

# Coarse-Grained Reconfigurable Computing for Power Aware Applications

Paul M. Heysters

## Recore Systems

P.O. Box 217, 7500 AE, Enschede, The Netherlands  
paul.heysters@recoresystems.com

**Abstract** – Reconfigurable architectures find the middle ground between flexibility and efficiency by limiting their flexibility to a particular algorithm domain. A domain specific reconfigurable architecture is both efficient and flexible. Recore Systems provides semiconductor IP solutions for programmable systems-on-chip. The coarse-grained reconfigurable technology of Recore promises to solve the flexibility, performance, power consumption and cost requirements of semiconductor businesses.

**Keywords** – DSP core, low power, system-on-chip, coarse-grained reconfigurable, Montium™

## I. INTRODUCTION

Recore Systems develops processor cores, programming tools and applications. The processor cores of Recore offer low power, high performance and flexibility. This is realized by using coarse-grained reconfigurable technology. Complex programmable systems-on-chip can be developed seamlessly by using the processor cores as Intellectual Property (IP) building blocks.

Wireless communication, multimedia and digital radio/TV applications require increasing processor power and are subject to ever changing standards. It becomes less and less attractive to develop Application Specific Integrated Circuits (ASICs) for these applications, as ASICs take a long time to develop and are very costly. Programmable architectures like Digital Signal Processors (DSPs) do not have enough processing power for these demanding applications and are extremely power hungry. Field Programmable Gate Arrays (FPGAs) often perform better, but are still too expensive and power hungry for many (portable) applications.

The challenges of the semiconductor industry can be summarized as follows:

- Exponentially increasing development costs due to shrinking of technology.
- Increasing design complexity puts great pressure on design costs and time-to-market, while at the same time product life cycles (i.e. volumes) become shorter.
- Very costly or impossible changes in inflexible ASIC designs.
- General purpose processors use too much energy and have insufficient computing power.

To counteract these problems, upcoming system-on-chip platforms will be built using interlocking pieces of IP: processing cores, peripherals, specialized I/O modules, operating systems and development tools. More complex systems can be built in a shorter timeframe by purchasing off-the-shelf IP blocks. In particular, this will increase the design productivity by 200% [7].

Conventional processing architectures, such as general purpose processors, DSPs, FPGAs and ASICs, cannot satisfy the combination of extremely low power consumption, high performance, flexibility and low costs. Yet, the upcoming need for processors that meet all these requirements is undeniable. Recore's reconfigurable computing technology combines the best of both worlds: high performance, programmability, low power, and a small footprint.

What exactly is reconfigurable computing? The term reconfigurable computing is used in many different contexts. This paper discusses the concepts and benefits of reconfigurable computing and more specifically Recore Systems' coarse-grained reconfigurable Montium technology.

## II. RECONFIGURABLE COMPUTING

Fully programmable architectures – like general-purpose processors – can be used to compute virtually any algorithm. Unfortunately, the overhead caused by this flexibility makes them very inefficient. Application specific architectures, on the other hand, are very efficient but offer little flexibility as they are not programmable by definition. The concession made by reconfigurable architectures is to limit the flexibility to a particular algorithm domain. The ambition of reconfigurable hardware is that the hardware adapts to the algorithm, instead of adapting the algorithm to the hardware.

### A. Domain specific

A key property of Digital Signal Processing (DSP) algorithms is that they exhibit high levels of both spatial and temporal concurrency. Spatial concurrency indicates that multiple identical computations occur in parallel, whereas temporal concurrency indicates that identical computations are repeated in time. A regular and repetitive computational part of a DSP algorithm that accounts for a large fraction of the execution time and the energy consumption is called a kernel. For example, the kernel of a finite impulse response (FIR) filter contains the multiply-accumulate (MAC) operation and the kernel of the fast Fourier transform (FFT) contains the FFT butterfly.

Algorithms belonging to the same algorithm domain have similar kernels and operate on similar data structures. For example, the butterfly computations within the FFT kernel resemble the MAC computations in the kernel of FIR filters and both kernels work on blocks or streams of samples. Algorithms within an algorithm domain can be implemented on similar hardware architectures, due to similarities between the algorithms of a particular domain. A domain specific architecture that is flexible enough to implement all algorithms within a particular domain can be highly optimized and requires only a modest amount of control overhead. That is, a domain specific architecture is both efficient and flexible within its algorithm domain.

### B. Reconfigurability

Before a domain specific processor can be used, it needs to be configured to implement a specific algorithm from its algorithm domain. In essence, reconfigurability is another word for the programmability of a reconfigurable processor.

The degree of programmability of a reconfigurable processor determines its flexibility.

Programmability enables supporting multiple standards (e.g., for wireless communication or multimedia) using the same hardware. When existing standards change or new standards are introduced, reconfigurable hardware does not necessarily become obsolete. Instead, it can be reconfigured to adapt to the revised or new standard.

Programmable processors also help application developers to focus on their core business rather than on designing integrated circuits. In fact, algorithms can be programmed directly on off-the-shelf reconfigurable processors. This can also reduce development costs, as developing Application Specific Integrated Circuits (ASICs) becomes increasingly expensive and is, for that reason, only lucrative for very large volumes. Besides, a company can protect its proprietary algorithms better, since it does not have to disclose implementation information to chip makers.

#### 1) Dynamic reconfiguration

Field Programmable Gate Arrays (FPGAs) were originally developed for prototyping and glue logic purposes. They needed to reconfigure at rates ranging from hours to weeks. Such infrequent reconfiguration is known as static reconfiguration. Reconfigurable hardware used in applications like mobile computing needs to be able to reconfigure at rates ranging from hours to milliseconds. Frequent reconfiguration is known as dynamic reconfiguration.

For dynamic reconfiguration it is important that a reconfigurable chip can change quickly from one configuration to another. Therefore, the amount of configuration data required to reconfigure a chip should be small. A technique to speed up the switching of configurations is to equip a reconfigurable chip with an active and a background configuration plane that can be swapped almost instantly. The penalty for the extra plane is the increased area overhead. For efficiency, the configuration time should always be amortized by the execution time of an algorithm.

In an application specific approach every function of a system requires its own implementation. An ASIC implementation of an individual function is very area and energy-efficient. In practice, many functions of a system are not used all the time. For example, some

functions are only invoked during initialization or upon user interaction. In other words, it is very well possible that, at a particular moment in time, parts of an ASIC are not involved in any computation. This waste of resources is neither area nor energy-efficient. Dynamic reconfiguration allows time-sharing of hardware resources by pipelining algorithms. A single domain specific reconfigurable architecture can replace a set of separate ASIC functions. Therefore, it is possible that the silicon area of a system incorporating reconfigurable processors is smaller than a system composed entirely of ASICs.

### C. Granularity

The granularity of a reconfigurable architecture is determined by the width of the components in its datapath. Here, an architecture is considered fine-grained when its datapath width is four bits or less. Otherwise, it is considered coarse-grained.

#### 1) Fine-grained

In fine-grained reconfigurable architectures the functionality of the hardware is specified at the bit-level and the programmable interconnect is manipulated as individual wires. The fine-grained flexibility comes at the cost of additional silicon and this overhead hampers the performance of word-level algorithms. Word-level operations like multiplications become relatively large (and slow) when implemented on fine-grained architectures. Fine-grained architectures are efficient for complex bit-oriented computations or bit-level masking and filtering. FPGAs such as the Altera Cyclone [2] and the Xilinx Virtex-E [15] are typical examples of fine-grained reconfigurable architectures.

#### 2) Coarse-grained

Coarse-grained reconfigurable architectures contain word-level function units, such as multipliers and Arithmetic Logic Units (ALUs). The programmable interconnect in coarse-grained architectures is manipulated as buses. Coarse-grained architectures require less configuration data than fine-grained architectures and therefore their configuration time is shorter. Fine-grained algorithms execute very inefficiently on coarse-grained architectures.

#### 3) Hybrid

A hybrid architecture comprising both fine and coarse-grained elements is also possible. Hybrid architectures can implement word-level algorithms much more efficiently than fine-grained architectures and they can implement bit-level algorithms much more efficiently than coarse-grained architectures. Yet, there is still an inherent overhead present in this organization of hardware due to the general-purpose nature of these hybrid architectures. Modern FPGA devices embed coarse-grained components into their fine-grained architecture. Examples of such devices are the Altera Stratix II [2] and the Xilinx Virtex-II Pro [15].

### D. Design tools

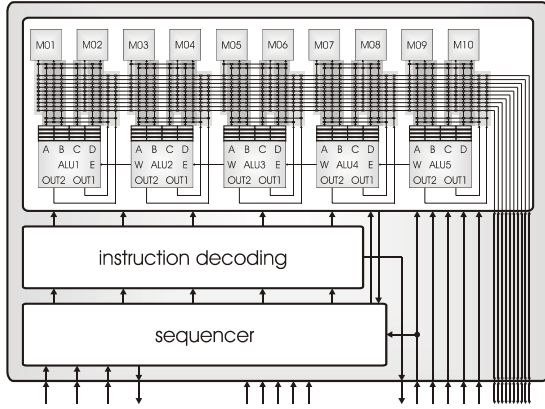
Good design tools are the most important requirement for the viability of coarse-grained reconfigurable architectures. Such tools reduce the design cycle (i.e. costs and time-to-market) of new applications, especially when compared to ASICs.

Hardware description languages (HDLs) and assemblers allow a programmer to exploit all parallelism in a reconfigurable architecture, but require him to understand the underlying hardware organization.

High-level design entry languages like C provide an abstraction from the underlying hardware organization, which reduces the design effort (i.e. costs and time). However, compilers are unable to extract the maximum amount of parallelism achievable on a reconfigurable target architecture from a sequential description. To overcome this problem, high-level languages are often extended with constructs for parallelism and scheduling. Unfortunately, these extensions introduce the need to have knowledge of the underlying architecture again.

## III. MONTIUM TECHNOLOGY

Recore's first coarse-grained reconfigurable processor core is called Montium. This core is based on technology developed at the University of Twente, The Netherlands [6]. Recore adopted the concepts of the proven Montium technology and redeveloped and reengineered the technology into products for chip developers. Besides a processing core, the Montium technology also constitutes other hardware modules, program development tools and application designs.



**Figure 1: Montium Tile Processor**

### A. Montium Processor Core

The Montium Tile Processor (TP) is a programmable architecture that obtains significant lower energy consumption than DSPs for fixed-point digital signal processing algorithms. The Montium TP targets computational intensive algorithm kernels that are dominant in both power consumption and execution time.

In contrast to a conventional DSP, the Montium TP does not have a fixed instruction set, but is configured with the functionality required by the algorithm at hand. In particular, the Montium TP does not have to fetch instructions and, hence, does not suffer from the Von Neumann bottleneck. Once configured, the Montium TP resembles more an ASIC than a DSP.

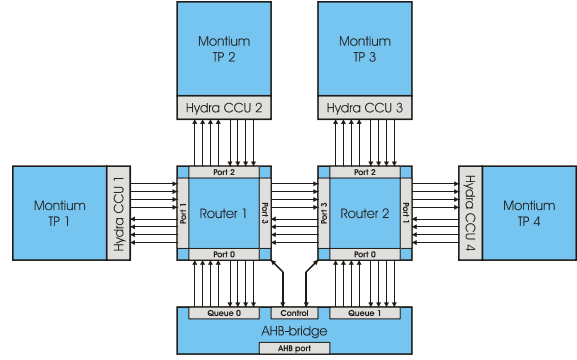
The Montium TP can be reconfigured virtually instantly, as the size of the configuration binaries is very small. The size of a typical configuration is less than 1 KB and reconfiguration typically takes less than 5  $\mu$ s.

The Montium TP has a low silicon cost, as the core is very small. For instance, the silicon area of a single Montium TP with 10 KB of embedded SRAM is 2 mm<sup>2</sup> in 0.13  $\mu$ m CMOS technology. The power consumption in this technology is less than 600  $\mu$ W/MHz (including all memory access).

The Montium TP is programmed using the proprietary Montium Configuration Description Language.

### B. Architecture

A diagram of the Montium TP is shown in Figure 1. The hardware organization is very regular. Five identical processing parts in a tile exploit spatial concurrency to enhance



**Figure 2: Example reconfigurable subsystem**

performance. This parallelism demands a very high memory bandwidth, which is obtained by having 10 local memories in parallel. The local memories are also motivated by the locality of reference principle, which is a guiding principle to obtain energy-efficiency.

The datapath width of a Montium core and the memory capacity are customizable at design time and depend on the computational requirements. The Arithmetic Logic Units (ALUs) support both signed integer and signed fixed-point arithmetic. Input registers provide the most local level of storage.

The five processing parts together are called the Processing Part Array (PPA). A relatively simple sequencer controls the entire PPA by selecting configurable PPA instructions. This control paradigm is very different from a VLIW architecture, despite a deceptive resemblance.

## IV. MULTIPROCESSOR SYSTEM

The Montium TP is typically used to perform digital signal processing tasks for a lightweight general purpose processor. The Montium is either used as a single accelerator core or in a heterogeneous multiprocessor system.

Figure 2 shows an example of a simple reconfigurable subsystem. Four Montium processing tiles are connected by a network-on-chip. A processing tile consists of a Tile Processor (TP) and a Communication and Configuration Unit (CCU). Each processing tile is connected to a router of the network-on-chip. There are two routers in the reconfigurable subsystem of Figure 2. Routers can either be packet or circuit switched. For this tiny network, a circuit switched router is used. Both routers are connected to the AHB bridge, which connects the reconfigurable subsystem to the general purpose processor and the rest of the system-on-chip.

### A. Hydra CCU

The Communication and Configuration Unit (CCU), called Hydra, implements the network interface controller between the network-on-chip and the Montium TP. The Hydra CCU provides configuration and communications services to the Montium TP. These services include:

- *Configuration* of the Montium TP and parts of the Hydra CCU itself
- *Frame based communication* to move data into or from the Montium TP memories and registers (using direct memory access)
- *Streaming communication* to stream data into and/or out of the Montium TP while computing

In streaming communication mode, data words (e.g. samples) are processed as they become available from the network. A streaming implementation of an algorithm has a higher throughput than its frame based alter ego. More information on the Hydra can be found in [5].

### B. Circuit Switched Router

A network-on-chip router is used to build a Montium TP reconfigurable subsystem (i.e. a multiprocessor). Routers are physically connected to CCUs and to other routers. Routers can also be used to create a heterogeneous system-on-chip comprising a variety of hardware modules.

The network router interface consists of two identical unidirectional physical channels. Each physical channel contains four lanes. Lanes contain data and network control signals.

In order to send data over the network, a processing tile has to send a network packet onto one of its output lanes. The packets sent by a source tile are received on an input lane of the destination tile. Up to four outgoing packets can be sent in parallel by a single tile, by using all four outgoing lanes.

### C. AHB Bridge

The Advanced High performance Bus (AHB) bridge connects the reconfigurable subsystem to embedded processors, high performance peripherals, DMA controllers, on-chip memory and I/O interfaces. The AHB protocol is part of the AMBA on-chip bus specification. AMBA is an open de facto standard for the interconnection

and management of functional blocks that make up a system-on-chip [3].

## V. MONTIUM DESIGN TOOLS

New (reconfigurable) processor cores are only viable when accompanied by solid program development tools. Such tools allow application engineers to implement new applications swiftly.

Recore develops program development tools in synergy with its hardware cores. The program development tools for the Montium core are called the Montium Sensation Suite. The Montium Sensation Suite comprises the Synsation Compiler, the Simsation Simulator and the Insation Editor.

The University of Twente, The Netherlands, also conducts research on high-level compilers for the Montium TP. Details on a near optimal column arrangement algorithm that reduces the number of configurations for each reconfigurable ALU is presented in [9]. This algorithm is useful for automatically scheduling relatively large applications on the Montium TP.

### A. Synsation Compiler

The Montium processor core is programmed in Recore's proprietary Montium Configuration Description Language (CDL). This language enables swift and structured application development. The Montium CDL programs are compiled using the Synsation Compiler. Part of this compiler is Recore's ALU mapping technology that instantly finds an efficient mapping of a complex computational expression on the Montium TP hardware.

The CDL language can be used by customers to develop their own applications. A library of CDL generator scripts is available for common DSP algorithms. These scripts instantly generate a Montium configuration, compliant with the algorithm parameters specified by the user.

#### 1) Programming example

Suppose the simple 5-tap FIR filter depicted in Figure 3 is to be implemented using CDL. The communication method is streaming, i.e. every clock cycle an input sample is consumed and a

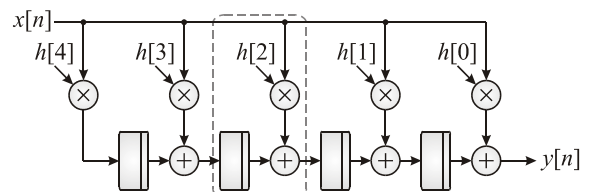


Figure 3: 5-tap FIR filter

```

// Multiply-accumulate operation
proc mac
  rep i <- 1 2 3 4 5
  alu (p.i.a1 fmul p.i.c1) sadd p.i.d1
  -> p.i.o1
end
end

// Write MAC results
proc moveresults
  rep i <- 1 2 3 4
  mov p.i.o1 -> p.(i+1).d1
end
mov data p5o1 -> ext1
end

main_loop:
mov ext1 -> p1a1 p2a1 p3a1 p4a1 p5a1
//load next sample in registers
call mac //compute multiply-accumulates
call moveresults //write results
jnc eof main_loop //end-of-frame test

```

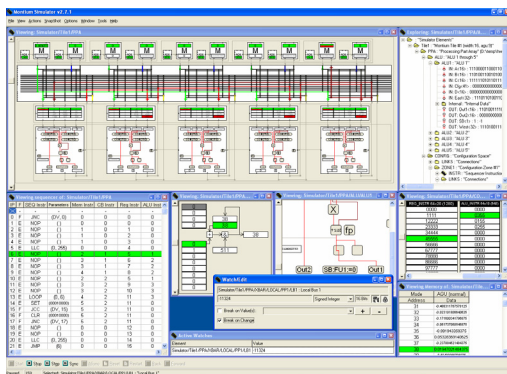
**Figure 4: FIR filter configuration description**

result sample is produced. From the figure can be observed that the FIR filter is composed of five identical multiply-accumulate operations, of which one is highlighted by the dotted rectangle.

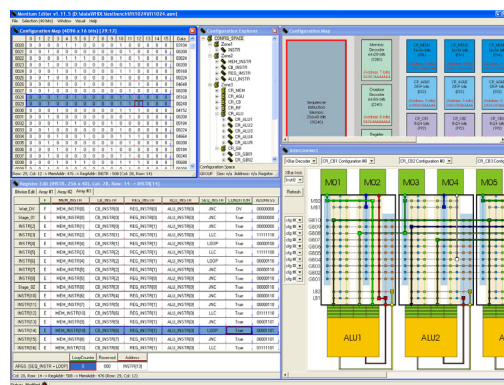
Figure 4 shows the Montium configuration description that implements the FIR filter of Figure 3. In the main loop, first the current input sample from external input ‘ext1’ is moved to the input register ‘a1’ of every ALU. Next, the multiply-accumulates are computed by calling procedure ‘mac’.

The filter coefficient for each ALU is preconfigured in input registers ‘c1’. The procedure ‘mac’ specifies that every ALU computes ‘a1\*c1+d1’, where ‘d1’ is the pipeline register in Figure 3. The result of the computation is sent to output ‘o1’ of an ALU.

Finally, the results of the computation need to be moved to the next inputs. This is done by calling the procedure ‘moveresults’ in the main loop. The procedure moves output ‘o1’ of the first four ALUs to the ‘d1’ input register of the next ALU. Output ‘o1’ of the fifth ALU is streamed out to external output ‘ext1’.



**Figure 5: Screenshot of the Simsation Simulator**



**Figure 6: Screenshot of the Insation Editor**

The statements in the main loop are executed concurrently. The loop is repeated until the last sample of a frame is encountered. The Hydra CCU can detect this sample and generates an end-of-frame (‘eof’) flag that the Montium can test. Typically, the main loop from Figure 4 is surrounded by code for synchronization with the environment.

### B. Simsation Simulator

The cycle-accurate Montium Simsation Simulator (see Figure 5) is a valuable tool for functional verification and debugging. The simulator gives the user as much detail as desired in various representations. It can be used for both debugging and system level simulations. Standard interfaces allow seamless integration in existing development flows. The simulation engine achieves great simulation speeds.

### C. Insation Editor

The Insation Editor (see Figure 6) gives full insight in all the configurable entities of the Montium processor core and Hydra network interface controller. This flexible tool gives critical users absolute control of configuring the Montium processing core. The graphical user interface can also be used for visual inspection of configuration files.

## VI. APPLICATION EXAMPLES

The Montium TP is a flexible core that can be used for digital signal processing applications. Typical target applications are (wireless) communication systems (UMTS, WLAN) and digital broadcasting systems (DAB, DVB).

Reference implementations of digital baseband processing and channel (de)coding for communication systems are built using a library of common DSP kernels. This library of DSP kernels includes algorithms such as FFT, FIR,

DDC, Viterbi decoding, Turbo decoding, Reed-Solomon decoding, and many more.

For example, the entire digital baseband processing of OFDM based systems can be done on a reconfigurable subsystem comprising one or multiple reconfigurable processor cores. The output of the baseband processing part (the channel bits) can be processed by Recore's channel decoding kernels.

Below a number of typical Montium applications are discussed.

## A. Baseband processing

### 1) Wideband CDMA

The Universal Mobile Telecommunications System (UMTS) standard is an example of a 3<sup>rd</sup> generation mobile communication system. The communication system has an air interface that is based on Code Division Multiple Access (CDMA) [11].

In a UMTS communication system, the signals from the strongest multi-paths are received individually. The receiver estimates the path delays for these strongest paths. For every delayed signal the receiver performs de-scrambling and de-spreading operations; this is called as a RAKE finger. The received soft-values of the individual RAKE fingers are combined and individually weighted by a channel estimator to provide optimal signal-to-noise ratio. Such a receiver is known as a RAKE receiver.

A flexible RAKE receiver is implemented using a single Montium processor. The total configuration file of the RAKE receiver is only 858 bytes. So, in about 4.29  $\mu$ s the Montium can be configured for RAKE processing. Using partial reconfiguration the number of fingers in the RAKE receiver can be adapted on run-time. For example, adjusting the number of fingers from 4 to 2 only takes 120 ns.

### 2) Bluetooth

The Bluetooth radio technology [12] provides a universal radio interface for electronic devices to communicate via short-range ad hoc radio connections.

A Bluetooth receiver contains an FM-discriminator, a Low Pass Filter (LPF) and a threshold detector. The FM-discriminator performs FM-to-AM conversion by multiplying the received signal with its delayed version. After FM-to-AM conversion, the signal is passed through the LPF, which is implemented as a FIR

filter, in order to block all high frequencies that occur due to multiplication. Finally, the consecutive bits are detected by a threshold detector. Since the signal bandwidth is limited to 1 MHz, the data rates are limited to 1 Mbps. This performance can be achieved easily by a single Montium tile implementing all baseband functionality.

### 3) HiperLAN/2 baseband processing

HiperLAN/2 [8] is a wireless local area network access technology similar to IEEE 802.11a. In [10] a Montium TP implementation of a HiperLAN/2 receiver and simulation results are presented. The implementation uses three Montium TPs. The simulation results show that the implementation can realize the minimum required bit error rate of  $2.4 \cdot 10^{-3}$  after error correction. These tile processors can meet the real-time performance requirements at fairly low clock frequencies: typically ranging from 25 to 72 MHz, depending on the function. Also, the one-time configuration overhead is low: ranging from 274 to 946 bytes. This is small enough to enable dynamic reconfiguration.

## B. Channel decoding

In [13] two kinds of channel decoders are implemented in the same Montium core: Viterbi and Turbo decoding.

### 1) Viterbi decoding

Viterbi forward error correction [14] is used in many wireless standards, including DRM, DAB, IEEE 802.11 and DVB. An adaptive Viterbi decoder is implemented on a single Montium TP. The implemented Viterbi decoder is adaptive in many ways. Parameters such as rate and constraint length can be reconfigured dynamically, depending on the communication system required by a specific application. The decision depth of the Viterbi decoder is also configurable and can be dynamically reconfigured while the Viterbi decoder is operating. Implementation details can be found in [13].

### 2) Turbo decoding

Turbo code schemes have been adopted by many wireless communication standards. In the 3<sup>rd</sup> generation UMTS system, Turbo coding is used for data channels (and Viterbi coding for voice channels) [1]. The SISO decoders in the Turbo decoder are implemented for the Montium using the Max-Log-MAP algorithm. This algorithm

has a regular optimized structure and achieves near optimal bit error rate. More information on the implementation of Turbo decoding on the Montium can be found in [13].

### C. Digital Down Conversion

Digital Down Conversion (DDC) is a method to reduce the sample rate by selecting a limited frequency band out of a stream of samples. By attenuating the unwanted frequencies, the signal can be resampled at a lower rate. The DDC algorithm used consists of a 2-stage Cascading Integrating Comb (CIC) filter, a 5-stage CIC filter and a 125-tap Finite Impulse Response (FIR) filter. The input sample rate is 64.512 MHz and the output is 24 KHz. This DDC configuration is suitable for DRM. More information on the implementation of this DDC can be found in [4].

## VII. CONCLUSION

Fully programmable architectures – like general-purpose processors – are flexible, but inefficient. Application specific architectures are efficient, but inflexible. The concession done by reconfigurable architectures is to limit the flexibility to a particular algorithm domain. A domain specific reconfigurable architecture is both efficient and flexible within its algorithm domain. The ambition of reconfigurable hardware is that the hardware adapts to the algorithm, instead of adapting the algorithm to the hardware.

Shortening time-to-market constraints will drive semiconductor companies to acquire complete and flexible IP solutions that can be integrated instantly. The market for DSP cores that combine low power, high performance, flexibility and low costs is growing rapidly.

The Montium TP is a DSP core that satisfies these requirements. The Montium TP can be used as a single DSP accelerator core or clustered in a (large) reconfigurable subsystem. The core is in particular suitable for battery operated devices and embedded systems that may not generate a lot of heat, such as (portable) consumer electronics, digital communications and top-of-the-hood automotive systems.

## REFERENCES

- [1] 3GPP TSG RAN WG1, “Multiplexing and Channel Coding (FDD)”, *Technical Specification 3GPP TS 25.212 V6.0.0 (2003-12)*, December 2003.
- [2] <http://www.altera.com/>.
- [3] ARM, “AMBA Specification (Rev 2.0)”, *ARM IHI 0011A*, ARM Limited, 1999.
- [4] T. Bijlsma, P.T. Wolkotte and G.J.M. Smit, “An Optimal Architecture for a DDC”, *Proceedings of the 20th International Parallel & Distributed Processing Symposium Reconfigurable Architectures Workshop (RAW 2006)*, Rhodes, Greece, April 2006.
- [5] M.D. van de Burgwal, G.J.M. Smit, G.K. Rauwerda and P.M. Heysters, “Hydra: an Energy-efficient and Reconfigurable Network Interface”, *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'06)*, Las Vegas, Nevada, USA, June 2006.
- [6] <http://chameleon.ctit.utwente.nl/>.
- [7] D. Edenfeld, A.B. Kahng, M. Rodgers, Y. Zorian, “2003 Technology Roadmap for Semiconductors”, *IEEE Computer*, vol. 37 issue 1, pp. 47-56, January 2004.
- [8] ETSI, “Broadband Radio Access Networks (BRAN); HiperLAN type 2; Physical (PHY) Layer”, *Technical Specification ETSI TS 101 475 V1.2.2 (2001-02)*, February 2001.
- [9] Y. Guo, C. Hoede and G.J.M. Smit, “A Column Arrangement Algorithm for a Coarse-Grained Reconfigurable Architecture”, *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'06)*, Las Vegas, Nevada, USA, June 2006.
- [10] P.M. Heysters, G.K. Rauwerda and G.J.M. Smit, “Implementation of a HiperLAN/2 Receiver on the Reconfigurable Montium Architecture”, *Proceedings of the 18th International Parallel & Distributed Processing Symposium Reconfigurable Architectures Workshop (RAW 2004)*, Santa Fé, New Mexico, U.S.A., April 2004, ISBN 0-7695-2132-0.
- [11] H. Holma and A. Toskala, “WCDMA for UMTS: Radio Access for Third Generation Mobile Communications”, *John Wiley & Sons*, 2001.
- [12] IEEE, “Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)”, *IEEE Standard 802.15.1(tm)-2002*, 2002.
- [13] G.K. Rauwerda, G.J.M. Smit, C.R.W. van Benthem, P.M. Heysters, “Reconfigurable Turbo/Viterbi Channel Decoder in the Coarse-Grained Montium Architecture”, *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'06)*, Las Vegas, Nevada, USA, June 2006.
- [14] A.J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm”, *IEEE Transactions on Information Theory*, 13(2):260-269, April 1967.
- [15] <http://www.xilinx.com/>.