2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Hybrid Emulation for faster Android Home screen bring up and Software Development

Rinkesh Yadav, Manoj Khandelwal, Sarang Kalbande, Garima Srivastava

Samsung Semiconductor India Research Bangalore

(rinkesh.y@samsung.com, manoj.k@samsung.com, sarang.mk@samsung.com, s.garima@samsung.com)

Hyundon Kim

Samsung Electronics, 1-1, Samsungjeonja-ro, Hwaseong-si, Gyeonggi-do, Korea

hyundon.kim@samsung.com

***Abstract*— SoC Software bring-up and its validation are usually done at post-silicon stage, which can affect time to market a product, and cost of silicon re-spin. It is also difficult to debug hardware/software design bugs at post-silicon level because of less RTL design visibility. So early software bring-up and validation is very crucial. Therefore, Platform Teams need a high-speed model of the SoC months ahead of silicon availability to begin Software bring-up and application driver development. Because of Less Emulator driverClk frequency software bring-up cannot be done in FPGA based Emulation Verification environment. Therefore, Platform Teams need a high-speed model of the SoC, months ahead of silicon availability to begin Software bring-up and application driver development. To overcome less emulator speed and to accelerate Software development, Hybrid Emulation is developed. Hybrid emulation combines virtual prototyping and hardware emulation. To speed up the hybrid emulation, some components of SoC move from emulator to virtual side. So in hybrid Emulation, one part of SoC design runs at emulator and other part of design run in virtual platform. Virtual prototypes are high performance, System C models of a particular IP, block or an entire SoC. These are modelling the behavior and inter-block communication at transaction level (TLM), which makes these faster than equivalent cycle-accurate RTL.**

Keywords—FPGA; Pure Emulation; Hybrid Emulation; Virtual Platform; System C Model; Software development;

I. INTRODUCTION

In today's competitive era, time-to-market a product is very important. SoCs are becoming increasingly complex, so early Software development and Verification is very crucial. Design teams cannot afford mistakes when making critical architecture decisions affecting performance or other requirements early in the development processes. SoC Software bring-up and its validation are usually done at post-silicon stage which can affect time to market a product and cost of silicon re-spin. It is also difficult to debug hardware/software design bugs at post-silicon level because of less RTL design visibility. Therefore, Software teams need a high-speed model of the SoC months ahead of silicon availability to begin application development. So it is viable to develop these Software much early at pre-silicon.

With the increasing complexity of SoC, the validation phase now has even more importance during the development. The risk of shipping buggy SoCs must be minimized through careful analysis and the use of different verification and validation processes, starting from the very beginning of the project and continuing through each design phase until the end of the life cycle of the chip.

The design of the SoC is usually just a small part of the job compared to the entire effort necessary to develop a new product. The majority of the costs are in the verification of the product and developing the software to run on it.

The following chart shows an analysis of the costs involved in various SoC Validation platform with respect to increase of design complexity:

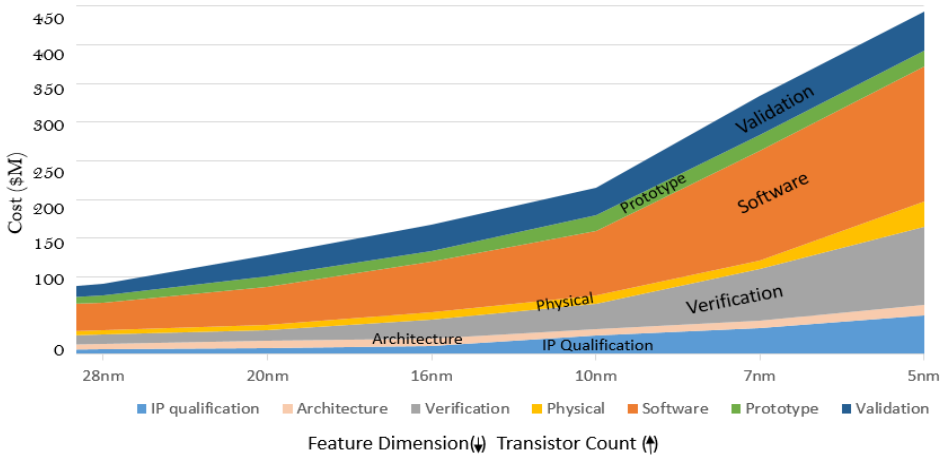


Figure 1. SoC development stages and their cost comparison based on scaling transistor size

The chart shows that the growth of the costs associated with validation and software have been growing exponentially when compared with the process technology scale; this is the reason why they both see a dramatic rise in importance, not only at the end of the design phase but from the beginning of the project, where hardware design, verification and software are performed simultaneously.

II. High performance Emulation methodology: Need & Impact

There are mainly four methods to test a design before the tape-out of chip; each of them has different advantages and disadvantages thus all four are used during the development of a SoC.

Validation Platfotm	Performance	Debug visiblity	Cost	SW Development	Use
Simulator	Low(in Hz)	High	Low	No	functional verification of the design
Emulator	Moderate(in KHz)	Moderate	Moderate	Still Not prefable	pre-silicon validation on bare metal
FPGA	High(in Mhz)	Low	High	Yes	silicon validation and software development
Silicon	High(in Ghz)	Very Low	High	Yes	silicon validation and software development

Table 1. Comparison of different Verification platform with respect to various aspects

i. Simulator

Simulation is used from the very beginning of the design because it gives a complete view of the signals in the RTL and it is often used to verify functionality at the module level. When the structure of the chip is close to being completed, the limits of this method start to rise quickly the huge computational power required for the simulation limits the speed of the execution to **100Hz** for the latest cores, requiring too much time to execute a boot of an operating system for example.

ii. Emulation

The performance problem could be partially solved by using an emulator, which, with specialized hardware, can reach a speed of **~5MHz (ZeBu Server 5)**, still keeping the same complete view of the internal status of the system. The main disadvantages of emulators are the cost of ownership, which is an order of magnitude bigger than the other methods, and again the time required to boot an actual operating system is not suitable for software developers who needs to the system reboot few times during the day when developing kernel and device drivers.

iii. FPGA

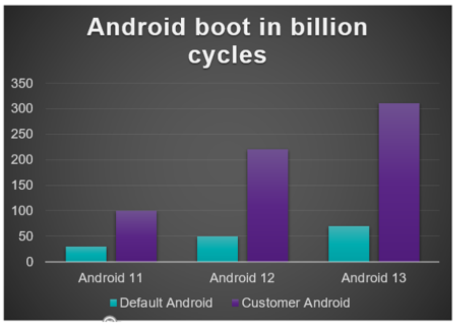
Software development and validation could be done at FPGA prototyping boards but because of high cost and less debug visibility, it is also not preferable for it.

iv. Silicon

Usually Software development and validation happen at post silicon stage. As increasing the SoC complexity and short schedule time, software development is also becoming time consuming and it is very challenging to complete it on time. Also less RTL debug visibility of silicon it is very difficult to debug and identify type of bugs either they are hardware bug or software bug. So to avoid time risk and silicon re-spin cost it is better to start software development and validation on pre-silicon stage. Therefore, we can avoid this risk.

A. Challenges to bring-up software in Pure Emulation.

Software development can be done using pure emulation but achieved emulator driver clock frequency is too less to bring-up Linux kernel, Android OS and to develop system level drivers for various complex IPs. It takes around 4 days' time to bring up complete android home screen on emulator where design is running at 1 MHz driver clock. So performing software development iterations on pure emulation is no more realistic.



android 13 boot time(cpu @3Mhz, 4cycles/ instruction):

(310Bx4/3 MHz)/3600s = 114 hours (~ 4 days)

Figure 2. Full Android 13 boot time on Emulator

To overcome less emulator speed and to accelerate software development, Hybrid emulation is developed. As Figure 3 shows hybrid emulation sits in between FPGA and Silicon with respect of debug to visibility and performance. So hybrid emulation has better performance to emulation and FPGA. It also helps easier platform upgradability and debug software bring-up due to better RTL design debug visibility compared to post-silicon. Furthermore, because of the full design visibility of hybrid emulation, it significantly helps users to debug any hardware/software issues and verify corresponding fixes against the accurate virtual model with a quick turnaround time. Hybrid emulation supports all emulator debug features along with software debugger like waveform dump, Trace32, TLM debugger, Transactor and Adaptor debugger through log levels.

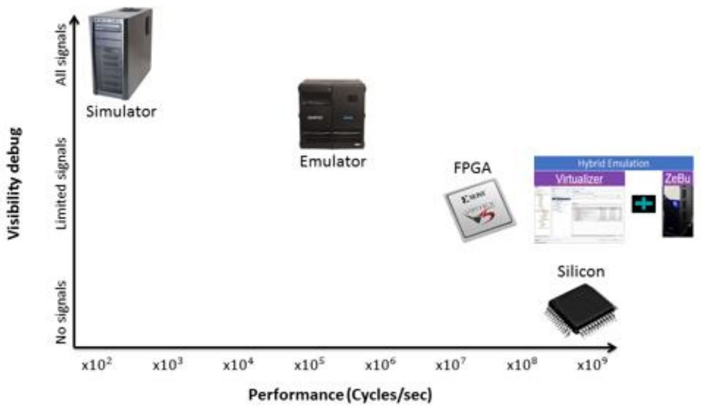


Figure 3. Hybrid Emulation Debug visibility VS Performance

III. Hybrid Emulation: Introduction

Hybrid emulation combines virtual prototyping and hardware emulation. Hybrid emulation is done when part of the design runs on the emulator and the other part runs in virtual prototype.

Hybrid platform has two domains:

Virtual Platform Side (TLM, VDK) :	CPU, GIC, MCT ,UART, PERI, Storages, DRAM
Emulation Side (RTL) :	Bus, Memory Controller, M/M IPs and others.

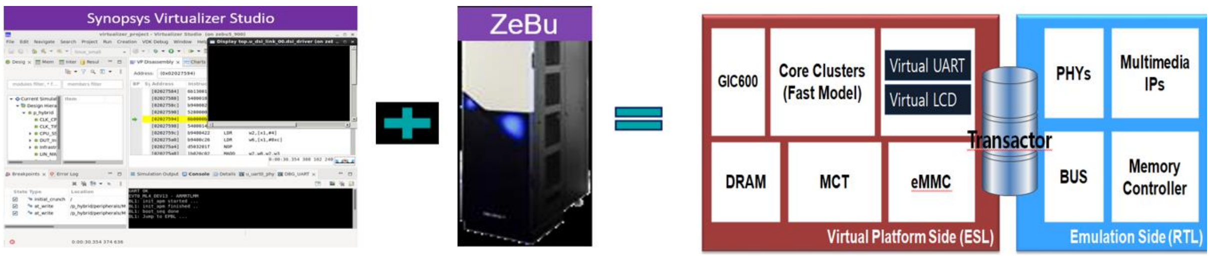


Figure 4. Hybrid Emulation

A. *The Virtual Prototype component*

Virtual prototypes are high performance, SystemC models of an entire SoC, a particular block or a system model. They have primarily been used by project architects to estimate SoC power, performance and for early software development.

Architects will use them for architectural design optimization and to check that their designs meet certain goals, such as performance and/or power. Architects may also use virtual prototypes to assess the correctness of the bus architecture.

B. *The Hardware Emulation component:*

An emulator runs much faster than a software-based RTL simulator and has a simulation-like use model. All the debug features available in simulation are present in emulation. The user can add dump bars, assertions, monitors and more.

Emulators make it possible to bring up a full SoC design before even an FPGA-based prototype is available. The design team can then begin hardware-software integration. The RTL probably is not stable at this stage, and this is why the full debug capability of the hardware emulator is so important.

IV. Development of Hybrid Platform

Hybrid emulation enables SoC development teams to take full advantage of their investments in a high-performance emulator across their entire project to meet increasing software, verification and competitive challenges. Typically, a model of the processor is run in the virtual platform and then the rest of the RTL design runs on the emulator.

The task of a virtual platform is to have just enough accuracy to support the level of software being run on it. This is achieved by modelling the behavior and inter-block communications at transaction level, which makes these faster than equivalent cycle-accurate hardware. If user has the necessary models or libraries of models usually written in SystemC, then user can create a virtual platform for any SoC. Such libraries of SystemC models are available as open source or commercial packages such as the system-level library.

Usually ARM IP dominates SoC designs, so you will need models of ARM cores and bus sub-systems. ARM supplies such models by the name of Fast Models, and these models are already in wide use in virtual platforms.

A. *Partitioning an SoC across virtual and hardware platforms*

In Hybrid emulation, there will be some blocks of the SoC design running in FPGA hardware and the rest in virtual models. In some cases, the choice will be governed by the availability of models or RTL. If you have both, however, then it will depend on the side of the virtual-hardware boundary where that particular block is placed. This leads to the critical question of where to place the boundary between virtual and physical domains.

In pure emulation model, there are many transactions between CPU and DRAM. This is bottleneck for Android/Linux booting and Software development. To overcome this, CPU and related components are moved to virtual side (S/W side) for OS boot and run acceleration in hybrid platform. This acceleration is achieved in Hybrid because of virtual model of the CPU runs Instruction Set Simulation (ISS) to perform much faster compared to CPU's gate-level behavior in emulator. Hybrid emulation has Fastmem model of DRAM instead of RTL memory model. Hybrid fast memory is shared memory between virtualizer and emulator, which is properly sync between them.

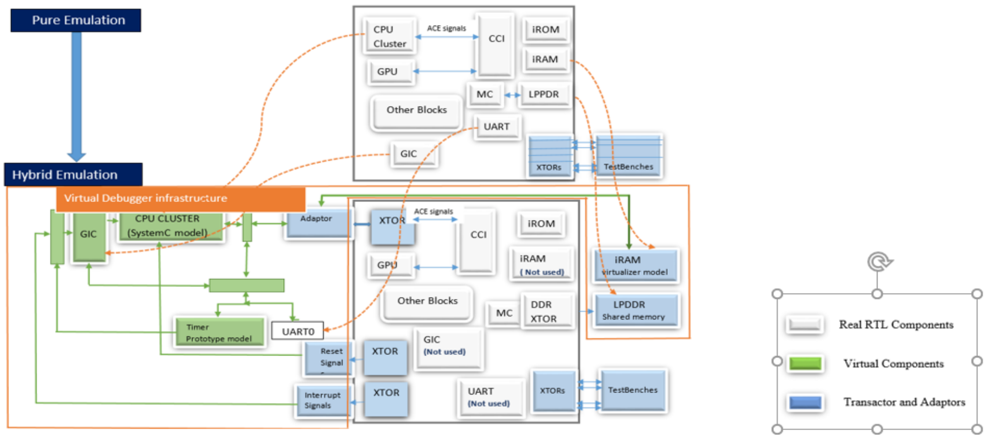


Figure 5. Partitioning an SoC across virtual and hardware platforms

B. *Hybrid Platform Architecture*

Figure 6 shows Hybrid Platform architecture on ZeBu 5 Server. To connect virtual component to hardware on emulator various Adaptors and Transactor are getting used. Adaptors are integrated at Virtualizer studio and these are used to connect virtual component to hardware through different Transactor implemented in Emulation testbench. ZHAL is Zebu hybrid adaptors library which provide wrapper of all require adaptors. These wrappers can be used inside virtualizer as an TLM model.

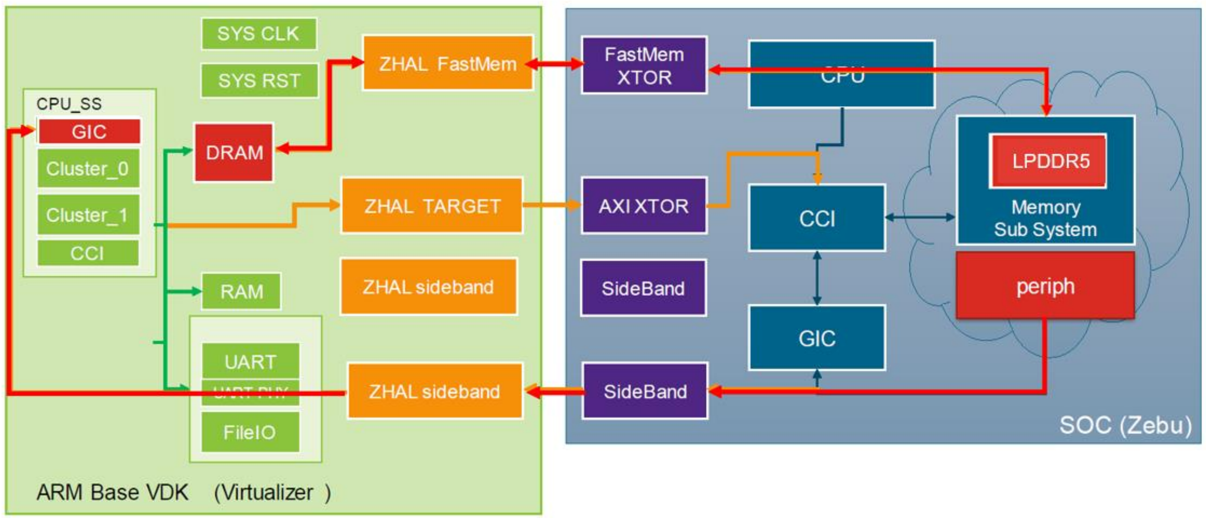


Figure 6. Hybrid Platform Architecture

As discussed above, while booting Linux kernel most of the transaction happens between CPU and DRAM. To reduce CPU to DRAM memory access latency, shared dram memory is developed in hybrid platform. In shared memory, we have dram ZRM memory on emulator and shadow memory at Virtual Side, which are properly synced through Transactor to minimize software virtual and emulator transaction.

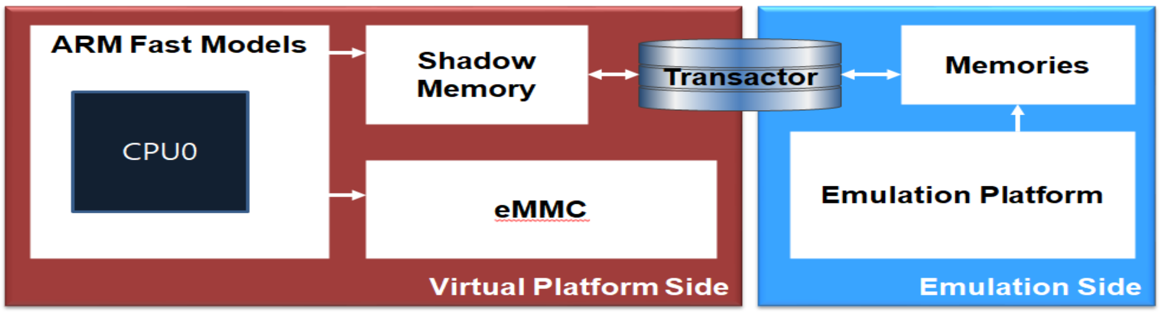


Figure 7. Shared dram memory model in hybrid Emulation

C. *Debugging Methods*

In any validation platform, supported debugging methods are very important. Hybrid Emulation supports various software and hardware debugging methods to debug any encountered bugs while developing software.
Below are few debugging way which hybrid emulation supports:

i. TLM transaction debug through Chart View
ii. SW debugger through VP disassembly feature and t32 debugger
iii. Transactor and Adaptor debug through different log levels support
iv. Hardware debug through waveform dump in Verdi

First Two debugging methods usually used to debug Software bugs by putting breakpoints, bypass the function, forcing the register and dumping the memory through T32 debugger. Third debugging method is used to debug transaction between software and hardware through various Adaptors supported log levels. Fourth method is used to debug RTL functional behavior through waveform dump in Verdi.

D. *Challenges and Limitations*

One challenge with the hybrid emulation approach is that it can generate insane amounts of data very fast. As a result, there is often a window of a million or so cycles that are recorded and can be used when a bug is encountered to investigate the root cause. However, running software and emulation and doing a system boot or bringing up a software may take billions of instructions. The point at which the bug was injected by an unexpected behavior and the point at which it is first observed may be too far apart for the million-cycle window to be useful. Therefore, it is little bit difficult to debug Software bugs in System C models. While in Pure Zebu, Software debugging is easy because whole SoC runs on hardware.

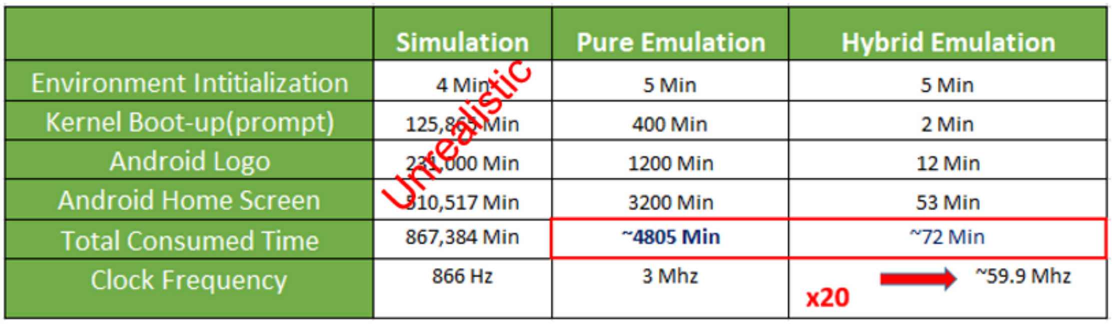
Hybrid Emulation has some limitations. As in hybrid emulation, we use System C and TLM model of some SoC components, which does not support Cache coherency. So while software development through hybrid emulation we need to disable cache coherency. One more limitation ARM Fast models has, it does not support multi-core simulation so we also need to disable SMP while developing Linux Kernel and Android home screen.

V. Results and Benefits: Pure Emulation V/S Hybrid Emulation

Software brought-up times shows in Table 2 taken from our Complex Exynos SoC, which has almost 2 billion gate count.

As Table 2 shows, In Simulation, software development is unrealistic because of less run time performance. In Emulation platform takes almost 4 days to bring-up full Android home screen. Therefore, Pure emulation platform is also not preferable to bring-up software where we need to run many iterations while developing software and drivers.

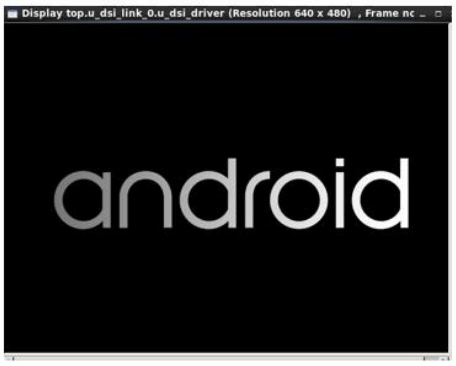
Hybrid Emulation takes almost 2 Minutes to boot Linux kernel, 12 Minutes to get Android Logo and total 72 Minutes to brought-up Android Home Screen on display panel.



	Simulation	Pure Emulation	Hybrid Emulation
Environment Intitialization	4 Min	5 Min	5 Min
Kernel Boot-up(prompt)	125,865 Min	400 Min	2 Min
Android Logo	231,000 Min	1200 Min	12 Min
Android Home Screen	510,517 Min	3200 Min	53 Min
Total Consumed Time	867,384 Min	**~4805 Min**	~72 Min
Clock Frequency	866 Hz	3 Mhz	x20 → ~59.9 Mhz

Unrealistic

Table 2. Software bring-up time comparison in Various Platforms


Display top.u_dsi_link_0.u_dsi_driver (Resolution 640 x 480) , Frame nc
android

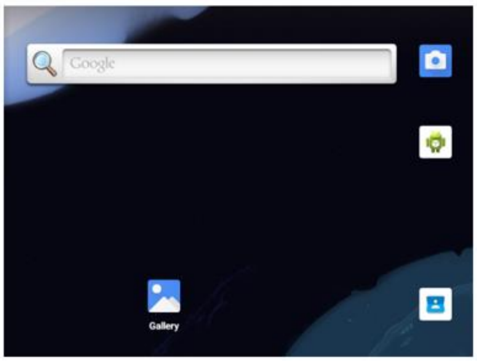


Figure 7. Snapshot of Android Logo and Home Screen on display panel

VI. FUTURE SCOPE:

A. *Enabling CPU Benchmarks:*

In Hybrid Emulation, to Enhance performance we black box real RTL CPU from emulator and integrate ARM released System C fast model of corresponding CPU at virtual side. So it is not possible to perform any CPU benchmark in hybrid emulation.
So we are trying to enable, perform CPU benchmarks in hybrid emulation through Gear shift methodology. In this methodology, once Linux/Android is brought-up with virtual side System C CPU model then left shift all CPU STATES from virtual CPU to RTL CPU. As **Linux/Android** is already **booted up**, **CPU Benchmark** can be run with Real RTL cpu on emulator.

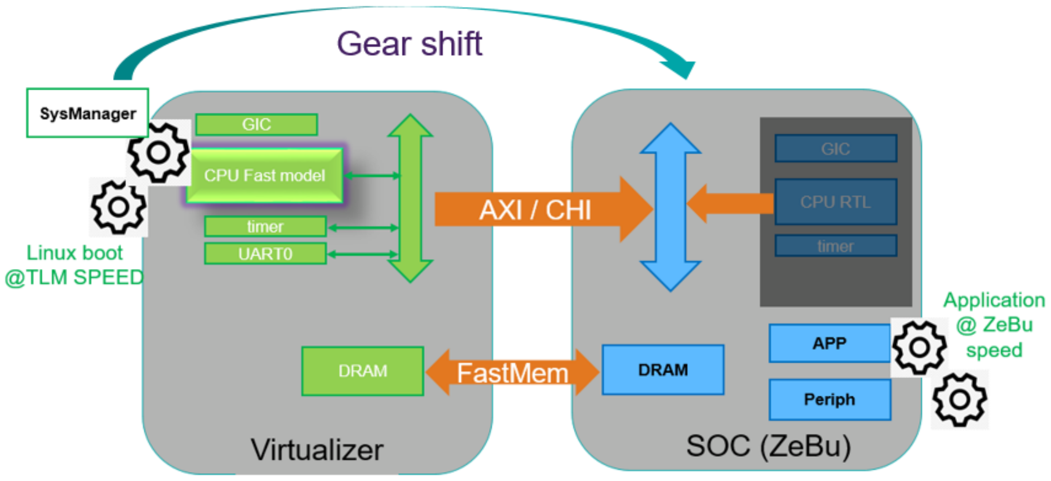


Figure 7. Gear shift methodology in hybrid emulation to perform CPU benchmark

B. *Multicore support enablement in Hybrid Emulation:*

As we mentioned section iv(d), In hybrid emulation challenges multicore Linux boot (SMP) and Cache Coherency does not support. Therefore, we are trying to find alternate way to support SMP in hybrid emulation.

C. *Multichip support enablement in Hybrid emulation:*

As SoCs are becoming more complex so it is hard to fabricate them in single chip, so Semiconductor industries are exploring another solution and they are moving to multi-die and multi-chip based chips. Therefore, we are also working on to enabling hybrid emulation to support our multi-chip base SoCs.

ACKNOWLEDGMENT

We would like to express my gratitude to Mr. HwanSung Park(Samsung Semiconductor Korea) and Synopsys Korea Support Team specially Mr. Tae-Hyung Kim, Mr. Jay Seo and DongSu Park for their guidance and support.

REFERENCES

[1] Issac P Zacharia and Jitendra Aggarwal "Hybrid Emulation: Accelerating Software driven verification and debug" Presented at DVCON India 2021 conference.

[2] Article "How FPGA boards help to validate ARM processors" (https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/how-fpga-boards-help-to-validate-arm-processors) [Page 1-3]

[3] Synopsys Hybrid Emulation Documents and Quick Start User Guide.