

Epics7B: A lean and mean concept

Ron Schiffelers, René van den Berg, John van den Braak,
Harpreet S. Bhullar, Simon Thijs de Feber, Michiel Klaarwater

Philips Semiconductors
Gerstweg 2, 6534 AE, Nijmegen, The Netherlands
+31 24 353 3411

Ron.Schiffelers@philips.com

rene.van.den, john.van.den, harpreet.bhullar, st.de.feber, michiel.klaarwater @Philips.com

ABSTRACT

This paper focuses on the architecture of the Epics7B and its implementation in a sea-of-DSP application. The Epics7B DSP core is optimized for usage in audio applications for Car, TV and portable audio being low cost and low power. The multi core approach offers maximum flexibility towards platform design and product specific derivations from these platforms. Besides operation in a multi DSP environment the Epics7B also interfaces easily to MIPS and ARM micro controller cores.

Categories and Subject Descriptors

System on Chip: DSP architecture and the use of this architecture in a multi core environment.

General Terms

Algorithms, Performance, Design, Theory.

Keywords

DSP, System, Sea of DSP, multi-core, multi-processor, audio, low power, low cost.

1. INTRODUCTION

Main requirements for the present and future DSP based hardware and software developments are short time to market in combination with a low cost solution, maximum processing power, low power performance and a flexible and easy application specific integration. These requirements are contradicting as such that high performance (processing power) and flexibility often has a larger silicon area (cost) and power

dissipation as a result. Optimizing for cost and power, in both the hardware and the embedded software, where the last square millimeter and MIPS should be squeezed out, does not correspond to short time to market of any product, unless one has an unlimited amount of resources, which is not really the case in most situations.

The platform-based approach is one of the ways to reduce the time to market. Making a general platform that can be used to derive products from and to develop the embedded software on in parallel is a popular way to go. 'Platform' is however a very broad definition which doesn't tell us what it really means. A lot of ingredients define the final shape of a platform. Examples are the target applications, processor architectures, the internal bus structure, the peripherals, the software framework and library and the tools.

Looking at processor architectures and the applications they should serve in the case of car, TV and portable audio a deliberate choice has been made not to go for the 'one big core can serve it all' approach. Current applications demand more complex algorithms and more important a combination of a lot of features to be executed in one application. The above mentioned applications do have contradicting requirements as the Car and TV applications ask for as much processing power as possible and the Portable Audio applications require low power solutions. The Epics7B architecture has been chosen as a solution for these wide- ranging products, being very compact in size, showing low power behavior, being capable of achieving high clock speeds and still having an instruction set powerful enough to keep the overall system area and power consumption very competitive, when more than 1 core is needed to support the complete application. The short time to market for the silicon development of the product is achieved through the plug and play capabilities of the DSP sub system. This allows to add and remove area and processing power from the system architecture in a fast and easy way, thereby making it possible to use the same architecture base for both high and low end products. Low end products will then be a sub set of the mid end products, which again submit as sub sets of the high end products. One of the ways to accomplish short time to market of the software development is the efficient C compilability of software for the Epics7B. An other reason is the software library approach that has been taken, allowing easy and fast integration of software modules into an application. The approach used for the hardware for the development of products in the complete spectrum between high and low end products is also applicable to the software. Smart partitioning of the software over the several

cores will demand a minimum of changes when migrating from one product to the other and across the whole range of target products.

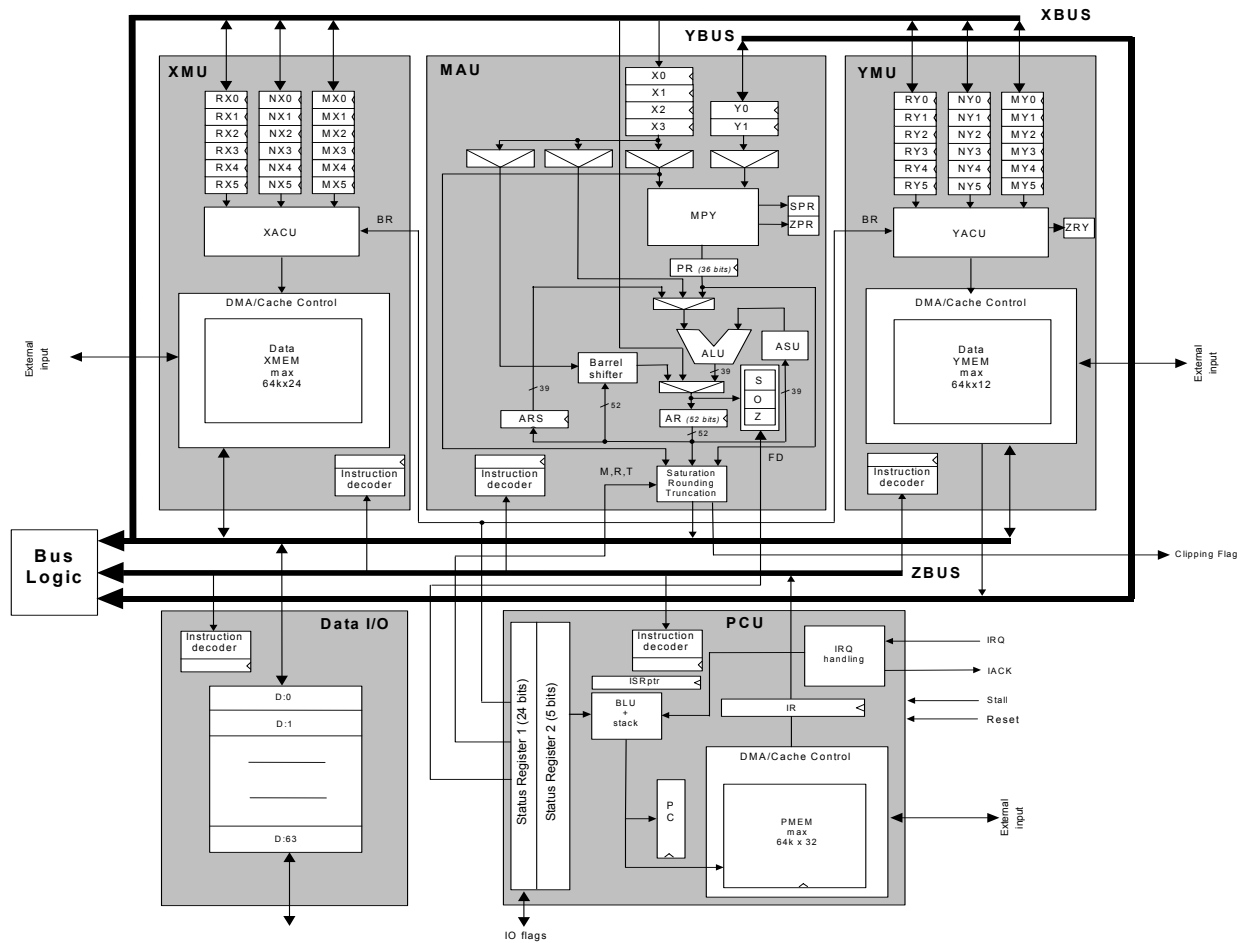


Figure 1, The Epics7B architecture

2. THE EPICS7B ARCHITECTURE

Philips Semiconductors has a long and rich history in embedded DSP, but which has been flying below the radar of most people for a long time. Better-known examples are the spin offs from Philips Semiconductors like TriMedia and R.E.A.L. [1]. A big part and the basis of that history is the Epics DSP family which is being used widely in Philips Semiconductors CarDSP and Audio DSP (TV, PC and portable & home audio) products. The latest development in this family is the Epics7B DSP, which should find its way to first silicon in the first half of 2003.

2.1 The DSP core

The Epics7B is the successor of the Philips Semiconductors Epics7A DSP. This DSP is currently integrated into 13 products

in the before mentioned application areas. Several of these products contain more than 1 Epics7A DSP, their numbers ranging from 2 to 5. It is also being used in combination with an ARM or MIPS micro controller. In these products the DSP mostly performs algorithms like Sample Rate Conversion, Stereo Decoding, RDS decoding, Dolby Pro Logic decoding and a lot of acoustical functions like stereo widening, bass boost and 3D virtualization.

For these kind of products more and more requests were made to execute more complex functions like for example MP3 decoding, Dolby Digital decoding and Acoustic Echo Cancellation. The main feature that is lacking from the Epics7A architecture is an interrupt in order to be able to do block based processing rather than sample based processing. Not using an interrupt is also possible, but this will involve unacceptable cycle overhead to be able to meet the real time requirements of the application.

The need for the interrupt functionality was the main reason for the design of the Epics7B DSP core.

The block diagram of the Epics7B architecture is given in Figure 1. The Epics7B has a double Harvard architecture with a 24 bits wide data bus and a 12 bits wide coefficient bus. The processing heart of the DSP is formed by the Multiply Accumulate Unit (MAU) which contains a 24x12-bit multiplier and a 39-bit ALU. The multiplier takes its input from one of the 4 data registers (X0, X1, X2 or X3) and from one of the 2 coefficient registers (Y0 or Y1). The multiplier result is always stored in the product register PR (36 bits wide). The ALU takes its operands on one side from the AR register, that can first be scaled through the ASU unit, and on the other side from a choice between the XR, PR or the ARS (Accumulator Save) registers. The result is stored in the upper 39 bits of the accumulator register (AR). The AR register itself is 52 bits wide and subdivided into 3 parts that can each be loaded from the data memory separately: ARL24 (24 LSB bits), ARH (next 24 bits) and ARO (4 MSB bits). A barrel shifter which is located next to the ALU is able to take all 52 bits of the AR register as input and perform a shift to the left or right (arithmetic) using one of the XR registers as input for determination of the shift distance and shift direction.

The Epics7B DSP supports data (X) and coefficient (Y) memories up to 64k words each. Each memory can be addressed through its own dedicated address calculation unit (ACU). Both memory control units (XMU/YMU) contain 6 address pointers, 6 modulo registers and 6 step registers. The ACU is able to perform circular buffering in the memories using the pointers in combination with the modulo registers. The modulo protection is active on the pointers whenever an update of a pointer register is executed, either directly (+1, +2 or -1) or through offset addressing using the step registers.

The program flow is controlled through the Program Control Unit (PCU), that takes care of the instruction fetching, branches, subroutines, interrupt servicing and setting and clearing of the user flags and modes. The PCU can handle 64k words of program memory. The architecture shows a three-stage pipeline existing out of fetching, decoding and execution stages. All instructions in the Epics7B instruction set are single cycle execution. Only the program jumps show a different behavior, as they are delayed branches.

The Epics7B has one interrupt line, but is able to serve several interrupt sources through a special mechanism which is implemented in the DSP sub system. All sources can be enabled and disabled (masked) individually.

The Epics7B core does not have a hardware looping system, but it does have special features to implement low overhead software loops. For this the ZRY flag can be used. The ZRY flag is the zero flag for the Y memory address pointers. Whenever one of the Y memory address pointers becomes zero after a post update, this flag will be set to one. It is assumed that one of the pointers will be initialized with the loop count value at the start of the loop. The counter is then decrement by 1 each time the loop is executed. This information can then be used to determine if the end of a loop has been reached,

2.2 The DSP subsystem

The Epics7B DSP core described in the previous section has a fixed design. All application specific aspects of the DSP design, like the memories, the control registers, the data I/O interface registers etc. are located in the DSP subsystem.

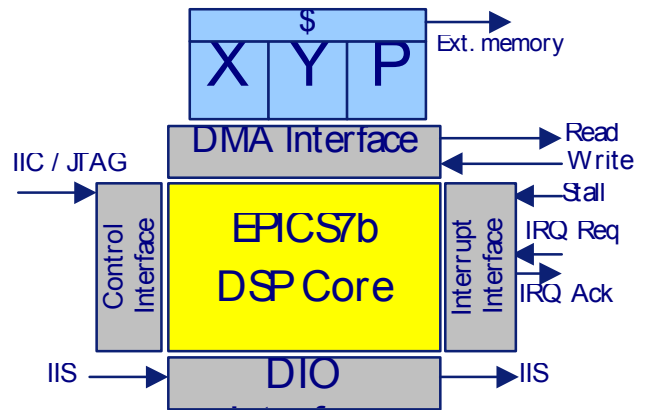


Figure 2, The DSP Sub System

Figure 2 shows an example of the DSP sub system. The application specific blocks on this level of the design are amongst others the memories with the caches and the corresponding slave DMA interface. The DMA interface arbitrates between the core and any external block that wants to access the X, Y or P memories. The interrupt defines the number and features of interrupt sources that are supported. Next to that this block translates the several interrupts to the single interrupt line and the user flags of the DSP core. In the DIO interface the data IO connections are made, which are usually used for direct data communication to the external peripherals like IIS of the IC.

2.3 The Epics7B tile and the sea of DSP

As mentioned in the introduction, it should be easy to add or remove DSP cores to/from the system architecture depending on the criteria (performance/cost/power) that are set for the specific application. This requires that the DSP cores interface to each other and to other blocks in a standard way. For this reason the concept of Epics7B tiles has been developed. This provides the integrator of a system with DSP tiles that use standard communication channels that can be connected to other DSP tiles, peripherals or a micro controller. An example of such a tile is given in Figure 3.

The power of the sea of DSP concept is based on multiple tiles each running on their own clock frequency, combined with standard communication channels between them, also known as Inter-Tile communication channels (ITC-channels).

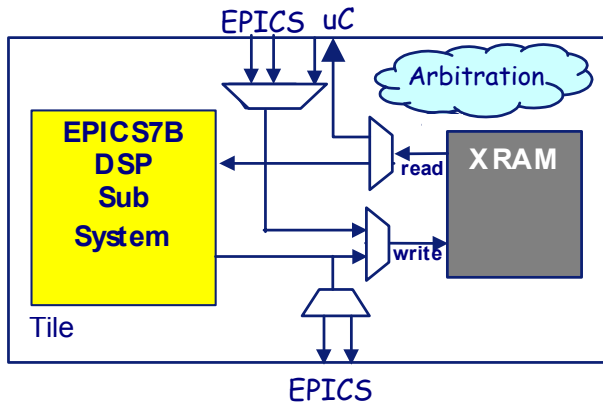


Figure 3, Epics7B tile

Through the ports at the tile boundary, communication between the different tiles and the on-chip micro controller can be established. To keep the total system simple and to limit the communication load on the communication channels, it is assumed that each Epics7B DSP core will execute complete algorithms and that only audio or control data that does not require a high communication bandwidth, is transferred between the tiles. It is currently assumed that algorithms are not split up over several cores, unless the communication between the sub blocks of an algorithm is not too demanding for the busload.

To keep the sea of DSP system flexible and high performance, the Inter-Tile communication channels are designed such, that they provide an optimal communication protocol. The latency of the data transfers is fixed and determined by the communication channels itself (like FIFO depth and clock domain crossing). To avoid that sources are blocked due to access collisions to the same memory for long periods of time, a kind of Round Robin arbitration is applied to the DSP cores, the uC-DMA channel and the Inter-Tile communication channels. With this scheduling approach the minimum available bus bandwidth for each processor can be determined and guaranteed. This arbitration protocol can be fine-tuned separately for each application in order to optimize the system for the given constraints, which may be different for each system. Examples of such constraints are latency, data throughput and required channel bandwidth. To check a system based on multiple tiles a performance visualization tool can be used to verify e.g. the ITC channel load and memory access.

Another way of limiting the complexity of the system is to allow communication from one Epics7B tile to a maximum of 4 other Epics7B tiles. If communication has to take place between tiles that are not tightly coupled, the data will have to be transferred either via one of the tightly coupled tiles or a specific tile interconnect network has to come in place.

2.4 Tooling

The tool chain for the Epics7B architecture has been developed by the company Target Compiler Technologies in Belgium and features the following tools [2]:

- C-compiler;
- assembler/disassembler;
- linker;
- instruction set simulator.

The high efficiency of the C-compiler allows for a good time to market performance of the software development.

The way the instruction set simulator API interfaces to the outside world also allows to use the graphical user interface for silicon debugging. Furthermore it allows for the definition of a multi core application that can be simulated using several simulators, communicating to each other as defined by the Epics7B tile concept.

3. THE FUTURE

Currently the system described in the previous section is being applied to systems containing up to 5 or 6 DSP cores. These systems can however be characterized by a clear-cut software partitioning due to the fact that each DSP core contains the software for a clearly marked sub application. For example, in a car application, where the processing of the radio functions is grouped together and embedded on their specific DSPs and the other DSPs are used for the audio post processing and sample rate conversion. Within Philips Research it is being investigated how this concept can be extended to a complete network of Epics7B processors that could be used as a big pool of computing resources. Investigations also include how algorithms can then be mapped in a very tool automated way to this resource pool, either through partitioning per algorithm or even through partitioning of algorithms themselves distributed over several cores.

An example of a future sea of DSP could then look as given in Figure 1.

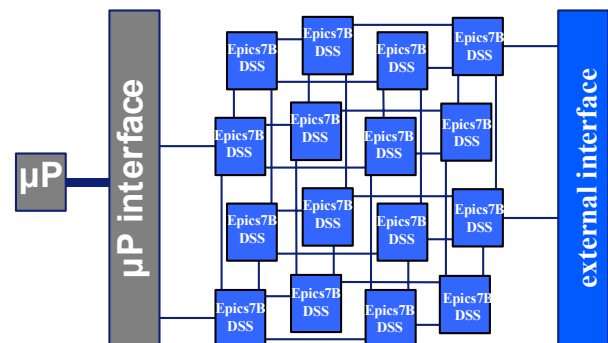


Figure 4, A sea of Epics7B system

4. REFERENCES

- [1] K. Moerman, R.E.A.L. DSP: Reconfigurable Embedded DSP Architecture for Low-Power/Low-Cost Applications in Proceedings of ICSPAT (San Diego, CA, Sept 1997), 814-818.
- [2] Target Compiler Technologies, "Chess/Checkers: a retargetable tool-suite for embedded processors", technical white paper, <http://www.target.com/doc/whitepaper.pdf>, January 2002