Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 05, May -2018

TEST AUTOMATION ON PRE-SILICON ENVIRONMENT

[1]Divya Yadav N,

[1]*Computer Networking Engineering, Department of Information Science, BMS college of Engineering, Bangalore, India*

Abstract— *Empowering SW environment by early enabling the software before the Power-On crosswise over Servers, Clients and Devices. and testing of basic benchmarks, programming workloads, Key Performance Indicators on the Pre-Silicon frameworks to empower "Product release capability" by control on by utilizing Shit-Left system. One of the ways to achieve this goal is to enable automation and continuous integration in the pre-silicon development/validation phase. The way to secure this goal is by initiating some of the activities in Pre-silicon phase. Automate the various software workloads and benchmark release's run on simulated SOC's enabling software stack to adapt to upcoming processors, and SOC's Connect the simulated environment to existing post-silicon to an automated environment. The intention of test automation on pre-silicon environment is having a scope in enhancing the time to bring up the mature silicon SOC's to the market which is an important activity, due to fact of change in market dynamics. The automation framework that will reduce the timeframe of environment setup and execution of the workloads. The function of automation framework will be to use technology to efficiently in order align to the shit-left strategy, The objective of pre-silicon validation is to verify and validate the correctness of design before the power on could happen and bridge a gap between pre-silicon and post silicon, thereby providing fast and accurate platforms.*

Keywords—*Silicon debug, pre-silicon, virtual platform, validation*

I. INTRODUCTION

Few companies are really working hard to reduce its products time to market, and thereby build shift left mindset. The term "shift left" refer to exercise software development where people focus on quality, mainly concentrate on problem prevention rather than detection, thereby start testing prior than interminably before in pre-silicon. The objective of the automation on pre-silicon is to rise the quality, reduce the elongated test cycles and also reduce the probability of unpleasant amazement at end of development cycle or even inferior in production. One of the ways to achieve this goal is enabling automation and continuous integration in pre-silicon development/validation phase. The way to secure this goal is by initiating some of the activities in Pre-silicon phase.

Validation through Design Automation will provide for the business in converging tools, flows and methodologies, accelerating productive innovation and while providing employees a new platform in order to have career growth. Design Automation teams are always an integral part of design groups they support and also there is an incredible legacy of exceptional support and shared responsibility for ensuring the quality of products.

Validation team pull together to develop & support critical programs and each other. We ignore organizational boundaries do the best thing. The software maximum frequently technologically advanced in the pre-silicon simulated platform are bootloaders and basic input/productivity organizations (BIOS), silicon verification and test software, drivers, firmware, and operating system support. All these are tested on the Virtual Platform and verified for the system optimization and functional behavior.

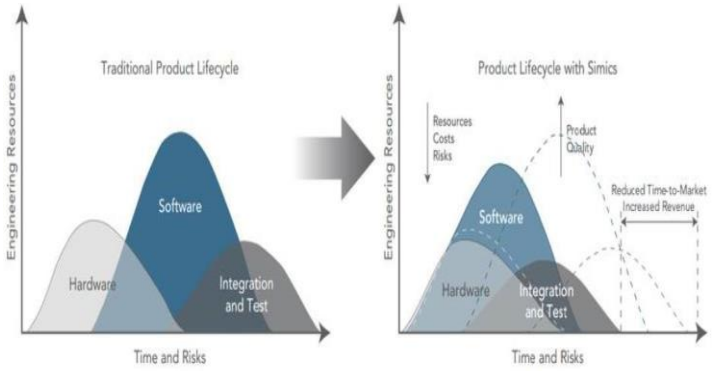

Figure1 left-shift strategy of pre-silicon system.

Figure 1 depicts left-shift strategy of pre-silicon system that reduce risks, costs, time to product during product development.

The project aims at minimizing the work force needed for the execution and reducing the cost needed for this process. Efficiency and accuracy will increase when repeated process is done on the system and its components and the manual errors will be detected easily. The automation of this process also means that existing implementation method can be easily changed and implemented on hundreds of platforms simultaneously without any physical intervention.
The rest of the paper is structured as follows. Section II summarizes background work that had made to what know what exactly the term. Section III gives description of proposed system and Section IV gives the description of system architecture and conclusion are given in Sections respectively.

II.BACKGROUND

In this we summarize background material for given methodology, It is a foundation from where inquire about thoughts are drawn and formed into ideas lastly speculations. It likewise gives the analyst an elevated view about the exploration done around there up until this point.

A. SOFTWARE READINESS QUALIFICATION:

Present framework designers are looked with two huge challenges: convey new arrangements speedier, and create, investigate, and keep up always intricate frameworks.

Exercises, for example, block bring, framework combination, and framework testing can start previously physical equipment is accessible. The equipment and programming groups work firmly together, permitting the product group to give input to the equipment creators as of now before the outline is solidified. This can stay away from exorbitant oversights as far as excessively complex programming models what's more, execution bottlenecks that show up result of an absence of framework improvement This enables the whole undertaking to have its calendar "move left," viably diminishing the opportunity to market and time to income for another item

B. SIMULATION:

Simulation is nothing but imitation of task of a true process or structure after some time. The demo of recreating somewhat initially involves that a ideal be created; this exemplary speaks to the key attributes, rehearses and features of the indicated physical or unique framework or process. The model speaks to the framework itself, though the reenactment speaks to the activity of the basis after just time. Recreation is employed as a part of numerous specific environments, for example, reproduction of modernization for implementation streamlining, security construction, analysis, formulating, exercising, and mainframe games. Frequently, computer tests been developed to think about reenactment models. Reproduction is in additionally been utilized along with logical displaying of regular frameworks or social frameworks to pickup understanding into their working, as in economic aspects. Reproduction will be utilized to demonstrate the possible genuine impacts of elective circumstances what's more, approaches. Reproduction is additionally operated when the unpretentious framework can't be protected in light of the circumstance that it may not be open, or it might be unsafe or unsatisfactory to draw in, or it is being outlined nevertheless not yet fabricated, or it might just not exist.

C. VIRTUAL PLATFORM TECHNOLOGY

Virtual Platform is a software based system that can fully mirror the functionality of a target System-on-Chip or board. These virtual platforms combine high-speed processor simulators and high-level, fully functional models of the hardware building blocks, to provide an abstract, executable representation of the hardware to software developers and to system architects Virtual platform to qualify its hardware and software running on the platform, before it getting it to tape-in

Software stack and Hardware components on a VP:

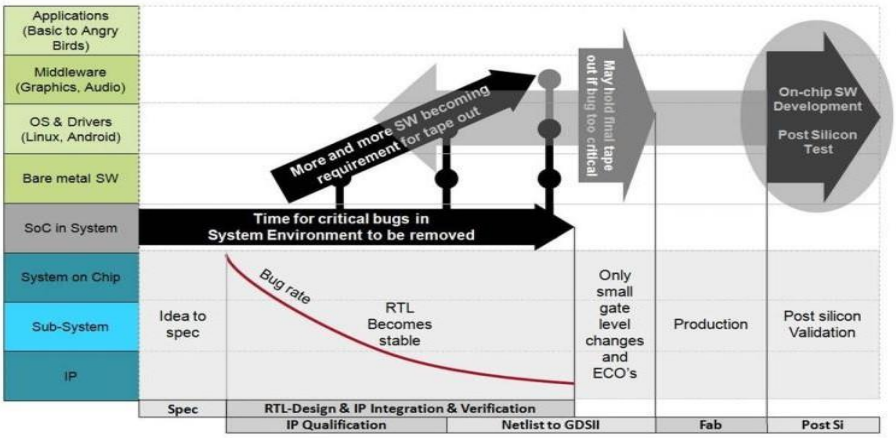

Figure 2 software stack and hardware components on VP

III. PROPOSED SYSTEM

This section provides a detailed description of the need for simulation and continuous integration of the automation framework.

An automation framework that will reduce the timeframe of the environment setup and execution of the workloads. The function of the automation framework will be to use the technology to efficiently in order align with the shit-left strategy.

Test Automation (CI) is an improvement hone that expects designers to incorporate code into a common archive a few times each day. Each registration is then confirmed by a robotized manufacture, enabling groups to identify issues early.

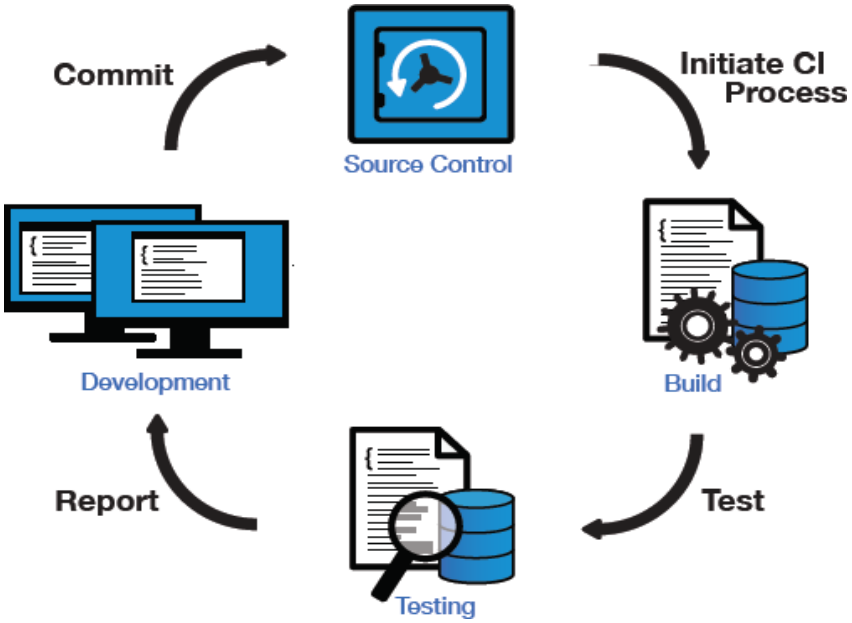

Figure3 Deployment diagram of test automation

Figure 3 depicts the deployment diagram how the test automation is executed. From a high level, a CI pipeline usually consists of the following discrete steps:

1. **Commit**. When a developer finishes a change to an application, he or she commits it to a central swece code repository.
2. **Build.** The change is checked out from the repository and the software is built so that it can be run by a computer. This step depends a lot on what language is used and for interpreted languages, this step can even be absent.
3. **Automated tests**. The change is tested from multiple angles to ensure it works and that it doesn't break anything else.
4. **Deploy**. The built version is deployed to production.

IV. SYSTEM ARCHITECTURE

The architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that comprise the system, and the interrelationships that occur among all architectural components of a system

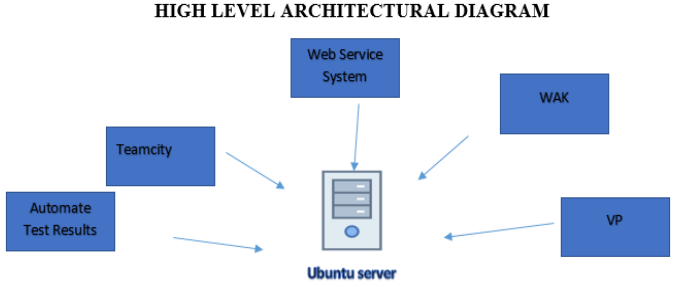

Figure 4. High level architecture of test Automation.

Figure 4 depicts the overall architectural components used in the project. The architectural design consists of Virtual Platform Windows assessment and development kit, Web service system, TeamCity, Linux Server. The Linux server acts as the host machine for the virtual platform that is used to validate and debug the software on the pre-silicon system. WAK is a collection of tools and technologies produced by Microsoft designed to help deploy Microsoft Windows operating system images to target computers or to a virtual hard disk image in format. This is primarily used to load or boot the target system. The web service system and the WAK are used to for the implementation of the automation framework. Whenever a test is built it is pushed into the TeamCity server for continuous integration. VP runs only a sub-system level test with a suite of integrated Programs. Any failure that hiders the execution is considered a bug in the pre-silicone when all the other Steps in the execution flow are followed. The Results are collected from the Web service system and stored in the Ubuntu host machine.

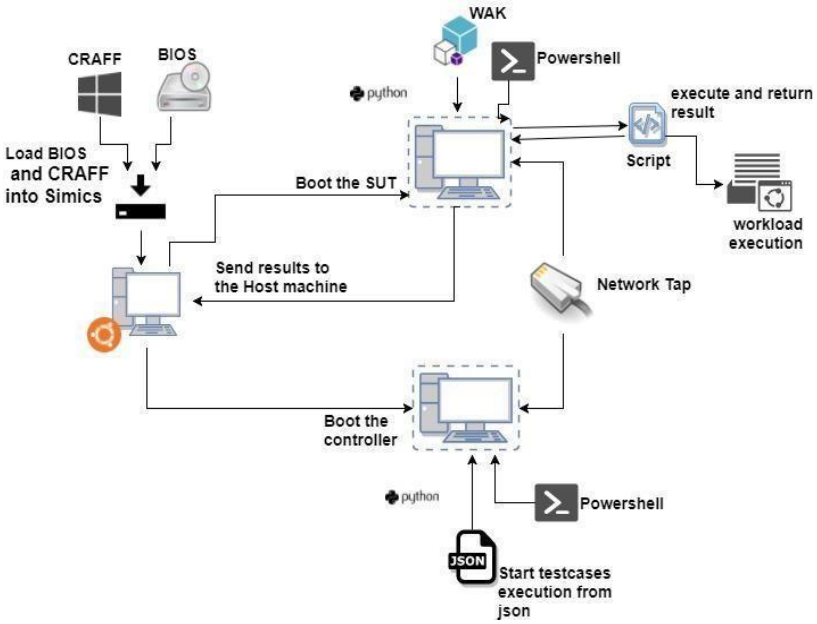

Figure 5 work flow of Automation framework

Figure 5 depicts the work flow execution of automation framework. Where necessary packages of CRAFF and BIOS have been loaded into Virtual platform in the host machine, and there by script is written to have controllers connected via network and made to boot the target. The communication is made to controller is made through JSON there by test is executed test results are collected automatically.

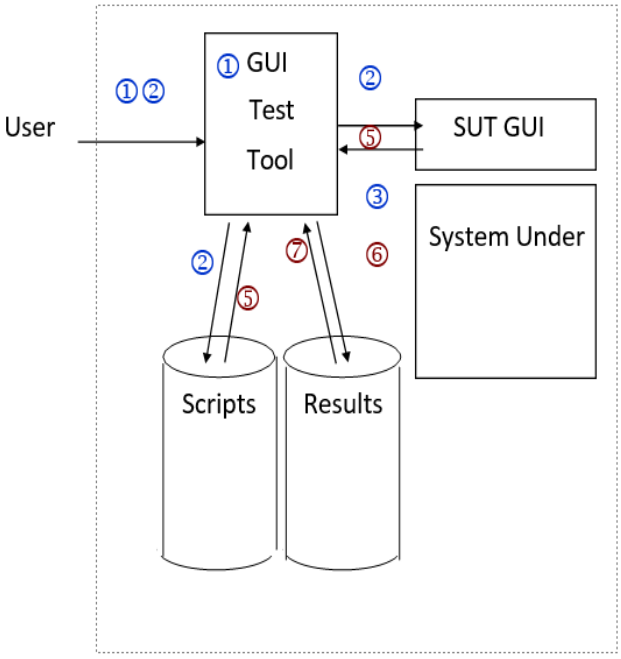

Figure 6 Test Automation procedure

Figure 6 depicts test automation procedure that is how automation flow happens when a test case executed.

The following steps are followed when automating based test case is executed :

1. Launch Tool
2. Test: Capture scripts
3. Test: capture results
4. Launch automated run
5. Play script
6. Capture SUT response
7. Read recorded result
8. Compare and report

Test workload is a computer analysis, diagnostic and benchmarking software used for validation of the software used in the silicon. Auto It tool is installed in the target where the workload execution needs to be done, the test workload is then installed on the target. The desired test case is chosen and recorded for the mouse clicks. The events are played back to check if the events happen correctly. Once the script is recorded then it is downloaded through the matic agent of the virtual platform these scripts are run through a bat file that is triggered from a JSON file. All these scripts are then mounted to a Virtual Hard Disk (VHD) after mounting the target will boot with all the files necessary for the workload to execute automatically through the automation framework. The results of the workload are collected and sent to the host machine that runs the Virtual Platform

V. EXPERIMENT RESULTS

This chapter gives details about the project after testing different models of the Critical Software. The results after testing shows if the requirements are met in the project, and if the requirements are not satisfied, the reasons for the same is listed below. Thoughts for future developments is done after study of results.

Figure 7 Result of Test workload executed through automation The qualification of critical software in pre-silicon using virtual environment. The input is a critical software to validate whether it runs on given binaries or not. The is a virtual environment tool to simulate the Operating system and running the critical software's on targeted system. The different workloads are allowed to run on the target systems to check whether the given workloads are run on the critical software

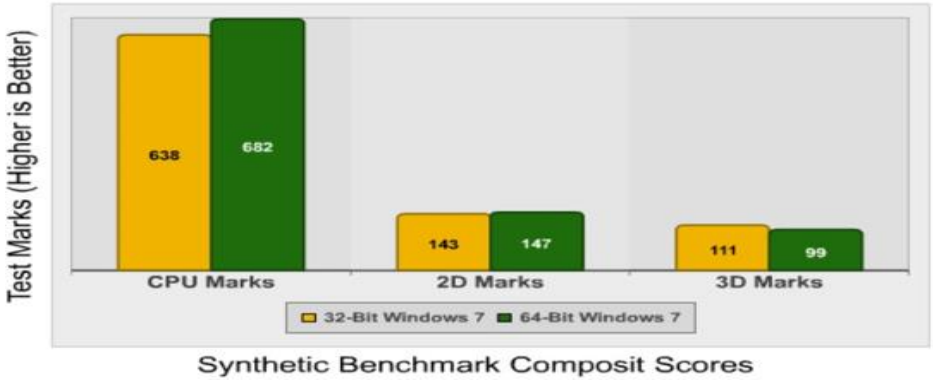

Figure8 Results of benchmark scores

Figure 8 depicts the Results that have been obtained through Automated test Executed on pre-silicon environment that tell benchmark scores that is obtained When the test case have runned

VI. CONCLUSIONS

The automation framework is used for tracking the regular updates of the packages, setting up the virtual platform with the scripts, with the help of the Continuous Integration server. It also can run the test cases on a different platform with some changes since it isn't generic. The automation happens in iterations, the first is collecting the BKC for the virtual platform along with the BIOS. The second is to run the Benchmarks and the Key Performance Indicators (KPI's) on the desired platform, when the run is over results are collated. These results are then utilized for analyzing if the test is a pass /fail. Thus by leveraging the automation framework developed the time taken to build the base setup and environment bring up can be reduced very effectively, the Benchmarks and KPI's can also be run with less manual intervention. And helps the team increase the productivity and work more efficiently by focusing on debugging the pre-silicon platform software validations.

FUTURE ENHANCEMENT

- The system can be scaled out to many servers.
- The results of the workload and Benchmark can be sent out through the mail.
- A GUI tool with a dashboard containing the current status of the execution with details specific to the test case can be developed.

REFERENCES

[1] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in *Proc. of Design, Automation and Test in Europe*, April 2007, pp. 1–6.
[2] L. Kumar, S. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in 2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), 2017, pp. 1–7.
[3] Eggert,A.(2001): Konzeptionelle Grundlagen deslektronischen Kundenbe ziehungs managements, in: Eggert, A.; Fassott, G. (editors): eCRM Electronic Customer Relationship Management, Stuttgart 2017, p. 87-106.
[4] Hippner, H.; Martin, S.; and Wilde, K. D. (2001): Customer Relationship Management, WiSt Heft 8, August 2010, pp. 417- 422.
[5] A. Abramovici and Y.C.Hsu, "A new approach to silicon debug,"in *IEEE International Silicon Debug and Diagnosis Workshop*,Nov. 2005.
[6] A. Gupta, S. Malik, and P. Ashar, "Toward formalizing a validation methodology using simulation coverage," in *Design Automation Conf.*, June 1997, pp. 740–745
[7] Y. S. Yang, N. Nicolici, and A. Veneris, "Automated data analysis solutions to silicon debug," in *Proc. of Design, Automation and Test in Europe*, April 2009, pp. 982–987.
[8] II. F. Ko and N. Nieolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," in *Proc. of Design, Automation and Test in Europe,* 2008, pp. 1298 - 1303
[9] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in *Proc. of Design, Automation and Test in Europe,* April 2007, pp. 225-230.
[10] Y.-S. Yang, A. Veneris, and N. Nicolici, "Automating data analysis andacquisition setup in a silicon debug environment," *IEEE Trans. on VLSISystems*, 2012.
[11] M. F. Ali, S. Safarpour, A. Veneris, M. S. Abadir, and R. Drechsler, "Post-verification debugging of hierarchical designs," in *Proc. of Int'l Conf. on CAD*, Nov. 2005, pp. 871–876
[12] S. Safarpour and A. Veneris, "Abstraction and refinement techniques in automated design debugging," in *Proc. Design, Autom., Test Euro.*, 2007, pp. 16–20

