communicate among the cores.

C. Validating the IPC

The IPC software supports various SoCs having different configurations (numbers of cores, memory, services etc). This module is used by various customers in different applications such as wireless, automotive, communication infrastructure, video etc.

Validation of IPC software includes creating different categories of test cases as shown in the figure and which can be run across multiple platforms and operating systems.

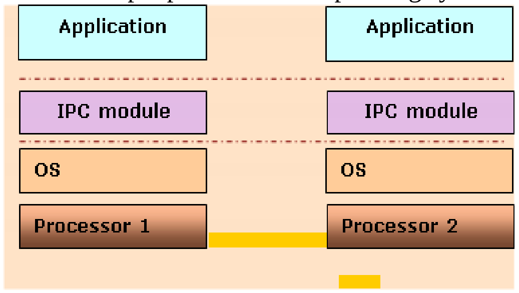

Fig.3: Application usage of IPC

II. LITERATURE REVIEW

The current work done on the test bench for IPC can work in scenarios where the SoC has only two processors. But if the number of processors increase, the test bench cannot scale.

III. PROPOSED WORK

Test bench architecture

The following figure represents different modules of the test bench.

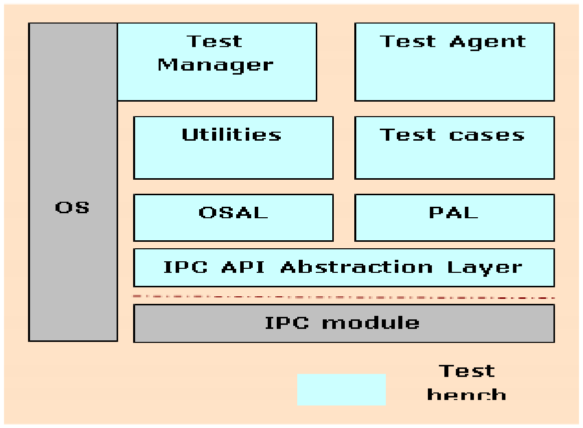

Fig.4: Test-bench architecture

The functionality of each module is explained below:

- **Test manager:** is instantiated on only one processor. It has access to the test scripts. Test manager is responsible for framework initialization, test script parsing, distribution of the test information to test agents present on each processor, triggering and monitoring the test execution and composing the test result.
- **Test agent:** is instantiated on each of the processors. It is responsible for taking the processor specific input test parameters and executing the operations which may involve creating different execution contexts and executing specific instructions in each of the context. It also communicates the result back to the test manager.
- **Test case:** files contain the actual test logic implementation for running a test.
- **Utilities:** contain common modules which can be used by different test cases. For example log module can be used as a utility for printing the logs and test information while executing the test case.
- **OSAL:** is an abstraction layer for the OS services.
- **PAL:** helps in abstracting the platform specific code.
- **IPC API Abstraction Layer:** is an abstraction to the IPC APIs. The test application will not directly call the IPC APIs but instead call into this abstraction layer. The advantage of having this layer is to minimize the changes to the test code if the IPC APIs or parameters change.

Test execution

The following diagram represents simple test execution flow.

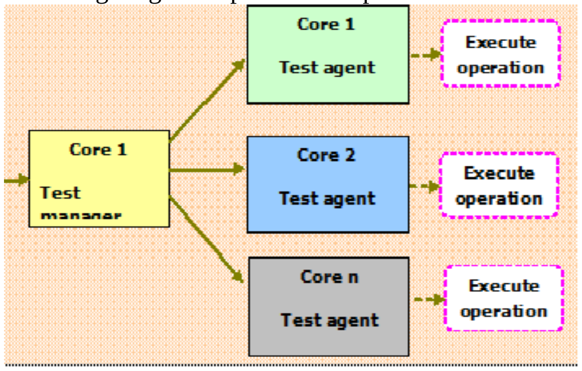

Fig.5: Important blocks of the test framework

The sequence of test execution is split into the following phases:

(i). Framework Initialization

During this phase, the test manager and test agents are initialized. The test bench itself is dependent on certain IPC resources and shared memory. All these resources are acquired. The system is ready for consuming the test scripts.

(ii).Parsing the test script

Test script contains the test information. It contains the details regarding the operations that need to be performed on each processor. The parser module, which is part of the Test manager, parses the script and collects the information required for executing the test on each processor. This information is passed to the respective Test agent.

(iii). Test execution

Once the Test agent receives the test information, the Test manager instructs all Test agents to start the testing. Test agents residing on each processor will create the required contexts and run the test operations as per the test information. The result is communicated to the Test manager.

(iv).Test Result

The Test manager collates all information coming from the Test agents and will present the final test result. After this step the test manager is ready to execute another test.

(v).Framework exit

This is the final phase. This phase is entered after all the tests are executed. All the resources acquired for test bench are released.

IV. RESULT

Here are the improved results w.r.t old test bench and methodology

Features	**Old Test Bench**	**New Test Bench**
Execute test cases between 2 processors	Yes	Yes
Can execute the actual use cases and scenario	No	Yes
Can run test scenario across more than 2 processors	No	Yes
Emulate Customer use case	No	Yes

Other benefits of the new test framework

A. Ease of test addition and execution

New test cases can be created using the test script. Test code need not always be changed for executing new test cases. The test flow will vary based on the sequence of operations and parameters specified in the test script file.

B. Faster and efficient porting

The test bench architecture enables easy porting of test cases across different operating system and silicon platforms.

C. Test bench product

The test bench is released as a product to the development team and the customer. Developers can use the test bench for regression testing. Customers can use the testbench and execute the tests on their custom boards and setup. Users can also add custom test applications to the testbench.

V. CONCLUSION AND IMPROVEMENTS

In this paper we have elaborated the need of inter-process communication, some of the IPC methods, mechanisms and their implementation. We have also discussed the reasons on why it is important to have a solid validation plan for IPC. Without proper validation, any bug in the IPC will break the complete system.

The existing test methodology can work only if the SoC has two processor. As we are moving towards the latest SoCs having multiple processors (multiple ARM cores and DSP cores), its important to have the Test bench that is capable of handling the practical scenarios, scenarios span across multiple processors and be able to measure the Functionality, Performance, stability and also the Power of each of the core and at system level. The new test bench framework we have proposed address all these requirements and will help the system developers to effectively validate the IPC and that inturn helps in less number of issues and improved Time to Market.

Currently the intermediate logs generated on all the processors during the test are not routed to a common point. We are currently working on a solution which can route the log information to a single processor.

Fault handling and recovery mechanism for the test frame work needs to be added to make it robust.

Overall, the inter-process communication is great for introducing reliable communication, maximum performance, great functionality, application modularity and support and also secure communication in our multi-process environment.

VI. REFERENCES

[1]. Zoran. S and Madevska. B "Inter-Process Communication, Analysis, Guidelines And Its Impact On Computer Security" The 7th International Conference for Informatics and Information Technology , pp: 46-50, 2010

[2]. Inter Processor Communication training-TI. https://training.ti.com/system/files/docs/keystone-intro-ipc-slides.pdf

[3]. Microsoft MSDN library, online material, http://msdn.microsoft.com.

[4]. N. Lekic ; Z. Mijanovic ; D. Gobovic ; R. Dragovic-Ivanovic "The simple multiprocessor communication system" https://ieeexplore.ieee.org/document/1046428

[5]. Dr.Vasuki "Design and Implementation of Multiprocessor Communication In Embedded Processors for Real Time and Industrial Automation" http://www.internationaljournalssrg.org/IJEEE/2014/Volume1-Issue1/IJEEE-V1I1P104.pdf

[6]. Gregory T. Byrd, Bruce A. Delagi, Michael J. Flynn "Communication Mechanisms In Shared Memory Multiprocessors" http://i.stanford.edu/pub/cstr/reports/csl/tr/94/623/CSL-TR-94-623.pdf

[7]. P. Nagabhushan Reddy, Dr. T. Bhaskara Reddy, "Latest Power Management Technologies for Mobile Computing Devices".

AUTHORS

Nagabhushan Reddy is currently working as Platform Architect at Intel Technology India Pvt. Ltd. He has been working on the Power Management (Modern Standby) Enabling, Debug and Validation Activities on Intel Core Platforms from last several years. He did his B.Tech in ECE from S.K.University and Masters from BITS, Pilani. He is also a Research Student at Rayalaseema University, Kurnool. He started his career as Product Engineer at Havells India Pvt Ltd working on the development of Static Energy Meters and later worked with various companies (across various domains) Bharat Electronics Ltd (RADAR division), Philips Innovation Centre (worked on UPNP, DLNA, Car DVD players), Texas Instruments (Validation-Post Silicion).

Dr.T.BhaskaraReddy is a Professor in the department of Computer Science and Technology at S.K University, Anantapur A.P. He holds the post of Deputy Director of Distance education at S.K.University and He also the CSE Coordinator of Engineering at S.K.University. He has completed his M.Sc and Ph.D in computer science from S.K.University.He has acquired M.Tech from Nagarjuna University. He has been continuously imparting his knowledge to several students from the last 17 years. He has published 55 National and International publications. 10 International conferences. 13 National conferences. One UGC Major Research Project. Attended several seminars in 3 countries. He has completed major research project (UGC).