Virtex-5 Configuration Guide

1. page 11
2. page 15 (especially sections on Master/slave & JTAG at bottom of page)
3. page 16-17
4. page 18-19
5. page 23 (will only comment on a few of these steps.)
6. page 26
7. page 28 (read paragraph below fig 1-8 caption)
8. page 29
9. page 30
10. page 33 (to prevent reverse engineering a design)
11. page 34 (note JTAG usage)
12. page 37
13. page 62 (point out SPI differs from Master serial, which uses Platform flash. Compare with figure 2-3; the interface is different.).
14. figure 2-3 (which is on page 39).
15. page 67
16. page 77
17. page 89-91
18. Page 93 (discusses partial reconfig)
19. page 135

1. Introduction to Embedded Systems

An Embedded system is any electronic device that incorporates a computer in its implementation. The user of an embedded device is often not even aware that a computer is present in the device. The computer is used primarily to provide flexibility and to simplify the system design. Unlike a PC, program code is usually stored in ROM and not a hard disk drive. Typically, the end user does not develop new software for the embedded device. With advances in VLSI technology, embedded systems have become so inexpensive that they are found in most of today's electronic devices.

1.1 Examples of Embedded Systems

A robot such as the Mars rover seen in Figure 1.1 is an embedded system. A cell phone, PDA, or portable multimedia player as seen in Figure 1.2 is an embedded device. Even the electric toothbrush seen in Figure 1.2 is an embedded system. A small microcontroller in the toothbrush provides programmable speed control and a battery state of charge indication. High end automobiles can contain over fifty embedded microcontrollers. A typical middle class household has over fifty embedded devices. For every PC in the world there are over one hundred embedded devices. All told, embedded devices account for most of the world's production of microprocessors.

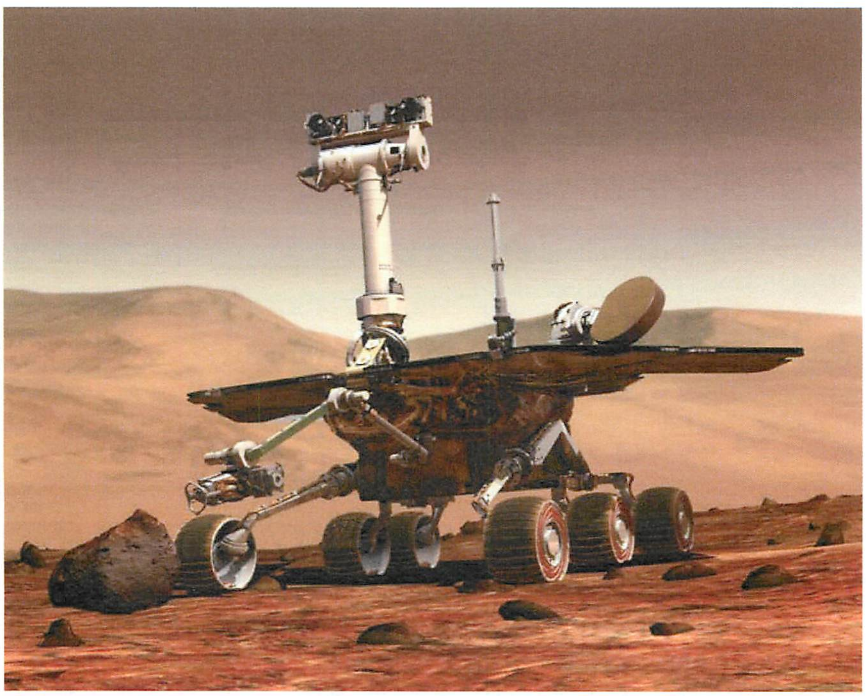
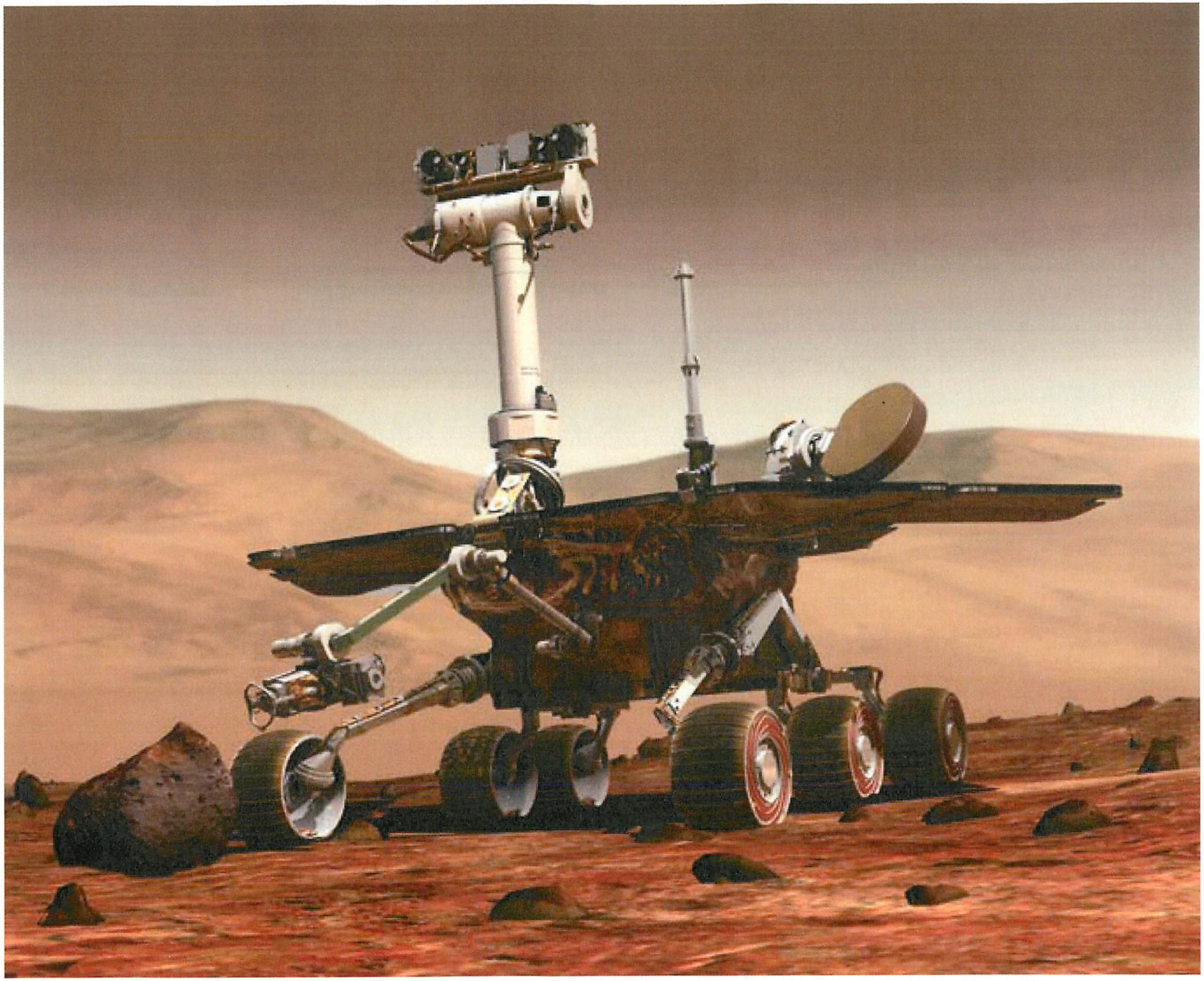


Figure 1.1 Robots such as the Mars Rovers are Embedded Systems. Photograph courtesy of NASA/JPL CALTECH.



”

Aircraft & Military Systems	Aircraft,,autopilots,,,avionics,,and,,navigation,,systems,,,automatic,,landing,,systems,,,guidance,,systems,,,engine,,controls.,,,
Biomedical Systems	CAT,,scan,,and,,Ultrasound,,imaging,,systems,,,patient,,monitors,,heart,pacers.,,,
Cars	Engine,, control,,, anti-lock,, braking,, systems,,, traction,,control,, systems,,, air,, bag,, controls,,, heating,, and,, air,,conditioning,,controls,,,GPS,,mapping,,,Satellite,,Radio,,,On-board,Diagnostics.,,,
Communications	Communication,,Satellites,,,network,,routers,,,switches,,,hubs.,,,
Consumer Electronics	TVs,,, ovens,,, dishwashers,,, DVD,, players,,, ,, stereos,,,security,,systems,,,lawn,,sprinkler,,controls,,,thermostats,,,cameras,,, clock,, radios,,, answering,, machines,,, set,, top,,boxes,,other,appliances.,,,
Computer I/O Devices	Keyboards,,mice,,printers,,scanners,,displays,,modems,,,hard,,disk,,drives,,,DVD,,drives,,,graphics,,cards,,,USB,,devices.,,
Electronic Instrumentation	Data,, acquisition,, systems,,, oscilloscopes,,, voltmeters,,,signal,generators,,logic,analyzers.,,,
Industrial Equipment	Elevator,,controls,,,surveillance,,systems,,,robots,,,CNC,,machines,,,Programmable,,Logic,,Controllers,,,industrial,,automation,and,control,systems.,,,
Office Machines	FAX,,machines,,,copiers,,,telephones,,,calculators,,,cash,,registers.,,,
Personal Devices	Cell,, phones,,, portable,, MP3,, players,,, Video,, players,,,Personal,, Digital,, Assistants,, (PDAs),,, electronic,, wrist,,watches,,,handheld,,video,,games,,,digital,,cameras,,,GPS,,systems.,,,
Robots	Industrial,, robots,,, autonomous,, vehicles,,, space,,exploration,robots,(i.e.,,Mars,robots),,,
Toys	Video,,Game,,systems,,,“Aibo”,,,"Furby“,,,and,,“Elmo”,,type,robot,toys.,,,

As seen in table 1.1, embedded devices can be found in a wide array of products including aircraft and military systems, biomedical systems, cars, communications, computer I/O devices, electronic instrumentation, home electronics, industrial equipment, office machines, personal devices, robots, and smart toys. Embedded devices can be found everywhere.

Embedded systems designers often face challenging design goals. Embedded systems must be reliable. Many embedded devices can't crash, and may not be able to reboot. Software can't be updated in many embedded devices. Many devices have critical performance & power design constraints. Real-time constraints occur in many applications and many devices have limited memory and processing power. Some devices may need to run on battery power for long periods of time. In addition, consumer devices typically have a fast time to market on new products and are very cost competitive.

Figure 1.2 Cell phones, PDAs, and Multimedia Players are all embedded systems. Even this electric toothbrush! Photographs courtesy of Motorola, Microsoft, and Philips Sonicare.

1.2 Real-Time Operating Systems

A real-time system must respond to external inputs and produce new outputs in a limited amount of time as seen in Figure 1.3. The response time needs to be bounded. Response times that are too long can cause real-time systems to fail.

An illustrative example of a real-time system is the automobile airbag controller in your car. When the airbag's motion sensors (accelerometers) detect a collision, the system needs to respond by deploying the airbag within 10ms or the system fails. At high speeds, with any delay longer than 10ms the driver will have already impacted the steering wheel before the airbag deploys.

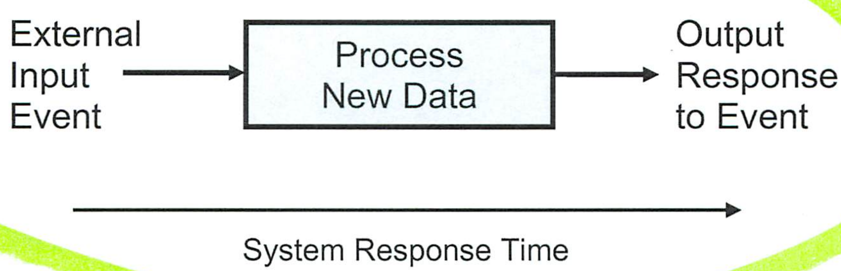


Figure 1.3 A real-time system must respond to external inputs and produce new outputs in a limited amount of time or the system fails. Current response times are around 0.5 to 10ms.

In a soft Real-Time System, critical tasks get priority. A soft real-time system normally meets the real-time response constraint. A typical soft real-time example is a multimedia player. The player could occasionally skip a video frame or audio sample and a user might not even notice as long as it ran correctly the vast majority of the time.

In a hard real-time system, a new output response must always be computed by the specified time bound or the system will fail. For a hard real-time example, consider a fly-by-wire (i.e. computer controlled) aircraft control system. In an aircraft flight control system when the pilot moves the control yoke, the flight control surfaces need to move very quickly in response or the aircraft would become unstable and crash. To insure safety, the FAA actually tests and certifies the real-time response of computer controlled flight simulators and aircraft.

Virtual memory page swapping and garbage collection routines needed for object oriented languages can cause problems in hard real-time systems. Even caching is sometimes an issue since it can cause program execution times to vary.

Many embedded systems are real-time systems with several inputs and outputs. Multiple events are occurring independently. Programming is simplified by separating the tasks, but this requires the CPU to switch back and forth among

1.4 Processors and Software used in new Embedded System Designs

Annual surveys of designers working on new embedded devices are conducted by several of the popular trade publications. It is interesting to examine the general trends uncovered by these surveys and see what types of new embedded systems are currently being developed. Figure 1.4 shows the bit size of processors currently being used in new embedded designs. Most current design activity appears to be centered around 32-bit processors. It is not surprising that given the continuing advances in VLSI technology the trend has always been toward larger and more powerful processors in embedded devices. The program code and applications running on new embedded devices also continues to increase both in complexity and memory requirements.

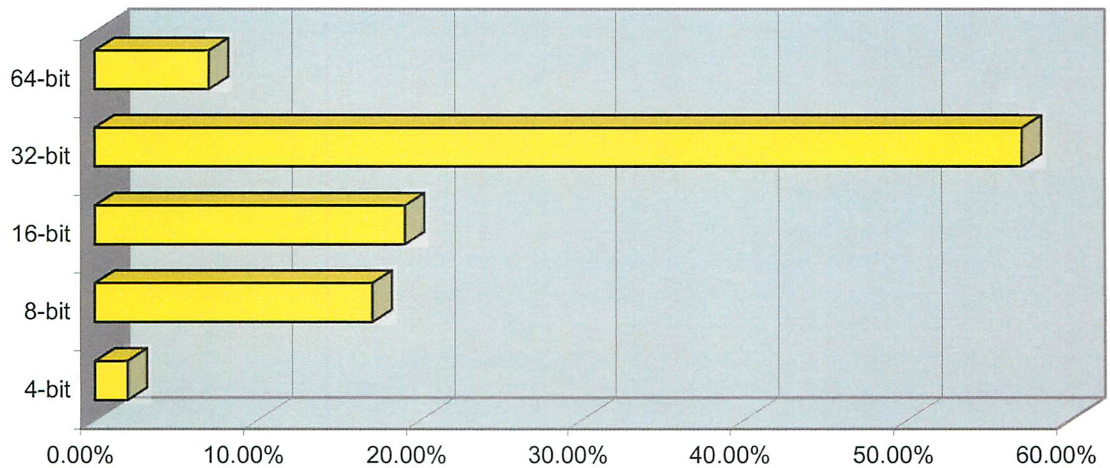


Figure 1.4 Processor bit size used in new embedded designs.

Figure 1.5 shows the annual 32 and 64-bit microprocessor sales data for 2002. ARM and X86 based processors have the largest market share followed by several other RISC processors, MIPS, SuperH, and PowerPC. Current data is believed to be similar, but is not publicly available. The ARM processor is a 32-bit RISC low-power design from an English IP company, ARM ltd (Advanced RISC Machines) http://www.arm.com/.

ARM does not make any processor chips, instead they produce their revenue by licensing the ARM IP processor designs to semiconductor manufacturers that produce their own version of an ARM processor. ARM's processor designs are licensed to over 100 chip manufacturers. ARM is used in many devices such as cell phones, iPod Nano, Cameras, Handheld Games, HDTVs, and Set-Top boxes. 80% of ARM processors are in cell phones. A good performance/power ratio makes it a very popular choice in low power and battery operated devices.

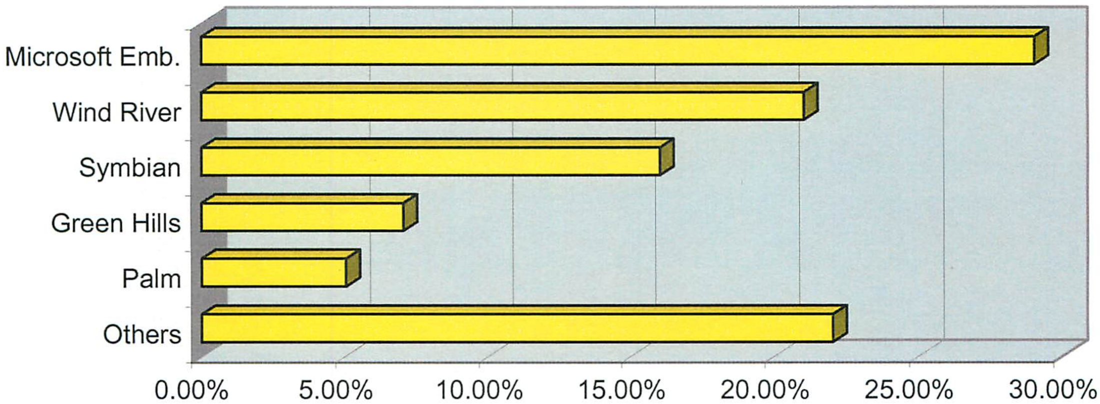


Figure 1.8 Commercial Operating Systems used in New Embedded Designs.

Figure 1.9 shows the response embedded designers gave to the question "what languages do you use to develop embedded systems". The C family of languages is clearly used for the majority of embedded systems development. For assembly language, the response indicated that around one third of embedded systems designers still had to use assembly language for some small portion of their designs.

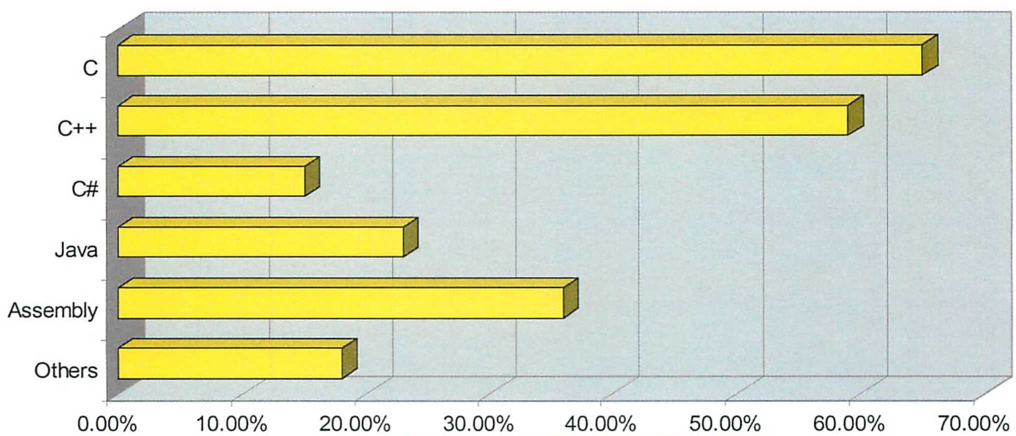


Figure 1.9 Programming Languages used in Embedded Devices.

Note that the way this question was posed in the survey, this response does not mean that one third of the code is written in assembly language. Rather it means that one third of the designers still need to use assembly language somewhere in their design. Other surveys indicate the amount of assembly

The software development tools are typically provided with the OS. Since the OS is written in C/C++ a compiler, linker, debugger, and binary image tools are needed to generate a new OS. These same tools are typically used for application development.

Software development occurs in parallel with hardware development to reduce the total product development time. This has become more important given the ever shortening product life cycles of current embedded devices. Emulation tools and embedded computer boards with similar hardware running the same OS can be used to develop and test software before the new hardware platform is available. Since the majority of the code is written in C/C++/C#, a large portion of the software can even be developed and tested on a different processor or an emulator. Code is then recompiled to target a new processor for the final round of development and testing once the new hardware becomes available.

In Windows Embedded CE, an ARM emulator is provided with the development tools, an X86 PC-based device (called a CEPC), or an embedded computer board (called a target device) can be used for initial software development and testing before the new hardware platform is available.

1.7 Memory Technologies used in Embedded Devices

Most embedded devices currently use two types of memory, SDRAM or occasionally perhaps SRAM for main memory and Flash or ROM memory for non-volatile storage. SDRAM has a significantly lower cost per bit than SRAM, but requires a more complex hardware controller for periodic dynamic memory refresh cycles. One important decision that must be made early in the design process is how much memory of each type is needed for the device.

The operating system and application programs are typically stored in flash, since most embedded devices do not have a hard disk drive. Hard disks have a higher failure rate, they require more space, and power, so they are not a viable option for many embedded designs especially in small mobile and battery operated devices.

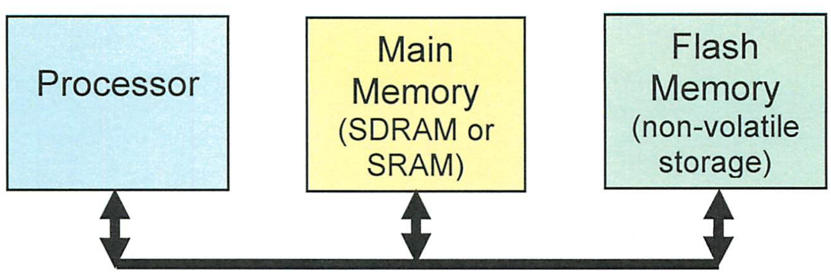


Figure 1.11 Most Embedded systems use two types of memory. SDRAM for main memory and Flash memory instead of a hard disk to store the OS and application programs.

NOW SHOW THE XILINX EDK TUTORIAL