EARLY SOFTWARE DEVELOPMENT AND VERIFICATION METHODOLOGY USING HYBRID EMULATION PLATFORM

Woojoo Kim, Haemin Park, Hyundon Kim, Seonil Brian Choi, SukWon Kim[1]
[1]Samsung Electronics Co., LTD. (space.kim@samsung.com)

Abstract
This paper proposes a novel Hybrid Emulation Platform for SW (software) development and verification of mobile AP(Application Processor) designs at the early stage of RTL(Register Transfer Level) design. The proposed platform adopts co-emulation of ESL (Electronic System Level) simulation and HW emulation. Most of the IPs are allocated to the HW emulator side to help start SW development with RTL designs. CPU, memories and peripherals, which are used frequently for Android platform boot-up, are allocated to the ESL virtual platform side, which maximizes run-time performance of SW. Proposed transactors for the IPs for external communications such as memory storage, camera modules and display modules, operate in the same way as real HW devices work, and proposed shadow memory reduces emulation time by supporting back-door and front-door transactions with a cache scheme. Also, automation of the platform build process minimizes time overhead of building a Hybrid Emulation Platform. Application results show that booting an Android platform can be completed within 54 minutes on the proposed Hybrid Emulation Platform, and show that an Android platform can be successfully developed before a silicon based development board is ready. Finally, the proposed platform also helps enhance quality of SW by saving more time for design and validation for SW developers.

1. Introduction

As the competition in smart device industry is intensified, importance of TAT(Turn-Around-Time) reduction of a mobile AP development has been greatly increased. To shorten overall TAT of mobile AP development, it is crucial to shorten SW development and verification TAT. However, SW development process usually starts after a development board is delivered to SW engineers, though there exists a fully functional RTL design a few months earlier than the board delivery. This is because full chip level simulations cannot provide enough execution speed required at SW development stage. There have been many studies to accelerate simulation speed by adopting emulation technology but the approaches are not sufficient for SW development. Therefore, this paper proposes a new methodology to provide sufficient execution speed to develop and validate SW at RTL design stage by adopting hybrid platform design methodology which uses HW emulation and virtual platform technology.

2. Related Research

In developing a mobile AP SoC, SW such as device drivers, linux kernel and Android platform used to be developed on its real HW platform. The HW is designed with an RTL designs after definition of specification. Verified RTL designs are fed to implementation stage for making silicon chip. This chapter introduces SW development and verification environments at each design stage of a HW design from specification to silicon.

2-1. Specification stage of HW platform

In general, the first step of mobile AP development is a specification definition. In order to define the specification, estimation of power consumption or performance of a target mobile AP is used. ESL environments are used to measure power and performance values of the target mobile AP in this case. The ESL environment is SystemC-based system and has advantages in easy adjustment of abstraction level of the design. The concept of the ESL is defined in [1] and described as below, which is a quote of [2].

> The definition of ESL (Electronic System Level) is an emerging electronic design methodology which focuses on the higher abstraction level concerns first and foremost. ESL is now an established approach at most of the world's leading System-on-a-chip (SoC) design companies, and is being used increasingly in system design. From its genesis as an algorithm modeling methodology with "no links to implementation," ESL is evolving into a set of complementary methodologies that enable embedded system design, verification, and debugging through to the hardware and software implementation of custom SoC, system-on-FPGA, system-on-board,

and entire multi-board system. ESL can be accomplished through the use of SystemC as an abstract modeling language [1][2].

Therefore, in ESL environments, a HW platform can be built easily by adopting high abstraction-level models in early design stage. ESL environment is useful in SW development and verification due to its rapid simulation speed. HW models which are used in ESL environments can have various abstraction levels with different level of cycle accuracy. As shown at figure 1, there are four kinds of abstraction levels according to cycle accuracies. CA (Cycle Accurate) models have highest accuracy same as RTL design. AT (Approximately Timed) models have 80%~90% accuracy level and LT (Loosely Timed) models have 60%~80% accuracy level. FA (Function Accurate) models guarantee 100% of function accuracy but do not have timing information. Higher cycle accuracy of HW models can provide higher timing accuracy in overall simulation but it results in more slow simulation speed. However, it is in inverse proportion to simulation speed. If only FA models are used in the HW platform, a simulation speed can be from a few dozen MHz to several hundred MHz. On the other hand, if only CA models are used in a HW platform, a simulation speed can be from a few dozen KHz to several hundred KHz [3]. At the specification stage, AT or LT models are more often used than CA models because they can be made more quickly and easily than CA models. Furthermore, TLM (Transaction Level Model) models are used extensively in order to reduce overheads for modeling because there is no need to implement transaction protocols in detail. Therefore, TLM AT and LT models are used widely in ESL.

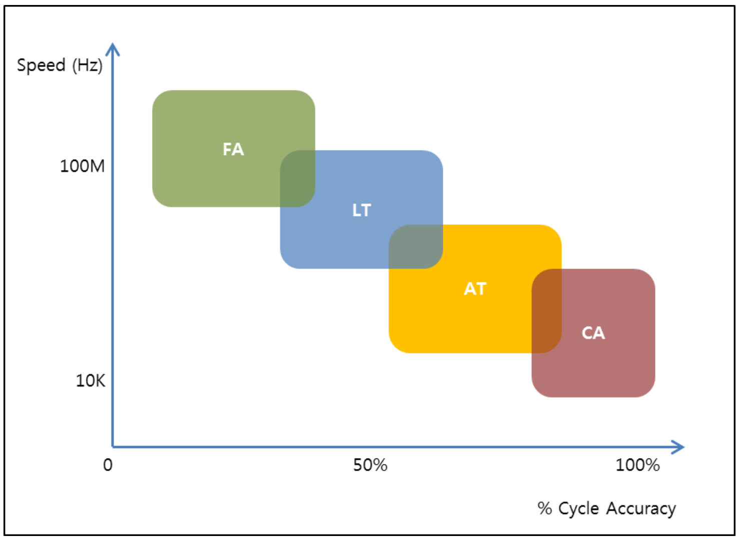

Figure 1. ESL model: Cycle Accuracy and Simulation Performance.

2-2. RTL Design stage of HW Platform

After specification of mobile AP is defined, RTL design for each IP and module starts and RTL simulation is used for RTL verification. These RTL netlists are used for GL (Gate Level) synthesis and HW implementation. Therefore RTL netlists are initial real design. RTL simulation is used for accurate function and timing verification. However, simulation speed is inversely proportional to design-scale. As the design size grows, simulation speed reduces exponentially. In case of several hundred million gate scale, simulation speed can be dropped to a few dozen KHz [4]. This simulation speed is 10^6 times slower than silicon speed. Therefore, the SW which runs during one second on silicon can consume 10^6 seconds (=11.6 days) for simulation. It is almost impossible for simulation to be used for SW development and verification with long-time duration.

Recently, emulation methodology is used to accelerate speed of RTL simulation in many SoC design companies [5]. It is known that speed of emulation is 1000 times faster than that of simulation. The emulation speed can be from several hundred KHz to several MHz [6][7]. The emulation speed is 10^3 times slower than silicon however, it can be applied to SW development and verification. SW which runs during one second on silicon can consume 10^3 seconds (=16.7 minutes) for running. The emulation can be adopted to develop SW such as device drivers which are relatively simple. However, it is not fast enough to be used for the development of linux kernel and Android Platform which are relatively large scale and complex. Because it can take more than ten hours to boot linux and Android using emulation platform.

2-3. Advanced RTL simulation environment

SW development and verification need to be started as early stage of RTL design as possible to secure enough time to develop. In order to start SW development at the early stage of RTL design, RTL simulation environment can be used for development platform. According to a previous research for acceleration of RTL simulation, it is shown that FPGA based test board has about 10 MHz simulation speed [7]. However, due to a limitation of FPGA scale, it is difficult to port full mobile AP to the FPGA test board. Platform build time can increase due to RTL modifications to fit to the FPGA board, or due to a fine tuning for synthesis. In order to avoid the above limitation of FPGA test board and to secure fastest RTL simulation environment, ESL co-simulation and co-emulation methodology were proposed in [8]. This methodology uses SystemC based TLM models instead of RTLs for some IPs. A simulation speed is bounded by emulation speed when IPs allocated to emulator are accessed. On the other hand, simulation speed is bounded by simulation speed of TLM models [8][9]. When ESL co-emulation methodology is applied, additional time is required for IP modeling and virtual platform build. However, total simulation time can be reduced by a trade-off between modeling period and ESL performance.
In the previous research [10], ESL co-emulation methodology was used for GPU verification. The GPU for which verification was required was allocated to an emulation platform and other IPs such as CPU/memory sub-systems and peripherals for boot sequences are allocated to an ESL simulator. In the research, OS boot-up was completed very quickly because CPU and memory sub-systems were allocated to ESL side. SW for OS booting and pre-defined HW platform which had been already applied to ESL environment was used. Since the research only focus on the functional verification of GPU device driver, the pre-defined HW platform and OS booting SW were suitable. As a result, the OS boot SW used in our mechanism cannot be applied to a real mobile AP device.
Furthermore, another existing research [11] used ESL co-emulation methodology to develop and verify SW. They focused on CPU core operation and they used VIP (verification IP) for external devices. Full system of mobile AP was not a focus in the research and it was difficult to adopt the co-emulation platform for SW developments due to a limitation of VIP.
VIP can operate with only pre-defined IP configurations. If all kinds of IP configurations are defined as same as real devices, there are no problems to be used for OS boot-up. The VIPs which are modeled on all SFR (Special Function Register) and its function should be called transactors. Therefore the platform has a limitation in developing full Android Platform with their co-emulation platform.

3. Hybrid Emulation Platform

As shown in the previous chapter, rapid simulation environment should be secured to start SW development and verification at RTL design stage. At the RTL design stage, ESL-RTL co-emulation methodology should be applied to guarantee enough run time performance for SW development. Furthermore, the methodology should support the same environment as that of real silicon mobile AP, and overheads to set up the platform should be minimized.
Therefore, this paper proposes a hybrid emulation platform which enables SW development and verification at the early RTL design stage. The proposed Hybrid Emulation Platform can reduce SW development time also. It uses co-emulation methodology. CPU and memory components are allocated to ESL virtual platform side with TLM LT models, because they issue a lot of transactions during boot up time of Android Platform. Other IPs are allocated to a HW emulator side. Furthermore, virtual external storages, displays, modems and camera models are connected to PHY models through the transactors which were developed in this research. The proposed Hybrid Emulation Platform can provide same environment of silicon-based development board, and enables development and verification of SW related to the external devices.
By following the proposed automatic platform generation flow, timing overheads to build a virtual platform and a HW emulation platform can be minimized. This chapter shows the proposed algorithms.

3-1. Architecture of Hybrid Emulation Platform

The proposed hybrid emulation platform is composed of virtual platform and emulation platform. A virtual platform is an ESL HW platform based on TLM LT models. CPU, memories, and external devices are allocated to virtual platform due to a lot of memory transactions. During Android Platform boot-up sequences, CPU, memories, external storages and display modules are operated very frequently. Therefore, they should be located on virtual platform.
Figure 2 shows an overall view of architecture of the Hybrid Emulation Platform in the experiment.

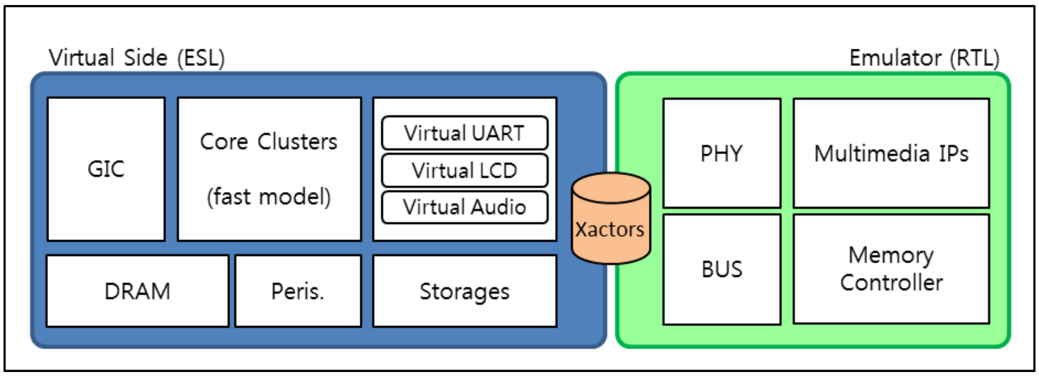

Figure 2. Hybrid Platform Overview.

IPs with high speed requirements should be allocated to virtual platform with TLM LT models to increase a performance of the Hybrid Emulation Platform. Any IP which is already modeled can be allocated to virtual platform side. Other IPs would be allocated to emulation platform side. TLM LT models in the virtual platform are connected to IPs in the emulation platform using transactors, and these transactors are used for communication between virtual platform and emulation platform. The transactors use signal hooking and forcing method to transfer data to other domain. In the view point of emulator, virtual platform looked to be one of RTL modules. On the other hand, emulation platform looks to be one of TLM LT models in the view point of virtual platform.

3-2 Transactor for Domain Conversion

As described in 3-1, adopted so called 'transactors' which enable communications between virtual platform and emulation platform. A transactor is implemented in the emulation platform but does not change original functionality of given DUT because it only performs signal forcing and signal hooking to transfer data between two platforms in different domain. Transactor performs protocol conversion between transaction-level protocol used in the virtual platform and pin-level protocol used in the emulation platform. Transactor has also an internal buffer to overcome speed gap between virtual platform and emulation platform. In the proposed Hybrid Emulation Platform, we adopted transactors for AMBA ACE protocol interface, sideband signals and PHYS for eMMC, UFS, Display and Camera. With these transactors, it was possible to provide user interface which is identical to that of physical mobile set and hence it enabled SW development/validation for full-chip mobile AP.

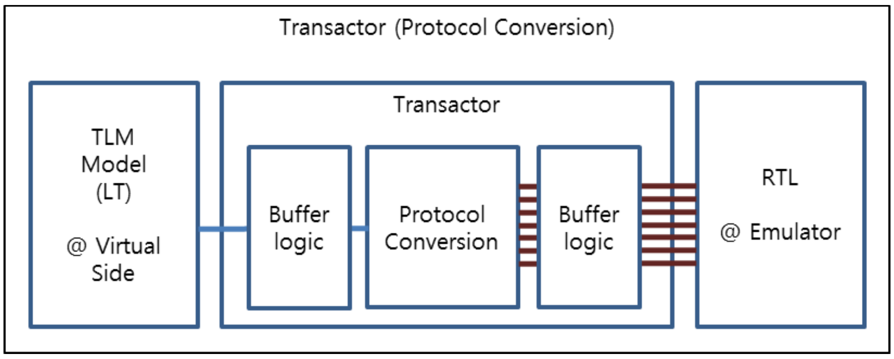

Figure 3. Transactor Overview.

3-3 Timing Synchronization Framework

The proposed novel timing synchronization framework to avoid the phenomena that maximum system speed is limited by the relatively slow emulation platform when synchronized at every cycle. As shown in Figure 3, a cache scheme in the internal buffer of each transactor is adopted, and it helped to enhance overall system operation speed by reducing synchronization frequency between two platforms.

3-4 Memory Shadowing

In our Hybrid Emulation Platform, memory shadowing scheme has been adopted to improve memory access speed. When an IP such as CPU, in the virtual platform requests memory access, the request must be passed through the main backbone in the emulator platform and it gradually limits overall system speed at emulation platform speed. To avoid this speed degradation, we adopted memory shadowing scheme. Detail of the scheme is described in Figure 4. This scheme is located between virtual platform and emulation platform. It supports fast memory access from virtual platform and emulation platform also. With the proposed scheme, a huge amount of memory access in Android boot process is performed well and overall simulation speed enhanced drastically.

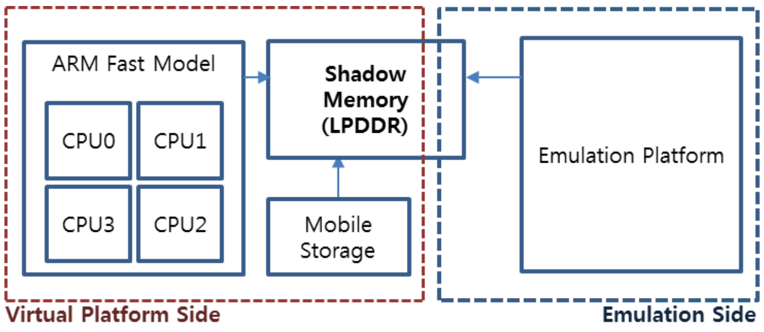

Figure 4. Memory Shadowing Architecture.

3-5 Automation Flow for Hybrid Platform Build

It requires extra design work to create a virtual platform and emulation platform with transactors to build Hybrid Emulation Platform. Users need to set a memory map for proper access control, and also need to set an interrupt map to guarantee correct behavior of GIC located in virtual platform. Moreover, CPU, memory, clock, reset, virtual storage and all peripheral models require connection between virtual platform and emulation platform. These processes generally started with pure emulation platform and incremental refinement for the emulation platform DB (database) is applied. Users need to exclude IPs in emulation platform by marking it as black-box when the IP is created in virtual platform and place a transactor to communicate the IP in virtual platform and surrounding blocks in emulation platform. User need to rewrite test-bench which includes the process described in previous sentence and need to perform emulator porting again. In our experiments, we found building hybrid platform is time consuming and error prone process. This is mainly because traditional validation/verification methodology covers virtual platform or emulation platform and does not cover the crossing boundary. Therefore, we proposed automation flow to minimize manual design efforts which will also minimize human errors as well. Automation flow includes modified emulation platform (DB creation process and virtual platform creation process. The proposed automation flow generates an emulation platform DB creation script with transactors and IP information written in predefined input format. Also virtual platform creation is automated with memory map and interrupt map written in predefined format. In our experiments, we identified we can reduce 37% of hybrid platform creation time with proposed automation flow when compared to that of full manual design work. Though time to create hybrid platform has 3% overhead when compared to that of pure emulation platform creation, it is only 10 minutes and is negligible.

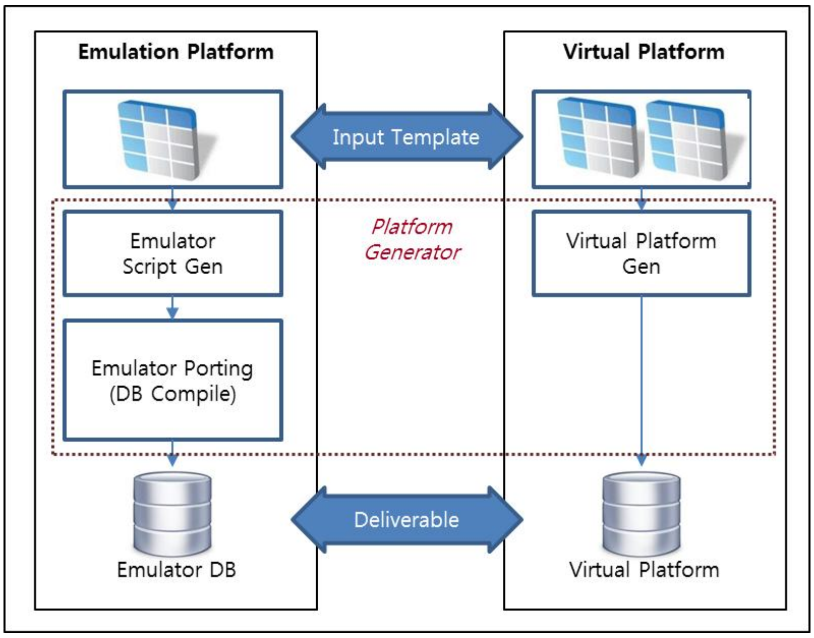

Figure 5. Automatic Hybrid Emulation Platform Generation Flow.

4. Application Results

For methodology validation, we applied the hybrid platform design methodology using our co-emulation framework for up-to-date flagship mobile AP. In our experiments, full SOC including peripherals and storage interfaces along with corresponding transactor and device models are also implemented in hybrid platform for complete device driver development and validation. Target AP consists of 2.5 billion logic gates and has 30 peripherals and 7 external storage devices. Full chip mobile AP was used for validation of the Hybrid Emulation Platform.

4-1. Platform Build Time

The proposed Hybrid Emulation Platform has two additional steps for platform build compared to pure emulation platform. One is transactor integration and the other is virtual platform build. Transactors should be connected to top design using RTL netlist. Therefore, some RTL files can be modified and re-compiled . During the build of virtual platform, IP instantiation, signal/interface connection, model configuration should be performed. Therefore, timing overhead for these steps is not small. Table 1 shows that comparison results for platform build time. In case of pure emulation platform, platform build was completed relatively quickly because virtual platform build and transactor integration is not required. In case of Hybrid Emulation Platform, platform build time increased to 63% by adding time for the manual virtual platform build and transactor integration. In order to minimize this timing overhead, automatic platform generation flow was applied. By the result data, automatic platform generation flow reduced platform build time to 37% compared with manual platform build time. In conclusion, the platform build time of the proposed platform was increased by 3% compared with pure emulation platform build time.

Table 1. Comparison for Platform Build Time.

	Pure Emulation	Hybrid (Manual)	Hybrid (Automatic)	Diff. (manual vs. hybrid)
Env. / Test Bench	20 min	60 min	20 min	▽ 66.7 %
DB Compile Time	360 min	360 min	360 min	▽ 0 %
Virtual Platform Build	0 min	200 min	10 min	▽ 95.0 %
Total Consumed Time	380 min	620 min	390 min	▽ 37.1 %

4-2. Performance Comparison

Device drivers for all IPs and Android Platform setup were performed on the Hybrid Emulation Platform. Finally Android Platform porting was completed on the platform before design of development board. Table 2 shows that comparison result for consumed time at each SW step among simulation, pure emulation and Hybrid Emulation Platform. In case of RTL simulation, time values were estimated using total cycle number for Android Platform boot-up and cps (cycle per second) and the values were used for the comparison due to its low simulation speed. Otherwise, pure emulation and Hybrid Emulation Platform were compared using real experimental values. As shown in Table 2, initialization time of simulation was shortest because emulator initialization and DB image transfer are not required in a simulation. However, time difference with pure emulation and Hybrid Emulation Platform was within one minute. It may be referred to as meaningless difference based on the total consumption time. For Linux kernel boot, simulation speed has been increased by 48 times when compared to that of pure emulation platform. Overall Android OS boot process took less than 1 hour and can provide efficient OS and boot code validation environment for SW developers. It is 14 times faster than pure emulation platform. During the linux kernel boot-up, most of operations are done in virtual platform . Furthermore, communication between virtual platform and emulation platform also is not much. Therefore, a high timing gain of Hybrid Emulation Platform was shown. However, timing gain of Android Platform boot-up is lower than that of linux kernel boot-up because a lot of operations such as initialization, register programming, and real function processing for each IPs were done in emulation platform and a lot of communications was done during Android Platform boot-up.

Operating time for Android Platform boot-up has accounted for 87% of the overall operating time. The timing gain on Android Platform boot-up represents timing gain of overall Hybrid Emulation Platform.

Table 2. Performance Comparison.

	Simulation	Pure Emulator	Hybrid Platform	Diff. (Pure vs. Hybrid)
Environment Initialization	4 min	5 min	5 min	x 1
Linux Kernel	125,867 min*	96 min	2 min	x 48
Android Platform	741,517 min*	661 min	47 min	x 14.1
Total Consumed Time	867,384 min*	762 min	54 min	x 14.1

* Estimated Value

Figure 6 shows Android Home Screen as a result of Android Platform boot-up on the proposed Hybrid Emulation Platform. We were able to see the image using a virtual LCD in the virtual platform. The frame data which were generated by display modules in emulation platform was saved in the frame buffer. The virtual LCD displayed the image by reading frame buffer using display PHY transactor. The proposed platform provided a SW development environment similar to the development board.

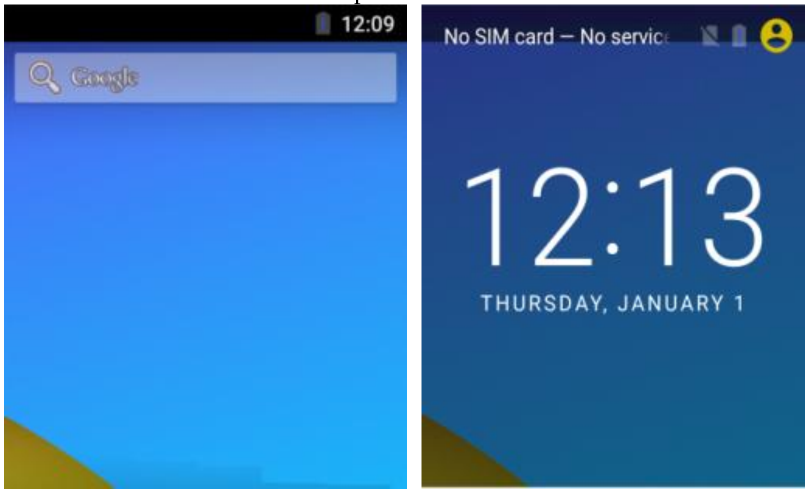

Figure 6. Android Home Screen at Hybrid Emulation Platform.

As shown in Table 2, a dozen of debug runs was possible when the hybrid platform is adopted, because only 7 minutes were enough for one linux kernel boot-up. It did not show any significant difference with development board. In case of Android Platform boot-up, simulation speed was slower than development board. However, multiple debug runs per work day was possible with this Hybrid Emulation Platform. We started developing and verifying SWs at the early RTL design stage using the Hybrid Emulation Platform. Display modules, camera modules and mobile storages such as eMMC, UFS, USB can be integrated to the proposed Hybrid Emulation Platform using various transactors. Most of the devices included in a mobile AP chip were developed and verified using the platform as the same way of development board.

5. Conclusion

This paper proposed a Hybrid HW Emulation Platform which enables early SW development at pre-silicon stage, where RTL design is available. To enable optimized SW development and validation environment for mobile-AP designs, transactors were designed to communicate with off-chip devices implemented in virtual platform. Moreover, overhead of access time of memory devices was minimized by placing CPU and memory subsystem in virtual platform. Application results showed 14 times faster simulation speed for Android boot-up compared to that of pure HW emulation systems. Also, automation of a hybrid platform creation flow reduced design time of a hybrid platform, and reduced human design errors as well. By applying the Hybrid Emulation Platform to a SW development and validation process, an Android platform was successfully developed before the delivery of a development board along with the fab-out of a mobile-AP chip-set. The hybrid platform also helped enhance quality of SW by allowing the SW developers to do more design and validation.
Future work will be to enhance the automation flow of the hybrid platforms with an IP-XACT based platform integration methodology. Also, more investigation on communication overhead between virtual and emulation platforms will enhance IP modeling methodology.

References

[1] G. Martin, B. Bailey, and A. Piziali, ESL DESIGN AND VERIFICATION: A Prescription for Electronic System Level Methodology, Morgan Kaufmann Publisher, 2010

[2] Wikipedia, Electronic system level; available at http://en,wikipedia.org/wiki/Electronic_system_level. Accessed July 5, 2006.

[3] K.H. Shim, W.J. Kim, K.H. Cho, and B. Min, “System-Level Simulation Acceleration forAarchitectural Performance Analysis using Hybrid Virtual Platform System,” Proc. ISOCC, pp.402-404, Jeju, Korea, Nov. 2012.

[4] Y. Nakamura, K. Hosokawa, I. Kuroda, K, Yoshikawa, and T. Yoshimura, “A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication ,” Proc. DAC, pp. 299-304, San Diego, CA, USA, June 2004.

[5] H. Wei, W. Xinan, D. Peng, and G. Zheng , “Implementation of high-speed verification platform based on emulator for ReDSP and ReMAP,” Proc. ASICON, pp. 682-685, Huan, China, Oct. 2009.

[6] C.Y. Huang, Y.F. Yin, C.J. Hsu, T. Huang, and T.M. Chang, “SoC HW/SW Verification and Validation,” Proc. ASP-DAC, pp. 297-300, Yokohama, Japan, Jan. 2011.

[7] M. Vavouras, K. Papadimitriou, and I. Papaefstathiou, “High-Speed FPGA-Based Implementations of a Generic Algorithm,” Proc. ICSAMOS, pp. 9-16, Samos, Greece, July, 2009.

[8] A. C. H. Ng, J. W. Weijers, M. Glassee, T. Schuster, B. Bougard, and L. Van der Perre, “ESL design and HW/SW co-verification of high-end Software Defined Radio platform,” Proc. CODES-ISSS, pp. 191-196, New York, NY, USA, May 2007.

[9] A. W. Ruan, Y. B. Liao, P. Li, Y. W. Wang, and W. C. Li, “A Stream-Mode Based HW/SW Co-Emulation System for SOC Test and Verification,” Proc. ICTD, pp. 1-4, Chengdu , China, April 2009.

[10] Sylvain Bayon de Noyer, “Hybrid Emulation Practical Use Cases,” Proc. DVCon India, ESL Poster, Bangalore, India, Sep. 2015

[11] Cadence, Palldium Hybrid; available at https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/acceleration-and-emulation/palladium-hybrid.html