EMBEDDED DSP TECHNOLOGIES IN CONSUMER APPLICATIONS

CLASS NOTES
DSP WORLD WORKSHOPS
SEPTEMBER 13-16 1998
TORONTO

C.M. Moerman, R. Woudsma, P. Kievits
Philips Semiconductors ASIC Service Group, Eindhoven, The Netherlands
P.O. Box 218, 5600 MD Eindhoven, The Netherlands

Introduction

With the advent of advanced ASIC technologies, a number of mainstream IC products-traditionally using analog circuit techniques-are now being transferred to the digital domain. This transfer of techniques yields increased circuit robustness, cost-efficiency, system integration, and flexibility. Such state-of-the-art ASIC methodologies are also enabling a continuous stream of completely new consumer applications, increasingly based on embedded (re)programmable processor cores like DSPs and microcontrollers.

This overview will focus on the use of embedded DSP technology in high-volume consumer applications. The design challenge for such highly integrated systems is to combine high-functional performance, low-power operation, and low-cost implementation. In this context, a highly flexible and reconfigurable embedded DSP technology is described in an ASIC context, where reuse and short development times are crucial design constraints. An overview of a family of embedded DSP architectures will be given, as well as the techniques applied to combine DSP and other system components in a single chip. A number of concrete IC product examples will be used to illustrate this approach in the application areas of digital audio and telecom terminals.

BACKGROUND

Within Philips Semiconductors, the Embedded Systems Technology Centre (ESTC) of the ASIC Service Group (ASG) has the responsibility to develop and supply reusable microcontroller and DSP cores for high-volume embedded applications. As such, ESTC has developed several generations of Philips-internal embedded DSP cores (EPICS[1], ***R.E.A.L.***[2]) for the audio and telecom application market, and is also tracking comparable cores from other manufacturers.

The Philips Semiconductors embedded DSP cores mentioned as examples in this article are intended as Philips in-house DSP cores, and are initially not directly available to external customers.

Embedded versus Stand-alone DSPs

A DSP *core* is characterised by being 'embedded' in a single-chip environment, e.g. a system-on-a-chip. Embedded cores are typically used within high-volume consumer applications. Stand-alone DSPs often can not serve these application areas due to several constraints, such as small component count, low power consumption, small product size (miniaturisation) or low product cost.

As an example, consider the Philips Semiconductors 'ABC chip' (PCD509x) [BO'96]. This chip is a single-chip baseband processor for Digital Enhanced Cordless Telephone (DECT). The chip, a highly integrated mixed analog/digital baseband processor, is called ABC for the integration of three major circuits: **A**DPCM codec, **B**urst mode control

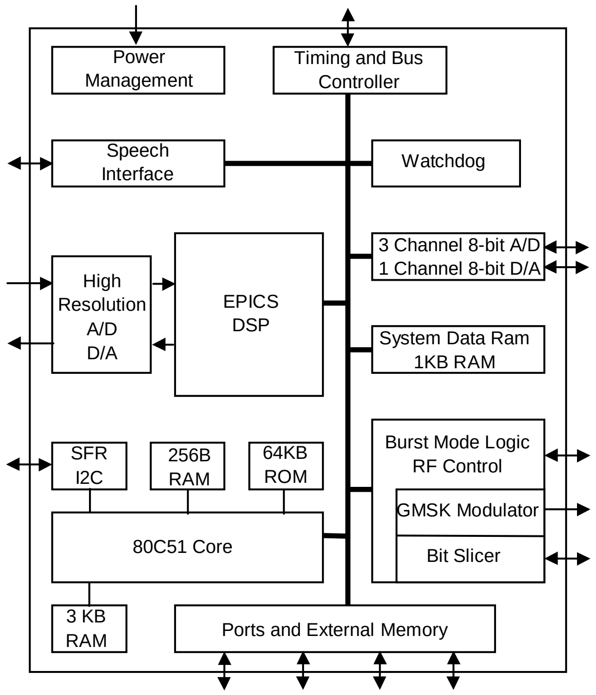

Figure 1. ABC Chip Architecture

[1] EPICS DSP: **E**conomic **P**arameterized **I**ntegrated DSP **C**ore**s**

[2] ***R.E.A.L.*** DSP: **R**econfigurable **E**mbedded DSP **A**rchitecture at **L**ow-power/**L**ow-cost.

and micro-**C**ontroller functions. Digital speech processing is performed on a customized EPICS DSP Core. Besides the embedded DSP, embedded micro-controller, and burst mode logic, the ABC chip also includes application specific peripherals (speech interface etc.) and has a variety of embedded memories (ROM, RAM). The chip represents the complete baseband functionality for both the DECT handset and basestation (see figure 1 on previous page).

From an application point of view, using embedded DSP cores has several advantages, the high integration level generally being the most important one[3]. This makes these cores different from stand-alone DSPs, which in most applications require extra chips (memory, glue logic, application-specific I/O devices). The choice for embedded DSP not only brings advantages (smaller chip count and thus greater reliability, higher bandwidth between components, lower power consumption) but also disadvantages due to the embedding (low observability w.r.t. testing).

In using embedded DSP cores, there are several design choices. One of the main selections to be made is the 'class' of embedded DSP:

- Using a 'fixed' core, often a stripped version of a standard, stand-alone DSP,
- Using a flexible reprogrammable core ('flexible' core) and tuning it to the specific application,
- Or using a hard-wired implementation given a fixed DSP algorithm.

All three options are viable depending on the specific application areas, and all involve different trade-offs regarding development time, tool availability, final product price (chip area), etc. (see Figure 2).

Fixed cores based on standard DSPs (such as can be licensed from e.g. Analog Devices) have the advantage of allowing prototyping on their stand-alone counterparts. This reduces risk and time to market. However, for certain applications, this may be inefficient w.r.t. to for example specific algorithmic requirements.

At the other end of the spectrum are the ASIPs, Application-Specific Instruction set Processors (non- or weakly reprogrammable architectures optimised for specific application tasks), and the hardwired architectures generated from e.g. signal flow graph descriptions (Mistral, DSP Station).

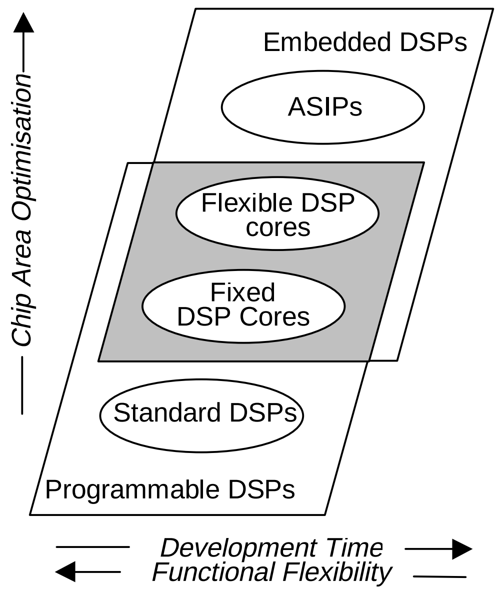

Figure 2. DSP Core Classification

[3] Note that integration does not automatically imply cost reduction, integrating memory and logic on one die may increase costs due to extra processing steps required (see [DI'98])

While being very well suited for applications with high bandwidth requirements such as video processing, they have the disadvantage of being more or less rigid w.r.t. modifications to the algorithms to be executed. This makes it difficult to follow changing standard specifications or adaptation to functional changes.

In between these two extremes are flexible DSP cores, which have the advantages of fixed DSP cores in that they have a well-known architecture and instruction set (also important for software development tools) and are fully reprogrammable, but which allow this instruction set and other architectural issues to be tuned towards the specific application area.

This approach yields a high functional performance without resulting in high clock speeds and associated high power consumption, and also without reducing reprogrammability and flexibility at least within the original target application area.

Standard DSP cores as Embedded Cores

Many DSPs available as standard stand-alone products are also available as embedded cores. In this case, two approaches can be seen: captive versus licensed cores.

In the case of licensed cores, the user of the DSP core licenses the right to use the core, and is (depending on the kind of license agreement) supplied with a model of the DSP core. This can be either a 'hard', 'firm' or 'soft' model. Hard models are fixed layout blocks, with no possibility to tune towards the final application area. Firm models are for example netlist based: the model can not be changed but is flexible w.r.t. placement and routing, and also to testing and clock strategy. Soft models are e.g. VHDL or Verilog models at RT-level. The designer is free to synthesize and implement this core according to its own requirements. The other end of the spectrum is an instruction set and/or architecture definition, with freedom for alterations towards the application. Examples of licensable cores are the DSP Group cores (Pine, Oak), Analog Devices (ADSP21xx), Tensleep and Clarkspur (CD2400/50) cores. Also, some companies have licensable versions of some TI cores.

In the case of manufacturers of captive cores, information is supplied the core regarding integration aspects etc., but the final fabrication of the chip will be done by the manufacturer. Examples of this approach are Lucent (DSP16xx) and Texas Instruments (TMS320C2x, C5x).

Flexible Cores: Application-specific Tuning

One of the major advantages of embedded cores is that they can be tuned towards the specific application area. A first step in this direction is to add application-specific I/O units and memory configuration. Of course, this would also be possible in case of standard DSPs. However, due to size restrictions, the allowed number of additional devices for making such extensions is often limited. Also, limitations in bandwidth, pin limitations etc. make this impractical.

As mentioned before, further customisation is possible for embedded DSPs. Where standard DSPs have fixed ALUs and instruction sets, due to limitations as the number of pins or the external interface speed required to add such extensions, embedded DSPs circumvent these problems by staying ‘on-chip’ completely.

This gives the opportunity to add tightly integrated 'application-specific execution units' (AXUs) as they are called within Philips Semiconductors. Also, instruction set optimisation with respect to the use of the available parallelism in the data path can be done per application area. In this way, DSP cores can fulfil the constraints of high-volume consumer DSP products like low cost, short time-to-market, and flexibility in functions and features.

As an example, the earlier mentioned EPICS DSP as used in the ABC chip [BO’96] is a flexible DSP core, which allows a high degree of customisation. In comparison to fixed cores, EPICS DSP cores can be customised in terms of data word lengths (12 .. 24 bits), addressing modes, and to a certain extend instruction set, while preserving full reprogrammability.

The proper balance between software and hardware is maintained through the use of application-specific ALU instructions. These special instructions, combined with the AXU concept, are used to facilitate the implementation of algorithms that do not map efficiently on standard DSP instruction sets. The addition of specific ALU instructions resulted in a significant reduction of the required execution cycles. For example, the complex G.726 algorithm has been coded into 488 cycles instead of 1,600 cycles on the unmodified architecture template. As a result, the operating frequency of the DSP in the DECT handset could be decreased which significantly reduces the power consumption. For the basestation, the freed cycles have been used for additional features.

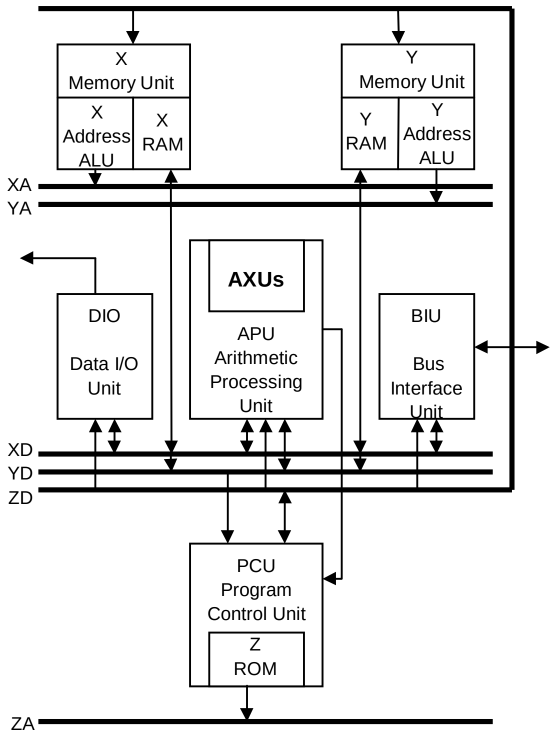

Figure 3. EPICS DSP Architecture Template

The customised EPICS product version for the ABC chip has been derived from a reconfigurable and parameterized VHDL architecture model. The architecture is configured and the parameters are set before VHDL synthesis is performed. This technology independent design flow allows a quick path to the implementation of the DSP core in new CMOS fabrication technologies. The application specific ALU instructions can be easily added to the standard instruction set by using this VHDL based technique.

TEST AND DEBUG

In general, for both fixed and flexible embedded DSP cores the software development tools are comparable to those found for standard DSPs, although due to the smaller user community for flexible DSPs, advanced features such as graphical user interfaces etc. are often not state of the art. However, due to the nature of embedded DSPs there are a few areas requiring extra attention.

Due to the embedded nature of the DSP, observability and controllability is weak compared to stand-alone DSPs. For example, memory buses are often not accessible externally, disabling the possibility to attach a logic analyser or program tracer in case of real time problems. Also the fact that the embedded DSP can have its program in ROM memory instead of RAM complicates the process of debugging in case an error is detected in a product version.

Therefore, extra features are needed for full DSP debugging and testing functionality even in deeply embedded (and pre-programmed) form. Due to pinout and timing constraints, debugging is not done using pods or ICE connectors, but debugging hardware is to be included in the final product, enabling on-chip debugging (e.g. Motorola’s OnCE approach or enhanced JTAG blocks). For example, break points can not be implemented via external hardware because program buses are not observable on external pins. Also, real-time tracing is not possible for this same reason, requiring solutions as local trace buffers connected directly to the core on silicon.

Additional complexity is added in case the system-on-a-chip has more than one processor core, as a combination of a DSP and a microcontroller core [ED’98]. Not only for debugging but also for production testing, special techniques are required due to the limited observability and controllability of internal nodes. Therefore, special core test techniques as the IEEE P1500 proposal [ZO’97] are needed for a more structural testing approach, by defining a hierarchical test structure based on the JTAG approach. This structure can then be re-used by the debugging tools, which need to take into account the possibility to ‘multiplex’ debugging streams, each intended for an individual core. This extends the problem of IP-combination from the hardware integration to the software development tools.

In case of flexible DSP cores with the ability to add application-specific extensions, early real-time emulation can be a problem due to the possible lack in methods to interface or emulate such blocks to the core in the development stages. Simulation can help in this situation, but will not obtain real-time speed. Especially for audio application, real-time testing may be required since artifacts of algorithms are often difficult to measure and actual listening tests are required. Other examples of applications requiring real-time testing include speech recognition and acoustic echo cancellation, which are sensitive to varying environmental conditions and background noise. In this case, solutions may be offered in the form of a bond-out version of the core, to which the customer may add logic by downloading this into FPGA or EPLD devices.

CURRENT TRENDS

Due to the increasing level of integration, which is enabled by the new submicron processes, DSPs will become more deeply embedded. At the same time, applications are merging; as PDA functions combined with phone functionality. Although cost is a dominant design constraint, the overhead (in execution cycles and code size) of high-level entry by means of C-compilation is becoming a necessity rather than luxury, both due to the growing complexity as to the shrinking Time-To-Market requirements.

Classical rule states that 10% of the code is executed 90% of the time (static vs. dynamic properties of program code). Complex, non-execution-time-critical parts of the program can be coded in C, whereas the time-critical portions (typically inner loops in for example FFT processing) should still be written in assembler, or be available as a library function, which can then be called from C-level. Still, significant overhead is imposed by the C compilation.

Also, due to the higher integration level, the functionality of the DSP programs is changing. At this moment, classic DSP functions, characterised by sum-of-products calculations, are still the major tasks of most embedded DSPs. A trend is visible towards more and more irregular, non-arithmetic functions such as bit stream manipulation (as needed to decode MPEG or AC-3 audio streams), and error detection and recovery. This has impact on the architecture of DSP systems, which are normally optimised to handle steady data streams by means of deep pipelining. Handling these irregular spaghetti-like algorithms often will break the pipe, at the result of taking extra cycles to refill the pipelines.

So, the request for performance is increasing rapidly. And although new submicron technologies still offer opportunities for increasing the clock frequency, this is not always a feasible solution due to limitations in power consumption or clock frequency. DSP manufacturers are thus looking into solutions to increase the amount of work being done per clock cycle so one of the most visible current trends in DSPs is the increase of parallelism.

Four major forms of parallelism can be distinguished:

· *Deeper pipelining*, in which case more instructions are executed in a concurrent sequence, each instruction being split into multiple stages. As an effect, the clock cycle can be reduced allowing the chip to run at higher frequencies. Alternatively, the supply voltage may be decreased at a limited clock frequency, yielding improved power efficiency. Although maybe the easiest way to increase performance, the negative effect is the visibility of these pipelines for the programmer e.g. w.r.t. branch penalty (pipeline refills) and scheduling problems in case of memory access latency. For example, in the new TI C62xx a memory access has a latency of 5 cycles. Although new accesses may be started each cycle, the programmer has to keep track of outstanding memory accesses.

· *Instruction-level parallelism*, by supplying parallel data paths driven by the same instruction sequencer. An example is the addition of a second multiplier-accumulator. The advantage is that more work can be done without increasing the clock frequency, thereby keeping power consumption within limits.

Examples of this approach can be found in the Lucent DSP16210 series, the Philips ***R.E.A.L.*** DSP (both dual-multiplier architectures, the last having 4 ALUs to speed up operations as Viterbi decoding) and the TI C62xx and Philips TriMedia, both having multiple (n>2) execution units. Major problem in this approach is to keep all data path elements busy. This is often difficult due to memory bottlenecks, data dependencies and intrinsic limitations in the parallelism found in the application. For example, a dual multiplier architecture does not give a factor of 2 overall performance increase, but will stay typically at a factor of 1.7 (***R.E.A.L.*** DSP), depending on the algorithm. Adding more multipliers will gradually reduce the performance increase depending on the application.

- *Data-level parallelism*, in which multiple data values are packed into single data words, as 32-bit data words interpreted as containing 2 16-bit data values. Examples of this approach can be found in the MMX extensions of Intel, and in the TI C62xx. However, this does require clever data packing, not possible (without major overhead in packing) in all application areas.
- *Task-level parallelism*, in which multiple processors are used to execute different part of the algorithm. This can either be done using a homogeneous approach (same processor type) or heterogeneous approach (dedicated processor types for specific tasks). For a homogeneous example, a recent product of Philips Semiconductors is the DUET AC3/MPEG multichannel audio decoder, which features a dual EPICS core. By using the dual core approach, clock frequency can be limited to 30..40 MHz depending on required features. A heterogeneous example can be found in the ARM/Piccolo microcontroller/DSP combination.

The advantage of increasing the parallelism without increasing the clock frequency, as is possible with the instruction level and data level approaches, is that also the power consumption increase is limited. This increase is linear with clock frequency, but less than linear with this increase in parallelism. This is due to the fact only parts of the core are duplicated (e.g. instruction fetch and decode is not duplicated). Also, inactive parts can be switched off rather easily on a per cycle basis using clock gating and by switching off functional units.

R.E.A.L. DSP example within Philips Semiconductors

As examples of especially the instruction level parallelism, different processors take different approaches to control the large number of operations allowed. This ranges from compact instruction sets with preselected parallel instructions for specific functions (Lucent DSP16210) to full VLIW for complete control of the available parallelism (TI TMS320C62xx). As an intermediate solution, the ***R.E.A.L.*** DSP core of Philips Semiconductors has a compact standard instruction set with the possibility to add customer-selected VLIW extensions [MO'97], [KI'98]. A comparable approach can be found in the Siemens Carmel DSP architecture [PR'98].

The ***R.E.A.L.*** DSP is a dual Harvard architecture with 16-bit wide data buses, see figure 4. The heart of the DSP is a Data Computation Unit (DCU) which contains two 16x16-bit signed multipliers and two 40-bit ALUs (32 data bits, 8 overflow bits). Depending on what type of instruction is executed, the ALUs can also be used as four independent 16-bit ALUs. Four input registers are available to store values fetched from X and Y data memories. ALU results are stored in four 40-bit accumulators, which can also be used as eight 16-bit registers. To provide good performance, the results of the multipliers are stored in pipeline registers, explicitly visible to the programmer. Shift and saturation units are available at a number of locations in the datapath. A stripped version of the ***R.E.A.L.*** DSP with only one multiplier and single 40-bit/dual 16-bit ALU has been derived from this architecture for low-cost applications.

A distinctive feature of the ***R.E.A.L.*** DSP (taken from the EPICS) is the possibility to add Application-specific eXecution Units (AXU) to the datapath. A part of the instruction set has been reserved to control such blocks. Among the AXUs already available are a 40-bit barrel shifter, a normalisation unit, and a division support unit.

The complex data path is controlled by the PCU (Program Control Unit). This unit fetches and decodes instructions from memory, and handles program flow instructions. These are performed with a minimum loss of cycles and include subroutine calls and relative, absolute, and calculated jumps. A loop controller based on an internal stack, with a depth specified by the customer, controls nested hardware looping. The contents of the loop stack (loop count, start and end address) can be saved to memory if an interrupt occurs.

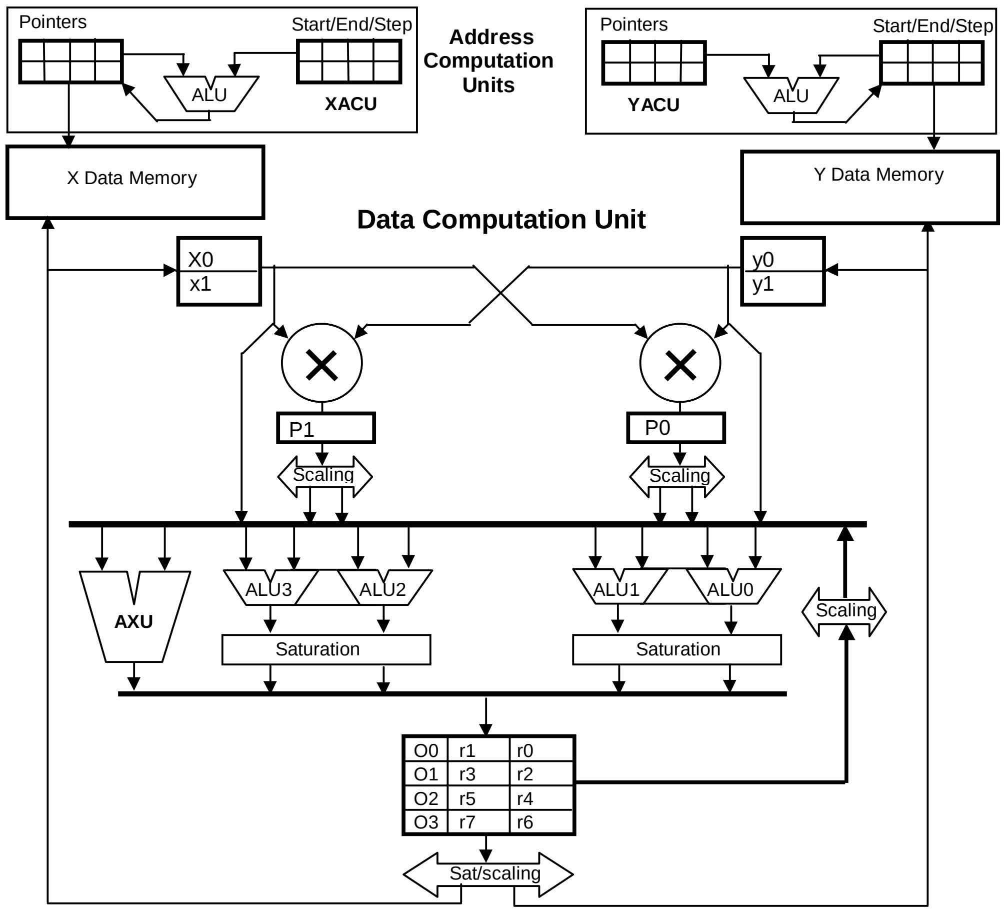

Figure 4. R.E.A.L. DSP Datapath Architecture Template

Due to the high level of parallelism, it is no longer feasible to control the data path using a 16-bit instruction. Within 16 bits, there is simply no space to encode all operands and operations to be performed in an 1-cycle instruction. To select 4 operations each having 2 input operands and one result register, at least approximately 4*(4+2*3+3) = 52 bits are needed, and additional bits will be needed for parallel memory fetch/store operations, multipler control and other functions. Therefore, to control this level of parallelism, a special area in the instruction set is reserved. This area can be used by the application programmer to define ASIs, ‘Application-Specific Instructions’, a VLIW-type of instructions stored in an internal lookup table and driven by the normal 16-bit instruction set (explained in more detail in [KI’98]).

To give an example, we can try to implement a FIR filter taking advantage of the dual multiplier structure. The main difficulty when implementing the block FIR filter using two multipliers is the memory bottleneck: each cycle only one sample and one coefficient can be fetched from memory. A straightforward FIR filter can thus only calculate one product term each cycle.

Different vendors have made different decisions how to circumvent this bottleneck. For example, in the Lucent DSP16210 the bus width to the data memory is 32 bits, so 2 16-bit values can be fetched in a single memory access. Due to certain constraints, this approach was not chosen for the ***R.E.A.L.*** DSP.

However, when doing block processing, two subsequent output samples can be calculated simultaneously, thereby reusing data by keeping the previous input value in a register. In that case, the two terms in the below formulas can be calculated at the same time, including the update of registers to set up for the next iteration.

```
$Out_n$ += $In_n$ * $Coeff_n$, $Out_{n+1}$ += $In_{n+1}$ * $Coeff_n$,  # calc terms
$In_n$= $In_{n+1}$, $In_{n+1}$=XMEM[n+2],            # shift data values
$Coeff_n$=YMEM[n+1];                          # next coeff
```

Formula: FIR filter algorithm (pseudo-code) calculating two output samples

A filter kernel would contain two single-cycle instructions as in the example, each calculating one term of sample [n] and one of sample [n+1]. Being too complex to code in the standard instruction set, the ‘asi’ keyword indicates a VLIW-like instruction. By switching roles for the x0 and x1 register with regard to being the 'old' or 'new' value, no data move is required to shift the sample as is shown in the below example:

```
do N/2
{
  asi a1=a1+p1, a0=a0+p0, p1=x0*y0, p0=x1*y0, x1=*px0++, y0=*py0++;
  asi a1=a1+p1, a0=a0+p0, p1=x1*y0, p0=x0*y0, x0=*px0++, y0=*py0++;
}
```

Example: FIR filter algorithm (code for ***R.E.A.L.*** DSP)

This way, the memory can keep up with the multiply-accumulate section. A 64-taps FIR filter with block size N implemented this way takes 8+37*N cycles to execute, i.e. a 40-sample block takes approx. 37 cycles per sample, including overhead. A single-multiplier DSP would take at least 64 cycles, excluding any overhead. For many algorithms, block-based processing is feasible, and the ***R.E.A.L.*** DSP architecture halves execution time for such algorithms.

Another example of a block-based algorithm is the FFT transform. Again the memory bandwidth is the bottleneck, and attention has to be paid to the effective use of the internal registers. However, due to the large register set (four 40-bit accumulators, each of which also can be used as two independent 16-bit accumulators), a radix-4 FFT implementation can make full use of the dual multiplier architecture. This reduces the time needed for a 1,024-points complex FFT to around 1,400 cycles.

To compare, the new TI TMS320C62xx processor takes approx. the same number of cycles for the same task. However, the ***R.E.A.L.*** DSP does not have the overhead of the VLIW program memory requirements of the TI C62xx DSP, or the resulting energy consumption. The ***R.E.A.L.*** DSP is optimized for low-cost consumer products requiring an embedded DSP, trading in full VLIW flexibility for a compact die and low-power operation. Of course this trade-off has also negative aspects, as in reduced instruction set orthogonality (also hindering the C compiler) and usability of the full parallelism only for specific time-critical application parts.

Based on this ***R.E.A.L.*** core concept, Philips has created a family concept dividing the ***R.E.A.L.*** family in four quadrants. At one hand, there is the distinction between cost-sensitive replacement markets versus the performance-critical new application markets. At the other hand there is a distinction between the telecom market, in which 16 bits is sufficient for the required signal to noise ration, and the high-end audio market where at least 20 bits is required for sufficient dynamic range.

This division is reflected by having a single multiplier version (for the low-cost market) and a parallel dual multiplier version (high-performance), both available as a 16-bit data width version and a planned 24-bit data width version. Having the same concept for these four areas results in a high reusability within the various application domains within Philips Semiconductors.

CONCLUSIONS

Embedded DSP-based consumer applications are characterised by high cost sensitivity, and consequently, DSP cores are designed taking this constraint into account. Cost minimisation is visible in the size of the DSP core, number and level of peripherals, and -more and more importantly- sizes of data and program memories. In particular, program code size as visible in (on-chip) program memory is getting a significant design factor. Hence, instruction compaction is getting a lot of attention.

Complementary to the design constraint in terms of costs of DSP implementation, processor performance is a crucial design issue as well. Rather than pure clock speed or raw MIPS performance, it is functional performance in the context of the target application domain that counts eventually.

Hence, the art of embedded DSP design for high-volume, consumer-type of products, is to find the optimal balance between cost and performance constraints. This balance can only be found while carefully analysing applications and their characteristics. For portable devices, obviously, low-power operation is a crucial design factor as well.

Another important design factor is short Time-to-Market, in order to enable fast introduction of new applications in the market, to swiftly add new features or functions, and to accommodate new or changing standards. This constraint generates a high drive on the level of software development tools, which are more difficult to make effectively as opposed to general-purpose, stand-alone DSP tools, since an embedded DSP core must be analysed and validated in its, often deeply, embedded system context. Availability of embedded software libraries and/or tools to bridge from one DSP platform to the other strongly contributes to short Time-to-Market.

These design constraints, characteristic for embedded DSPs in consumer products, can only be met on the basis of a flexible embedded DSP technology, which can be configured and tuned to a given application domain within the scope of a family concept, yielding high functionality at effective MIPS performance, and a cost-effective VLSI implementation. In addition, the availability of a powerful, complete, and user-friendly software development platform is indispensable. Re-use of ASIC design modules and embedded software modules is essential to be successful in the vastly growing embedded DSP consumer market.

REFERENCES

[BO'96] *The ABC Chip: Single Chip DECT Baseband Controller based on EPICS DSP Core*, A.J.P. Bogers, M.V. Arends, R.H.J. De Haas, R.A.M. Beltman, R. Woudsma, D. Wettstein, ICSPAT 1996 Proceedings

[MO'97] *R.E.A.L. DSP: Reconfigurable Embedded DSP Architecture for Low-Power / Low-Cost Applications*, K. Moerman, P. Kievits, E. Lambers, J. Walkier, R. Woudsma, ICSPAT 1997 Proceedings

[ZO'97] *Test Requirement for Embedded Core-based Systems and IEEE P1500*, Yervant Zorian, Proceedings of the 1997 International Test Conference, IEEE

[DI'98] *Embedded Memory, the All-purpose Core*, Brian Dipert, EDN March 13, 1998 (Background article on embedded systems from implementation technology point of view: it is not always cheaper to integrate)

[ED'98] Electronic Design, Jan 12, 1998: Thematic issue on IP (Intellectual Property), handling a number of general issues on embedding of system blocks such as DSP cores)

[KI'98] *R.E.A.L. DSP Technology for Telecom Baseband Processing*, P. Kievits, E. Lambers, C. Moerman, R. Woudsma, ICSPAT 1998 Proceedings

[PR'98] *DSP architecture focuses parallel instructions on key operations*, Graham Prophet, EDN June 4, 1998