DOWNLOAD PDF EMBEDDED DSP PROCESSOR DESIGN

Chapter 1 : Introduction to DSP Builder for Intel FPGAs

This book provides design methods for Digital Signal Processors and Application Specific Instruction set Processors, based on the author's extensive, industrial design experience. Top-down and bottom-up design methodologies are presented, providing valuable guidance for both students and practicing design engineers.

Embedded silicon features such as embedded memory, DSP blocks, and embedded processors are ideally suited for implementing DSP functions such as finite impulse response FIR filters, fast Fourier transforms FFTs , correlators, equalizers, encoders, and decoders. The embedded DSP blocks provide functionality such as addition, subtraction, and multiplication, which are common arithmetic operations in DSP functions. Many DSP applications use external memory devices to manage large amounts of data processing. The embedded memory in FPGAs meets these requirements and also eliminates the need for external memory devices in some cases. Embedded processors in FPGAs provide versatile system integration because of flexible partitioning of the system between hardware and software. Intel devices provide a choice between embedded soft core processors and embedded hard core processors. You can use multimaster buses to define as many buses and as much performance as needed for a particular application. Off-the-shelf DSP processors make compromises between size and performance when they choose the number of data buses on the chip, potentially limiting performance. DSP processors have predefined hardware accelerator blocks, but FPGAs can implement hardware accelerators for each application, allowing the best achievable performance from hardware acceleration. You can implement hardware accelerator blocks with parameterizable IP functions or from scratch using HDL. The flexibility of programmable logic and soft IP allows you to quickly adapt your designs to new standards without waiting for long lead times usually associated with DSP processors. Embedded processors and hardware acceleration offer the flexibility, performance, and cost effectiveness in a development flow that is familiar to software developers. You can combine a software design flow with hardware acceleration. In this flow, you first profile C code and identify the functions that are the most performance critical. Intel also provides system integration tools such as Platform Designer for system-level partitioning and interconnection. These tools enable hardware design, simulation, debug, and in-system verification of the DSP system. Intel recommends you do not use it for new designs, except as a wrapper for advanced blockset designs. Related Information Volume 3: In particular, you can use the basic Simulink blockset to create interactive testbenches. The automatic testbench flow runs a test and returns a result indicating whether or not the outputs match. It creates a memory-mapped interface only if the design contains interface blocks or external memory blocks.

Chapter 2 : Processors & DSP | Analog Devices

Publisher Summary. This chapter focuses on the design of DSP architectures. Architecture design is a specification of the top level of a processor, which includes defining hardware modules of the core and connections among them.

However, they may also use some more specific tools: In circuit debuggers or emulators see next section. Utilities to add a checksum or CRC to a program, so the embedded system can check if the program is valid. For systems using digital signal processing , developers may use a math workbench to simulate the mathematics. System level modeling and simulation tools help designers to construct simulation models of a system with hardware components such as processors , memories , DMA , interfaces , buses and software behavior flow as a state diagram or flow diagram using configurable library blocks. Simulation is conducted to select right components by performing power vs. Typical reports that helps designer to make architecture decisions includes application latency, device throughput, device utilization, power consumption of the full system as well as device-level power consumption. A model-based development tool creates and simulate graphical data flow and UML state chart diagrams of components like digital filters, motor controllers, communication protocol decoding and multi-rate tasks. Custom compilers and linkers may be used to optimize specialized hardware. An embedded system may have its own special language or design tool, or add enhancements to an existing language such as Forth or Basic. Another alternative is to add a real-time operating system or embedded operating system Modeling and code generating tools often based on state machines Software tools can come from several sources: Software companies that specialize in the embedded market Ported from the GNU software development tools Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor As the complexity of embedded systems grows, higher level tools and operating systems are migrating into machinery where it makes sense. For example, cellphones , personal digital assistants and other consumer computers often need significant software that is purchased or provided by a person other than the manufacturer of the electronics. Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical applications. Household appliances, such as microwave ovens, washing machines and dishwashers, include embedded systems to provide flexibility and efficiency. Debugging[edit] Embedded debugging may be performed at different levels, depending on the facilities available. The different metrics that characterize the different forms of embedded debugging are: From simplest to most sophisticated they can be roughly grouped into the following areas: Interactive resident debugging, using the simple shell provided by the embedded operating system e. Forth and Basic External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger that even works for heterogeneous multicore systems. An in-circuit emulator ICE replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor. A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified, and allowing debugging on a normal PC. The downsides are expense and slow operation, in some cases up to times slower than the final system. This is used to debug hardware, firmware and software interactions across multiple FPGA with capabilities similar to a logic analyzer. Software-only debuggers have the benefit that they do not need any hardware modification but have to carefully control what they record in order to conserve time and storage space. The view of the code may be as HLL source-code , assembly code or mixture of both. Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software- and microprocessor- centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals DSP, FPGA, and co-processor. An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. A graphical view is presented by a host PC tool, based on a recording of the system behavior. The trace recording can be performed in software, by the RTOS, or by special tracing hardware. RTOS tracing allows developers to understand timing and performance issues of the software system and gives a good

understanding of the high-level system behaviors. Reliability[edit] Embedded systems often reside in machines that are expected to run continuously for years without errors, and in some cases recover by themselves if an error occurs. Therefore, the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided. Specific reliability issues may include: The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles. The system must be kept running for safety reasons. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals. The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service. A variety of techniques are used, sometimes in combination, to recover from errorsâ€”both software bugs such as memory leaks , and also soft errors in the hardware: This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection. Immunity Aware Programming High vs. For low-volume or prototype embedded systems, general purpose computers may be adapted by limiting the programs or by replacing the operating system with a real-time operating system. Embedded software architectures[edit] There are several different types of software architecture in common use. Simple control loop[edit] In this design, the software simply has a loop. The loop calls subroutines , each of which manages a part of the hardware or software. Hence it is called a simple control loop or control loop. Interrupt-controlled system[edit] Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte. These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes. Cooperative multitasking[edit] A nonpreemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The advantages and disadvantages are similar to that of the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue. Preemptive multitasking or multi-threading[edit] In this type of system, a low-level piece of code switches between tasks or threads based on a timer connected to an interrupt. This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel. As any code can potentially damage the data of another task except in larger systems using an MMU programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues , semaphores or a non-blocking synchronization scheme. Because of these complexities, it is common for organizations to use a real-time operating system RTOS , allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real-time system, due to limitations regarding memory size, performance, or battery life. The choice that an RTOS is required brings in its own issues, however, as the selection must be done prior to starting to the application development process. This timing forces developers to choose the embedded operating system for their device based upon current requirements and so restricts future options to a large extent. These trends are leading to the uptake of embedded middleware in addition to a real-time operating system. Microkernels and exokernels[edit] A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc. In general, microkernels succeed when the task switching and intertask communication is fast and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are

available to and extensible by application programmers. Monolithic kernels[edit] In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment similar to a desktop operating system like Linux or Microsoft Windows , and is therefore very productive for development; on the downside, it requires considerably more hardware resources, is often more expensive, and, because of the complexity of these kernels, can be less predictable and reliable. Common examples of embedded monolithic kernels are embedded Linux and Windows CE. Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as wireless routers and GPS navigation systems. Here are some of the reasons: Ports to common embedded chip sets are available. They permit re-use of publicly available code for device drivers , web servers , firewalls , and other code. Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume. Many engineers believe that running application code in user mode is more reliable and easier to debug, thus making the development process easier and the code more portable. Additional software components[edit] In addition to the core operating system, many embedded systems have additional upper-layer software components. If the embedded device has audio and video capabilities, then the appropriate drivers and codecs will be present in the system. In the case of the monolithic kernels, many of these software layers are included. In the RTOS category, the availability of the additional software components depends upon the commercial offering.

Chapter 3 : Application-specific instruction set processor - Wikipedia

Embedded DSP Processor Design Application Specific Instruction Set Processors Dake Liu â€¢ i^: t\ AMSTERDAM â€¢ BOSTON â€¢ HEIDELBERG â€¢ LONDON.

History[edit] Prior to the advent of stand-alone DSP chips discussed below, most DSP applications were implemented using bit-slice processors. The AMD bit-slice chip with its family of components was a very popular choice. There were reference designs from AMD, but very often the specifics of a particular design were application specific. These bit slice architectures would sometimes include a peripheral multiplier chip. It also set other milestones, being the first chip to use Linear predictive coding to perform speech synthesis. In , AMI released the S It was designed as a microprocessor peripheral, and it had to be initialized by the host. The S was likewise not successful in the market. Both processors were inspired by the research in PSTN telecommunications. The Altamira DX-1 was another early DSP, utilizing quad integer pipelines with delayed branches and branch prediction. It was based on the Harvard architecture, and so had separate instruction and data memory. It already had a special instruction set, with instructions like load-and-accumulate or multiply-and-accumulate. TI is now the market leader in general-purpose DSPs. About five years later, the second generation of DSPs began to spread. They had 3 memories for storing two operands simultaneously and included hardware to accelerate tight loops ; they also had an addressing unit capable of loop-addressing. The main improvement in the third generation was the appearance of application-specific units and instructions in the data path, or sometimes as coprocessors. These units allowed direct hardware acceleration of very specific but complex mathematical problems, like the Fourier-transform or matrix operations. Some chips, like the Motorola MC, even included more than one processor core to work in parallel. Modern DSPs[edit] Modern signal processors yield greater performance; this is due in part to both technological and architectural advancements like lower design rules, fast-access two-level cache, E DMA circuitry and a wider bus system. TMSC chips each have three such DSPs, and the newest generation C chips support floating point as well as fixed point processing. The processors have a multi-threaded architecture that allows up to 8 real-time threads per core, meaning that a 4 core device would support up to 32 real time threads. The Blackfin family of embedded digital signal processors combine the features of a DSP with those of a general use processor. The TriMedia media processors support both fixed-point arithmetic as well as floating-point arithmetic , and have specific instructions to deal with complex filters and entropy coding. Introduced in [7] , the dsPIC is designed for applications needing a true DSP as well as a true microcontroller , such as motor control and in power supplies. Most DSPs use fixed-point arithmetic, because in real world signal processing the additional range provided by floating point is not needed, and there is a large speed benefit and cost benefit due to reduced hardware complexity. Floating point DSPs may be invaluable in applications where a wide dynamic range is required. Product developers might also use floating point DSPs to reduce the cost and complexity of software development in exchange for more expensive hardware, since it is generally easier to implement algorithms in floating point. With a processing speed of 0.

DOWNLOAD PDF EMBEDDED DSP PROCESSOR DESIGN

Chapter 4 : Osborn, Embedded Microcontrollers & Processor Design | Pearson

This book provides design methods for Digital Signal Processors and Application Specific Instruction set Processors, based on the author's extensive, industrial design experience.

The following document describes the basic concepts of Digital Signal Processing DSP and also contains a variety of Recommended Reading links for more in-depth information. What is a DSP? Digital Signal Processors DSP take real-world signals like voice, audio, video, temperature, pressure, or position that have been digitized and then mathematically manipulate them. A DSP is designed for performing mathematical functions like "add", "subtract", "multiply" and "divide" very quickly. Signals need to be processed so that the information that they contain can be displayed, analyzed, or converted to another type of signal that may be of use. In the real-world, analog products detect signals such as sound, light, temperature or pressure and manipulate them. From here, the DSP takes over by capturing the digitized information and processing it. It then feeds the digitized information back for use in the real world. It does this in one of two ways, either digitally or in an analog format by going through a Digital-to-Analog converter. All of this occurs at very high speeds. During the recording phase, analog audio is input through a receiver or other source. This analog signal is then converted to a digital signal by an analog-to-digital converter and passed to the DSP. During the playback phase, the file is taken from memory, decoded by the DSP and then converted back to an analog signal through the digital-to-analog converter so it can be output through the speaker system. In a more complex example, the DSP would perform other functions such as volume control, equalization and user interface. Signals may be compressed so that they can be transmitted quickly and more efficiently from one place to another e. Signals may also be enhanced or manipulated to improve their quality or provide information that is not sensed by humans e. Although real-world signals can be processed in their analog form, processing signals digitally provides the advantages of high speed and accuracy. You can create your own software or use software provided by ADI and its third parties to design a DSP solution for an application. For more detailed information about the advantages of using DSP to process real-world signals, please read Part 1 of the article from Analog Dialogue titled: A DSP contains these key components: Stores the information to be processed Compute Engine: Serves a range of functions to connect to the outside world Recommended Reading Digital Signal Processing is a complex subject that can overwhelm even the most experienced DSP professionals. Although we have provided a general overview, Analog Devices offers the following resources that contain more extensive information about Digital Signal Processing:

Chapter 5 : A Beginner's Guide to Digital Signal Processing (DSP) | Design Center | Analog Devices

Coverage includes design of internal-external data types, application specific instruction sets, micro architectures, including designs for datapath and control path, as well as memory sub systems. Integration and verification of a DSP-ASIP processor are discussed and reinforced with extensive examples.

Chapter 6 : Digital signal processor - Wikipedia

embedded dsp processor design Download embedded dsp processor design or read online books in PDF, EPUB, Tuebl, and Mobi Format. Click Download or Read Online button to get embedded dsp processor design book now.

Chapter 7 : embedded dsp processor design | Download eBook pdf, epub, tuebl, mobi

Digital Signal Processors (DSP) take real-world signals like voice, audio, video, temperature, pressure, or position that have been digitized and then mathematically manipulate them. A DSP is designed for performing mathematical functions like "add", "subtract", "multiply" and "divide" very quickly.

DOWNLOAD PDF EMBEDDED DSP PROCESSOR DESIGN

Chapter 8 : Embedded Dsp Processor Design Dake Liu by LibbyHoyle - Issuu

This design example exercises the memory mapped interfaces of the hard processor system (HPS) exposed to the FPGA fabric. The design performs memory tests by writing and reading the HPS memory using various ports of the HPS and measures the performance of the data movements.

Chapter 9 : Support Resources Design Examples

embedded dsp processor design application specific instructi see more like this.