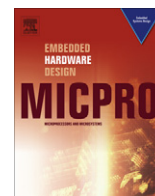


Contents lists available at [SciVerse ScienceDirect](#)

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Mobile satellite reception with a virtual satellite dish based on a reconfigurable multi-processor architecture [☆]

M.D. van de Burgwal ^{a,*}, K.C. Rovers ^a, K.C.H. Blom ^a, A.B.J. Kokkeler ^a, G.J.M. Smit ^a^a Computer Architectures for Embedded Systems Group, Faculty of Electrical Engineering, Math and Computer Science, University of Twente Enschede, The Netherlands

ARTICLE INFO

Article history:
Available online xxxxx

Keywords:

Adaptive beamforming
Phased array
Virtual satellite dish
Reconfigurable processor
MONTIUM TP
MPSoC

ABSTRACT

Traditionally, mechanically steered dishes or analog phased array beamforming systems have been used for radio frequency receivers, where strong directivity and high performance were much more important than low-cost requirements. Real-time controlled digital phased array beamforming could not be realized due to the high computational requirements and the implementation costs. Today, digital hardware has become powerful enough to perform the massive number of operations required for real-time digital beamforming. With the continuously decreasing price per transistor, high performance signal processing has become available by using multi-processor architectures. More and more applications are using beamforming to improve the spatial utilization of communication channels, resulting in many dedicated digital architectures for specific applications. By using a reconfigurable architecture, a single hardware platform can be used for different applications with different processing needs.

In this article, we show how a reconfigurable multi-processor system-on-chip based architecture can be used for phased array processing, including an advanced tracking mechanism to continuously receive signals with a mobile satellite receiver. An adaptive beamformer for DVB-S satellite reception is presented that uses an Extended Constant Modulus Algorithm to track satellites. The receiver consists of 8 antennas and is mapped on three reconfigurable MONTIUM TP processors. With a scenario based on a phased array antenna mounted on the roof of a car, we show that the adaptive steering algorithm is robust in dynamic scenarios and correctly demodulates the received signal.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Satellite reception requires accurate pointing of the receiving antenna. For home installations, this can be done relatively easy with a dish antenna that is mounted to a fixed plane like a wall or roof. In mobile situations, there is no fixed plane to mount the dish on, hence a complex actuator mechanism is required to continuously point a dish antenna towards the transmitter. A solution is to use an electronically steerable antenna, referred to as a 'virtual satellite dish'. To control such an antenna, advanced Digital Signal Processing (DSP) algorithms are required to compensate for the movement of the antenna and optimize the quality of the received signal. Typically, these algorithms require a vast amount of processing power and should be executed quickly to keep the receiver pointed at the transmitter.

Phased array beamforming techniques have been used in radar systems for many years already. The design of these systems is mainly driven by functional requirements (e.g., resolution, sensitivity, response time) where non-functional requirements (e.g., costs, power consumption) are of secondary concern [1]. For that reason, no low-cost, low-power phased array systems are available yet. However, in areas like Software Defined Radio (SDR) and satellite receivers, phased array antennas show great promise but their large scale introduction has been obstructed by the high costs involved. Our goal is to develop a low-cost, low-power phased array receiver platform. This can be realized by using a scalable architecture that is flexible enough to support multiple applications, such that the same architecture can be reused. Reconfigurable Multi-Processor System-on-Chip (MPSoC) based architectures seem to be promising, as they offer high performance (by enabling parallel processing through multiple processors) and are flexible within a certain application domain (reconfiguration enables efficient reuse of hardware by reconfiguring parts of the hardware or the application). Conventional phased array receivers typically use a large amount of dedicated central processing hardware, making the system neither scalable nor power efficient [2]. This article presents the implementation of an adaptive beam steering algorithm onto a reconfigurable multi-processor based architecture. We show

[☆] This research is done within the projects CMOS Beamforming Techniques (07620) and NEST (10346), supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

* Corresponding author. Tel.: +31 53 4894178.

E-mail addresses: m.d.vandeburgwal@utwente.nl (M.D. van de Burgwal), k.c.rovers@utwente.nl (K.C. Rovers), k.c.h.blom@utwente.nl (K.C.H. Blom), a.b.j.kokkeler@utwente.nl (A.B.J. Kokkeler), g.j.m.smit@utwente.nl (G.J.M. Smit).

how such an architecture can be utilized for efficient execution of beamforming, adaptive beam steering and symbol demodulation.

After an introduction to phased array systems and their different applications in Section 2, we will provide a more thorough discussion on the required processing and its complexity in Section 3. The target reconfigurable multi-processor architecture is introduced in Section 4 and the mapping of the processing blocks onto the reconfigurable processors in this platform is presented in Section 5. Finally, the scheduling of the entire application on the multi-processor architecture is discussed in Section 6, together with an example operation of the application to demonstrate the performance.

2. Mobile satellite signal reception

Digital television broadcasts are transmitted by many different satellites in different frequency ranges. The most commonly used range is the K_u band (11.7–12.7 GHz in The Americas and Australia and 10.7–12.75 GHz in Europe) where most systems use Digital Video Broadcast for Satellite (DVB-S) for transmission [3]. The DVB-S standard specifies a channel bandwidth of 36 MHz (effective bandwidth used is 50 MHz due to pulse shaping filter roll-off), transmitted within the 10.7–12.75 GHz frequency range. The modulation technique used for individual DVB-S channels is Quadrature Phase Shift Keying (QPSK). QPSK uses four different phases to represent transmitted information, equally distributed on the unit circle of the IQ plane. Each of these four phases represents a symbol, which is represented by two data bits. Since the transmission of two subsequent symbols requires instantaneous phase shifts in the transmitted output signal, high frequency components are introduced. A pulse shaping filter is used to decrease the effects these phase shifts by spreading the signal into a slightly larger frequency band such that the high frequency components are attenuated. For correct demodulation, QPSK requires a minimum Signal to Noise Ratio (SNR) at the receiver of 16 dB.

2.1. Conventional satellite dish

Conventionally, DVB-S receivers use a parabolic dish antenna. Such an antenna can be constructed easily and has a high efficiency. A parabolic dish antenna focuses a wavefront incoming from a single direction to one focal point. By mounting a Low Noise Block (LNB) at the focal point, the Radio Frequency (RF) signal is captured, amplified, downconverted to an Intermediate Frequency (IF) and transmitted to the remote modem that applies channel selection and demodulation of the IF signal. This construction requires the parabolic dish antenna to be tightly aligned with the transmitter, otherwise the wavefront is not efficiently focused and the LNB cannot successfully capture the transmitted signal.

A dish antenna could be used for reception of DVB-S signals in mobile environments (for example, in a moving car or on a yacht), but mechanical control is required to continuously steer the dish. Moreover, since a dish antenna would cause considerable air resistance when mounted on a car, it is not an efficient solution for mobile satellite reception. Hybrid solutions are commercially available, where the dish is reduced to multiple waveguide antennas which are mechanically steered [4]. Such systems can be embedded in a relatively compact housing, but still rely on mechanical parts (which are sensitive to wear and possibly consume much energy).

2.2. Virtual satellite dish

The virtual satellite dish discussed in this article is based on a phased array antenna. Such an antenna is fully steered electronically, which means it does not rely on mechanical control. An

advantage of such an antenna is the possibility of receiving broadcasts from multiple satellites simultaneously, by applying DSP techniques. This is useful, when multiple users want to receive signals from different satellites simultaneously.

Phased array systems are based on the principle of interference using multiple antennas in an array to make a transceiver directional (see Fig. 1). Interference is the pattern resulting from the addition of two or more (partly) correlated waves. For in-phase signals, the waves add up constructively and for out-of-phase signals the waves add up destructively.

Assume a single omni-directional wave source, emitting a spherical waveform s in time and space:

$$s(t, l) = A \cdot \cos(\omega t \pm kl) \quad (1)$$

with A the amplitude, ω the frequency, k the wave number, t time and l the path length from the source. For a source in the far field perpendicular to the array, the wavefront is considered planar and the received signals add up constructively. From other directions, the wavefront arrives at different times at the antennas. Typically, the antennas are placed a distance $d = \lambda/2$ apart (where λ is the wavelength of the received signal). If the wavefront arrives at an angle θ incident to the array, the wavefront travels a distance $\Delta l = d \cdot \sin(\theta)$ further to the next antenna, which results in a time delay:

$$\Delta t = \frac{\Delta l}{c} = \frac{d \cdot \sin(\theta)}{c} \quad (2)$$

where c is the propagation speed of radio waves. If the signal is a narrowband signal, this time delay can be considered as a phase shift:

$$\Delta\psi = \omega \cdot \Delta t \quad (3)$$

hence by applying the inverse phase shift, the time delay is corrected. Therefore, such an antenna array is usually referred to as 'phased array antenna'.

By correcting the phase, the direction of maximum sensitivity is steered [1]. After the RF front end for each antenna, sampled signals are combined by the beamforming processing to create a resulting signal with for example a maximum sensitivity in a direction of interest or a minimum sensitivity (a *null*) in the direction of an interfering signal. Beam steering refers to changing the shape and direction of the formed beam by changing the gain and phase of complex multiplier stages, such that a certain angular sensitivity is created.

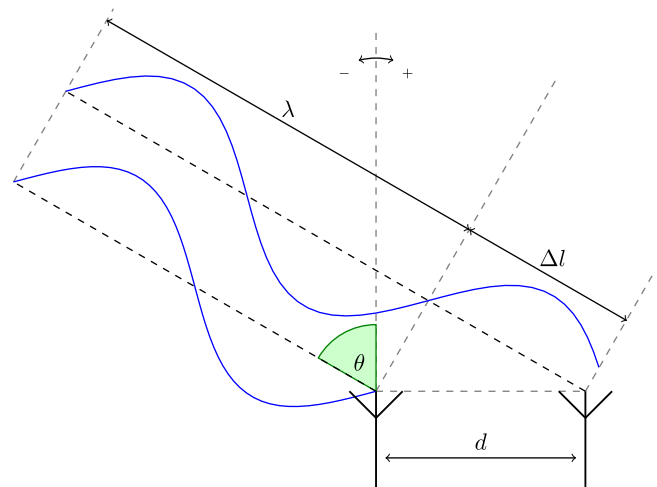


Fig. 1. Wavefront received by multiple antennas in a phased array.

Note that the same antenna signals can be used to form multiple beams in different directions simultaneously. Therefore, for each beam, the antenna signals are combined with different correction parameters. This requires a dedicated instance of the beamformer and beam control parts for each beam.

Antenna design is done based on a worst case signal reception. For the mobile satellite receiver, the phased array is mounted horizontally on the roof of a car. Therefore, the maximum angle θ with respect to the orthonormal of the horizontal plane (see Fig. 1) is determined by the position of the car (its maximum tilt in either direction, for example when driving up a mountain) and the elevation of the satellite currently received. In Europe, satellites are positioned between 27° and 45° above the equator, so θ is in the range of $45\text{--}63^\circ$. To have similar performance as a dish antenna (i.e. gain and directivity), the projected dimensions in the direction of the satellite should be equal to the dish; this means the phased array should be about $125\text{ cm} \times 125\text{ cm}$. Having $\lambda = 2.5\text{ cm}$, this means an array of about 100×100 elements is required.

In this article, a simplified array consisting of eight antenna elements is presented. Although there is a large difference with the requirements for a real antenna, we use this simplified array to show that an adaptive phased array can be implemented fairly efficiently. Since the computational complexity scales linearly with the number of elements, this also holds for the full-size array.

3. Digital signal processing

By replacing the conventional dish antenna in a DVB-S receiver system by a phased array antenna, the DVB-S receiver can be used in dynamic environments where the relative receiver location is continuously changing. Fig. 2 shows the phased array processing chain that is used for the DVB-S reception case. For each of the blocks in the chain after the Analog-to-Digital Converter (ADC), we will shortly describe its functionality and complexity.

3.1. Beamforming

The beamforming operation is defined as the inner product of the current antenna snapshot \vec{x} (the set of samples taken from the different antennas at one time instant) and the steering vector $\vec{\phi}$ that applies phase shifts to each of the antenna streams:

$$y = \vec{\phi} \cdot \vec{x} = \sum_{i=1}^N \phi_i x_i \quad (4)$$

where N denotes the number of antenna elements. Since both \vec{x} and $\vec{\phi}$ consist of complex values, the beamforming operation consists of N complex multiplications and additions. If multiple beams are formed, the same antenna snapshot can be used for each beam

while each beam can be steered individually by using a separate steering vector.

3.2. DVB-S baseband processing

As mentioned in Section 2, the QPSK modulated symbols are filtered by a pulse shaping filter at the transmitter to suppress high frequency components in the transmitted signal [3]. At the receiver side, the beamformer output is filtered by a Matched Filter to restore the high frequency components, such that the original QPSK symbols can be demodulated (see Fig. 3). A Root-Raised-Cosine (RRC) filter is used as Matched Filter, and can be implemented by a 25-tap Finite Impulse Response (FIR) filter [5].

3.3. Adaptive beam steering

Since the receiver might be continuously moving, an adaptive beam steering algorithm is required to correct the beam direction. There are three classes of adaptive beam steering algorithms [6].

- *Spatial beamforming* algorithms use correlation between the data streams received by individual antennas. This requires a considerable amount of processing, as the correlation may be done over long data streams and over multiple antennas.
- Algorithms of the *temporal beamforming* class rely on correlation between the received data stream and a known reference stream. For example, pilot symbols or streams of symbols are added as a reference to determine the channel input response.
- If structural or statistical properties of the received signal are known, the beam direction can be corrected by algorithms in the class of *blind beamforming*.

Because fast adaptive beam steering is desired with low computational complexity, the most suitable class is the blind beamforming class. As required for such adaptive steering algorithms, QPSK modulation adds both structural and statistical properties to the transmitted signal. This will be shown in the next subsection.

In the initial situation where the wanted satellite has not been detected yet, a search action has to be done to find its location. This can be done with a Direction of Arrival (DOA) estimation algorithm. Since there is no reference signal available in the initial situation, only a spatial beamforming algorithm can be used for DOA estimation. Examples of suitable DOA algorithms are ESPRIT [7] and MUSIC [8]. The disadvantage of these algorithms is their high complexity of $O(N^4)$ due to the correlation operations calculated for all possible antenna pairs. Therefore, a real-time implementation of such algorithms is very computational intensive and should be avoided. Since DOA estimation is outside the scope of this arti-

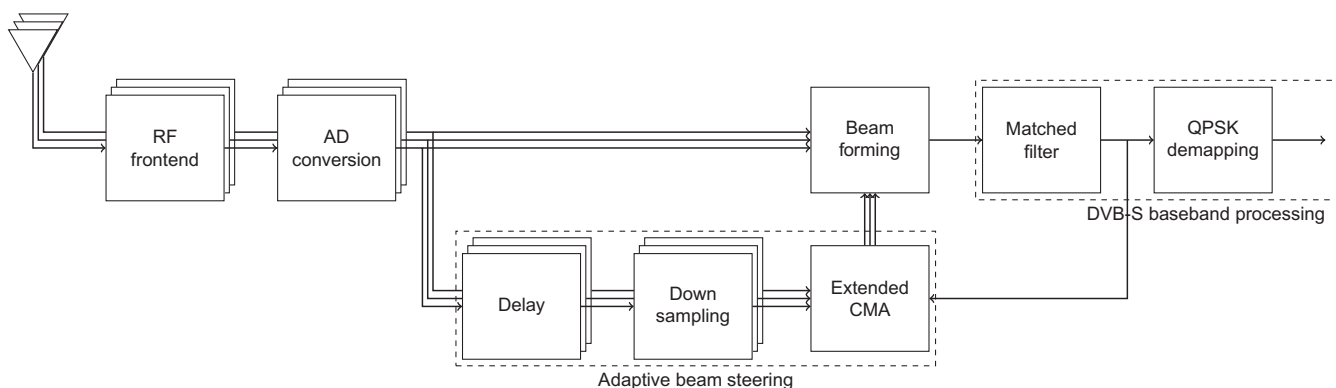


Fig. 2. Structure of the DVB-S phased array receiver.

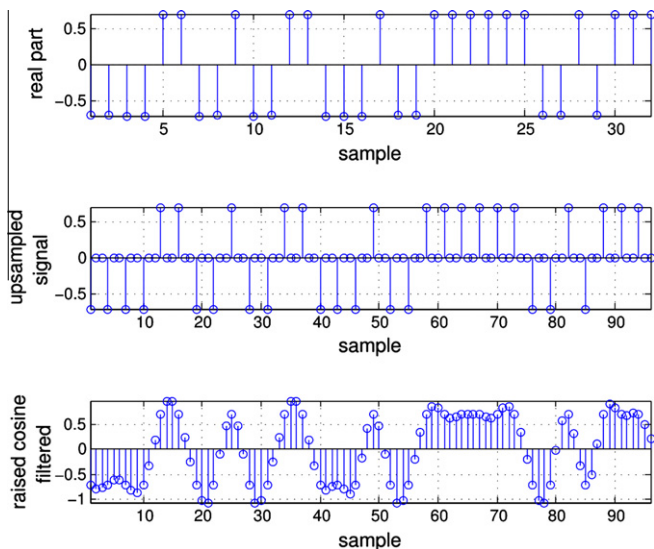


Fig. 3. Effects of pulse shaping applied to the transmitted signal to filter high frequency components.

cle, for the remainder of this article we will assume that initial locations of transmitters are known.

3.3.1. Tracking algorithm

Once the initial locations of the satellites are known, a tracking algorithm is enabled. As mentioned in the previous section, QPSK is used for transmitting DVB-S symbols. This modulation technique has well-defined structural and statistical properties. The signal is modulated in phase only, which is a strict structural property. The highest utilization of the channel can be reached when the usage of all constellation points is uniformly distributed, so transmitted symbols have a clear statistical property. Since the gain is assumed to be constant, a Constant Modulus Algorithm (CMA) can be used efficiently [9]. Xu proposed an extension to CMA that allows for correction of phase deviations [10]. Extended CMA uses both the antenna samples \vec{x} as well as the output of the beamformer $y = \vec{\phi} * \vec{x}$ to adjust the current steering vector $\vec{\phi}$ such that the modulus error of the beamformer output with respect to the expected modulus used by QPSK is minimized. The deviation of the current beamformer output with respect to the expected QPSK modulation points is defined by the cost function J [10], which is given by Eq. (5):

$$J(\vec{\phi}) = E(|y|^2 - 1)^2 + E(\sin^2(2\angle y)) \quad (5)$$

where $\angle y$ denotes the angle of y (in radians) and $E(p)$ denotes the expected value of p .

The optimal steering vector in terms of J is found when the cost function is minimized. This can be done by applying an iterative gradient descent method, such that Eq. (5) can be rewritten to Eq. (6):

$$\vec{\phi}[n+1] = \vec{\phi}[n] - \mu \nabla_{\vec{\phi}} J = \vec{\phi}[n] - \mu \cdot \frac{8j(|y|^4 - |y|^2) + 4\sin(4\angle y)}{4j \cdot y} \cdot \vec{x} \quad (6)$$

where μ indicates the update stepsize (typically, $\mu = 0.005$ for stable operation [10]) and $\nabla_{\vec{\phi}} J$ indicates the gradient of J .

After optimization, we get:

$$\vec{\phi}[n+1] = \vec{\phi}[n] - \mu \cdot \frac{2(|y|^4 - |y|^2) - j\sin(4\angle y)}{y} \cdot \vec{x} \quad (7)$$

A block diagram of the computations involved in Eq. (7) is shown in Fig. 4. As can be seen, the calculation of $\mu \cdot \frac{2(|y|^4 - |y|^2) - j\sin(4\angle y)}{y}$ only consists of scalar operations and therefore

has a fixed computational complexity. Because the calculation of $\vec{\phi}$ consists of a multiplication of two vectors of length N , the complexity of the entire Extended CMA algorithm scales linearly with the number of antennas N . Using the steering vector $\vec{\phi}$, the beam pattern tracks the transmitter while interferers are automatically suppressed. Multiple transmitters can be tracked by adding a beamformer and an Extended CMA algorithm for each transmitter.

3.3.2. Downsampling

Although the computational complexity of blind beamforming algorithms is much lower than the complexity of the spatial and temporal beamforming classes, the Extended CMA algorithm is too computationally intensive to operate on each new antenna snapshot. Moreover, since antenna snapshots are sampled at a rate that is orders of magnitude higher than the rate of movement of the transmitter, operating the Extended CMA algorithm once every few antenna snapshots considerably saves in processing requirements. Typically, the update rate of the Extended CMA algorithm can be in the order of hundreds of times lower than the sample rate [11]. For each Extended CMA update, only one snapshot is used and the remaining snapshots are not used.

4. Reconfigurable tiled architectures

The goal is to execute the Extended CMA algorithm on a reconfigurable tiled architecture. In this section, we describe the targeted architecture in more detail.

Phased array processing can be characterized as a streaming application with high data rates and processing requirements, but a regular processing structure. Because a scalable and regular solution is needed, a reconfigurable multi-processor architecture is proposed [12]. Such a reconfigurable processing architecture provides flexibility and has a number of advantages. For example, we can use only part of the antenna array or create multiple sub-arrays to save energy or increase the lifetime. Moreover, graceful degradation can be provided since individual tiles in a reconfigurable architecture might fail due to aging. Reconfigurability inherently leads to an adaptable system, that can be adapted to changing environments while maintaining the quality of service.

In this article, we use the reconfigurable System-on-Chip (SoC) as shown in Fig. 5. This SoC is based on 3 reconfigurable MONTIUM Tile Processor (TPs) that are used for data processing and a LEON2 processor [13] for general purpose control tasks. The MONTIUM TP processors are connected to a circuit-switched Network-on-Chip (NoC), that provides configurable dedicated interconnect between the processors and external interfaces. As a result, high communication bandwidth between the processors is available with a fixed latency. The LEON2 processor controls the MONTIUM TPs, NoC configuration and external interfacing with the USB and serial ports. Once the interfaces have been initialized, the LEON2 processor configures the NoC and suspends until the MONTIUM TPs have finished processing or other I/O actions are required. Since the LEON2 is not responsible for the phased array processing, its performance and overhead are not discussed in this article.

The SoC shown in Fig. 5 is prototyped on a Xilinx Virtex 4 LX200 FPGA. In this prototype platform, the LEON2 processor and NoC operate at 45 MHz and the 3 MONTIUM TP processors are clocked at 15 MHz. A similar architecture consisting of an ARM-926 processor and 4 MONTIUM TPs was implemented in 130 nm CMOS technology [14,15].

4.1. Network-on-chip

For the communication between the MONTIUM TP processors, a predictable circuit-switched NoC was used [16] that interconnects

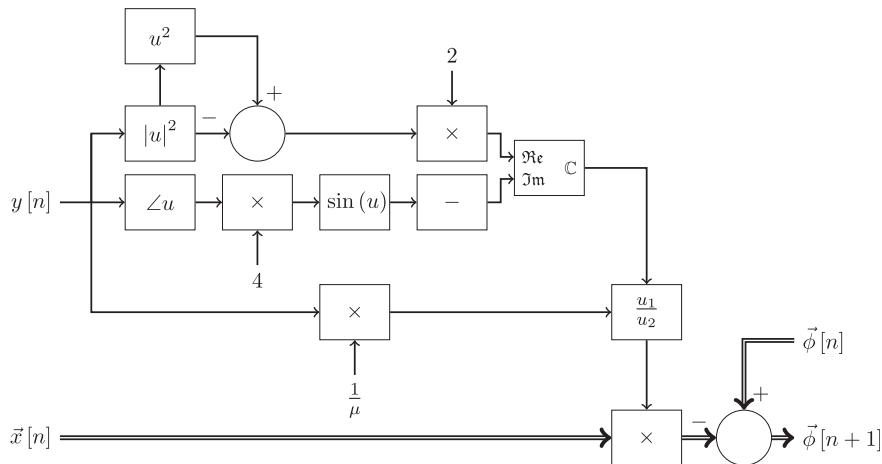


Fig. 4. Block diagram of the Extended CMA adaptive beam steering algorithm.

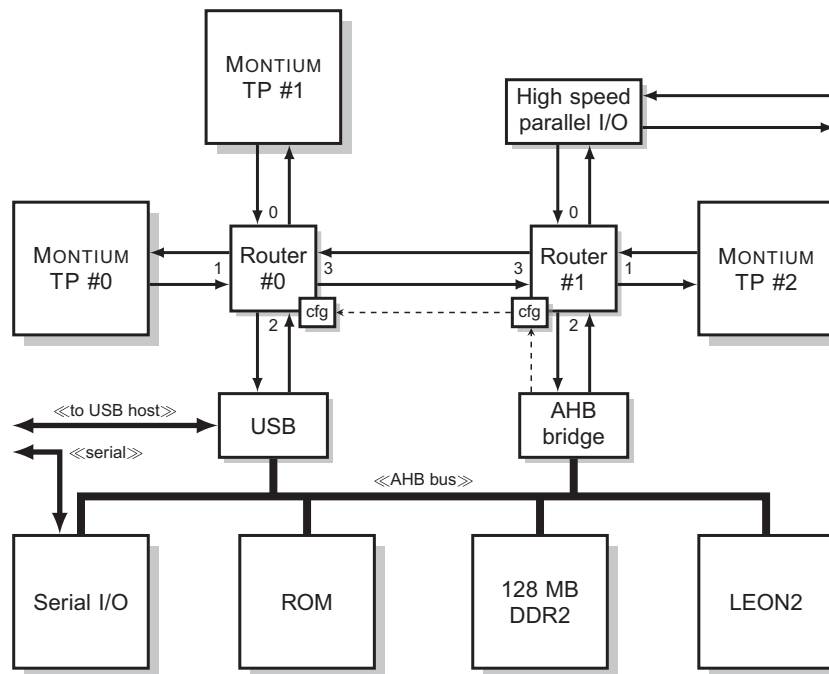


Fig. 5. LEON2 SoC.

the three MONTIUM TP cores. Circuit switching has been chosen as it simplifies the Network Interface (NI) of the connected processors, because it allows for a simplistic network protocol that does not require to include the routing information inside sent packets. This is an advantage for both the sender and receiver, since there is no overhead during the communication for packaging of data (assembly or re-assembly of packets). In the case of streaming applications, it was expected that the mapping would be rather static with fixed information streams. Due to the semi-static behavior of streaming applications, the connections for the input data and output data remain open during their execution. The connections in the NoC, i.e. the routers' configuration, are controlled via the Advanced High-performance Bus (AHB) bridge. Routing of communication channels is done by dedicated router configuration interfaces, which are included in the memory map of the bridge. Each network link between a router and another router or processor (see Fig. 5) consists of 4 parallel 16-bit lanes that can be used simultaneously. The routers in the LEON2 SoC operate at

47.52 MHz, resulting in a gross bandwidth of about 380 MByte/s per link.

4.2. Montium Tile Processor

The MONTIUM TP is an example of a coarse-grained reconfigurable processor [17] developed by Recore Systems.¹ The MONTIUM TP targets the DSP algorithm domain. Several core operations (called kernels) used in DSP applications like baseband processing and channel decoding in wireless communications receivers have been efficiently mapped on the MONTIUM TP architecture [18,19]. Its template based design allows for customization of architectural properties like data path width (16-bit by default), targeted clock frequency (100 MHz for 90 nm technology) and processing capacity (5 parallel Arithmetic Logic Unit (ALUs) for the reference design). This reference

¹ <http://www.recoresystems.com>.

design has a silicon area of approximately 2 mm^2 and a power consumption of approximately $550 \mu \text{ W/MHz}$. The MONTIUM TP is configured by the Communication and Configuration Unit (CCU) [20], which is implemented in the Network Interface shown in Fig. 6.

4.2.1. Program control

Inside the MONTIUM TP, shown in Fig. 6, three regions can be identified. The lower region consists of a sequencer in which the kernel is stored. It contains a programmable Static Random Access Memory (SRAM) to store the kernel instructions. A program counter is used for the program flow. It is directly connected to the SRAM to select the next instruction to be executed. Hence, an instruction can be fetched immediately as it is not affected by typical memory access delays as encountered in conventional architectures.

4.2.2. Configuration memory

The instruction selected by the sequencer is decoded by a number of decoders. A memory decoder decodes the instruction that is used to generate the address patterns for the memories in the data path. The interconnect decoder decodes the part of the instruction required to control the crossbar and local buses around the ALUs and the memories. The instruction that selects ALU operands stored in the register files is generated by the register decoder and the ALU instruction itself is decoded using the ALU decoder. Such decoding enables efficient storage of the kernel, as similar parts of two instructions can be stored at the same decoder address, thus resulting in lower memory area requirements. Moreover, since only few wires are required to control the decoders, only a few bit-flips occur in the control wires when a new instruction is selected. Hence, the energy consumption is decreased.

4.2.3. Processing part array

The decompressed instructions that are generated by the decoders are sent to the upper region, the Processing Part Array

(PPA). It consists of 5 identical ALU together with a local interconnect and a large crossbar consisting of 10 global buses that provides a high bandwidth to 10 memory units. Each ALU can be connected to two of the memories via a local interconnect or to the 8 other memories via the global buses. The operands for the ALU are stored in 4 register files which can be read simultaneously. In addition, each ALU can receive an intermediate value from its right neighbor ALU via an east–west connection. Using these 5 inputs, multiple operations can be executed simultaneously and from each ALU at most 3 results can be generated (one to the west output and two to the bottom outputs, which are connected to the interconnect). Fig. 7 shows the internal structure of one ALU.

The upper part, level 1, contains 4 function units, each of which can execute bitwise and logic operations or simple arithmetic operations. Each function unit generates status flags to indicate the occurrence of overflow, a negative result or whether its result equals zero. These status flags may be used by the sequencer, for example for conditional jumps. In the second level a Multiply Accumulate (MAC) operation can be executed. Its multiplier operates on either the outputs of the first level, named Z_1A and Z_1B , or the register files A to D. Next, the operands for the adder can be selected from the result of the multiplication, the register files A to D, the outputs of level 1, or the east input. In addition, depending on the value of the status bit (SB), the right operand for the adder can be dynamically selected from inputs B, D, Z_1A and Z_1B . The output of the adder is made available to the next ALU at the left side through the west output and can be used in the butterfly unit located in the lower part of the ALU. Two results can be returned via outputs o_1 and o_2 .

Since the ALU is not pipelined, the entire operation from the register file inputs to the ALU outputs can be done within one clock cycle. Almost all arithmetic operations in the ALUs can be executed in either integer modulus (i.e., they operate on the 16 rightmost bits) or in 1.15 fixed point modulus (i.e. the leftmost bit is used as sign bit whereas the other bits contain the fixed point fraction). In order to avoid overflow, the intermediate values can be saturated.

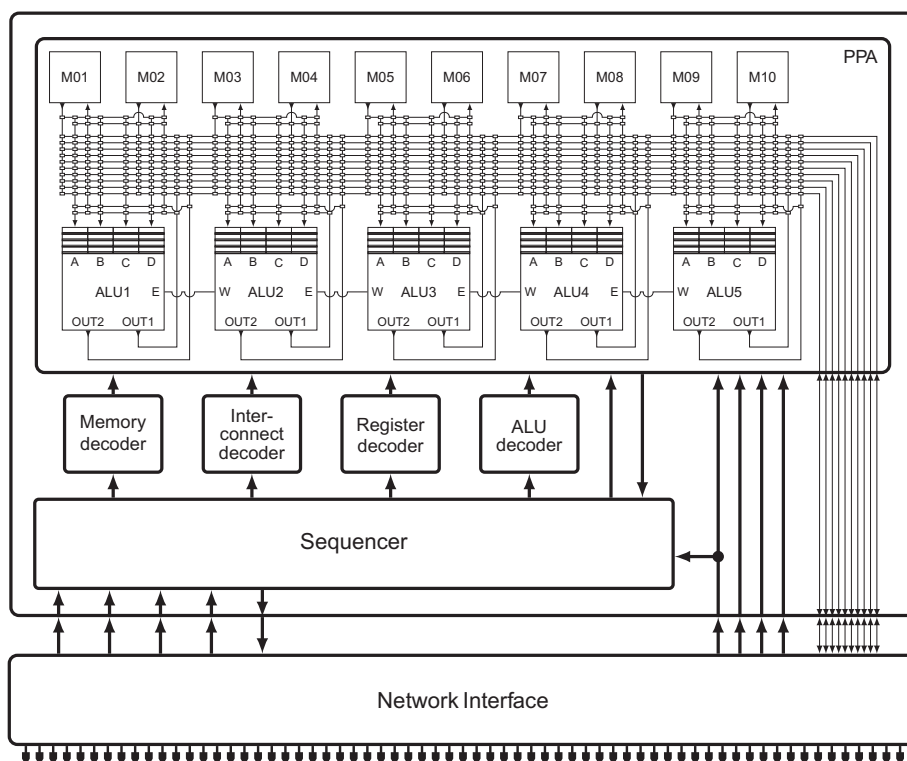


Fig. 6. Montium Tile Processor.

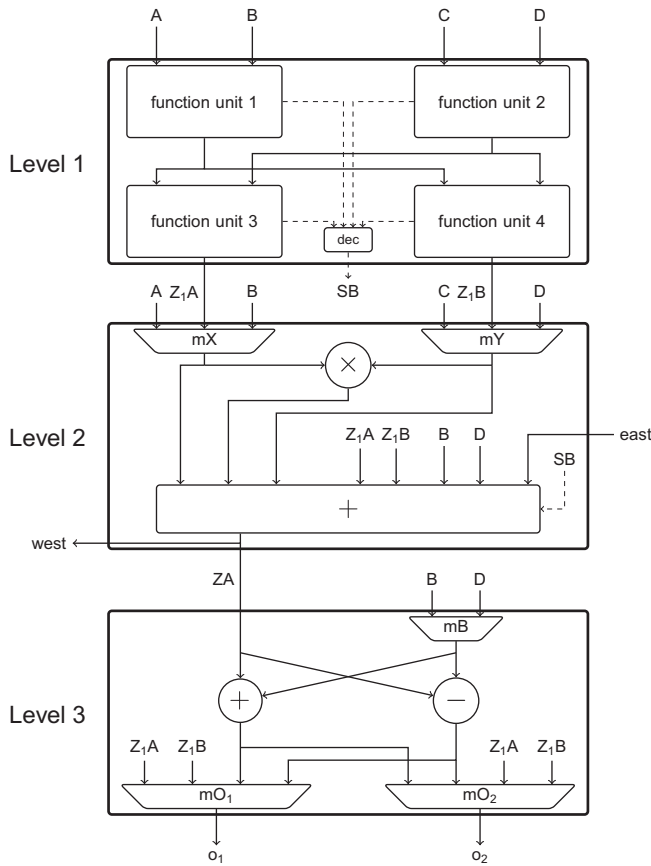


Fig. 7. Structure of one MONTIUM TP ALU.

Many DSP applications are based on vector or matrix operations. Hence, the memory addressing can be done in a regular way. Typically, these addressing patterns consist of linear addressing, stride-by- n (i.e., the address is incremented by n after each read or write operation), bit reversing (an output reordering technique that is typically used for Fast Fourier Transform (FFT) algorithms) and modulo counting (e.g. for creating circular buffers). By supporting these operations in hardware, the address calculation can be done separately from other ALU operations such that the ALU performance is optimized. Per memory, the MONTIUM TP contains an Address Generation Unit (AGU) that provides the hardware support for memory addressing.

4.2.4. NoC network interface

The MONTIUM TP connects to the NoC via a local NI. Each of the 4 NoC lanes can be connected to any Global Bus (GB) of the MONTIUM TP and vice versa. This NI is implemented by the Communication and Configuration Unit (CCU), which is responsible for control, memory initialization, synchronization and NoC protocol conversion. The CCU provides two mechanisms for communication between the MONTIUM TP and other processors: block mode and streaming mode communication. In the block mode, the input samples are stored in the memories by means of a DMA transfer by the CCU. The execution of the configured algorithm is enabled by using a start command. When the execution has finished, the CCU retrieves the results from the memories with another DMA transfer. The streaming mode operation requires less explicit control overhead by the CCU. In this mode, a program configured in the MONTIUM TP can generate a read request for reading data from any input NoC lane or generate a write request for writing data to any output NoC lane. These requests are executed by CCU, or the

MONTIUM TP is temporarily halted if the NoC lanes cannot be accessed (for example, when no input data is available or when the output buffer is full). Simultaneously with generating read and write requests, the MONTIUM TP can continue its computation. This enables the overlap of communication and computation, avoiding costly wait cycles during the computation. Due to the data driven behavior of streaming mode applications, the communication requests can be embedded within the program itself such that communication and computation are automatically synchronized.

5. Mapping kernels to the MONTIUM TP

The receiver chain for a phased array based DVB-S receiver was presented in Section 3. In this section, we present the implementation of the beamforming, adaptive beam steering and DVB-S demodulation. For the adaptive beam steering algorithm, we assume that the initial positions of transmitters are known.

First, the digital beam steering and beamforming are mapped to an MPSoC architecture based on the MONTIUM TP architecture as presented in Section 4. Then, the processing requirements are derived for a DVB-S receiver using $N = 8$ antennas. Each antenna is equipped with a 1.5 MHz IQ ADC, resulting in 1.5 Msamples/s complex data.

5.1. Beamforming

The beamforming operation is the operation with the highest computational complexity, because it combines the samples from all antennas with a vector multiplication. Therefore, the number of (complex) operations scales linearly with the number of antennas. To minimize the processing latency, the beamforming operation defined in Eq. (4) is partitioned in smaller operations, as given in Eq. (8):

$$y = \sum_{i=1}^4 \vec{\phi}_i \cdot \vec{x}_i + \sum_{i=5}^8 \vec{\phi}_i \cdot \vec{x}_i \tag{8}$$

Here, two MONTIUM TPs each perform the partial multiply-sum operations on four antenna samples and the third MONTIUM TP sums the results of the other two processors, resulting in the mapping shown in Fig. 8. With this partitioning, both partial sums are calculated in 4 clock cycles and the results are added in a single clock cycle.

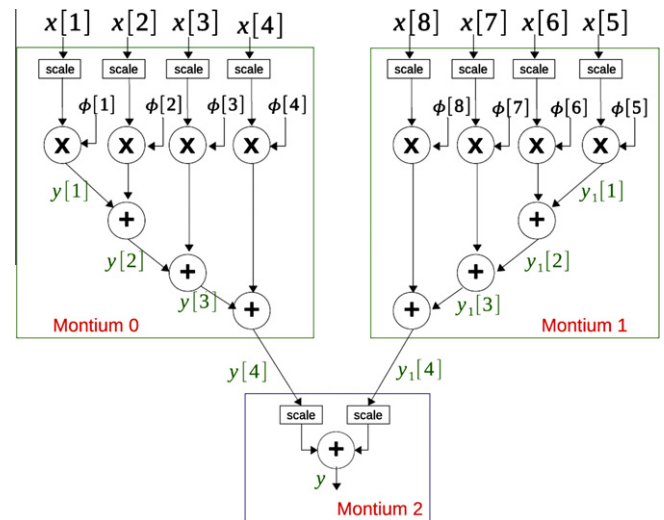


Fig. 8. Mapping of the beamforming operation performed by 3 MONTIUM ALUs.

5.2. Baseband processing

The RRC Matched Filter is implemented using two 25-taps FIR filters for the I and Q parts of the beamformer output. An N -taps FIR filter can be mapped on the MONTIUM TP in $N/5$ clock cycles [18]. Hence, each of the two filters can be executed by a MONTIUM TP in $25/5 = 5$ clock cycles.

5.3. Beam steering

A schematic overview of the Extended CMA algorithm is given in Fig. 4. All blocks in Fig. 4 can be mapped on the MONTIUM TP ALUs directly, except for the coordinate transform operations ($|u|$ and $\angle u$) and the sine calculations, which do not directly fit. These operations are explained in the following sections.

5.3.1. Transforming Cartesian coordinates to polar coordinates

The conversion from Cartesian coordinates (x, y) to polar coordinates (r, θ) and back requires some goniometric operations.² These could be implemented by a large Lookup Table (LUT), at the consequence of limited accuracy. A more accurate approach is the COordinate Rotation DIgital Computer (CORDIC) algorithm. It was originally proposed by Volder [21] as an iterative approach based on shift and add operations to apply coordinate transformations, which heavily rely on trigonometric functions. Extensions were proposed by Walther [22] to implement other operations like division, square root, Fourier transforms and many others. We mapped the algorithm, as described in [23], to the MONTIUM TP. For the conversion from Cartesian to polar coordinates, the vectoring mode is used. The equations for the vectoring mode are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \quad (9)$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i} \quad (10)$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (11)$$

where $d_i = +1$ if $y_i < 0$, -1 otherwise. When the number of iterations n is increased, the final values will approximate to:

$$x_n = A_n \sqrt{x_0^2 + y_0^2} \quad (12)$$

$$y_n = 0 \quad (13)$$

$$z_n = z_0 + \tan^{-1} \left(\frac{y_0}{x_0} \right) \quad (14)$$

$$A_n = \prod_n \sqrt{1 + 2^{-2i}} \quad (15)$$

such that $r = \frac{x_n}{A_n}$ and $\theta = z_n$.

The mapping of these equations is shown in Fig. 9. The decision variable d_i , which depends on the sign of y_i , is generated using the status bits of the function units. Then, based on the status bit the new values of x_{i+1} , y_{i+1} and z_{i+1} can be calculated. For the calculation of x_{i+1} , the value of $y_i \cdot 2^{-i}$ is calculated by shifting y_i over i bits to the right. The calculation of x_{i+1} , y_{i+1} and z_{i+1} depends on the sign of y_i . For x_{i+1} , d_i is a by-product of the $y_i \gg i$ operation. For y_{i+1} and z_{i+1} , d_i is determined explicitly by the sign of y_i . Both the positive and negative values are calculated for the left operand. For example, for calculating x_{i+1} the value of $y_i \cdot 2^{-i}$ and its negative value are both calculated and based on the decision variable d_i one of them is subtracted from x_i . The values for $\tan^{-1} \left(\frac{y_0}{x_0} \right)$ are calculated offline and stored in a read-only memory. During the calculation of the CORDIC equations, an AGU reads the memory and writes the value to the register file for ALU 3. Hence, reading these constants does not require any additional clock cycles. Using this mapping, a

single CORDIC iteration consisting of the three equations can be calculated in a single clock cycle.

The results of the implemented algorithm are shown in Fig. 10, which gives the accuracy of the result as a function of the iteration number. As can be seen, each iteration yields approximately one additional bit of precision. Due to the bitshift operations and the limited word-width of the MONTIUM TP, the smallest possible error is reached after 14 iterations. The CORDIC equations (Eqs. (9)–(11)) are only valid for rotation angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ [21,23]. For larger angles, an initial rotation over $\pm \frac{\pi}{2}$ should be applied, which can be realized with a set of equations that is slightly different from Eqs. (9)–(11). The MONTIUM TP implementation of the initial equations is comparable to the mapping presented for the regular CORDIC operations. Hence, the calculation of the initial equations can be done in one additional iteration.

5.3.2. Sine calculation

The sine function could be calculated very accurately using CORDIC. However, this requires an additional CORDIC operation which is expensive in terms of clock cycles. Instead, we chose to map the sine function to a LUT which is stored inside one of the memories. The upper 10 bits of the 16-bit fixed point angle are used as address for the lookup.³ Such a lookup only requires 2 clock cycles, which is much less compared to the 14 cycles required for running a complete CORDIC operation.

5.3.3. Complex division

In processor architectures where multiplications are scarce (i.e., if no multiplier is available or when the latency of a multiplier is very high), the complex division could be implemented by 2 CORDIC operations, one real division and 2 multiplications [24]. The MONTIUM TP, however, contains multipliers and therefore, more efficient implementations of the complex division can be made. Assume a division between the complex numbers $X = a + jb$ and $Y = c + jd$. The division can be rewritten as follows:

$$\frac{a + jb}{c + jd} = \frac{a + jb}{c + jd} \cdot \frac{c - jd}{c - jd} = \frac{ac + bd}{c^2 + d^2} + j \frac{bc - ad}{c^2 + d^2} \quad (16)$$

Now define $e = \frac{1}{c^2 + d^2}$. After substitution in Eq. (16), we get:

$$\frac{a + jb}{c + jd} = \dots = (ac + bd) \cdot e + j(bc - ad) \cdot e \quad (17)$$

which can be implemented by 6 multiplications, 2 additions and the costs for the calculation of e .

Note that $e = \frac{1}{c^2 + d^2} = \frac{1}{|y|^2}$. For the division used in Eq. (7), X corresponds with the nominator and Y corresponds with the denominator which is y , so $e = \frac{1}{|y|^2}$. As can be seen in Fig. 4, the calculation of $|y|^2$ is already done. An option is to calculate its inverse by using a LUT, similar to the sine calculation. However, the values of $\frac{1}{|y|^2}$ with $|y|^2 \in [0, \dots, 1)$ are in the range of $(1, \dots, \infty)$, which cannot be represented in a 1.15 fixed point notation. A straight-forward LUT based implementation is therefore not useful. In order to solve this problem, the multiplication by a step factor μ (see Fig. 4) is included in the LUT. Typically, $\mu = 0.005$ is used for the best tracking results. So, instead of using a LUT containing values $\frac{1}{|y|^2}$, the LUT consists of values $\frac{\mu}{|y|^2}$ which contains unsaturated values for all $\mu \leq |y|^2 < 1$ (see Fig. 11). We use a LUT with 512 entries to calculate $\frac{\mu}{|y|^2}$. For such a LUT, the first 3 entries are saturated (since $\frac{0.005}{512} < \mu$). However, since the Extended CMA algorithm is used to normalize $|y|^2$

³ Since the output of the coordinate transform operation is squared due to the $|u|^2$ blocks in Fig. 4, the quantization noise caused by a 10-bit lookup operation is amplified considerably. The output of the sine calculation, however, is hardly amplified by further operations in Fig. 4.

² Note that in this section, the symbols x and y are used differently from the rest of the article.

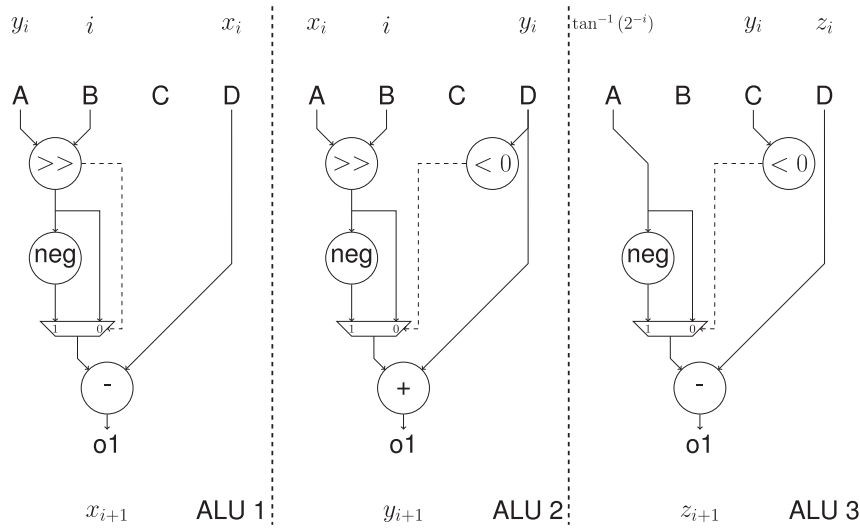


Fig. 9. Mapping of CORDIC equations on 3 MONTIUM TP ALUs.

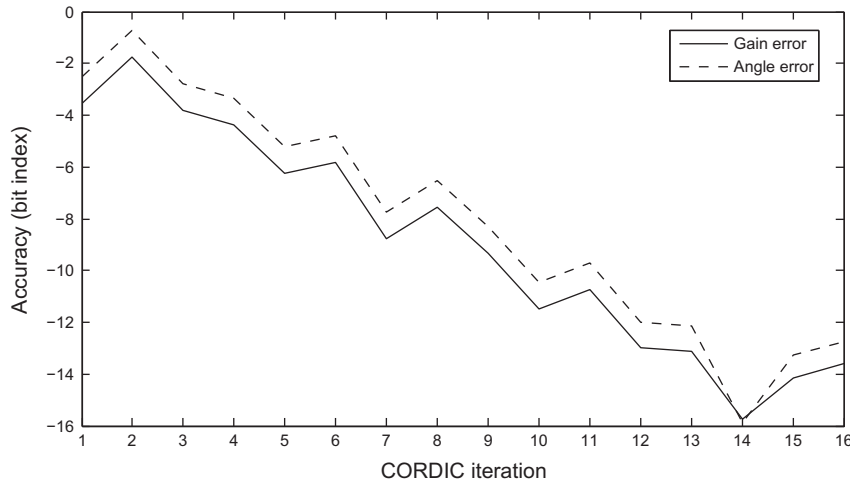


Fig. 10. Error between MONTIUM TP result and input value (in bits) after each CORDIC iteration.

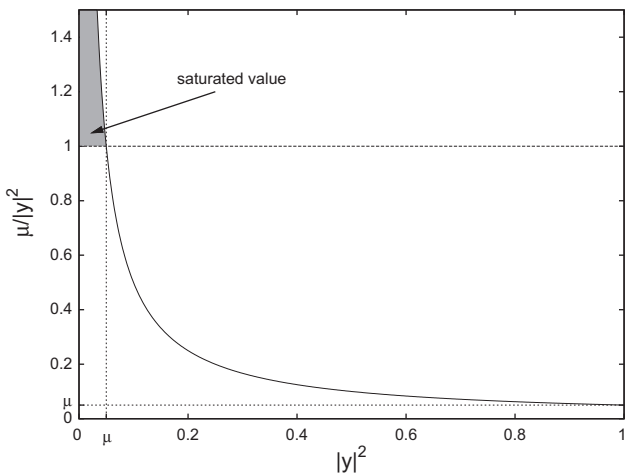


Fig. 11. Contents of the LUT for calculation of $\frac{\mu}{|y|^2}$.

to 1, the probability of a lookup of one of these saturated values is very low. Hence, in total the calculation of a complex division re-

quires 6 multiplications, 2 additions and 2 clock cycles for one lookup operation.

5.3.4. Pipelining

Several operations mentioned in the previous sections can be executed simultaneously or in a pipelined fashion. For example, while performing the sine lookup operation, calculating the denominator of the complex division ($e = \frac{1}{|y|^2}$) can be done already. As a result, the scalar part of the Extended CMA algorithm can be executed in 21 clock cycles (see Table 1 for the allocation on the 5 ALUs). By distributing the scalar result to the other 2 MONTIUM TPs, the new steering vector $\vec{\phi}$ can be calculated by those processors simultaneously in 4 clock cycles (i.e., $\vec{\phi}_{1...4}$ calculated by one MONTIUM TP while the other calculates $\vec{\phi}_{5...8}$).

6. System level implementation

The operations explained in Section 5 are scheduled onto the platform presented in Section 4. The operations were partitioned such that the available processing power would be used as much as possible. Since the update frequency of all operations in a control loop depends on the processing delay caused by those opera-

Table 1
ALU mapping of the implemented Extended CMA algorithm.

	ALU1	ALU2	ALU3	ALU4	ALU5
1	CORDIC	CORDIC	CORDIC	CORDIC	
2	CORDIC	CORDIC	CORDIC	CORDIC	
⋮	⋮	⋮	⋮	⋮	
15	CORDIC	CORDIC	CORDIC	CORDIC	
16	CORDIC			×4	
17				sin	$ y ^2$
18				$\frac{\mu}{ y ^2}$	$ y ^4 - y ^2$
19					×2
20	Cplx div	Cplx div	Cplx div	Cplx div	
21	Cplx div	Cplx div	Cplx div	Cplx div	

tions, the processing should be optimized for low latency. Therefore, all operations in the adaptive beam steering loop are scheduled on the 3 MONTIUM TPs such that the overall latency is as small as possible.

6.1. Multi-processor scheduling

The scheduling of the partitioned operations is depicted in Fig. 12. The lighter shaded blocks indicate the processing in normal operation (i.e., beamforming and Matched Filter operation, when no adaptive steering is applied) and the darker shaded blocks indicate the processing required for the adaptive beam steering algorithm. Note that the Extended CMA operation can be started after the first clock cycle of the Matched Filter. This is possible because the last 5 taps of the FIR filter are calculated first, such that the filter output can be obtained after one clock cycle already. In the remaining 4 cycles, the other 20 filter taps are processed.

Since the MONTIUM TPs communicate via the NoC which is clocked at a 3 times higher frequency, the output of one MONTIUM TP is already available at the next MONTIUM TP at the next clock cycle. Therefore, communication does not introduce latency in the schedule. Because the Extended CMA operation cannot be parallelized further (the iterative behavior of the CORDIC algorithm requires a sequential implementation), its processing latency halts the other two MONTIUM TPs. Processor utilization could be improved by buffering antenna snapshots, such that the halted MONTIUM TPs can continue with the beamforming and matched filtering until the Extended CMA algorithm has finished.

In normal operation, where the Extended CMA algorithm is not executed for each antenna snapshot, 10 clock cycles are required for the beamforming and Matched Filter operations. Since the MONTIUM TPs in the prototyping platform presented in Section 4 are operated at 15 MHz, this means that about 1.5 Msamples per second can be processed.

6.2. Performance

To assess the performance of the implementation presented in the previous section, a synthetic scenario was defined. Assume the virtual satellite dish is mounted on the roof of a car, which has a velocity of 20 m/s (72 km/h). While moving, the virtual satellite dish is pointed at a specific satellite, receiving the broadcast currently transmitted by the satellite. The channel is assumed to be an Additive White Gaussian Noise (AWGN) channel, where the SNR equals 16 dB. Then, at a sudden moment, the car driver decides to change the driving direction such that the car gets an angular velocity of 100 Hz (80° peak to peak).⁴ The pointing of the virtual satellite dish should be updated as fast as possible to as-

⁴ Although a 80° peak to peak rotation at 100 Hz is extreme for car movement, such a scenario is great to test the robustness of the adaptive steering algorithm.

Table 2
Parameters for the moving car scenario.

Number of antenna elements	N	8
Convergence rate	μ	0.005
AWGN channel SNR		16 dB
Vehicle speed		20 m/s
Vehicle angular velocity		100 Hz sine (80° peak to peak)

sure correct reception of the broadcast stream. Table 2 presents the relevant parameters for this scenario.

The performance of the steering algorithm is analyzed by using the following three criteria:

- Extended CMA costs, as defined in Eq. (5).
- Antenna radiation pattern using the generated steering vector $\vec{\phi}$.
- Demodulation at the output of the Matched Filter.

Fig. 13 shows the results of the first two criteria. The cost function J during the scenario is shown in Fig. 13a. Here, the costs are in the range of (0.08, 0.20), which means that the maximum amplitude error equals $\sqrt{0.20} = 0.45$ and the maximum phase error equals $\arcsin(\sqrt{0.20}) = \pm 27^\circ$.⁵ Since different QPSK modulated symbols have a phase difference of at least 90°, as can be seen in Fig. 13b, with these maximum errors all symbols are still in the correct quadrant of the constellation diagram, leading to successful demodulation of the received symbols in our experiments.

Another criterion to qualify the adaptive steering algorithm is the radiation pattern of the virtual satellite dish, based on the calculated steering vector $\vec{\phi}$. The radiation pattern shows the directional sensitivity of the array antenna, as shown in Fig. 14. It can be seen that the main sensitivity is accurately steered towards the transmitter, while the sensitivity to signals impinging from other directions is 15–20 dB lower. This resembles the channel noise (16 dB) and the side-lobe suppression is enough for correct demodulation of the QPSK symbols.

6.3. Scalability

The beamforming operation consists of a vector multiplication, where a length of both vectors equals the number of antennas. Therefore, the number computations required for the beamforming operation scales linearly with the number of antennas.

Matched filtering is performed on the output of the beamformer; the number of operations is independent of the number of antennas. As a result, matched filtering has a fixed computational complexity.

The adaptive beam steering algorithm consists of a scalar part that uses the beamformer output (and therefore, has constant computational complexity) and a vector multiply-add operation of the steering vector (linearly depending on the number of antennas).

Overall, it is safe to conclude that the processing requirements of the DVB-S receiver scale linearly with the number of antennas.

7. Related work

Many reconfigurable beamforming architectures are based on bit-level programmable Field Programmable Gate Array (FPGAs), like [25,26], which perform very well as they can be optimized to the application. However, configuration times are long (millisecond to second scale), so no processing is possible during that period. The reconfigurable tiled architecture approach used in this article can be reconfigured much faster (nano-second to micro-second).

⁵ In practice, both amplitude and phase errors contribute to the costs, hence the maximum amplitude and phase errors are very likely to be lower than the figures presented here.

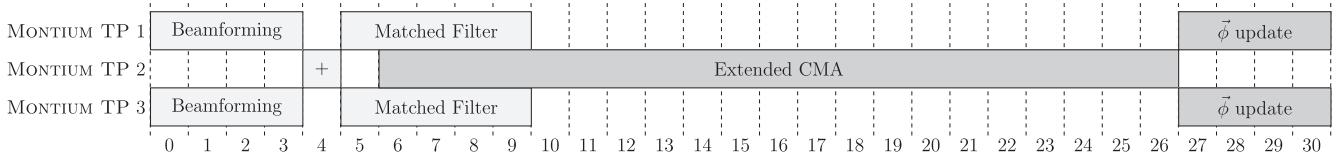


Fig. 12. Scheduling of the processing blocks on 3 MONTIUM TPs.

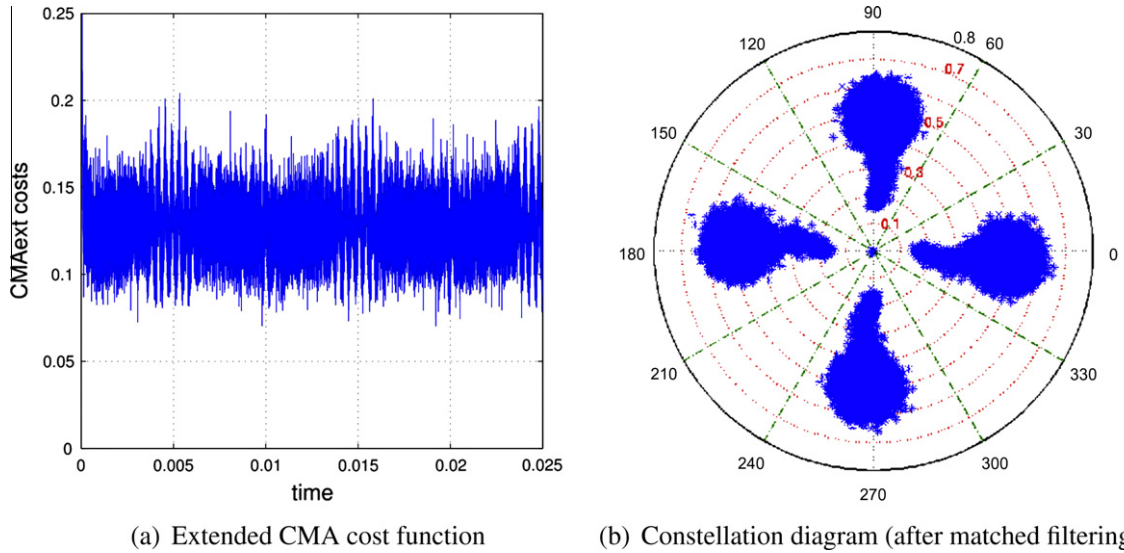


Fig. 13. Results of adaptive beam steering during the moving car scenario. For representation purposes, all amplitudes in the constellation diagram were multiplied by 0.5.

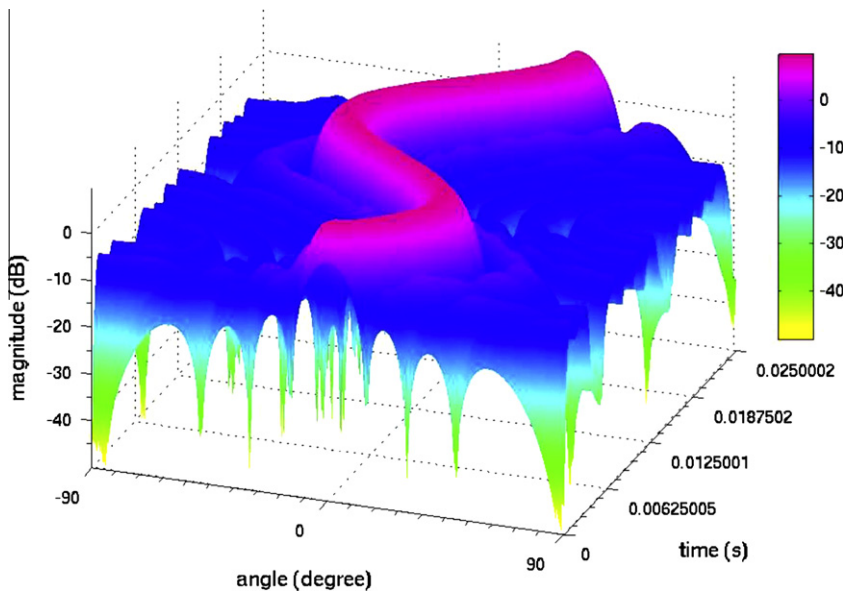


Fig. 14. Visualization of the radiation pattern during the scenario.

ond scale), where it is possible to reprogram individual processors instead of the entire chip.

The RaPiD reconfigurable architecture [27] has some similarities compared to the MONTIUM TP, but it is not designed as a tile processor. The reconfigurable beamformer processor proposed by Hwang [28] can be reconfigured in case processors become faulty. However, they present no performance figures. The beamforming architecture proposed by Sarrigeorgidis [29] shows some resemblance with Hwang’s architecture. Its performance normalized to power is about 20 MOPS/mW. With its 5 large ALUs, the MONTIUM

TP can typically execute about 15 operations per clock cycle, resulting in a normalized performance of about 30 MOPS/mW. The CA μ S architecture presented in [30] is a hybrid architecture consisting of an FPGA and a Digital Signal Processor (DSP). The execution times of the kernels presented are close to those of the MONTIUM TP. However, an FPGA has a much higher energy dissipation than the MONTIUM TP [17]. Raytheon’s MONARCH processor [31] was designed to focus on maximum achievable processing power. It consists of a reconfigurable data path which is controlled by 6 Reduced Instruction Set Computer (RISC) processors. If its pro-

cessing power is insufficient, multiple MONARCH processors can be connected to form a larger processing array. Although it is operated at a clock frequency several times higher than the MONTIUM TP, the MONARCH's normalized performance equals 3–6 MOPS/mW, which is much lower than the MONTIUM TP.

In contrast to each of the multi-processor architectures mentioned above, the architecture presented in this article was developed as a generic stream processing platform that has been used for consumer electronics applications. Therefore, it was not optimized for the phased array processing application, which is the case for each of the above architectures. By showing the efficiency of our generic stream processing platform, its applicability for a variety of algorithms is proven.

8. Conclusion

Phased array processing requires a high performance architecture that is capable of combining many input data streams at high data rates. In this article, we proposed a reconfigurable multi-processor architecture consisting of 3 MONTIUM TP processors and a LEON2 host processor, as a generic beamforming platform.

The modulation scheme used for Digital Video Broadcast for Satellite (DVB-S) enables tracking of transmitters using a blind beamforming algorithm. We used the Extended Constant Modulus Algorithm (CMA) to create an adaptive beamforming application that can be used for satellite tracking in mobile situations. The entire digital processing chain (consisting of antenna processing, beamforming and beam steering) was mapped to the generic beamforming platform. On average, the beamforming operation and matched filtering can be done in 10 clock cycles on the prototype platform, allowing for a symbol rate of 1.5 Msymbol/s. An ASIC realization of the prototype could operate at a higher frequency, increasing the maximum symbol rate considerably.

The processing requirements roughly scale linearly with the number of antennas and the number of beams. Due to the use of a Network-on-Chip, our reconfigurable multi-processor architecture can be scaled to provide the processing required for either applications using small number of antennas (e.g. wireless communications) or large phased array architectures (e.g. next generation radar systems). With the scalable implementation presented in this article, the goal to design a low-cost and low-power phased array system has been met partially. Using identical chips as small building blocks for our architecture, the production costs for individual chips can be decreased due to large volumes. However, note that the main costs for a phased array antenna are added due the analog front-ends, each of which must have the same noise figure as the single dish antenna front-end. Therefore, the low-cost design goal for a phased array antenna cannot be shown without considering the analog front-end.

9. Future research

The Extended CMA algorithm iteratively generates a steering vector by minimizing the error in the received symbols after beamforming. If, however, it is desired to use a customized radiation pattern, the current algorithm cannot be used. Other implementations of the CMA algorithm might be useful for this purpose.

Although the current implementation of the Extended CMA algorithm proved to be not very computationally intensive, the use of a COordinate Rotation DIgital Computer (CORDIC) implementation causes latency of the algorithm and therefore, two of the three MONTIUM TPs are temporarily stalled during the calculation of the new steering vector. Future research will focus on reducing the latency of the Extended CMA algorithm.

Modern satellite receivers are already supporting the DVB-S2 standard [32], which is backwards compatible with the DVB-S standard used in this work. The maximum channel size was scaled up to 86 Mbps, requiring advanced data compression and error coding. It supports adaptive channel coding that depends on the quality of the transmission channel. Therefore, next to Quadrature Phase Shift Keying (QPSK) also 8PSK, 16APSK and 32APSK modulation are used. Since these modulation techniques do not have the same properties as QPSK does, Extended CMA cannot be used in its current form. Further research is needed to determine if CMA can also be used for the remaining modulation schemes of DVB-S2.

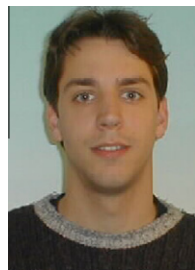
Acknowledgment

The authors thank Fasil Tadesse for his contribution in the system level implementation of the processing chain.

References

- [1] H.J. Visser, *Array and Phased Array Antenna Basics*, Wiley, Chichester, West Sussex, UK, 2005.
- [2] M.I. Skolnik, *Introduction to Radar Systems*, third ed., McGraw-Hill, New York, NY, USA, 2001.
- [3] European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services*, EN 300 421 v1.1.2, August 1997 [cited November 25, 2009]. <http://www.etsi.org/deliver/etsi_en/300400_300499/300421/01.01.02_60/en_300421v01010102p.pdf>.
- [4] KVH Industries, Inc., *Making the world as mobile as you are*, January 2010 [cited January 25, 2010]. <<http://www.kvh.com>>.
- [5] K.C.H. Blom, *DVB-S signal tracking techniques for mobile phased arrays*, Master's thesis, University of Twente, Enschede, The Netherlands, December 2009.
- [6] B.H. Allen, M. Ghavami, *Adaptive Array Systems, Fundamentals and Applications*, Wiley, Chichester, West Sussex, UK, 2005.
- [7] R.H. Roy, A.J. Paulraj, T. Kailath, *Esprit – a subspace rotation approach to estimation of parameters of cisoids in noise*, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34 (5) (1986) 1340–1342.
- [8] R.O. Schmidt, *Multiple emitter location and signal parameter estimation*, *IEEE Transactions on Antennas and Propagation* AP-34 (3) (1986) 276–280, doi:10.1109/TAP.1986.1143830.
- [9] J.R. Treichler, B.G. Agee, *A new approach to multipath correction of constant modulus signals*, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 31 (2) (1983) 459–472.
- [10] Z. Xu, *New cost function for blind estimation of M-PSK signals*, in: *IEEE Wireless Communications and Networking Conference*, vol. 3, 2000, pp. 1501–1505. doi:10.1109/WCNC.2000.904857.
- [11] K.C.H. Blom, M.D. van de Burgwal, K.C. Rovers, A.B.J. Kokkeler, G.J.M. Smit, *DVB-S signal tracking techniques for mobile phased arrays*, in: *Proceedings of the IEEE 72nd Vehicular Technology Conference*, 2010, 2010, p. 5. doi:10.1109/VETEFC.2010.5594146.
- [12] K.C. Rovers, M.D. van de Burgwal, A.B.J. Kokkeler, G.J.M. Smit, *Rationale for and design of a generic tiled hierarchical phased array beamforming architecture*, in: *Proceedings of the 18th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2007)*, Technology Foundation, Utrecht, The Netherlands, 2007, pp. 160–168. doi:http://purl.org/utwente/64539.
- [13] Gaisler, Processors – aeroflex gaisler [cited January 6, 2011]. <<http://www.gaisler.com/leonmain.html>>.
- [14] G.J.M. Smit, A.B.J. Kokkeler, P.T. Wolkotte, P.K.F. Hölzenspies, M.D. van de Burgwal, P.M. Heysters, *The Chameleon architecture for streaming DSP applications*, *EURASIP Journal on Embedded Systems* (2007) 78082, doi:10.1155/2007/78082.
- [15] G.J.M. Smit, A.B.J. Kokkeler, P.T. Wolkotte, M.D. van de Burgwal, *Multi-core architectures and streaming applications*, in: I.I. Mandou, A.A. Kennings (Eds.), *Proceedings of the Tenth International Workshop on System-Level Interconnect Prediction (SLIP 2008)*, ACM, New York, NY, USA, 2008, pp. 35–42. doi:10.1145/1353610.1353618.
- [16] P.T. Wolkotte, G.J.M. Smit, G.K. Rauwerda, L.T. Smit, *An energy-efficient reconfigurable circuit switched network-on-chip*, in: *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 155–162. doi:10.1109/IPDPS.2005.95.
- [17] P.M. Heysters, *Coarse-grained reconfigurable processors – flexibility meets efficiency*, Ph.D. thesis, University of Twente, Enschede, The Netherlands, September 2004.
- [18] P.M. Heysters, G.J.M. Smit, *Mapping of DSP algorithms on the Montium architecture*, in: *Proceedings of the 17th IEEE International Parallel and Distributed Processing Symposium (IPDPS'03)* [33], p. 6. doi:10.1109/IPDPS.2003.1213333.

- [19] G.K. Rauwerda, P.M. Heysters, G.J.M. Smit, Towards software defined radios using coarse-grained reconfigurable hardware, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16 (1) (2008) 3–13, doi:10.1109/TVLSI.2007.912075.
- [20] M.D. van de Burgwal, G.J.M. Smit, G.K. Rauwerda, P.M. Heysters, Hydra: an energy-efficient and reconfigurable network interface, in: T.P. Plaks (Ed.), *Proceedings of the International Conference on Engineering of Reconfigurable Systems & Algorithms (ERSA'06)*, CSREA Press, Las Vegas, NV, USA, 2006, pp. 171–177. doi:http://purl.org/utwente/62883.
- [21] J. Volder, The CORDIC computing technique, in: *Proceedings of the AFIPS Joint Computer Conferences*, ACM, New York, NY, USA, 1959, pp. 257–261. doi:10.1145/1457838.145788.
- [22] J.S. Walther, A unified algorithm for elementary functions, in: *Proceedings of the AFIPS Spring Joint Computer Conferences*, ACM, New York, NY, USA, 1971, pp. 379–385. doi:10.1145/1478786.1478840.
- [23] R. Andracka, A survey of CORDIC algorithms for FPGA based computers, in: *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, ACM, New York, NY, USA, 1998, pp. 191–200. doi:10.1145/275107.275139.
- [24] S. Hitotumatu, Complex arithmetic through CORDIC, *Kōdai Mathematical Seminar Reports* 26 (2–3) (1975) 176–186, doi:10.2996/kmj/1138846999.
- [25] J. Ou, V.K. Prasanna, Parameterized and energy efficient adaptive beamforming on FPGAs using MATLAB/Simulink, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 5, IEEE Signal Processing Society, Los Alamitos, CA, USA, 2004, pp. V – 181–184. doi:10.1109/ICASSP.2004.1327077.
- [26] D. Theodoropoulos, G. Kuzmanov, G. Gaydadjiev, A reconfigurable beamformer for audio applications, in: *IEEE 7th Symposium on Application Specific Processors (SASP '09)*, IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 80–87. doi:10.1109/SASP.2009.5226341.
- [27] C. Ebeling, C. Fisher, G. Xing, M. Shen, H. Liu, Implementing an OFDM receiver on the RaPiD reconfigurable architecture, *IEEE Transactions on Computers* 53 (11) (2004) 1436–1448, doi:10.1109/TC.2004.98.
- [28] D.K. Hwang, T.L. Wernimont, W.K. Fuchs, Evaluation of a reconfigurable architecture for digital beamforming using the OODRA workbench, in: *Proceedings of the 26th ACM/IEEE Design Automation Conference (DAC '89)*, ACM, New York, NY, USA, 1989, pp. 614–617. doi:10.1145/74382.7448.
- [29] K. Sarrigeorgidis, J. Rabaey, Massively parallel wireless reconfigurable processor architecture and programming, in: *Proceedings of the 17th IEEE International Parallel and Distributed Processing Symposium (IPDPS'03)* [33], p. 170.2. doi:10.1109/IPDPS.2003.1213313.
- [30] S. Scalera, M. Falco, B.E. Nelson, A reconfigurable computing architecture for microsensors, in: *Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, IEEE Computer Society, Los Alamitos, CA, USA, 2000, pp. 59–67, doi:10.1109/FPGA.2000.903393.
- [31] M.D. Vahey, J.J. Granacki, L.J. Lewins, D. Davidoff, J.T. Draper, C.S. Steele, G.K. Groves, M. Kramer, J. LaCoss, K. Prager, J. Kulp, C. Channell, MONARCH: a first generation polymorphic computing processor, in: *10th High Performance Embedded Computing Workshop*, 2006.
- [32] European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), EN 302 307 v1.2.1, August 2009 [cited April 21, 2010]. <http://www.etsi.org/deliver/etsi_en/302300_302399/302307/01_02_01_60/en_302307v010201p.pdf>.
- [33] IEEE Computer Society, Washington, DC, USA, 2003.



Marcel van de Burgwal received his M.Sc. degree in Computer Science from the University of Twente, Enschede, The Netherlands, in 2005. He pursued his Ph.D. studies in the Computer Architectures for Embedded Systems (CAES) group at the University of Twente from 2005 to 2010. During this period he cooperated in different science projects, focusing on hardware/software co-design of multi-processor systems-on-chip. His research interests include energy-efficient architectures, networks-on-chip, multi-media applications design and tools. Since 2011, he works at Locamation, Hengelo, The Netherlands as a digital

hardware designer.



Kenneth C. Rovers was born in Rotterdam, The Netherlands in 1978. He received the M.Sc. degree in Electrical Engineering and the M.Sc. in Computer Science in 2006 from the University of Twente, The Netherlands. His masters thesis was on the system design of RF front-ends. He is currently working towards a Ph.D. degree in the area of model-based design of embedded systems at the Computer Architecture for Embedded Systems (CAES) group at the University of Twente. His current research interests include functional programming, reconfigurable tiled architectures and dataflow processors.



Koen Blom received his Master's degree in Embedded Systems from the University of Twente in 2009. Currently he is working at the University of Twente as a Ph.D. in the field of Digital Signal Processing for (acoustic) underwater communication. His research activities are part of the SeaSTAR Underwater Monitoring Platform project.



Andr Kokkeler has worked more than 6 years at Ericsson as a system engineer and 8 years at The Netherlands foundation for research in astronomy (ASTRON) as a scientific project manager. In 2003 he joined the University of Twente. In 2005 he obtained his Ph.D. degree on the thesis Analog-digital Codesign using Coarse Quantization. He has a background in telecommunication, mixed-signal design and signal processing. Currently, his main interest lies in the area of applying low-power design techniques for computationally intensive applications. The emphasis is on reconfigurable architectures for streaming applications. He is

involved in research projects, sponsored by the Dutch and European governments and industry.



Gerard J.M. Smit received his M.Sc. degree in Electrical Engineering from the University of Twente, Enschede, The Netherlands. He finished his Ph.D. thesis entitled *The Design of Central Switch Communication Systems for Multimedia Applications* in 1994. He currently is a Full Professor with the faculty of EEMCS, University of Twente, leading the CAES chair, where he is responsible for a number of research projects sponsored by the EC, industry and Dutch government in the field of streaming applications and efficient reconfigurable systems. After receiving the M.Sc. degree, he worked for four years at the Research Laboratory of Oce, Venlo, the

Netherlands. In 1994, he was a Visiting Researcher with the Computer Laboratory, Cambridge University, Cambridge, UK and, in 1998, he was a Visiting Researcher with Lucent Technologies Bell Labs Innovations, Murray Hill, NJ. Since 1999, he has been leading the Chameleon group, which investigates new hardware and software architectures for energy-efficient systems. Currently his research interests include low-power communication, and reconfigurable architectures for energy reduction. He co-author of more than 200 research papers (see caes.ewi.utwente.nl). The research has resulted in three start-up companies: Recore systems, Smart Sign solutions and HOMA Software.