DOWNLOAD PDF DSP CORE PERIPHERALS

Chapter 1 : Infineon TriCore - Wikipedia

Unified Core. With an abundance of acoustic echo cancellation (AEC), raw processing power and networked audio channel count, the Q-SYS Core f offers the best cost-to performance ratio of any in-room processing solution available.

The CMSIS enables consistent device support and simple software interfaces to the processor and the peripherals, simplifying software re-use, reducing the learning curve for microcontroller developers, and reducing the time to market for new devices. The CMSIS is defined in close cooperation with various silicon and software vendors and provides a common approach to interface to peripherals, real-time operating systems, and middleware components. The CMSIS is intended to enable the combination of software components from multiple middleware vendors. API for the Cortex-M processor core and peripherals. The API is RTOS independent and connects microcontroller peripherals with middleware that implements for example communication stacks, file systems, or graphic user interfaces. The library is available for all Cortex-M cores. CMSIS-NN is a collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Cortex-M processor cores. It provides a standardized programming interface that is portable to many RTOS and enables software components that can work across multiple RTOS systems. Development tools and web infrastructures use the PDSC file to extract device parameters, software components, and evaluation board configurations. System View Description for Peripherals. Describes the peripherals of a device in an XML file and can be used to create peripheral awareness in debuggers or header files with peripheral register and interrupt definitions. This component is provided as separate download. System resource definition and partitioning. Defines methods to describe system resources and to partition these resources into multiple projects and execution areas. Note Refer to ARM:: It enables consistent software layers and device support across a wide range of development tools and microcontrollers. CMSIS is not a huge software layer that introduces overhead and does not define standard peripherals. The silicon industry can therefore support the wide variations of Cortex-M processor-based devices with this common standard. Developers can write software quicker through a variety of easy-to-use, standardized software interfaces. Consistent software interfaces improve the software portability and re-usability. Generic software libraries and interfaces provide consistent software framework. Provides interfaces for debug connectivity, debug peripheral views, software delivery, and device support to reduce time-to-market for new microcontroller deployment. Provides a compiler independent layer that allows using different compilers. CMSIS is supported by mainstream compilers. Enhances program debugging with peripheral information for debuggers and ITM channels for printf-style output and RTOS kernel awareness. CMSIS-Zone will simplify system resource and partitioning as it manages the configuration of multiple processors, memory areas, and peripherals. Variables and parameters have a complete data type. Expressions for define constants are enclosed in parenthesis. MISRA rule violations are documented. CamelCase names to identify function names and interrupt functions.

Chapter 2 : ADSP-SC Datasheet and Product Info | Analog Devices

Programmability is provided by dual-core ARM Cortex-A15 RISC CPUs with Neonâ„¢ extension, TI C66x VLIW floating-point DSP core, and Vision AccelerationPac (with one or more EVEs). The ARM allows developers to keep control functions separate from other algorithms programmed on the DSP and coprocessors, thus reducing the complexity of the system.

Training Cortex Microcontroller Software Interface Standard CMSIS enables consistent device support and simple software interfaces to the processor and its peripherals, simplifying software reuse, reducing the learning curve for microcontroller developers, and reducing the time to market for new devices. Creation of software is a major cost factor in the embedded industry. Standardizing the software interfaces across all Cortex-M silicon vendor products, especially when creating new projects or migrating existing software to a new device, means significant cost reductions. CMSIS is defined in close cooperation with various silicon and software vendors and provides a common approach to interface to peripherals, real-time operating systems, and middleware components. It simplifies software reuse, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices. Consistent system startup and peripheral access System startup, processor core access, and peripheral definitions are essential for every embedded application. The standardized CMSIS-CORE is implemented for over different devices and makes it easy to get started with a new device or migrate software across microcontrollers. Deterministic Real-Time Software Execution A super-loop concept is only adequate for simple embedded applications. Cortex-M microcontrollers are designed for real-time operating systems that give you resource and time control. Generic peripheral interfaces for middleware and application code Interfacing microcontroller peripherals with middleware or generic application code can be challenging as each device is different. Ready-to use CMSIS-Driver interfaces are today available for many microcontroller families and avoid cumbersome and time consuming driver porting. Easy access to reusable software components Previously, software modules were hard to integrate as the source and header files had unclear requirements, inconsistent documentation, or missing license information. Because CMSIS-Pack defines the structure of a software pack containing software components, these issues are addressed. Software components are easily selectable, and any dependencies on other software are highlighted. Consistent view to device and peripherals For every supported microcontroller, debuggers can provide detailed views to the device peripherals that display the current register state. Connectivity to low-cost evaluation hardware Inexpensive development boards are available from many microcontroller vendors. Frequently, a low-cost debug unit is included, but different interfaces need a specific tool setup. Efficient neural network kernels Neural network-based solutions are becoming increasingly popular for embedded machine learning applications. CMSIS-NN is a collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Cortex-M processor cores. Stay Informed Sign up for news and updates. First Name Please enter your first name. Last Name Please enter your last name.

Chapter 3 : TMSDMSCYE0 | Audio & media processors | TI Store

DSP Cores A core is the "guts" of the processor, i.e. the CPU, and may include peripherals. Many current DSP cores do not include memory, whereas the first LSI DSP core (circa) and older TI DSP cores (circa) do.

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.

Chapter 4 : STSW-STM - STM32F4 DSP and standard peripherals library - STMicroelectronics

The combination of an application-specific DSP core and a set of application-centric peripherals allows the DSP device to perform extremely fast calculations and algorithms specialized for image processing. By integrating the key audio/visual connectivity peripherals on chip, overall system cost is reduced.

Robert Oshana Texas Instruments - Leave a Comment Embedded systems interact with the environment and other components within a larger system. Much of the bottleneck in meeting real-time demands comes from delays in getting data in and out of the system, rather than processor speed or software implementation. Knowing how to select the right mix of peripherals for embedded applications and how to utilize these peripherals to optimize system performance can alleviate this bottleneck. Choosing peripherals DSP developers must understand when and where peripheral trade-offs can be advantageous for a given application. Using inefficient or poorly chosen peripherals for the application can significantly affect system performance. Embedded processors vary in their customization for the specific problem at hand. These processor types will range from general purpose processors that handle a wide variety of applications, to application-specific processors like DSPs, which are specific to a particular application class such as signal processing , to single purpose processors, which are customized to a very specific function. A single purpose processor is a digital circuit designed and implemented to execute a very precise program. In a digital camera, for example, a single purpose processor is often used to implement a JPEG codec, which can then be used to perform compression and decompression on video frames. Many so-called single purpose processors are predesigned and integrated onto a more complicated System on a Chip SoC. Other common names for these single purpose processors are accelerators or peripherals 1. There are benefits, as well as drawbacks, to using peripherals in embedded systems processors. On the other hand, a custom peripheral implementation is less flexible across applications, and unit cost may be higher due to the increased customization. Modern DSPs often have several peripherals integrated on chip. Some of these peripherals require significant software support to operate efficiently, so engineers should consider whether the vendor supplies peripheral device drivers to interface with the peripherals to ease overall system development time. Because DSPs are designed to support hard real-time processing with streaming data samples as the critical path for many applications, these processors are optimized to move this data quickly and efficiently from peripherals to the DSP core for processing and then back out to the environment 2. The interaction between the peripheral and the serial port is synchronous and controlled mostly in hardware. DSPs also provides one or more Direct Memory Access DMA controllers to reduce the overhead of multiple interrupts being generated over the serial port. Intelligent use of the DMA using software control allows multiple samples to be buffered and then transferred to the on-chip memory without involving the DSP. The DSP is interrupted when the buffer is full of data, rather than getting interrupted on every data sample. Peripherals â€" performance and function Peripherals are often referred to as hardware accelerators. In general, this refers to the replacement of a software algorithm with a hardware component, taking advantage of the intrinsic speed of hardware. Interfacing to a hardware accelerator is similar to calling a function. The main difference is that the function is implemented in hardware and is transparent to the calling function. Using hardware-accelerated peripherals can result in an order of magnitude improvement in execution time, up to x for some algorithms. Hardware-accelerated peripherals are more efficient at performing certain mathematical functions, moving data from one place to another, and repetitively performing the same operation many times. Using a peripheral requires the programmer to write data to memory-mapped registers most peripherals have memory-mapped registers. Peripheral computation is then performed outside of the CPU so the CPU can continue executing code while the peripheral is processing in parallel. Setup and initialization of a peripheral is not too difficult, and usually requires several instructions to write to the control registers, status registers, and data registers, as well as a few instructions to read the result 3. As an example of the power of peripherals, consider the latest generation of DSP processors for the video application space. Until recently, DSPs have not been fast enough to process images and video in real time. The main requirement for video applications is to be fast enough to keep up with smooth, continuous

video and still be able to extract useful information. The combination of an application-specific DSP core and a set of application-centric peripherals allows the DSP device to perform extremely fast calculations and algorithms specialized for image processing. These video port peripherals provide a glueless interface to common video decoder and encoder devices. The ITU-BT standard describes the fundamentals of the video digitization process, while SMPTE M defines the parameters required to generate and distribute component video signals on a parallel interface. In addition to the video port peripherals, the DM also has a multichannel audio serial port peripheral to support up to eight stereo lines over 16 channels, an Ethernet MAC to connect the system to IP packet networks, and a PCI peripheral to connect to a backplane chassis or PCI bus. As shown in Figure 3, video port data transfers take place using the DMA. DMA requests are based on buffer thresholds. The preferred transfer size is often one entire line of data because this allows the most flexibility in terms of frame buffer line pitch. Some modes of operation for the highest display rates may require more frequent DMA requests, such as on a half or quarter line basis 4. Embedded engineers must select the right processor for the right application. Even though this rarely results in a perfect match, the engineer must be able to make intelligent decisions based on some important data, such as:

Chapter 5 : Selecting the right peripheral for DSP applications - Signal Processing Design

The Q-SYSâ„¢ Core f processor is the latest addition to the Q-SYS Core family, providing a solution for small, single room projects up to the largest Enterprise scale deployments. The continuity of the Q-SYS software based DSP platform the Q-SYS Core f to leverage all the features that are.

Before diving in to this content, we recommend reading the following introductory material: In StarterWare, the system configuration library provides simple APIs for interrupt support and cache management. Programming ARM applications can execute in privileged or non-privileged user mode. While executing in user mode, the application cannot access system resources which require privileged access. Note that all ISRs are executed in privileged mode. Separate APIs are provided for enabling and disabling instruction and data cache. Also, APIs are given for invalidating and cleaning cache memory. Flushing a cache will clear it of any stored data. Cleaning a cache will force it to write back the cached values to main memory. Note that the MMU must be enabled to use data cache. The below sequence can be used to configure the MMU. Create a page table. The page table starting address must be aligned to 16kB. Set the translation table base register to the starting address of the page table using the TtbSet API. Upon reaching main , the execution is in privileged mode. System interrupts are generated by device peripherals. The AINTC supports up to system interrupts, each of which can be prioritized by mapping to one of 32 channels. The application developer must decide how to map system interrupts to channel numbers. A single system interrupt cannot be mapped to multiple channels. However, more than one system interrupt can be mapped to a single channel. This can be done by mapping each system interrupt used to a particular channel. An ISR should be registered for all system interrupts enabled for processing. This will make sure that all the fault system interrupt pending status is cleared before we enable any system interrupt. After the configuring AINTC, the application can enable interrupt processing for one or more peripherals. Any system interrupt generated by a peripheral after this point will cause the corresponding registered ISR to be called. The application uses the Timer64P2 peripheral to demonstrate interrupt processing. The system interrupt number for Timer64P2 is 68, which is mapped to channel number 2. TimerIsr is the ISR registered for this system interrupt. This processor acts as co-master with the ARM in the overall system. The DSP has its own interrupt controller and cache system. System events are generated by device peripherals. Programming The application developer must decide how to map system events to DSP interrupts. This can be done on a per-event basis with a single API call. DSP interrupts are then mapped to interrupt service routines ISRs that are provided by the application. Since there are only 16 DSP interrupts counting 4 reserved interrupts but over system events, the DSP subsystem also provides an event combiner module ECM to combine multiple system events into DSP interrupt s. Using the ECM requires a slightly different procedure: ECM is typcially only used by applications that use more system events than there are DSP interrupts, but this application was intentionally written to use ECM as an example. Cache Management The C DSP core also includes comprehensive cache support that can be used to speed up program or data memory access. All or part of internal L1 and L2 memories can be dedicated to cache. Caching must be explicitly enabled for specific memory ranges using the appropriate MAR bits. The following steps are required to enable cache for a particular memory region: Writing back cached memory copies the current cache lines into physical memory. Invalidating cached memory throws out the cached values. It also demonstrates the importance of maintaining cache coherency when working with another memory master i. This is a useful feature for debugging purposes. Writing one to the bit will generate the signal or event. These events are fed to the respective interrupt controller INTC to get mapped to the core interrupt inputs. To pass data, any of the KB internal or MB external memory areas can be used as shared memory. Mutually exclusivity can be controlled using the mutex or semaphore mechanisms provided with the operating system. The Notify module contains APIs to set up events, send notification to the other CPU as well as wait for an event in case polling method is desirable over interrupt method. The number of events, as well as its associated handler is programmable for each CPU. By not tying any meaning to the bit value nor the event, your application can use the same call to pass a simple bit value for one event, or a bit pointer to a

shared buffer of a complex data structure containing commands and data for another event, with no buffer copying needed between the 2 CPUs. Under "File List", click on "ipc. On the host PC, run a serial terminal application ex. Teraterm, Hyperterminal, minicom and configure it for baud, no parity, 1 stop bit and no flow control. The ARM side application will output to this serial console. To run the executables without rebuilding, set up the serial terminal and start CCSv5 with an XDS emulator connected to the EVM or with the on-board emulator see here for more instructions. Each time the value is received, the application performs a left shift and sends it back to the other CPU. The loop is set 10 times with the receved value printed out each time by each CPU. You can modify this example to add more events and see how your application can be "event driven" in an OS-less programming environment. Once received, the ARM side application prints the received string buffer, refilled with another string and send it back to the DSP side. The StarterWare package includes a device abstraction layer DAL , or driver library, to make these peripherals easy to use. StarterWare also includes example applications to demonstrate the use of these peripherals. The UART module provides a byte FIFO for the transmitter and receiver sections for buffering the data, which improves performance under slight latency conditions. Please ensure that the local echo setting for the terminal is turned off. When the "uart" example application is loaded and run on the target, the text "Starterware UART echo application" is printed on the serial terminal. After this it loops indefinitely, waiting for input characters from the user on the serial terminal. The application echoes any input back to the terminal. Features used in this example: The application waits indefinitely for this input. Once the data is entered, the application echoes the same characters back to the terminal. The I2C module supports only Fast mode of operation up to kbps. The I2C module can be configured to multiple master-transmitters and slave-receivers mode and multiple slave-transmitters and master-receivers mode. I2C module requires that the module input frequency fall between 6. The actual output clock operating clock frequency is obtained by applying the clock divisor to this pre-scaled clock speed. This functionality is demonstrated either using interrupts only or using DMA as well, depending on which application is used. Features used in these examples: Then, the written data is read back from SPI flash and compared against the original data that was written. If they match, a message is displayed on the host PC serial terminal. There are two versions of this example application. Features use in these examples:

Chapter 6 : Embedded Software Development | Cortex Microcontroller Software Interface Standard â€“ Arm

TMSCx DSP Peripherals Overview This document provides an overview and briefly describes the peripherals available on the TMSCxâ„¢ digital signal processors (DSPs) of the TMSCâ„¢ family.

Chapter 7 : TMSCDZKB3 | Cx DSP | TI Store

A DSP peripheral module is an I/O device of the DSP core. It is located near the processor core within the processor chip. In the past, when a DSP appeared as a chip, there were always peripherals included on the chip.

Chapter 8 : Q-SYS Cores - Products, Peripherals & Accessories - Q-SYS Platform - Products - Systems - C

DSP core clock, CCLK; and the peripheral clock, HCLK. CCLK can sustain clock values of up to MHz, while HCLK can be equal to CCLK or CCLK/2 for values up to a maximum.

Chapter 9 : StarterWare User Guide - Texas Instruments Wiki

A digital signal processor (DSP) is a specialized microprocessor (or a SIP block), with its architecture optimized for the operational needs of digital signal processing. [1] [2] The goal of digital DSP signal processors is usually to measure, filter or compress continuous real-world analog signals.