Preface

Overview

The realm of embedded systems is quite large and is predominantly carried out around the general purpose processor and microcontrollers. The use of field programmable gate array (FPGA) in microprocessor-based embedded systems is often for glue logic or for off-loading the processor from tasks that require fast updates. The motivation for writing this text is to present a single source of information that can be used to understand how a FPGA and the Hardware Description Language (HDL) can be used in the design of embedded digital systems.

Digital design methodology has undergone several changes over the past three decades. The use of FPGA and HDL for implementing digital logic has become widespread in the last decade. The use of FPGA in embedded systems is still in its nascent stage. The majority of the embedded applications are divided between an 8-bit microcontroller implementation and a 32-bit processor-based real time operating system (RTOS) implementation. This text provides a starting point for the design of embedded system using FPGA and HDL. To give the text a common thread of thought from the application point of view, a design example of a hypothetical industrial robot controller is taken up. Different chapters of the text provide the necessary background on FPGA and HDL along with its use in designing an industrial robot controller.

Coverage

The first FPGA, introduced in 1985, consisted of 2000 gates. Since then, gate density has grown to tens of millions of gates. With increasing density of FPGAs, varied hardware resources have become a standard feature of contemporary FPGA-based devices. The text includes simulation of digital logic using Verilog HDL, synthesis of HDL code for a given FPGA device and processor-based FPGA devices. The focus of the HDL chapter is to emphasise the synthesizable area of Verilog constructs and to provide a basis to understand the application examples that follow in subsequent chapters. A chapter is devoted to the understanding of hardware–software partitioning in a FPGA device. Proprietary 8-bit and a 32-bit soft processors are discussed along with interfacing methodology using system-on-

chip interconnections. Basic technique for serial data communication, signal conditioning, motor control and hardware prototyping is covered using FPGA and HDL.

How to Use This Book

Moore's law has kept the semiconductor business in a constant state of flux. It is very difficult to write a book that uses FPGA and continues to be relevant despite ongoing technological changes. The author has presented basic concepts and techniques for using FPGA and hence should not change quickly. Since this book covers vast areas of HDL and FPGAs, some sections are brief and sketchy. For this the author recommends that the reader supplement the contents of each chapter with additional available literature. The chapter on HDL coding and simulation should be supplemented by standard textbooks on HDL coding and simulation. The FPGA resources and synthesis topic should be supplemented by EDA tools provided by different FPGA vendors and FPGA device datasheets. The contents on FPGA embedded processors can be supplemented by application notes on interfacing processors to custom codes and datasheets of soft processors.

FPGA Device and Tools Used

For purposes of illustration and consistency, Xilinx ISE™ software and SPARTAN™3E FPGA have been used throughout the book. Though the exemplars are specific to this device, the concepts can be applied to FPGA devices available from other FPGA vendors.

Gandhinagar *Rahul Dubey*
October 2008

1

Introduction

Digital systems and their design have evolved greatly over the last four decades. Rising densities and speed have provided designers a huge canvas to create complex digital systems. Present-day embedded systems use single-chip microcontrollers. Contemporary microcontrollers are available with 8-, 16- and 32-bit processing capability along with a peripheral set containing ADC, timer/counter and networks (I^2C, CAN, SPI, and UART). For most applications the microcontroller-based board is adequate. For applications where there is a need to integrate custom logic for faster control and additional peripherals, the microcontroller or microprocessor board is augmented by a FPGA or an application specific standard product (ASSP) device. The focus of this chapter is to understand different digital design methodologies before embarking on a full fledged description of the use of a custom digital design based on a FPGA.

1.1 Embedded System Overview

Embedded systems are usually single function applications. Various functional constraints associated with embedded systems are low cost, single-to-fewer components, low power, provide real-time response and support of hardware-software co-existence. A general methodology used in designing an embedded system is shown in Table 1.1.

The decision on the kind of digital platform to be used takes place during the system architecture phase as each embedded application is linked with its unique operational constraints. Some of the constraints of a digital controller of embedded system hardware include (in no particular order) the following:

- Real-time update rate
- Power
- Cost
- Single chip solution
- Ease of programming
- Portability of code

- Libraries of re-usable code
- Programming tools.

Table 1.1. Embedded system design flow [1]

Design phase	Design phase details
Requirements	Functional requirements and non-functional requirements (size, weight, power consumption and cost)
User specifications	User interface details along with operations needed to satisfy user request
Architecture	Hardware components (processor, peripherals, programmable logic and ASSPs), software components (major programs and their operations)
Component design	Pre-designed components, modified components and new components
System integration (hardware and software)	Verification scheme to uncover bugs quickly

1.2 Hypothetical Robot Control System

For understanding different digital design platforms, this text uses the design of a digital controller for a robot as a case study. The robot is a hypothetical, vertically articulated robot system for an automated assembly line. The process of designing this controller will help in understanding various digital design concepts. Figure 1.1 shows the various components of an assembly line robot. Each robot consists of five electric motors that work as actuators for different joints of the robot. A programming pendant or workstation is used to program the movements of the robot along with a communications network to link this robot to other robots on the assembly line. Various sensors are interfaced to the robot control system.

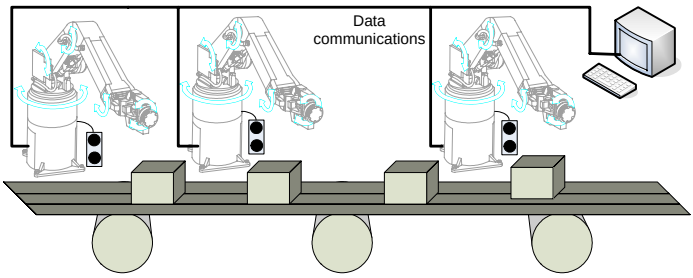


Fig. 1.1. Vertically articulated robot system used in an assembly line environment

The typical requirements of an Industrial robot controller include

- Control method for point-to-point control using servomotors
- Position detection using incremental or absolute encoder system
- Return to origin using limit switches and encoder
- Trajectory control
- Programming using a personal computer.

Table 1.2. Tasks for robot digital controller

Task	**Subtask**	**Update time**
Control of joint motors	Gate Driver, protection and current sensing	Fraction of a microsecond
	Dead time	Microseconds
	Closed-loop torque control	Tens of microseconds
	Closed-loop speed control	Hundreds of microseconds
	Position coordinate interpolation	Milliseconds
	Host communications	Tens of milliseconds
Sensor signal processing	ADC, DAC	Tens of milliseconds
Networking applications	Low-speed network	Milliseconds

Control Strategy for the Robot Controller

For implementing the robot controller on a digital system, a list of controller tasks is created in Table 1.2 along with the update time requirements. The major tasks for the robot controller for an articulated factory robot are

- Simultaneous control of five motors with details shown in Table 1.3.
- Signal processing of sensor inputs coming from robot environment — encoders, limit switches, proximity sensors, vision sensor
- Communication of robot co-ordinates to other robots in the vicinity, using CAN bus or Modbus®
- Communicating with host controller over serial port
- Computation of trajectory for robot movement.

Table 1.3. Specifications of a micro articulated robot Mitsubishi Movemaster RV-M1 [2]

Axis	Description	Encoder pulses per revolution (PPR)[1]	Gear ratio	Working range in degrees
J1	Waist	200	1:100	300°
J2	Shoulder	200	1:170	130°
J3	Elbow	200	1:110	110°
J4	Wrist pitch	96	1:180	90°
J5	Wrist roll	96	1:110	± 180°

The tasks and their update times are shown graphically in Fig. 1.2.

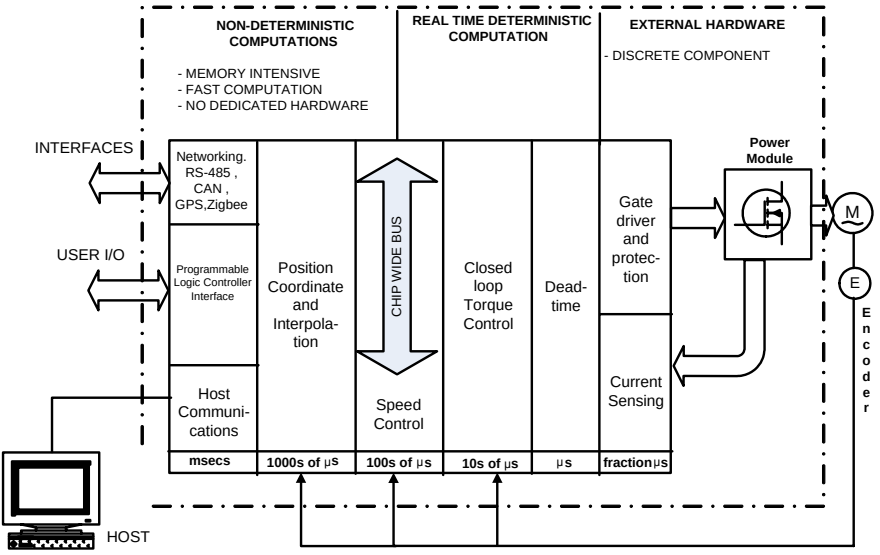


Fig. 1.2. Update times needed for various control functions of a robot control system [3]

1.3 Digital Design Platforms

Till the 1970s, electronic system designs were based on discrete analogue components such as transistors, operational amplifiers, resistors, capacitors and inductors. These circuits offered concurrent processing but had problems of parameter drift with temperature and ageing. The coming of TTL-based

[1] The encoder is used to find the position and speed of the robot joint. The working of the encoder is explained in Chap. 2.

components laid the foundation of digital design. The Intel 4004 microprocessor became the first digital platform which was configurable using software. Table 1.4 lists the major contemporary digital designs along with their relative merit.

Table 1.4. Digital design platforms

Digital design platform	Merit
Microprocessors	Reconfigurable using software. Good for computations
Microcontrollers, digital signal controllers	Combination of peripherals and CPU
Application specific standard product (ASSP)	A specialized peripheral with the ability to communicate with a host processor
Field programmable gate array (FPGA)	Ability to combine the strengths of processor, controller and ASSP

1.3.1 Microprocessor-based Design

The microprocessor has changed digital design methodology like no other digital component. It started out as a 4[2] bit programmable CPU in 1971 and still continues to be the digital controller of choice across several application areas. The microprocessor brought the concept of instruction set architecture (ISA), assembler and compiler. There are many real-time applications, with fast update rates require programming the microprocessor in its native assembly language. This is usually done when the size of available memory is a constraint. Even though most commercial microprocessors used today cater to data-centric applications, there are microprocessor cores embedded in microcontrollers for real-time control applications.

Digital control systems, like the robot application use a processor by using interrupts for real-time processing. There are interrupts for calculation of robot arm trajectory, encoder and sensor feedback, control of motors and networks. Each interrupt will occur based on the update time requirement of the given task. Figure 1.3 shows the generic nature of interrupt processing, where an interrupting device seeks CPU attention. A microprocessor-based robot controller carries out the task of arm positioning based on the flowchart shown in Fig. 1.4.

[2] The early Intel 4004 and the 8086 processor had close to 2300 and 29000 transistors. A basic 2 input NAND gate consists of 4 transistors. Effectively the early Intel processors 4004 and 8086 used only 575 and 7250 gates. This helps to put in perspective the amount of digital logic that can be accomodated in a 500,000 gate FPGA.

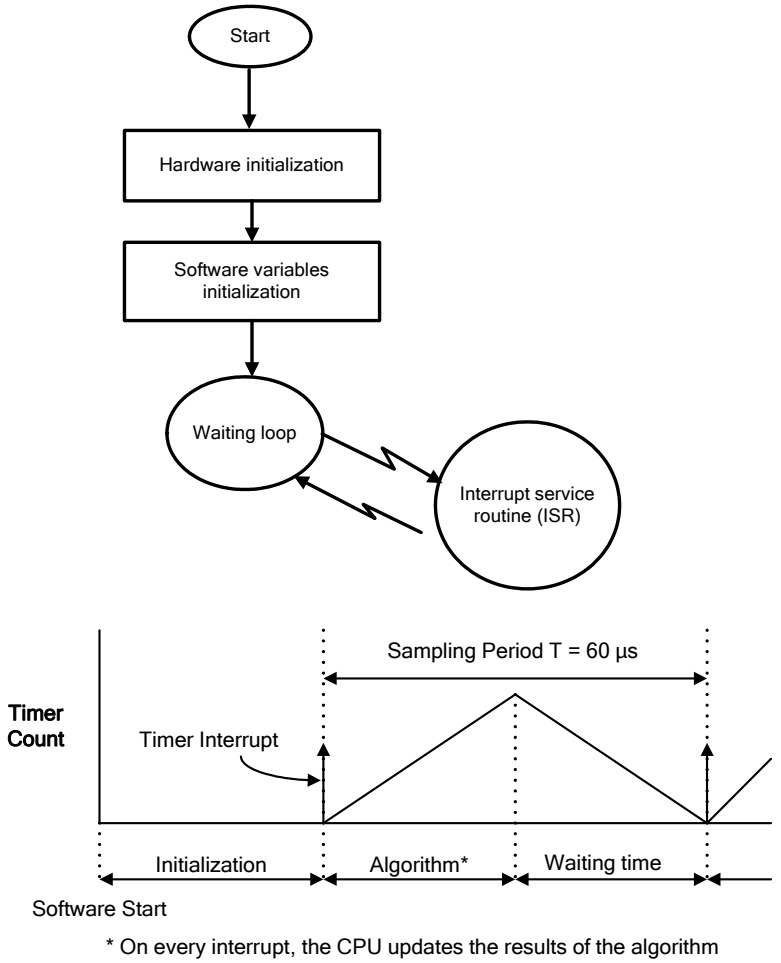


Fig. 1.3. Interrupt service routine (ISR) based processing scheme of processor-controller control scheme

Because most single core general purpose processors (GPP) are single-threaded (can process one instruction at a time), the processor use can become very high when managing multiple interrupts from different tasks of the robot controller. This can be seen from Fig. 1.5, where processor CPU use increases linearly with each motor.

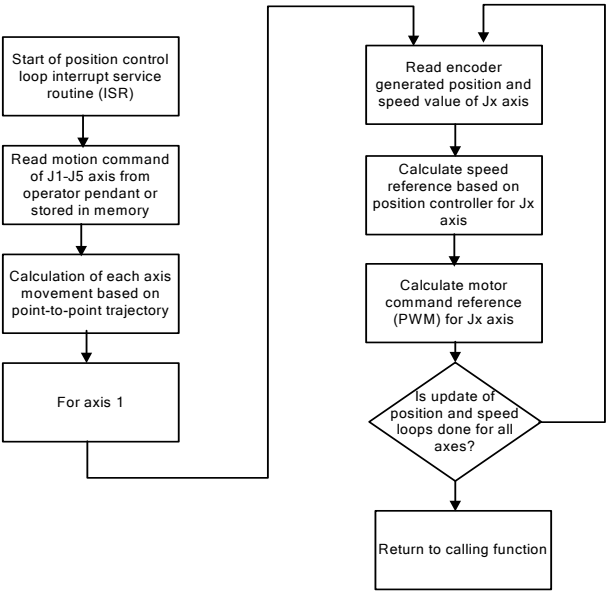


Fig. 1.4. Processor-interrupt-based flowchart needed for computing a control action [4]

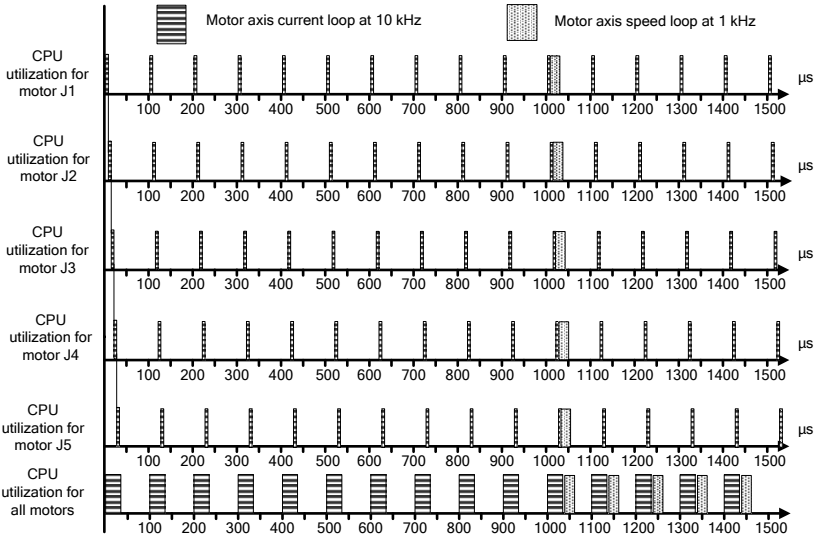


Fig. 1.5. CPU use for axis motor control for a single-threaded controller

1.3.2 Single-chip Computer/Microcontroller-based Design

The microcontroller represents the next generation of controllers for embedded systems. It allows creating systems with fewer numbers of components by

incorporating peripherals that were earlier externally interfaced with the general purpose processor. A block diagram of a typical single-chip controller, which is used as a robot motor controller, is shown in Fig. 1.6.

Like the microprocessor, tasks in a microcontroller design environment are divided as per the update rates required. For tasks requiring low update rates, coding is accomplished using a software programming language such as C. Tasks that need to have high deterministic update rates are coded using the native assembly language for a particular microcontroller. In the robot application at hand, many of the motor control routines require update rates of a few kilohertz. Traditionally, these routines are written in assembly language. It is difficult to port routines written in assembly language as they are tied to a CPU's ISA. The other constraint with a microcontroller-based system is the fixed number of available peripherals. Though microcontroller vendors offer a wide range of devices with different numbers and types of peripherals, it is not always possible to find one that matches the application requirements perfectly.

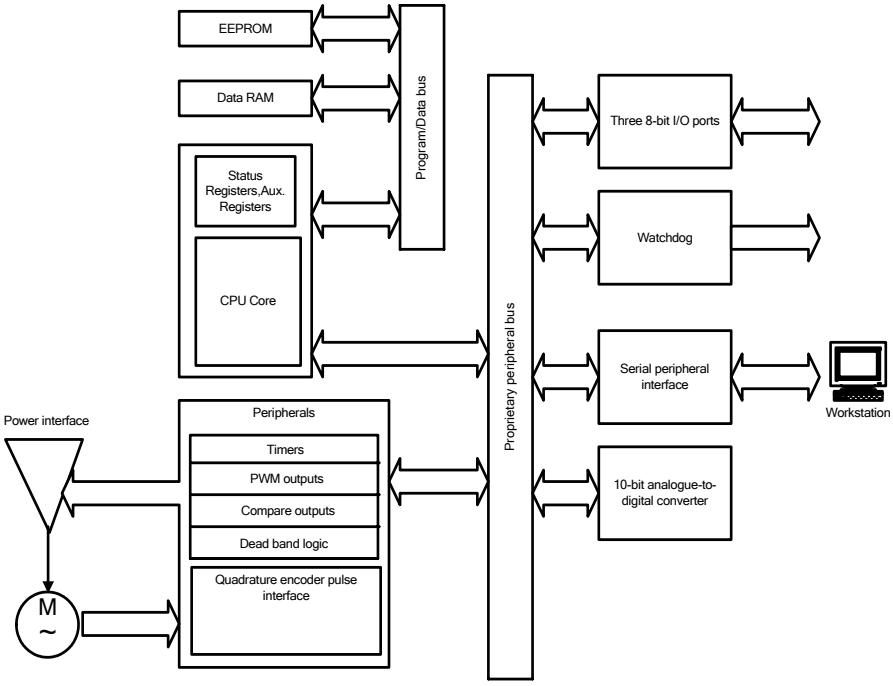
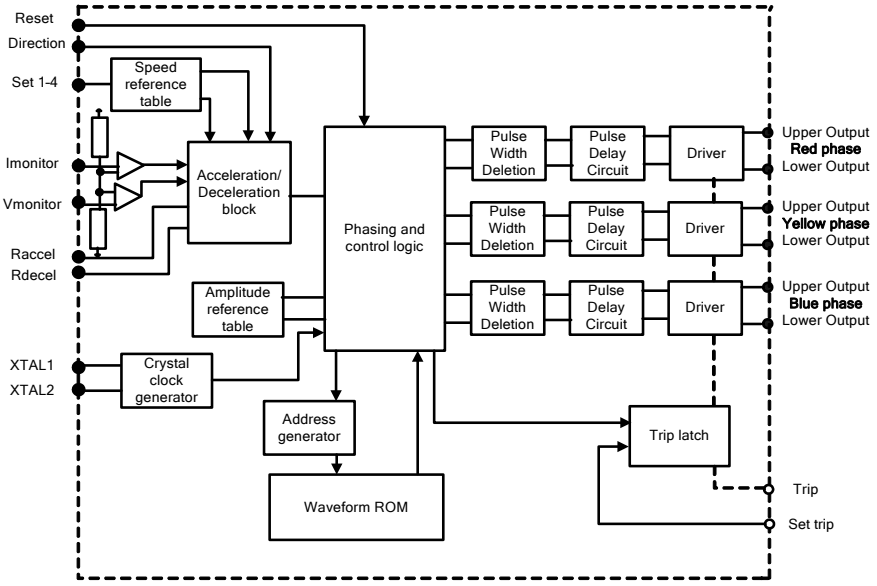


Fig. 1.6. Single-chip microcontroller environment for a motor control application

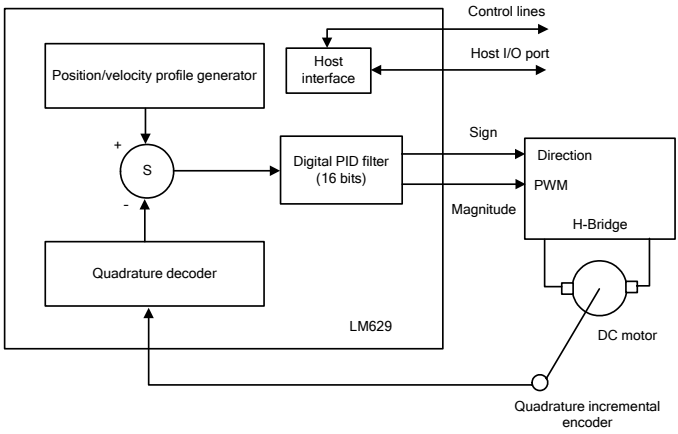
1.3.3 Application Specific Standard Products (ASSPs)

An ASSP is a configurable logic component for a specific application. The functionality of an ASSP is tweaked by specifying its control word. ASSPs are made in volumes and cater to the generic requirements of the application. Most of

the time, ASSP-based designs are used on a PCB. In the robot control application at hand, an ASSP can be used for controlling the motor for each axis of the robot. Based on the type of motor and control strategy used, a corresponding ASSP is chosen. Two examples of ASSPs for motor control include LM629 from National Semiconductor for control of a brushed DC motor and SA628 (see Fig. 1.7a and b) for three-phase motor control. Configurable ASSPs provide address, data and control bus connectivity for interfacing with the host processor.



a



b

Fig. 1.7. a ASSP chip SA628 for control of a three-phase AC Induction Motor [5]; **b** ASSP chip LM629 for control of a DC motor

1.3.4 Design Using FPGA

The present-day FPGA provides a platform that supports both processor and custom logic requirements. The microcontrollers currently have an edge over the FPGA in terms of power and cost. But FPGAs are catching up by offering portability of code across various FPGA vendors, libraries of re-usable code and availability of low-cost programming tools. Programmable devices that were traditionally low gate count devices are now in a position to support large parts of digital system logic. The digital designer today has a viable option of using only the FPGA device as the embedded system controller. The availability of high-density, low-cost FPGA devices has given digital designers lots of flexibility to design custom digital architectures using FPGA and HDLs. FPGA devices have evolved from their glue logic predecessor to a device that now contains a large variety of built-in digital components (memory, multipliers, transceivers and many more). FPGA device density has risen over the years and at the same time its cost has made it economically viable for use in several applications. Contemporary FPGAs contain thousands of look up tables (LUTs) and FFs for implementing complex digital logic.

Contemporary FPGAs offer

- Reconfigurability: Field programmable devices can be reconfigured at any time. Designers can integrate modifications or do complete personality changes.
- Software-defined design: The hardware is defined by software-like languages (HDL). Designers can develop, simulate and test a circuit fully before "running" it on a field programmable device.
- Parallelism: Circuits defined in an FPGA can be designed in a completely parallel fashion. This is similar to using multi-path analogue circuits. A user can instantiate multiple hardware implementations on the same chip without cross-module interference or computation loading. An example of FPGA-based concurrent processing is shown in Fig. 1.8.

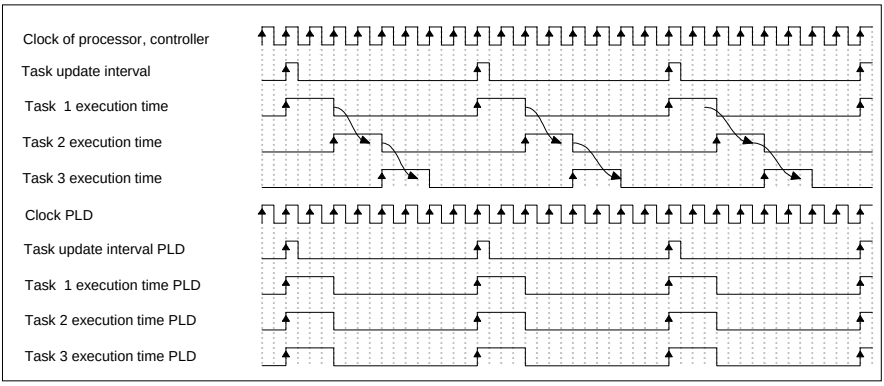


Fig. 1.8. Multi-tasking scheme using a GPP vis-à-vis a FPGA

- High speed: Because an FPGA is a hardware implementation running with fast clock rates, designers can achieve very high speeds. Coupled with parallelism, FPGA implementation can outperform processor-based systems.
- Reliability: Designers can expect true hardware reliability from FPGAs because there is no operating system or driver layer[3] that can affect system uptime.
- IP protection and re-use: Once compiled and downloaded to a FPGA, hardware implementation is difficult to reverse engineer. A tested hardware design can be re-used multiple times by instantiating.

FPGA-based systems are gaining acceptance because these systems integrate digital logic design, processors and communication interface on a single chip. The front end design flow of a FPGA is very similar to that of a custom logic design. Almost all FPGA vendors offer a suite of software tools that allows a designer to simulate, synthesize, place and route and program the FPGA. Table 1.5 shows the different design tools offered by two leading vendors. Once a designer feels comfortable in a particular design suite, it is easy to migrate to another vendor's design tools because they work in a similar fashion[4].

Table 1.5. Common design tools provided by two leading FPGA Vendors

Functionality	XILINX	ALTERA
Design synthesis, mapping, place and route	Integrated Software Environment (ISE)™	Quartus II®
FPGA embedded processor design tool	Embedded Design Kit (EDK)®	System on Programmable Chip (SoPC) builder®
Custom peripheral support	Yes	Yes
On-Chip signal logic analyzer	ChipScope™ Pro	SignalTap®
MATLAB® co-simulation and IP cores library	System Generator™	DSP Builder®

[3] Not applicable to FPGA-based processor systems.

[4] One of the strengths of HDL and associated synthesis software is to make the implementation option wider for the designer. For consistency, this book uses a contemporary Xilinx SPARTAN-3E™ 500K gate FPGA along with the Xilinx ISE™ for illustrating various examples. The author feels strongly that if the designer is able to master one vendor's specific design flow along with a given FPGA architecture, the same concepts can be applied to understand quickly and implement a digital design using FPGAs from other vendors.

From an implementation point of view, a robot controller using a FPGA device can be considered a viable alternative[5], as robots are usually low-volume application-specific systems. The FPGA allows for customization of servo-motor type for joint control, industrial communciation network, integration of custom peripherals and control algorithms.

Software-based design flows are suited for applications which are data centric and hardware design flow is suited for fast real-time applications.Table 1.6 provides a transition path for migrating from microprocessor/controller to FPGA-based design. The FPGA design process consists of design entry, which is accomplished by using either schematic or HDL. Following the design phase, digital logic is synthesized, mapped and placed on a FPGA[6].

Table 1.6. Transition path from a microcontroller-based system to a FPGA system

Existing microprocessor/microcontroller code	**Field programmable device**
Target independent 'C' Code	Embedded processor within the FPGA device
Target dependent assembly constructs for routines requiring fast update rates	Target independent HDL-based coding for routines requiring very fast update rates

1.4 Organization of the Book

The book is organized to weave together concepts, tools and techniques to help in designing FPGA-based embedded systems. This book does assume that the reader is versed in the basic concepts of embedded systems programming and interfaces. There are references at the end of each chapter where the reader can get more information on the topics covered in the chapter. This text is trying to put together many components of a system, so certain sections are not covered in detail but are used to convey the concept of system design.

The sequence of chapters is to introduce basic concepts and then build upon them. Table 1.7 details the contribution of each chapter in building up a FPGA-based digital system.

[5] The purpose of this text is to explain embedded hardware design using FPGA. It is not the intention of this text to prove that FPGA-based robot controller is the best digital platform for implementing the robot controller.

[6] The HDL design process is described in Chap. 2. The complete design flow of synthesis, mapping, place and route is described in Chap. 3.

Table 1.7. Preview of FPGA-based digital design implementation

FPGA design	**Chapter**						
	1	2	3	4	5	6	7
The case for using FPGAs	■						
Hardware description language (HDL)		■					
Synthesis of HDL design using FPGA as a target device			■				
FPGA embedded processors				■			
Serial communications and interfacing					■		
Motor control						■	
Prototyping using FPGA							■

Broadly, Chaps. 1 to 4 of the book introduce the technology and tools for implementing digital logic using a FPGA device. Chapters 5 to 7 discuss interfacing, motor control and prototyping using FPGA.

As shown in Fig. 1.9, different aspects of robot controller design are covered in chapter numbers mentioned in each component.

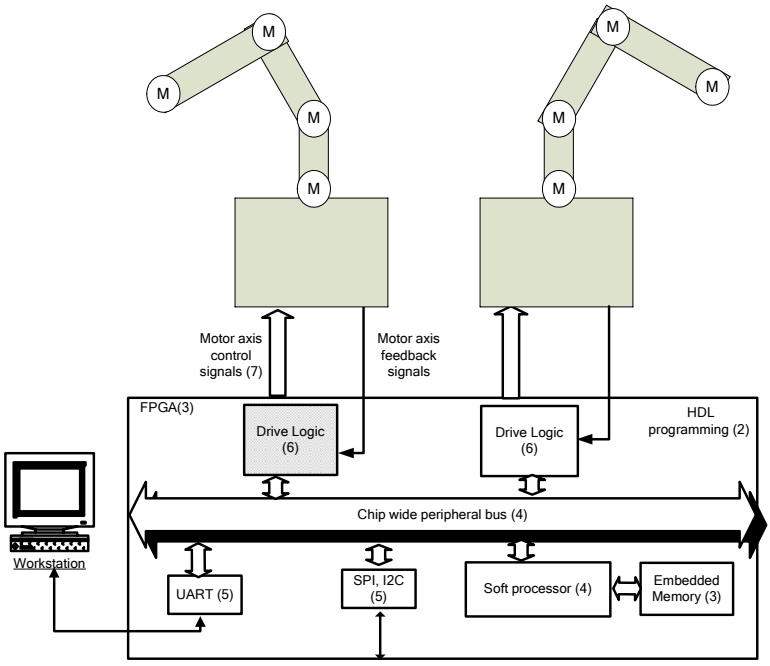


Fig. 1.9. Contribution of each chapter (shown in parentheses) for creating a robot controller

The second chapter is on simulation of digital systems using Verilog as the hardware description language (HDL). It introduces basic concepts of how a printed circuit board (PCB) containing digital components can be modelled using HDL and how it can be tested using software simulators. A simulation environment of an EDA tool is also explained.

Chapter 3 of the book introduces the architecture and resources of FPGA. Each building block of the programmable device such as embedded memory, phase-locked loops, logic blocks, multipliers and different interfacing I/O standards are explained along with their HDL based instantiation template. The chapter ends with examples of digital systems and their FPGA-based synthesis results.

FPGA-based embedded processors have made it possible to migrate from microcontroller-based embedded system design to FPGA-based embedded system design. FPGA-based designs give the designer an option to retain much of the skill set of high-level software programming. Now instead of coding in a native assembly language for a particular processor — deterministic tasks can be coded in HDL. Chapter 4 provides methodology on bringing together the software and the hardware worlds. FPGA immersed processors along with different interfacing buses connect to external standard and custom peripherals. A system-on-chip is created using this approach.

Chapter 5 discusses FPGA-based interfaces. It covers basic data communication using HDL and FPGA and protocols. The chapter also discusses asynchronous and synchronous serial data communications. The second section of the chapter discusses basic signal conditioning of the acquired signal.

The actuator is the last component of the control loop. In the robot example used in this book, the electric motor is the actuator for various joints of the robot. Chapter 6 discusses digital design and control implementation of different motors — stepper, permanent magnet DC motor, brushless DC motor, permanent magnet synchronous motor (PMSM) and permanent magnet reluctance motor.

The last chapter of the text is on prototyping the different schemes discussed using a FPGA-based board. It discusses various hardware verification and interfacing techniques, which are useful for hardware system integration.

Problems

1. Give an example of a application suited for a microcontroller and for a FPGA. Justify why one cannot replace the other.
2. What are the limitations of a FPGA-based system vis-à-vis a custom ASIC-based system.
3. How is real-time processing done on a GPP or a microcontroller based system by using interrupts?
4. What kind of power constraints are part of an articulated factory robot and that of a robotic rover shown in Fig. 1.10?
5. The robotic rover application (shown in Fig. 1.10) involves travel along terrains either by use of a remote link such as the Global Positioning System (GPS). The rover collects information about its surroundings using

sensors and relays it to a base station or operator console. A list of tasks for this rover includes

a. Power management for the rover
b. Control of six motors
c. Signal processing of sensor inputs coming from the robotic environment using a vision sensor.
d. Determining the robot position using GPS
e. Communicating with the host controller using ZigBee
f. Ability to interface with various payloads — new sensors, new actuators.

Partition the tasks as per their update time requirements and comment on the suitability of putting the task on a FPGA or a GPP.

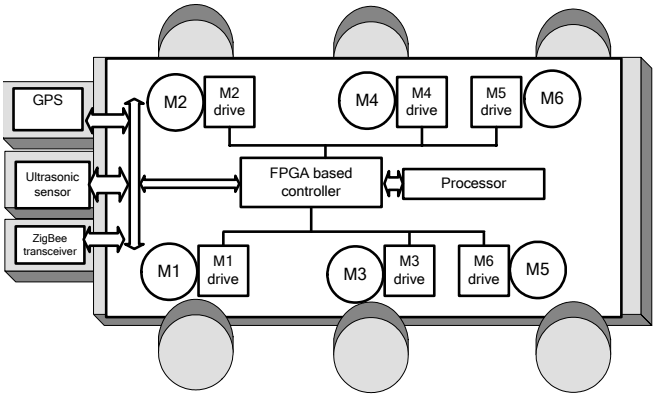


Fig. 1.10. Diagram of a robotic rover

References

1. Wolf W (2005) Computers as Components: Principles of Embedded Computer Systems Design. Morgan Kaufmann, San Francisco
2. Kung YS, Shu GS (2005) Development of a FPGA-based motion control IC for robot arm. Paper presented at IEEE Industrial conference on Industrial Technology (ICIT 2005), City University of Hong Kong, Hong Kong, December 2005
3. Goetz J, Takahashi TT (2003) A design platform optimized for inner loop motor control. Paper presented at power conversion and intelligent motion (PCIM 2003) conference. http://www.irf.com/technical-info/whitepaper/pcimeur03innerloop.pdf. Accessed 15 October 2008.
4. Kung YS, Shu GS (2005) Design and implementation of a control IC for vertical articulated robot arm using SOPC technology. Paper presented at IEEE Mechatronics ICM 2005, pp. 532–536
5. Mallinson N (1998) Plug and play single chip controllers for variable speed induction motor drives in white goods and HVAC systems. Paper presented at IEEE applied power electronics conference APEC 1998, 2:756–762

Further Reading

1. Maxfield C (2004) The design warrior's guide to FPGAs – devices, tools and flow. Newnes
2. Vahid F, Givargis T (2002) Embedded system design – A unified hardware/software introduction. John Wiley
3. Keramas JG (1999) Robot technology fundamentals. Thomson Delmar
4. Klafter RD et al (1989) Robotic engineering, an integrated approach. Prentice-Hall
5. Balch M (2003) Complete digital design, a comprehensive guide to digital electronics and computer architecture. McGraw Hill
6. Slater M (1989) Microprocessor-Based Design, A Comprehensive Guide to Hardware Design. Prentice-Hall
7. Monmasson E, Chapuis Y (2002) Contributions of FPGAs to the Control of Electrical Systems, a Review. IEEE Industrial Electronics Society Newsletter, 49(4)
8. Newman KE, Hamblen JO, Hall TS (2002) An Introductory Digital Design Course Using a Low-Cost Autonomous Robot. IEEE transactions on Education, 45(3):289–296
9. Kung YS et al (2006) FPGA-Implementation of Inverse Kinematics and Servo Controller for Robot Manipulator. Paper presented at IEEE Robotics and Biomimetics, (ROBIO 2006) at Kunming China, December 2006
10. Navabi Z (2007) Embedded Core Design with FPGAs. McGraw Hill
11. Navabi Z (2004) Digital Design and Implementation with Field Programmable Devices. Springer
12. Navabi Z (1999) Verilog Digital System Design. McGraw Hill