VIRTUALISED ECOSYSTEM TO ENVISAGE NUMEROUS APPLICATIONS ON AN AUTOMOTIVE MICROCONTROLLER

Meghashyam Ashwathnarayan, Vaishnavi J,
Ananth Kamath and Jayakrishna Guddeti

Infineon Technologies India Pvt Ltd, 11MG Road, Bengaluru, Karnataka, India

ABSTRACT

In automotive electronics, new technologies are getting integrated into basic framework creating ways for new software-defined architectures. Virtualization is one of most discussed technologies which will offer a functionality growth into architecture of automobile. This paper introduces concept of validating test cases from multiple IPs on virtualised framework and also investigate the feasibility of implementing protection mechanism on memory segment dedicated for a virtual machine (VM). We describe a proof-of-concept which can be used to promote the use of virtualisation to extend the coverage of post silicon validation. Experimental results are presented as a quantitative evaluation of using virtualization for different testcase scenarios.

KEYWORDS

Virtualisation, Automotive, Multi-Core Systems, Hypervisor, Post Silicon Validation.

1. INTRODUCTION

Automotive industry at present is seeing a tremendous change. The latest connected technology trends are forcing old, purpose-built embedded systems to be modified or replaced by new technologies that defines a new dimension for software-based architecture. As functionality of automobile increases, the number of functional hardware modules and software used by automakers grows [1] [2]. Handling the rise in number of ECUs and complexity in an automobile has become a challenge for manufacturers (OEMs) [3]. Recently, there have been a lot of discussions about trends that can offer reduction of ECUs and to develop safe modules without restricting costumer's requirement [4]. Implementing new architectures into old which can help us manage complexity, power consumption, cost and weight. One emerging architecture in this industry is virtualisation. In [6][4], Gernot Heiser briefly talks about automotive industry with virtualisation in near future and benefits of this approach.

Virtualisation, is a technology by which multiple virtual machines (VMs) are multiplexed on single hardware machine ensuing a logical division of available physical resources. Virtualisation makes it possible to assign the hardware resources to multiple isolated applications. This is an effective approach to restructure the existing architecture, take full benefit of the performance of processors and concentrate on the increasing complexity of software-defined functions in vehicle. This project aims to implement framework for development and validation of testcases using concept of virtualization. In testcases requiring the fulfilment of safety standards, it is important to include the related safety aspects in the workflow which is offered by the addition of protection mechanisms. Since high-quality testcases for post-silicon validation should be prepared before a silicon is available in order to reduce time spent on preparing the tests,

David C. Wyld et al. (Eds): EMSA, SEA, AIFU, NLCAI, BDML, BIoT, NCOM, CLOUD, CCSEA, SIPRO - 2022
pp. 17-26, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120602

debugging and fixing it after the silicon is available. We propose an approach of executing of post-silicon validation tests on virtual machines for improved test coverage. The proposed workflow should be able to provide parallel validation of multiple IP modules, while not decreasing the efficiency of the workflow. We also intend to understand potential savings in time and bill of materials used during validating both framework as well as IP modules.

2. WHAT IS VIRTUALISATION?

Virtualisation creates a virtual environment that replicates functionality to that of physical hardware machine. A virtual machine (VM) is an environment that is mounted on software, which synchronizes the execution of the VM's dedicated hardware. A software layer that resides between the VMs and the hardware is known as Hypervisor or Virtual Machine Manager (VMM). In [7], Popek and Goldberg highlights the following essential characteristics of a VMM:

[1] Program running under the VMM should display expected functioning that is identical to that of demonstrated behaviour when running directly on the underlying hardware platform.
[2] Overwhelming load should not degrade performance of VMM.
[3] VMM should have complete control of the physical hardware resources which are allocated to the guest OS at all times.

Virtualised framework typically consists of a real-time component which performs critical tasks within a certain deadline and a general-purpose component that may contain processing information, configuring or managing the system.

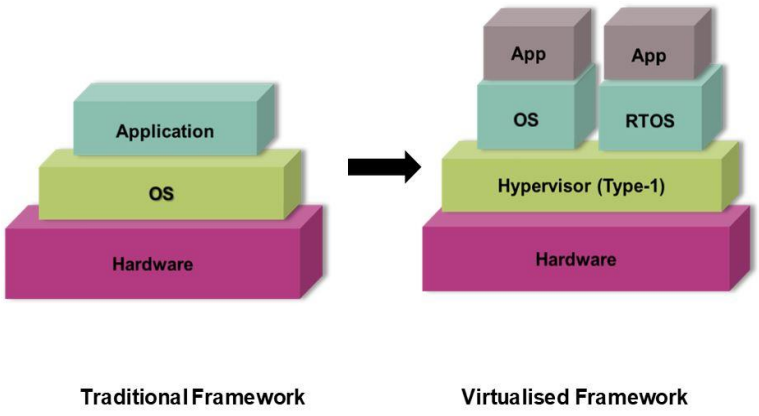

Fig. 1. Virtualised framework for Automotive domain

VMM software makes it possible to split the physical resources of a processor into safely separated segments for VMs mounted thus providing a near-complete isolation. Since VM is the logical replica of a physical hardware, multiple VMs can be installed on a same hardware and are logically separated providing isolation on the same hardware [5] [8]. The OS and application are completely unaware they are sharing hardware resources with other applications.

3. STATE OF THE ART

In the IT infrastructure, virtualisation is used to create multiple server instances from a physical server. Unlike a server that uses virtualisation to run many independent isolated servers on its VMs, embedded systems are highly integrated [9]. The prerequisites for a hypervisor for automotive systems is different from that of an IT domain. The requirements of hypervisor can be summarized as below [10] [9] [11].

VMM should incorporate processor architectures aiming embedded systems.

[1] Support secure encapsulation of subsystem components that interact with another subsystem strongly.
[2] Minimal impact on hardware resources and real-time performance such as fault-tolerance and time-related characteristics.
[3] Supports a scheduling policy between VMs and provide priority for real-time system components.
[4] Ensures strong spatial and temporal isolation between VMs by using concept of memory protection unit (MPU).

The prerequisite to integrate applications with very different requirements, modularly on a single processor that have the computing power to run these using a hypervisor [5].

4. VIRTUALISATION IN AUTOMOTIVE EMBEDDED SYSTEM

The total number of ECUs in the automotive network is growing due to the new functions which are implemented per ECU. Semiconductor vendors are also met with new challenges. Moore's Law defines that need for additional computing power and chip performance will double every 18–24 months. For the customer's comfort, demand for implementing cost-effective and innovative function in cars are growing. Over years of discussion on whether running multiple function on an ECU is a solution to reduce the ever-increasing digits of computing devices inside an automotive vehicle has started to find certainty in future architecture [12]. Since auto-mobile is a high-complexity device, introducing this approach requires thorough consideration about requirements regarding safety, availability, security, performance and resource management. This discussion often leads to virtualisation associated with a hypervisor which provides freedom from interference of different domains in auto mobile. The understanding of incorporation of such a technology on a single core is already complex. A challenge that we face is to integrate automotive customized functions into this new technology. Since Virtualisation presents the opportunity for multicore complexity and layer of abstraction for hardware, thus it offers multiple opportunities automotive industry [13].

A. Virtualised Framework

The framework envisages a virtualised isolated framework containing four VMs. In one of the derivatives of TC4x microcontroller, three hardware resource partitions (HRP) are utilised, the first HRP is used by the Hypervisor, the second HRP is used by the real-time VM1 and the third HRP is used by non-real-time VMs (VM2 to VMx). The figure virtualised ecosystem depicts a multicore System on Chip (SoC), a VMM and four VMs assigned to Core0. Each subsystem in a VM have four IPs being used to create an application. The IPs can be reused in different VMs, however a strict isolation of memories, sharing of resources should be enforced. In order to test the above concept of running multiple subsystem on a virtualised framework, a demo setup was created. Majority of the subsystems in independent VMs use Direct Memory Access (DMA) as

one of the IP used in the application. The DMA IP, as it is shared by all the VMs need to be partitioned in a way that the no channel is overlapped by another VM. A set of channels are assigned to a particular VM and are thoroughly isolated from each other using access protection offered in A3G.

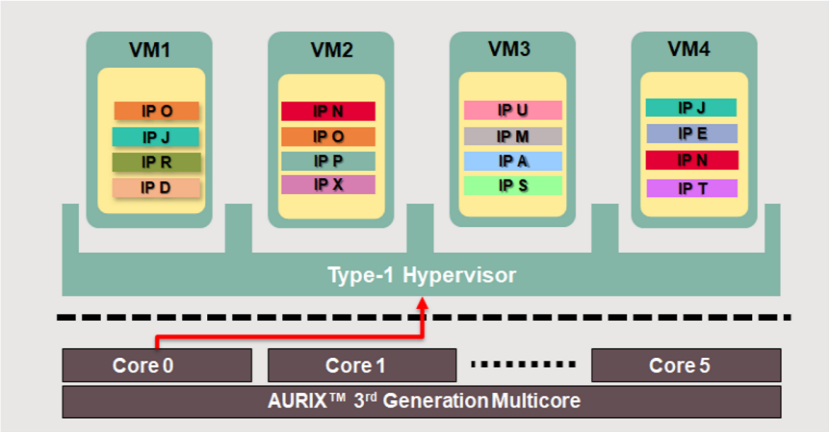

Fig. 2. Virtualised Ecosystem

5. VALIDATION IN VIRTUALISED FRAMEWORK

Post Silicon Validation commonly involves developing test cases for validating a design, analysing the results, involving in debug with assistance of designers. A significant amount of study is focused on detecting bugs in these silicon chips. With our approach of using virtualisation framework, it employs us to accelerate the validation with ease as various testcases can be deployed into multiple VMs and with no interference between the VMs, each testcase works independently. Every VM that is enabled will have application to be tested integrated on it. In figure 3 “Validation flow in AURIX™/TriCore™”, we can deploy four different testcases of different IPs on four different VMs. In our proposed framework, processor now has extended into four more VMs that copies processor functionality thus increased capability of software than before. With addition of protection mechanism for each VM, we deal with safety feature where it eliminates the possibility of the memories getting overlapped thus saving unnecessary traps while debugging. Near-complete isolation between application on the same hardware provides us advantage on testing multiple testcases. At present we are preparing the testcases by running on virtualised framework on emulator.

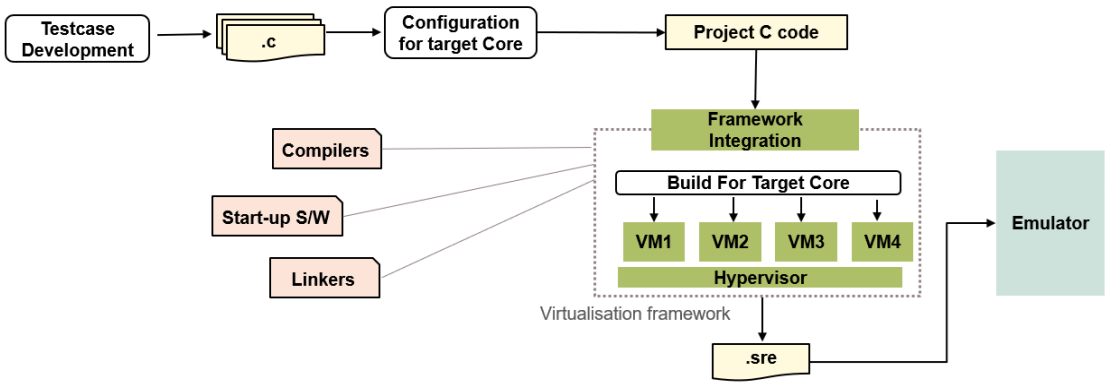

Fig. 3. Validation flow using virtualisation in AURIX™/TriCore™

We intend to run it on the chip once it is available. We have managed to find number of issues while integrating testcases into VMs. The main advantage that we have derived running these applications on an integrated platform was having a system level approach instead of having a directed test known to only validate a particular functionality/feature in an IP. Due to this, numerous issues lying in the interface of IPs have been found. The other positive is that the real-world applications can be deployed and validated in a virtualised environment there by increasing the confidence to the customer regarding virtualisation in the automotive field.

6. METHODOLOGY

Our main insight for this project is to enable application execution on a virtualised framework and to offer encapsulation to integrated applications to be tested and deployed in an automotive clustered environment. The main challenge when involving VMs lies in execution of an application with real-time constraints [5] and a framework that can support sharing of hardware resources between VMs. This requires a hypervisor with a scheduler which provides equal time slice for VMs enabled.

6.1. Hypervisor

AURIX™/TriCore™ uses Type-I VMM that is installed on the primary boot system of hardware and executes at the highest level of privilege with full control over any VMs that use it [14]. As Type 1 VMM often needs a scheduler to administer multiple VMs [15], a need for scheduling algorithm must be thoughtfully recommended in the hypervisor software where the priority is given to real time VM [16]. Generally, the VMM is executed in the privileged mode and hence has full access to hardware resources whereas VM is run under user processor mode. Multiple operating systems, serving the different needs of various subsystems, such as real-time environment and non-safety critical applications can be run on the virtualised framework [5]. With security sensitive aspect kept as a priority, we come up with best fit solution on how to use physical resources [8]. These set of channels service only that VM that it is assigned for. At no instance, the subsystems or application running in the VMs cause a conflict when requesting a DMA service.

When creating a subsystem, a section of memory range is assigned for the smooth operation of the VM. A high-level protection mechanism can also be setup such that when a non-assigned VM

tries to access a memory region which is out of its range an interrupt or an alarm is raised informing that it is an illegal access. These memory ranges are dedicated to a particular VM and is never shared by another VM hence is able to provide secure encapsulation for applications running on it. Another common IP that is predominantly used in the subsystems is the timer module, just like the DMA channel, each timer is setup to run independently by a VM without any conflict. Interrupt service routines (ISRs) are created independently for a VM. To avoid conflict of interrupt priority numbers, all the ISRs are assigned priority numbers mutually exclusive to one another.

6.2. Validation of DMA module in virtualised framework

Direct memory access (DMA) is a means of speeding up data transfer between peripheral device and processor's memory bus while reducing work overload on the CPU. The DMA implements virtualisation by extending the concept of HRPs. In this approach, HRPs are used to isolate sets of DMA channels from each other and provide isolation from interference of channels in other HRPs when running functional operation. In TC4x, we have a number of resource partitions that can be assigned to any of the channels being used. Table 1 provides details on how channels of DMA0 is segregated into different resource partition along with its access master, virtual machine number, transfer operation and memory region in use.

Table 1. Resource partitioning of DMA in virtualised framework.

Ch No	VM No	HRP	Master	Src	Write Access (src)	Read Access (src)	Dst	Write Access (dst)	Read Access (dst)
0-15	VM1	RP0	DMA0	LMU0	No	Yes	LMU2	Yes	No
16-31	VM2	RP1	DMA0	LMU1	No	Yes	LMU3	Yes	No
32-47	VM3	RP2	DMA0	LMU4	No	Yes	LMU6	Yes	No
48-63	VM4	RP3	DMA0	LMU5	No	Yes	LMU7	Yes	No
64-79	VM1	RP4	DMA0	LMU0	No	Yes	LMU2	Yes	No
80-95	VM2	RP5	DMA0	LMU1	No	Yes	LMU3	Yes	No
96-111	VM3	RP6	DMA0	LMU4	No	Yes	LMU6	Yes	No
112-127	VM4	RP7	DMA0	LMU5	No	Yes	LMU7	Yes	No

6.3. Use of protection mechanism for DMA module

Each HRP has its own protection mechanism register which provides write protection to the HRP access registers (AR), whereas AR provides read and write protection to HRP specific registers which controls data transfer operation of DMA. Whether or not a channel is allowed to write or read to or from memory range is allocated in the access registers. To prevent unexpected use of memory, the access register memory range can be modified to restrict accesses to a certain range per channel/HRP. The memory region in which the HRP is configured can generate alarms or trap and deny any illegal access from other channel, which lies in a different HRP.

Function of access protection in DMA in a nutshell:

1) Attempt to access for DMA by unauthorized master leads to a trap.
2) It allows transfer operations which obey the access policy.
3) DMA channels under HRP configured by protection mechanism can request for DMA service.

6.4. Sequence of execution

1) Start-up software and firmware gets executed on the microcontroller. Necessary configurations with respect to virtualisation are configured.
2) Once control is in the VMM, it initialises the states and allocates the resources of the VMs which are enabled.
3) The VMM starts scheduling VM1 for execution.
4) DMA0 channels which are assigned in HRP0 and HRP4 are executed to transfer data from respective source and destination addresses.
5) After the DMA0 channels have finished their respective transactions, PPUC is scheduled as service by VM1. PPUC executes a complex fast Fourier transform (FFT) algorithm. Once the service is relinquished the control goes back to VM1.
6) A VMM call is made from VM1 so that the control gets to VMM and can schedule the next VM i.e. VM2.
7) This sequence of scheduling tasks by VMs continue until all the enabled VM's have performed their operations.

7. ADVANTAGE OF VIRTUALISATION IN AUTOMOTIVE DOMAIN

Introducing virtualisation into framework enables advantage [17] for vehicular architectures such as:

A. Hardware isolation: Virtualisation provides the reusability of software. It also enables the upgrade of outdated infrastructure or application. It improves the portability of application or function to different hardware and OS platforms.

B. Hardware costs: Virtualisation reduces hardware costs by enabling consolidation into a shared hardware resource. Efficiency of the hardware is improved as processor time, memories as well as peripherals are being shared by application running on different virtual machine. It allows users to take full benefit of concurrency provided by a multicore architecture.

C. System management: Merging multiple ECUs will reduce hardware required which in turn decreases the production and operating costs. As multiple ECUs are combined together the power consumption in the vehicle is significantly reduced.

D. Isolation: As mentioned in [7], VMM should be able to provide isolation of VMs. Isolation is also provided to each VM to operate independently without causing an impact to the system whenever any of the VM fails. This can be achieved by using timers in VMM such that if certain VM is unable to complete its function at assigned execution time, interrupt can be generated to request VMM to take necessary actions. When an application running on a VM faces failure, this failure does not propagate to the system and interfere during execution of other VMs.

E. Reliability and robustness: The isolation and modularity provided by VMs improve reliability and robustness by reducing the effect of failure on a single VM.

F. Safety and security: With implementing appropriate protection access mechanisms, it is able to offer secure encapsulation where interference between subsystems is reduced.

8. RESULT

The results in this section is regarding the correctness and performance evaluation of the DMA functionality running on framework. Firstly, for framework validation, we have provided snapshot from emulator that illustrates basic print statements in VMs of core0 which then was customized as per fig 3. With this, we are trying to validate whether VMs enabled by user are getting switched once finished its application without any error.

Secondly, we added DMA testcases on to the VMs to evaluate on how framework works. The main objective of running DMA module would be ease of validating data packet transaction with function that checks the data stored in both src and dst addresses using protection mechanism for DMA accesses as mentioned on table 1.

Finally, to get the proof of the DMA transferring operation, status register of DMA has been checked along to verify data is sent accordingly for respective LMU addresses. The desired results were obtained and we also saw generation of LMUaccess error when channels are accessed out of set memory region.

```
[dut_tb.inst_performance_counter_dut:3] started
   53595 |   911.106 uS | TriCore CPU0.1  | % TC0 VM1
   54051 |   915.666 uS | TriCore CPU0.2  | % TC0 VM2
   56345 |   938.606 uS | TriCore CPU0.3  | % TC0 VM3
   58985 |   965.006 uS | TriCore CPU0.4  | % TC0 VM4
   61481 |   989.966 uS | TriCore CPU0.5  | % TC0 VM5
   64023 |  1015.386 uS | TriCore CPU0.6  | % TC0 VM6
   66491 |  1040.066 uS | TriCore CPU0.7  | % TC0 VM7

           V C S   S i m u l a t i o n   R e p o r t
Time: 30000000000 ps
CPU Time:    317.490 seconds;       Data structure size:   6.7Mb
```

Fig. 4. Emulator snap of 7 VM shifting

9. CONCLUSIONS AND FUTURE SCOPE

Virtualisation approach tends to be efficient, which provides a benefit to use minimalistic hardware furthermore improving its security and CPU utilization. With a brief introduction to virtualisation techniques, we have presented a discussion on how our framework proposes testcases to run on VMs with additional protection mechanism to avoid illegal access. In our future research, we will explore on how to use this framework to its fullest which might include execution of automotive IP application such as non-safety critical applications such as fuel gauge monitor, battery level indicator, tyre pressure indicator and temperature monitor in-place of testcases while reducing standard footmark of IPs being used furthermore reducing the bill of materials while creating these subsystems.

10. ACKNOWLEDGMENTS

The authors thank Mr. Shyam Kommajosyula, head of the post-silicon validation department, Infineon Technologies India, Bengaluru and Pankaj Moharikar, line manager for their support in publishing this paper.

Acronym List

Acronym	Definition
AR	Access Registers
APU	Access Protection Unit
A3G	AURIX™/Tricore™ 3rd Generation
CPU	Central Processing Unit
DMA	Direct Memory Access
ECU	Electronic Control Unit
FFT	Fast Fourier Transform
HRP	Hardware Resource Partition
IP	Intellectual Property
ISR	Interrupt Service Routine
IT	Information technology
LMU	Local memory unit
MPU	Memory Protection Unit
OEM	Original Equipment Manufacturer
OS	Operating system
PPUC	Parallel processing unit controller
PROT	Protection Register

REFERENCES

[1] M. Broy, "Challenges in automotive software engineering," in Proceedings of the 28th international conference on Software engineering, 2006, pp. 33–42.

[2] H. Hanselmann, "Challenges in automotive software engineering," in Companion of the 30th international conference on Software engineering, 2008, pp. 888–888.

[3] Y. Onuma, Y. Terashima, and R. Kiyohara, "Ecu software updating in future vehicle networks," in 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2017, pp. 35–40.

[4] D. Reinhardt, D. Kaule, and M. Kucera, "Achieving a scalable e/e□architecture using autosar and virtualization," SAE International Journal of Passenger Cars-Electronic and Electrical Systems, vol. 6, no. 2013- 01-1399, pp. 489–497, 2013.

[5] N. Navet, B. Delord, M. Baumeister et al., "Virtualization in automotive embedded systems: an outlook," in Seminar at RTS Embedded Systems. Citeseer, 2010.

[6] G. Heiser, "The role of virtualization in embedded systems," in Proceedings of the 1st workshop on Isolation and integration in embedded systems, 2008, pp. 11–16.

[7] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," Communications of the ACM, vol. 17, no. 7, pp. 412–421, 1974.

[8] M. Strobl, M. Kucera, A. Foeldi, T. Waas, N. Balbierer, and C. Hilbert, "Towards automotive virtualization," in 2013 International Conference on Applied Electronics. IEEE, 2013, pp. 1–6.

[9] R. Mijat and A. Nightingale, "Virtualization is coming to a platform near you," ARM white paper, vol. 20, 2011.

[10] I. Pavic and H. Dzapo, "Virtualization in multicore real-time embedded ˇ systems for improvement of interrupt latency," 05 2018, pp. 1405–1410.

[11] M. Traub, A. Maier, and K. L. Barbehon, "Future automotive ¨ architecture and the impact of it trends," IEEE Software, vol. 34, no. 3, pp. 27–32, 2017.

[12] G. K. Thiruvathukal, K. Hinsen, K. Laufer, and J. Kaylor, "Virtualization ¨ for computational scientists," Computing in Science Engineering, vol. 12, no. 4, pp. 52–61, 2010.

[13] J. E. Smith and R. Nair, "The architecture of virtual machines," Computer, vol. 38, no. 5, pp. 32–38, 2005.

[14] D. Reinhardt and G. Morgan, "An embedded hypervisor for safety□relevant automotive e/e-systems," in Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014), 2014, pp. 189–198.

[15] A. Aguiar and F. Hessel, "Embedded systems' virtualization: The next challenge?" in Proceedings of 2010 21st IEEE International Symposium on Rapid System Protyping. IEEE, 2010, pp. 1–7.
[16] Z. Gu and Q. Zhao, "A state-of-the-art survey on real-time issues in embedded systems virtualization," 2012.
[17] J. Pelzl, M. Wolf, and T. Wollinger, "Virtualization technologies for cars," Tech. Rep., 2008.

AUTHORS

Meghashyam Ashwathnarayan Graduated from Electronics and Communication in 2006-07. Following this, worked at McAfee for 3 years before pursuing MS in VLSI at MIT, Manipal. Currently working in Infineon Technologies, Bengaluru. Areas of expertise is in the field of Automotive Microcontrollers Post-silicon Validation and Compute architecture. Has numerous papers/publications/journals.

Jayakrishna Guddeti Lead Principal Engineer and design lead in microcontroller design and development across different functional areas, with total of 16 years of experience in Silicon Architecture, Digital Design and Post-Silicon Validation. Has 12 Patents and various IEEE publications.

Vaishnavi J received the M.E. degree in Embedded Systems from Manipal School of Information Science, Manipal in 2021. She currently works as a trainee engineer at Infineon Technologies, Bangalore. Her current interests include PCIe communication, virtualisation.

Ananth Kamath, Senior Specialist Software and Firmware Engineer at Infineon Technologies India. I am part of the Infrastructure Software team, my responsibilities includes low level drivers for AURIXTM microcontrollers, Application Start-up Software and reference demo applications.