ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

Comparison of Embedded System and FPGA Application for Universal Multi-Resource Bus

Nitin Waghmare, Prof. T.B. Mohite-Patil

Abstract— **This paper presents a survey on embedded systems design and applications. Several platforms for embedded systems, including microcontrollers, microprocessors, field-programmable gate arrays, digital signal processors, Processors for Software Defined-Radio (SDR). A survey of embedded system-based industrial applications is presented. Study examples include FPGA Application for Universal Multi-Resource Bus**

Index Terms—**Embedded System**

I. INTRODUCTION

Embedded systems can be found everywhere in daily life, from electrical commodities and appliances, to nonlinear compensation mechanism, complex automation systems and adaptive control systems. Comparing with computer platform, embedded systems normally presents much less power of computation and very limited memory size. However, for solving particular real-time tasks, the disadvantages above turn around to be merits: embedded systems are less expensive and much easier to design. The simpler design is reflected in both hardware and software. Fixed design or only limited variations of hardware allow for use of simplified operating systems (OS) that allow for predictable, real-time operation, or even direct implementation of applications without any formal OS. This paper is organized as follows. In Section II, four major platforms for embedded systems implementation, including microcontrollers and microprocessors, field-programmable gate arrays (FPGAs), digital signal processors, Processors for Software Defined-Radio (SDR) introduced in details.

II. DESCRIPTION OF EMBEDDED SYSTEM PLATFORMS

The embedded systems are normally defined as the software implemented in hardware in order to realize specified real-time functionalities. The normally used soft-core processing hardware includes microcontrollers, microprocessors, FPGAs, digital signal processors (DSPs), and application-specific integrated circuits (ASICs), each of which has its own properties.

A. Microcontrollers and Microprocessors

For many years only microcontrollers and microprocessors were applied as the only efficient way to implement embedded systems, because of their programmable functionalities. The hardware architecture of microprocessors is fixed and generic within a given subclass which makes the platform low cost. They are capable of executing sequences of basic instructions that are typically stored in persistent read-only memory (ROM) or more recently in on-chip FLASH memory. Microcontrollers (MCUs) are typically manufactured with memory and some digital and analog peripherals integrated with a processor core on one chip. In order to reduce manufacturing cost and operating power, some microcontrollers are designed to use very short word length, such as four-bit words. They have very little random-access memory (RAM) and run clocked at a kilo Hertz range frequency. Also, those microcontrollers are able to retain partial functionality while the reminder of their circuit is suspended when waiting for an event or interrupt. On the other end, microcontrollers may operate using 32 or even 64-bit words, be clocked at a hundred Mega Hertz range and have sufficient computational power to perform functionality of a DSP. In almost every case, however, internal ROM and minimal amount of RAM, some programmable interval timers, digital input–output circuitry, and some forms of serial communication interface are integrated into the chip while external memory bus might be left out from their design. Microprocessors were developed as a single chip implementation of central processing units. Early embedded system utilized them. However, with development of more advanced and efficient manufacturing technology, the main applications of microprocessors remain in computing technology. In many embedded system applications, microprocessors are replaced by already discussed microcontrollers. Advanced microprocessor designs

include several CPUs (multicore) on one chip, RAM memory cache due to latency caused by use of RAM external to the microprocessor chip, and some hardware support to implement virtual memory addressing.

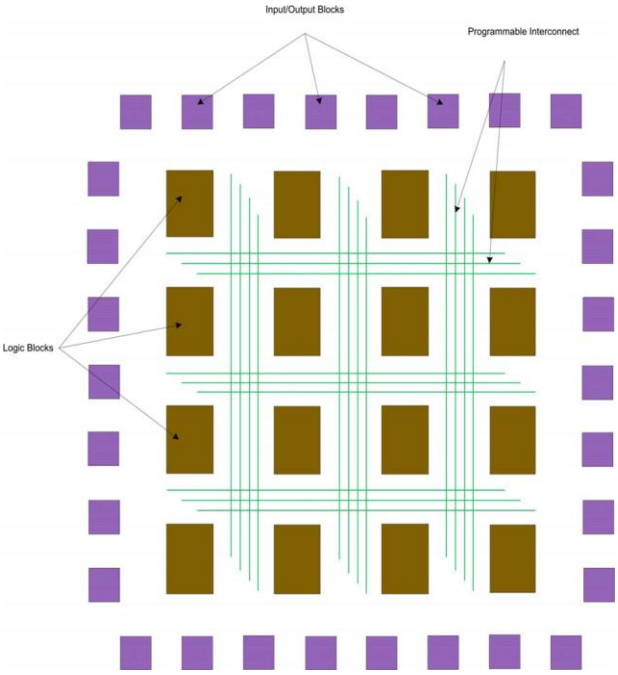

Figure 1. Architecture of FPGAs.

B. Field-Programmable Gate Arrays (FPGAS)

FPGAs were developed for digital embedded system based on the idea of using arrays of custom logic blocks (LBs) surrounded by a perimeter of I/O blocks (IOBs), all of which could be assembled arbitrarily (Fig. 1). FPGAs take the advantages of high operation speed, reconfiguration capability, very large number of components, and supported protocols. In embedded systems, FPGAs are used in two ways: either to implement the desired functionalities directly in the digital logic, or by implementing the architecture of a microprocessor—so called soft processor core, and desired microcontroller peripherals. The latter scenario became very popular in recent years as the FPGA prices reduced significantly, and could compete with microcontrollers. Use of FPGAs allows also for easy design of additional custom hardware accelerators that implement in hardware certain time consuming computations. Rodriguez- Andina *et al.* presented a thorough study of evolution of capabilities of FPGAs and design tools [1]. Monmasson *et al.*presented a very compact state-of-the-art tutorial demonstrating balancing use of hardware and software in design of a Kalman filter based AC driver controller [2]. In [3], Monmasson and Cirstea provided the overview of design techniques employed in design of various FPGA-based industrial controllers. Furthermore, capability to perform partial reprogramming on the fly of an FPGA leads to the practical implementation of an old idea of reconfigurable computing [4], [5].
FPGA technology is very valuable for one more reason. It makes possible to replace failed digital components in legacy systems [6]–[8]. Many industrial control systems were designed when expected lifetime of its components was about 25 years while presently the average design life cycle of used components is only about two years [9]. That makes it impossible to replace such failed components unless special steps are taken such as stock piling of them. Another, more costly and time consuming choice is to redesign and replace whole subsystems.

C. Digital Signal Processors (DSPs)

Digital signal processors (DSPs) are designed to have embedded multipliers and DSP blocks, which allow complex arithmetic operations to be performed, so they are easy for high-level programming implementation. When compared with microcontrollers, one of the primary advantages of DSP is availability of a single cycle multiply and accumulation operation. Also, DSPs have parallel processing capabilities and integrated memory blocks, which largely enhanced the processing speed. Some DSP architectures, called digital signal controllers (DSC), are optimized for control applications and contain control-oriented peripherals such as PWM generators, watchdog timers and fast response interrupts. However, DSPs require much higher cost comparing with FPGAs. Usually, DSPs are applied for image and audio signal processing when use of microcontrollers is not possible due to computational limitations. Their primary application type in industry is motor controller.

D. Processors for Software Defined-Radio (SDR)

Software-defined radio is an emerging technology that involves processing digital radio signals by means of software techniques. Today's mobile applications feature an abundance of radio standards with ever increasing bandwidth. This leads to a demand in both increased design productivity and

flexibility after device deployment. Software techniques are seen as a solution to quickly design flexible mobile applications. Software-defined radio was first motivated by moving from analogue to digital technology in radio systems. Because of the exponential increase in performance and productivity of digital technology compared to slow advances in analogue circuits, digital technology will continue to move closer to the antenna and replace much of the analogue front end. Figure 2 shows a receiver where the signal is digitised directly after being mixed to an intermediate frequency (IF). A digital radio however is not synonymous with a software-defined radio [11].

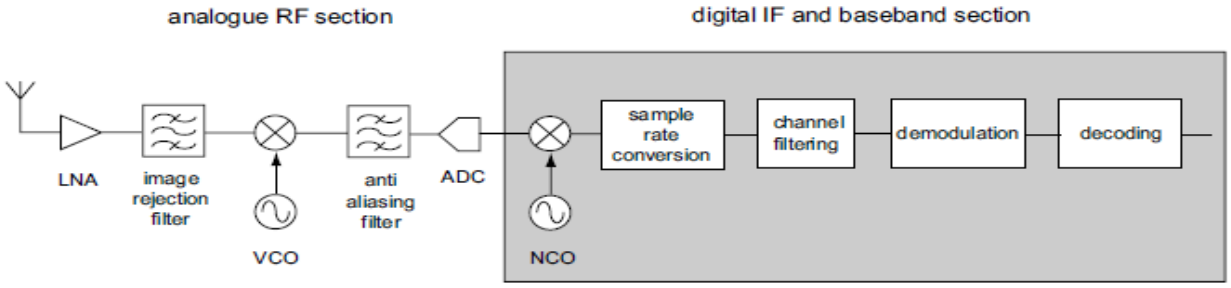

Figure 2. Receiver architecture of a software-defined radio

III. FPGA Application for Universal Multi-Resource Bus

In recent years FPGA-based prototyping has evolved into a widely used methodology that offers an at speed verification environment to dramatically accelerate the functional verification of complex SoCs, while simultaneously providing a pre-silicon platform for early software development. New generations of bigger and faster FPGAs have expanded the size of designs that can be successfully prototyped with both in-house and commercial prototyping systems. With the advent of Synopsys' HAPS-60 FPGA-based prototyping series, the technology gap compared to build-your-own systems as well as other commercial offerings has widened even further. In addition to increased size and capacity offered by new larger FPGAs, many users are now asking for additional and advanced use modes for their FPGA based prototypes to improve their overall design, verification, and software development productivity.

A. New Architecture Improvements

The most commonly requested enhancements involve increasing user accessibility to the system (including remote access), and ways to integrate the system with other verification tools and technologies. A broader list of productivity boosting enhancements is summarized here: Ease of use-Faster validation of the hardware prototype, Better design debugging, Easier prototype configuration.

Advanced use modes- Support of standard APIs, Support for RTL-based co-simulation and debugging, Support for accelerated transaction-based verification, Support for connecting to and co-simulating with virtual prototypes.

Rapid system initialization- Script-based configuration, SD-card design downloading to the FPGAs, Software-controlled memory pre- load and read-back.

Remote access and management of the FPGA-based prototype- Network/remote access to the prototype.

The UMR Bus architecture for the HAPS-60 series delivers programmability and flexibility without compromising "at-speed" system performance. It delivers the advanced verification tools and capabilities needed but missing from most build-your-own or commercial prototyping boards. The ultimate measure of success of the HAPS-60 series hinges upon delivering these capabilities at a price low enough to remain effective as an early software development system. In the next section, we take a closer look at the UMR Bus architecture and its implementation on a HAPS-60 system.

B. Communication System Architecture

The UMR Bus communication system is comprised of four key elements: The first is a programming

library of API (Application Programming Interface) calls and related software drivers for use on the host workstation. This communications software and API is provided to enable fast, transparent communication between software applications running on the host workstation and the DUT running on the HAPS system.

The second is a hardware interface module which provides a physical connection between the host workstation's PCI-Express bus and the UMR Bus on the HAPS system. Ethernet and USB connections are fast alternatives, but ultimately don't provide the low-latency performance required for co-simulation and transaction-based verification with HDL simulators.

The third element is the 8bit UMR Bus itself which has a 100MHz transfer rate enabling up to 100Mb/sec download and upload rates. The UMR Bus enables fast API access to the DUT on the HAPS system via IP (Intellectual Property) cores called CAPIM (Client Application Interface Modules).

CAPIM are the fourth and final element of the UMR Bus communications system. They are user instantiated IP cores, and are API accessible. They facilitate direct read and write access to the UMR Bus making all nodes, signals, and pins on the HAPS system accessible even if the signals are ultimately distributed across multiple FPGAs.

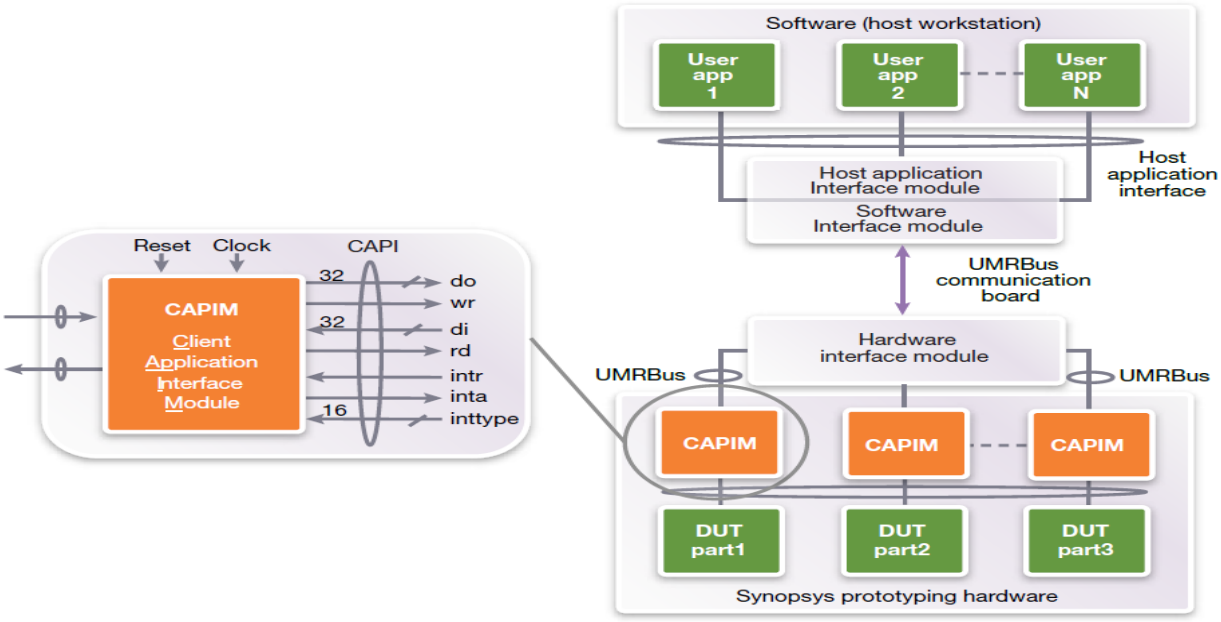

Figure 3. UMR Bus Architecture. The UMR Bus enables high speed, low latency communication between the DUT in the prototype and the host.

As is shown in Figure 3 above, the UMR Bus is the communications backbone of the system, providing connectivity between the host and the HAPS-60 FPGA-based prototyping system. Key features of the UMR Bus communication system are: Communication between the host workstation and the HAPS prototyping hardware, Pre-defined, API-accessible, CAPIM IP blocks, Up to 63, 32-bit channels (CAPIMS) for communication with the DUT, Interrupt support via hardware (DUT) and software (user's host application), Transfer of 16-bit interrupt messages from DUT to host application, Support for assertion messaging methods .The UMR Bus' wide, fast architecture is the key to providing users with direct (or remote) access to at-speed data, signals, and memory within the FPGA-based prototype. The next section highlights the new capabilities and related benefits enabled by the UMR Bus' unique architecture.

C . Multi-MHz Performance with Flexibility and Capability

The combination of the hardware interface module, the UMR Bus, and the client application interface

modules create a high-bandwidth, low-latency communication channel between the software on the host workstation and the DUT on the FPGA-based prototyping system. This high-speed connection is absolutely crucial for high bandwidth (many signals) and high frequency (many transfers) communication performance. Access via standard software APIs enable access and use by both the software developers and hardware designers on the project team, extending and expanding both the utility (features) and the productivity (access) of the FPGA based prototyping system. Figure 4 depicts some of the many capabilities enabled by the UMR Bus as it is implemented on the HAPS-60 system.

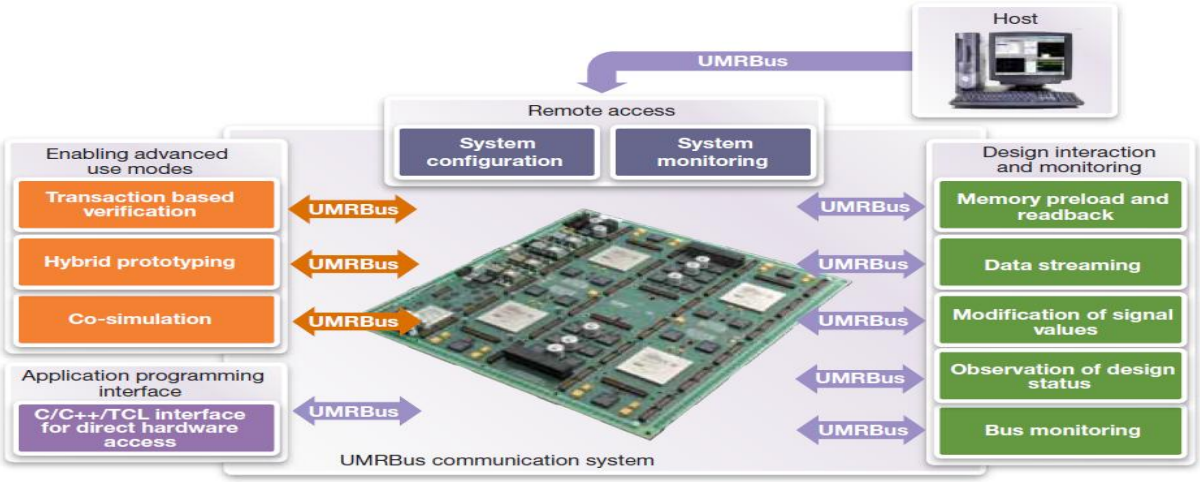

Figure 4: HAPS-60 UMR Bus Capabilities. New capabilities enable faster and deeper access within the prototype, boosting project team productivity.

Combined, the elements of the UMR Bus communications system provide users with an unprecedented range of features and capabilities, some of which are listed below: Remote access enables hardware and software team access to the prototype, Fast and flexible API.

Software-controllable remote configuration and management including: Remote design download, High speed configuration and re-configuration of the system with or without design download.

High performance memory access: High speed download and upload of memories, On-the-fly capture, reading, setting and re-setting of memories.

Faster and more powerful debugging and control: RTL debugging with Identify, Monitor design and bus status with the UMR Bus via workstation, Local or remote debugging, Debug within RTL; set triggers, breakpoints and sample nodes in RTL, View results in RTL source and via waveform viewer.

High-performance data streaming: Quick regression tests, Chip tester pattern validation with error injection.

D. High Performance Data Streaming

With a high-performance host workstation-to-prototype link, the capabilities and use modes for the prototype are greatly extended. High speed data streaming between the host workstation and the prototype is easily performed using the UMR Bus. The example in figure 4 below, highlights sending an encoded video stream residing on the host workstation directly to an H.264 decoder (the DUT) running on the HAPS system. To accomplish this task, the user simply instantiates two elements on the prototyping system: a CAPIM and a UMR2AHB component (The UMR2AHB is pre-defined UMR Bus IP element which enables direct access via UMR Bus to the AHB master). The encoded video stream can now be streamed directly from the host workstation via high-speed to the H.264 decoder DUT.

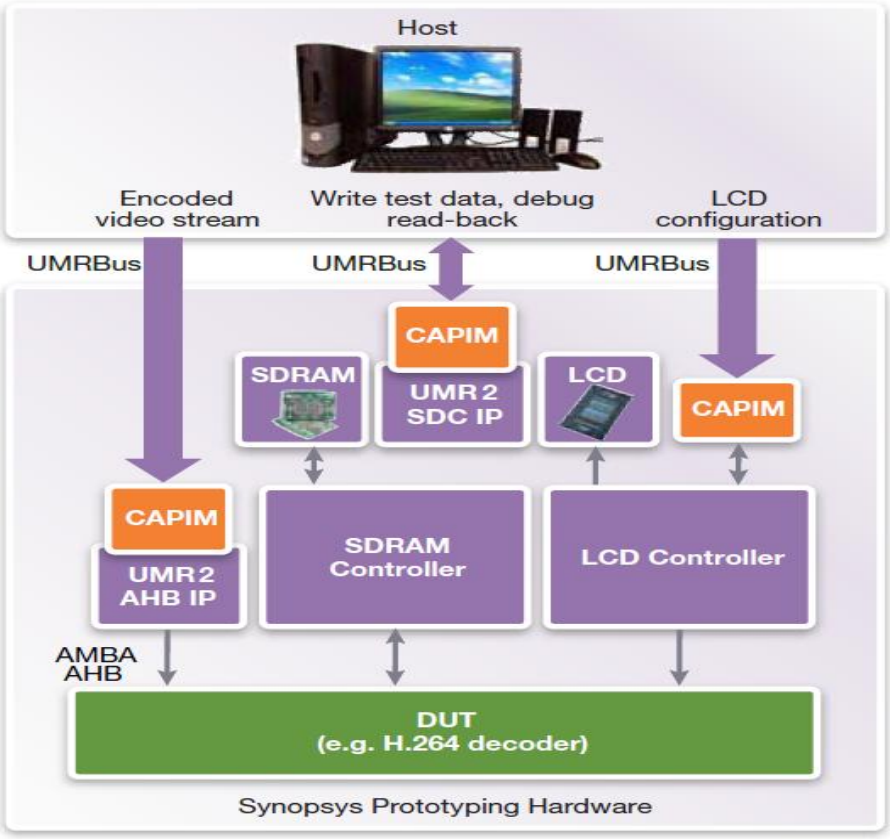

Figure 5: High performance data streaming using the HAPS-60 system

IV. CONCLUSION

This paper introduced the current development of embedded systems. Various implementation platforms, including microcontrollers, microprocessors, FPGAs, DSPs, and ASICs, are discussed and compared. Three real applications of embedded systems are presented in order to illustrate issues that need to be considered and problems that need to be solved when designing embedded systems. As sufficient work presented in the literature research above, it can be concluded that embedded systems have already maturely developed and can be designed to solve various very complex problems in industrial applications.

V. REFRENCES

[1] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," *IEEE Trans. Ind Electron.*, vol. 54, no. 4, pp. 1810–1823, Aug. 2007.

[2] E. Monmasson, L. Idkhajine, I. Bahri, M.-W. Naouar, and L. Charaabi, "Design methodology and FPGA-based controllers for power electronics and drive applications," in *Proc. 5th IEEE Conf. Ind. Electron.Appl. (ICIEA)*, Taichung, Jun. 15–17, 2010, pp. 2328–2338.

[3] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.

[4] G. Estrin, "Reconfigurable computer origins: The UCLA fixed-plusvariable structure computer," *IEEE Ann. Hist. Comput.*, 2002.

[5] G. Estrin, "Organization of computer systems—The fixed plus variable structure computer," in *Proc. Western Joint Computer Conf., Western Joint Comput. Conf.*, New York, 1960, pp. 33–40.

[6] L. Anghel, R. Velazco, S. Saleh, S. Deswaertes, and A. El Moucary, "Preliminary validation of an approach dealing with processor obsolescence," in *Prof. 18th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, 2003, p. 493.

[7] J . J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, " Features, design tools, and application domains of FPGAs," *IEEE Trans. Ind Electron.*, vol. 54, no. 4, pp. 1810–1823, Aug. 2007.

[8] B. W. Bomar, "Implementation of microprogrammed control in FPGAs," *IEEE Trans. Ind. Electron.*, vol. 49,

no. 2, pp. 415–422, Apr. 2002.
[9] P. Sandborn, "Trapped on technology's trailing edge," *IEEE Spectrum*, vol. 45, no. 5, pp. 42–45, Apr. 2008.
[11] Tobias Becker, Wayne Luk, and Peter Y.K. Cheung, "Parametric Design for Reconfigurable Software-Defined Radio"

Nitin Waghmare B.E. in Electronics from Shivaji University in 2005. Working as lecturer in Gourishankar Polytechnic Limb Satara, Maharashtra. Pursuing M.E. in Electronics and Telecommunication from DYPCET Kolhapur.

[10] P. P. Fasang, "Prototyping for industrial applications," *Ind. Electron. Mag.*, vol. 3, no. 1, pp. 4–7, Mar. 2009.

Prof. T. B. Mohite-Patil working as Professor in DYPCET Electronics & Telecommunication Department Kolhapur. B.E. from Electronics Department of DYPCET and M.E. in Control System from Walachand College of Engineering Sangali in 1990 and 2001 respectively. Presently pursuing Ph. D. from Shivaji University.