

# Adaptive Beamforming using the Reconfigurable MONTIUM TP

Marcel D. van de Burgwal, Kenneth C. Rovers, Koen C.H. Blom, André B.J. Kokkeler, and Gerard J.M. Smit  
 Faculty of Electrical Engineering, Math and Computer Science

University of Twente  
 Enschede, The Netherlands

Email: {m.d.vandeburgwal, k.c.rovers, k.c.blom, a.b.j.kokkeler, g.j.m.smit}@utwente.nl

**Abstract**—Until a decade ago, the concept of phased array beamforming was mainly implemented with mechanical or analog solutions. Today, digital hardware has become powerful enough to perform the massive number of operations required for real-time digital beamforming. While more and more applications are using beamforming to improve the communication channel utilization both in space and frequency, many dedicated digital architectures are proposed for the processing. By using a reconfigurable architecture, the same hardware platform can be reused for different applications with different processing needs. In this paper, we present a reconfigurable Multi-processor System-on-Chip based solution for phased array processing that supports advanced tracking mechanisms to continuously receive signals with a mobile receiver. An adaptive beamformer for DVB-S satellite reception is presented, that uses a Constant Modulus Algorithm to track satellites. The processing of a receiver with 64 antennas and 3 beams is mapped on a reconfigurable processor named MONTIUM TP. The total implementation of such a receiver requires about 570 clock cycles on a single MONTIUM TP, but can also be partitioned over multiple MONTIUM TPs to support larger phased arrays.

**Index Terms**—Adaptive beamforming; phased array; reconfigurable processor; MPSoC

## I. INTRODUCTION

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 and for 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 for more applications. 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 an application). Conventional phased array receivers typically use a large amount of dedicated central processing hardware, making

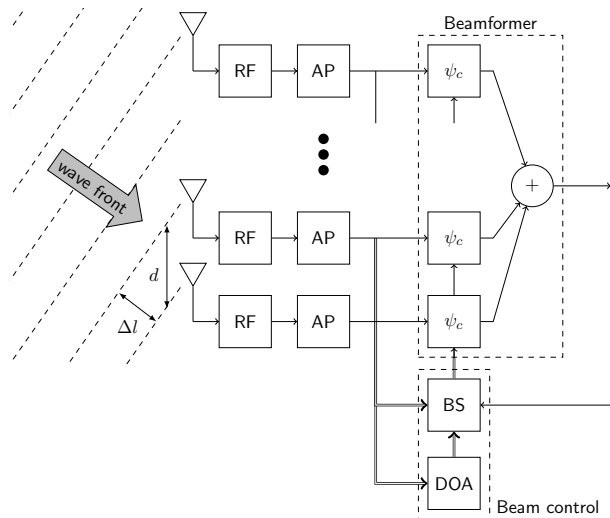


Fig. 1. Generic phased array receiver

the system neither scalable nor power efficient [2].

After an introduction to phased array systems and their different applications in Section II, we will provide a more thorough discussion on the required processing and its complexity in Section III. The processing is mapped to a tiled architecture using reconfigurable processors (presented in Section IV). Next, we will show how the algorithm is implemented in Section V.

## II. PHASED ARRAY SYSTEMS AND APPLICATIONS

The system blocks of a generic phased array system are shown in Fig. 1. In a phased array receiver, signals are received at multiple antennas. After the Radio Frequency (RF) front end for each antenna, Antenna Processing (AP) is applied for calibration or equalization purposes (to correct for electrical or mechanical distortions of the front-end and the wireless channel). The signals are then combined by the beamforming processing (beamformer) 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. Beamsteering (BS) refers to changing the shape and direction of the formed beam by changing the gain and delay of the antenna signals before summation.

Assume the phased array consists of antennas placed at a distance  $d$  apart, while a wavefront arrives at an angle  $\vartheta$  incident

TABLE I  
OVERVIEW OF APPLICATIONS WHERE BEAMFORMING CAN BE BENEFICIAL

	DVB-S	Radar	Radio astronomy	Wireless base stations
# Antennas	256	4096	8672	64
# Beams	3	20	24	32
Frequency (GHz)	10–13	7–13	0.4–2.8	2–6
Bandwidth (MHz)	50	100	100	10–100
SNR (dB)	16	100	70	30
AD bits	4	16	12	10

Figures for radar, radio astronomy and wireless base stations are based on current requirements and extrapolated to the near future.

to the array. The wavefront travels a distance  $\Delta l = d \cdot \sin(\vartheta)$  further to the next antenna, which results in a time delay  $\Delta t = \frac{\Delta l}{c}$  between the signals (where  $c$  is the propagation speed of radio waves). If the signal is a narrowband signal, this time delay results in a phase shift ( $\Delta\psi = \omega \cdot \Delta t$ ) giving rise to the term ‘phased array’. By correcting the delay, the direction of maximum sensitivity is steered [1].

### A. Applications

Beamforming can be beneficial for any RF system, such as Digital Video Broadcast for Satellite (DVB-S), radar, radio astronomy and wireless communications. A cost-effective solution could be to design a single generic beamforming architecture that is flexible and modular such that each of these applications could be supported. Therefore, a comparison of the requirements of these applications is given in Table I.

1) *DVB-S*: The DVB-S [3] standard specifies a frequency of 10.7 to 12.75 GHz, a Signal to Noise Ratio (SNR) of 16 dB, a channel bandwidth of 36 MHz (effective bandwidth used is 50 MHz due to pulse shaping filter roll-off) and satellites are at least  $5^\circ$  apart. Conventional dish-based systems must be steered mechanically. This makes the dish unsuitable for mounting on vehicles that are moving, such as cars or yachts. A phased array system can therefore be beneficial. Broadcasts from multiple satellites can be received simultaneously by enabling for two or three independent beams. This is useful, when multiple users want to receive signals from different satellites simultaneously.

2) *Radar*: Radar is a specialized application aimed at detecting and locating reflecting objects or targets. It is for example used for scanning and tracking or guiding objects. Future phased array radar systems require a large array size (a few thousand antennas), a high SNR (100 dB) and a high sample rate (200 MHz). A suitable band for realizing systems with such spectral bandwidth requirements is the region between 7 to 13 GHz. At each moment in time, there might be multiple interesting objects in sight. Therefore, a typical tracking radar should be able to track tens of objects, which requires as many independent beams.

3) *Radio Astronomy*: The aim of radio astronomy is to produce images of celestial objects. As the objects of interest are very distant, the beam width must be as narrow as possible. This requires a very large array with accurately calibrated

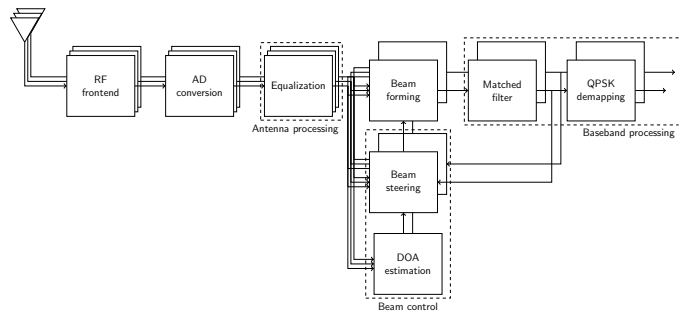


Fig. 2. Main system blocks in the phased array receiver

antenna processing. Furthermore as the received signals are very weak, low noise figures are important.

4) *Wireless base stations*: During the last few years, multi-antenna techniques have been included in new Software Defined Radio (SDR) standards. Such Multiple Input/Multiple Output (MIMO) systems allow for higher channel utilization, as their transmission techniques are designed for both spatial and spectral optimization. Beamforming is one of the techniques that can be used to optimize the spatial use of the spectrum. Nowadays, receivers typically use 2 or 3 antenna elements at most. Adding more antenna elements implies that multiple additional front-ends are to be included, which makes a portable receiver less compact and efficient. For the base station, however, beamforming is a useful technique as the transmitted power can be spatially controlled such that the beam is focused at receivers. Additionally, the same spectrum may be reused spatially by steering a beam to each receiver.

## III. PHASED ARRAY PROCESSING AND COMPLEXITY

In a dynamic environment where transmitters are continuously moving with respect to the array, adaptive processing is required to detect and follow transmitters. One example would be the application of the DVB-S satellite receiver mentioned in section II mounted on a vehicle, for example on the roof of a car or on the cabin of a yacht. 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 AD conversion, we will shortly describe functionality and complexity.

### A. Antenna processing

For accurate beamforming, it is important that the gain and delay of each antenna up to the beamforming is equal. To realize this, antenna calibration and/or channel equalization has to be applied. This can be implemented by an  $(\mathcal{F}_{\text{eq}} - 1)^{\text{th}}$  order Finite Impulse Response (FIR) filter, where the minimum order can be determined based on the required SNR and filter roll-off. For each filter tap, one multiply-accumulate has to be calculated. Hence, the total computational complexity of one FIR filter equals  $\mathcal{F}_{\text{eq}}$  multiply-accumulates. As a filter is needed for each antenna, the amount of processing can easily become as large as the beamforming itself. However, it is independent of the number of beams that are formed.

## B. Beamforming

To implement phase shifting in the digital domain, a FIR filter can be used, which consists of complex multiplications followed by addition. In the narrowband case, a single complex multiplication for each antenna can be used. The phase delay correction with a complex multiplication has the advantage that it is flexible with respect to the beam-shape and its angle. In case of multiple beams, each beam can be pointed in a different direction independently, while its shape may differ from other beams. However, the costs for this flexibility are large as processing costs are constant per beam.

## C. Baseband processing

The modulation technique used in DVB-S for transmitting symbols is Quadrature Phase Shift Keying (QPSK) [3]. QPSK modulation maintains a constant modulus, while the information is added by applying a multiple of  $\frac{\pi}{2}$  shift in phase to the carrier frequency. When switching instantaneously between two symbols, high frequency components occur in the transmitted signal due to discontinuity in the phase. The transmitter uses a pulse shaping filter to compress the signal into a slightly wider frequency band. At the receiving side, a matched filter is then used to decompress the signal such that the modulation information is reconstructed. The matched filter can be implemented using a FIR filter consisting of  $\mathcal{F}_{mf}$  taps. Since this filter is applied to the output of the beamformer, its complexity only depends on the number of beams  $\mathcal{B}$ .

## D. Beam control

The pointing direction and shape of the beam are steered by the beam control part. Since the receiver is continuously moving, an adaptive beamsteering algorithm is required. There are 3 classes of adaptive beamsteering algorithms [4]. *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, when multi-path effects can occur, often pilot symbols are added to synchronize with the received signal. The third class consists of so-called *blind beamforming* algorithms. These algorithms use structural or statistical properties of the received signal to correct the beam direction.

In the initial situation where the satellites have not been detected yet, a search action has to be done to find the location of possible transmitters. This can be done with a so-called 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 [5] and MUSIC [6]. The disadvantage of these algorithms is their high complexity of  $O(\mathcal{N}^4)$  (where  $\mathcal{N}$  is the number of antennas) 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

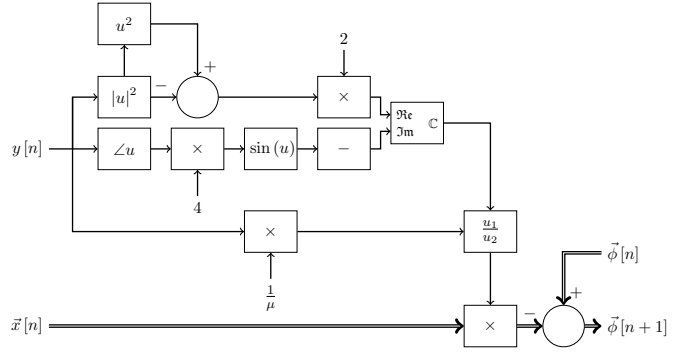


Fig. 3. Block diagram of the CMA adaptive beamsteering algorithm

avoided. Direction of Arrival is not in scope of this article and therefore, for the remainder of this article we will assume that initial locations of transmitters are known.

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 so-called Constant Modulus Algorithm (CMA) can be used efficiently [7]. Xu proposed an extension to CMA that allows for correction of phase deviations [8]. 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. This is done by applying an iterative gradient descent method to decrease the cost function  $J$ , which is given by Equation 1:

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

This can be rewritten to (2) [8]:

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

After optimization, we get:

$$\vec{\phi}[n+1] = \vec{\phi}[n] - \mu \cdot \frac{2(|y|^4 - |y|^2) - j\sin(4\angle y)}{y} \cdot \vec{x} \quad (3)$$

which is also shown in Fig. 3.

CMA scales linearly with the number of antennas  $\mathcal{N}$ , because  $\vec{x}$  and  $\vec{\phi}$  are vectors of length  $\mathcal{N}$ . The calculation of  $\mu \cdot \frac{2(|y|^4 - |y|^2) - j\sin(4\angle y)}{y}$  only consists of scalar operations and therefore requires a fixed number of operations.

Using the steering vector  $\vec{\phi}$ , the beam pattern tracks the transmitter while interferers are automatically suppressed.

TABLE II  
COMPLEXITY PER BASIC OPERATION

	Operation	Complexity
Antenna processing	Equalization	$O(\mathcal{N}\mathcal{F}_{\text{eq}})$
Beamforming	Phase shift	$O(\mathcal{N}\mathcal{B})$
Beam steering	CMA	$O(\mathcal{N}\mathcal{B})$
Baseband processing	Matched filter	$O(\mathcal{B}\mathcal{F}_{\text{mf}})$

Multiple transmitters can be tracked by adding a CMA algorithm for each transmitter. As a result, the total complexity for tracking  $\mathcal{B}$  beams with CMA equals  $O(\mathcal{N}\mathcal{B})$ .

### E. Algorithm complexity

An overview of the complexity of the antenna processing and beamforming is given in Table II. The table shows the parameter dependencies for the figures mentioned in the previous sections.

## IV. RECONFIGURABLE TILED ARCHITECTURES

Phased array processing can be characterized as a streaming application with high data rates and processing requirements, but a regular processing structure. Because of costs, complexity, dependability and scalability reasons a design with mostly identical components is preferred. Because a scalable and regular solution is needed, a tiled reconfigurable architecture is proposed [9]. We aim at a processing architecture which is flexible enough to support multiple methods of beamforming (based on phase shifting, time delay or Fast Fourier Transform (FFT)), as well as beamsteering and beam-control.

A reconfigurable processing array provides flexibility and has a number of advantages. For example, we can use only part of the array or create multiple sub-arrays to save energy or increase the lifetime. Moreover, graceful degradation is 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.

### A. Montium Tile Processor

The MONTIUM TP is an example of a coarse-grained reconfigurable processor [10] developed by Recore Systems<sup>1</sup>. The MONTIUM TP targets the Digital Signal Processing (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 [11]. It is typically used in multi-processor systems, for example in the Annabelle MPSoC [12]. Its template based design allows for customization of architectural properties like data path width (16-bit by default), targeted clock frequency (100 MHz for 90nm technology) and processing capacity (5 parallel Arithmetic Logic Units (ALUs) for the reference design). This reference design has a silicon area of approximately 2 mm<sup>2</sup> and a power consumption of approximately 550 $\mu$ W/MHz.

<sup>1</sup><http://www.recoresystems.com>

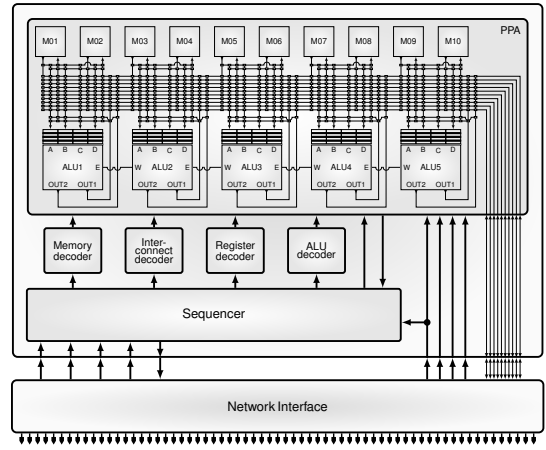


Fig. 4. Montium Tile Processor

1) *Program control*: Inside the MONTIUM TP, shown in Fig. 4, 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. As there is no interaction between the data path and the instruction program, the kernel execution is fully deterministic.

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 selection of ALU operands in the register file is done by the *register decoder* and the ALU instruction itself is decoded using the *ALU decoder*.

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 ALUs 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 sent to the west output and both outputs connected to the interconnect respectively. Fig. 5 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*

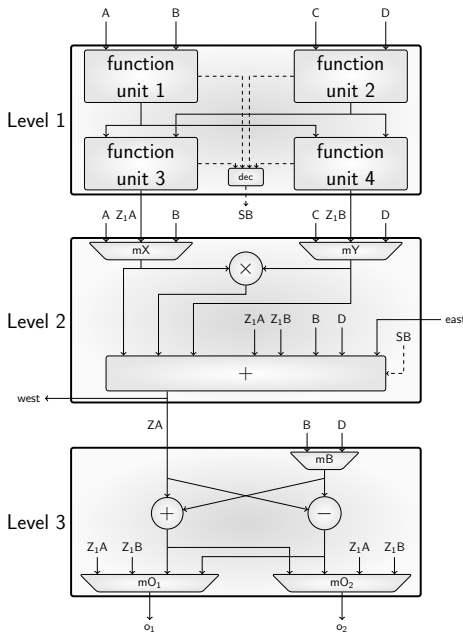


Fig. 5. Structure of one MONTIUM TP ALU

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* modus (i.e., they operate on the 16 rightmost bits) or in *1.15 fixed point* modus (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.

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 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.

## V. IMPLEMENTATION AND RESULTS

In the previous sections, we presented the algorithms that were selected for phased array processing. As can be seen in Fig. 2, the antenna processing needs to be performed for both the beamforming and beam control. By using reconfigurable hardware, the beamforming and beamsteering functionality can be replaced temporarily by DOA estimation. Here we present the implementation of beamforming and tracking, with the assumption that the initial positions of transmitters are known.

As shown in [13], the update rate of the beam control algorithms can be decreased considerably before symbol errors start to occur. Therefore, for the remainder of this article we assume a beam update rate of  $\rho = \frac{1}{250} = 0.004$  updates per antenna sample. We mapped the digital beamsteering and beamforming to an MPSoC architecture based on the MONTIUM TP architecture as presented in Section IV and observed the processing requirements for a DVB-S receiver using  $\mathcal{N} = 64$  antennas. The antenna signals are sampled by 50 MHz  $I/Q$  ADCs, such that they produce 50 Msamples/s. Antenna front-end errors are corrected by an equalization filter, which is implemented by a complex FIR filter of  $\mathcal{F}_{eq} = 5$  taps.

### A. Antenna processing and beamforming

The complex FIR filter required to implement the equalization filter can be executed on the MONTIUM TP in 5 clock cycles per sample per antenna (i.e. one tap mapped on 4 multipliers) [14]. So, in total for all antennas  $64 * 5 = 320$  clock cycles are required. The beamforming operation requires one complex multiplication and one addition per beam per sample per antenna (which can be executed in 1 clock cycle for the MONTIUM TP), so 64 clock cycles per beam. Hence, for 3 beams the antenna processing and beamforming together can be executed in  $320 + 3 * 64 = 512$  clock cycles per sample. Luckily, almost all operations can be done in parallel. Therefore, the 512 clock cycles required to process one sample can be divided over multiple MONTIUM TPs. Since the MONTIUM TP can execute 2 instructions per sampling period, at least 256 MONTIUM TPs are required to enable real-time processing.

### B. Baseband processing

The matched filter is implemented using two  $\mathcal{F}_{mf} = 9$  taps FIR filters for the  $I$  and  $Q$  parts of the beamformer output. Hence, for 3 beams this part of the baseband requires  $3 * 2 * 9 = 54$  MAC operations per antenna sample.

### C. Beam steering

A schematic overview of the CMA algorithm is given in Fig. 3. Some arithmetic operations can be mapped on the MONTIUM TP ALUs directly, but three operations are required which do not directly fit. These operations are explained in the following sections.

1) *Transforming Cartesian coordinates to polar coordinates:* The conversion from Cartesian coordinates  $(x, y)$  to polar coordinates  $(r, \theta)$  and back requires some goniometric operations. These could be implemented by a large Lookup Table (LUT), at the consequence of a limited accuracy. A more efficient and accurate approach is the COordinate Rotation DIGital Computer (CORDIC) algorithm [15]. We mapped the algorithm, as described in [16], to the MONTIUM TP. For the conversion from Cartesian to polar coordinates, the vectoring mode is used. The equations for the vectoring mode are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \quad (4)$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i} \quad (5)$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (6)$$

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

$$x_n = A_n \sqrt{x_0^2 + y_0^2} \quad (7)$$

$$y_n = 0 \quad (8)$$

$$z_n = z_0 + \tan^{-1}\left(\frac{y_0}{x_0}\right) \quad (9)$$

$$A_n = \prod_n \sqrt{1 + 2^{-2i}} \quad (10)$$

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

The mapping of these equations is shown in Fig. 6. 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 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 read value to the register file for ALU 3. Hence, reading these constants does not require any additional clock cycles. Using this mapping, all three CORDIC equations can be calculated in a single clock cycle.

The results of the implemented algorithm are shown in Fig. 7. As can be seen in Fig. 7, each iteration yields 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 Equation 4 to Equation 6 are only valid for rotation angles between  $-\frac{\pi}{2}$  and  $\frac{\pi}{2}$ . 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 Equation 4 to Equation 6. The MONTIUM TP implementation of the initial

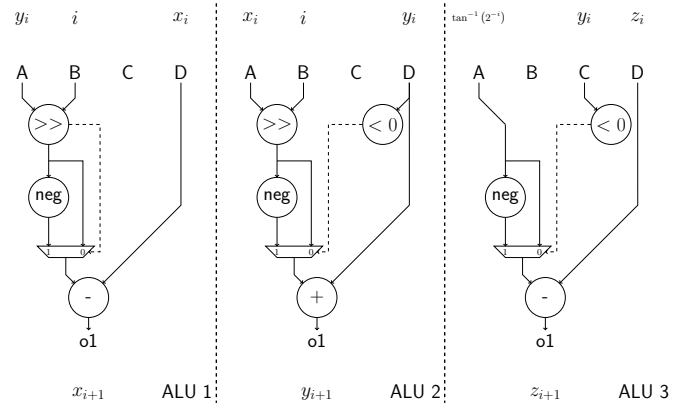


Fig. 6. Mapping of CORDIC equations on 3 MONTIUM TP ALUs

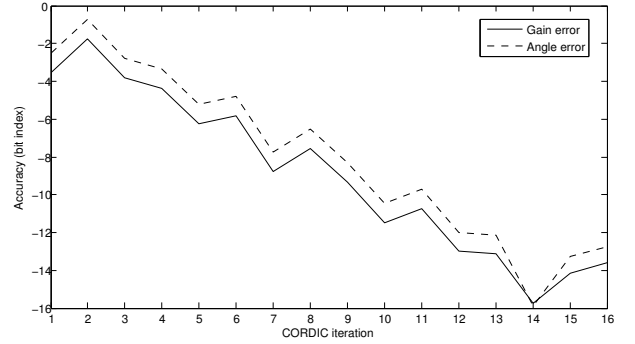


Fig. 7. Error between MONTIUM TP result and input value (in bits) after each CORDIC iteration

equations is comparable to the mapping presented for the regular CORDIC operations Equation 4 to Equation 6. Hence, the calculation of the initial equations can be done in one additional iteration.

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. Such a lookup only requires 2 clock cycles, which is much less compared to running a complete CORDIC operation.

3) *Complex division:* The complex division could be implemented by using 2 CORDIC operations, one real division and 2 multiplications [17], which is useful for implementation at multiplier-limited architectures. 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:

$$\frac{a + jb}{c + jd} = \frac{a + jb}{c + jd} \cdot \frac{c - jd}{c - jd} = \frac{ac + bd}{c^2 + d^2} + j \frac{bc - ad}{c^2 + d^2} \quad (11)$$

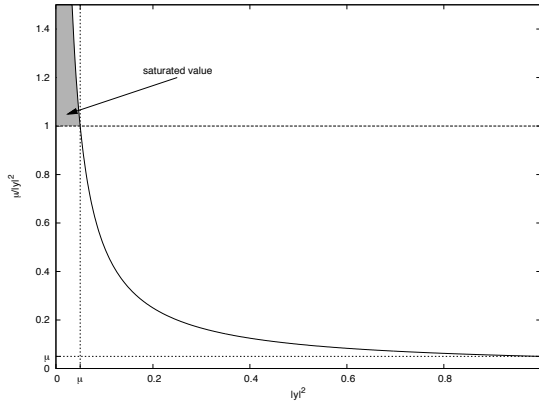


Fig. 8. Contents of the LUT for calculation of  $\frac{\mu}{|y|^2}$

Now define  $e = \frac{1}{c^2+d^2}$ . After substitution in (11), we get:

$$\frac{a + jb}{c + jd} = \dots = (ac + bd) \cdot e + j(bc - ad) \cdot e \quad (12)$$

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 (3),  $X$  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. 3, the calculation of  $|y|^2$  is already done. Its inverse can be calculated efficiently by using a LUT, similar to the sine calculation. The values of  $\frac{1}{|y|^2}$  with  $|y|^2 \in [0, \dots, 1)$  are in the range of  $\langle 1, \dots, \infty \rangle$ , 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  $\mu$  (see Fig.3) 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. 8). We use a LUT with 512 entries to calculate the inverse of  $|y|^2$ . For such a LUT, the first 3 entries are saturated (since  $\frac{0 \dots 2}{512} < \mu$ ). However, since the CMA algorithm is used to normalize  $|y|^2$  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 requires 6 multiplications, 2 additions and 2 clock cycles for one lookup operation.

#### D. Implementation Results

A summary of the numbers presented in the sections before is given in Table III. The table presents the processing costs in terms of clock cycles per antenna sample. Due to its low update rate, the beam steering costs are neglectible compared to the antenna processing, beamforming and baseband processing.

The implementation costs in terms of program memory size are also very low. The configuration data required to program CMA into the MONTIUM TP takes 710 bytes. Such a file can be loaded into the MONTIUM TP in 3.55  $\mu$ s. The precalculated LUTs for calculation of the sine and inverse both consist of

TABLE III  
REQUIRED MONTIUM TP CLOCK CYCLES PER INPUT SAMPLE  
USING 64 ANTENNAS AND FORMING 3 BEAMS

	Operation	Clock cycles
Antenna processing	Equalization filter	320
Beamforming	Phase shift	196
Beam steering	CMA	$\rho * 288^\dagger$
Baseband processing	Matched filter	54
	Total	$570 + \rho * 288$

<sup>†</sup> CMA implementation costs are 288 clock cycles when updated for each antenna sample. However, for a realistic update rate  $\rho = 0.0004$ , the cost per beam update is less than one clock cycle.

2048 bytes, whereas the  $\tan^{-1}$  read-only memory used for CORDIC takes 32 bytes of memory space. These tables can be loaded in 41  $\mu$ s. Hence, in less than 50  $\mu$ s the MONTIUM TP is ready for execution.

## VI. RELATED WORK

Many reconfigurable beamforming architectures are based on bit-level programmable Field Programmable Gate Arrays (FPGAs), like [18], [19], which perform very good as they can be optimized to the application. However, configuration times are long (milli-second to second scale), so no processing is possible during that period. The reconfigurable tiled architecture approach used in this paper can be reconfigured much faster (nano-second to micro-second scale), where it is possible to reprogram individual processors instead of the entire chip. The RaPiD reconfigurable architecture [20] has some similarities compared to the MONTIUM TP, but it is not designed as a tile processor. The reconfigurable beamformer processor proposed by Hwang [21] can be reconfigured in case processors become faulty. However, they present no performance figures. The beamforming architecture proposed by Sarrigeorgidis [22] 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 $\mu$ S architecture presented in [23] 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. Raytheon's MONARCH processor [24] 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 processing 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.

## VII. CONCLUSION

Phased array processing requires a high performance architecture that is capable of combining many input data streams at high data rates. In this paper, we proposed a tiled architecture consisting of reconfigurable processing elements as a generic beamforming platform. We presented a processing chain and analyzed the complexity of the operations that are to be performed in this chain.

The modulation scheme used for Digital Video Broadcast for Satellite (DVB-S) enables tracking of transmitters using a blind beamforming algorithm. We used the Constant Modulus Algorithm (CMA) and a Quadrature Phase Shift Keying (QPSK) optimized variant 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 reconfigurable MONTIUM TP architecture. For a receiver with 64 antennas and 3 beams, the implementation requires about 570 clock cycles on a single MONTIUM TP operating at 100 MHz and a full configuration of the algorithm into the MONTIUM TP can be completed within 50  $\mu$ s. Although the computational costs of beam steering (per antenna sample) is relatively high, the update rate of the CMA algorithm can be very low and therefore, the total processing costs are neglectable. Since a majority of the operations consist of vector operations, this processing load can be partitioned over a multi-processor architecture very well. The processing requirements roughly scale linearly with the number of antennas  $N$  and the number of beams  $B$ , which makes our reconfigurable tiled architecture very suitable for both applications using small number of antennas (e.g. wireless communications) as well as large phased array architectures (e.g. next generation radar systems).

Further research might cover the application of CMA to receivers using higher order modulation techniques like Quadrature Amplitude Modulation (QAM) or to receivers based on adaptive modulation techniques, for example DVB-S2.

## ACKNOWLEDGEMENT

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.

## REFERENCES

- [1] H. J. Visser, *Array and phased array antenna basics*. Chichester, West Sussex, UK: Wiley, Sep. 2005.
- [2] M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY, USA: McGraw-Hill, 2001.
- [3] *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services*, European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, Aug. 1997, EN 300 421 v1.1.2. [Online]. Available: [http://www.etsi.org/deliver/etsi\\_en/300400\\_300499/300421/01\\_01\\_02\\_60/en\\_300421v010102p.pdf](http://www.etsi.org/deliver/etsi_en/300400_300499/300421/01_01_02_60/en_300421v010102p.pdf)
- [4] B. H. Allen and M. Ghavami, *Adaptive Array Systems, Fundamentals and Applications*. Chichester, West Sussex, UK: Wiley, 2005.

- [5] R. H. Roy, A. J. Paulraj, and T. Kailath, "Esprit – a subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1340–1342, Oct. 1986.
- [6] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986.
- [7] J. R. Treichler and B. G. Agee, "A new approach to multipath correction of constant modulus signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 2, pp. 459–472, Apr. 1983.
- [8] Z. Xu, "New cost function for blind estimation of M-PSK signals," in *IEEE Wireless Communications and Networking Conference*, vol. 3, Sep. 2000, pp. 1501–1505.
- [9] K. C. Rovers, M. D. van de Burgwal, A. B. J. Kokkeler, and G. J. M. Smit, "Rationale for and design of a generic tiled hierarchical phased array beamforming architecture," in *Proc. 18th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2007)*. Utrecht, The Netherlands: Technology Foundation, Nov. 2007, pp. 160–168.
- [10] P. M. Heysters, "Coarse-grained reconfigurable processors – flexibility meets efficiency," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, Sep. 2004.
- [11] G. K. Rauwerda, P. M. Heysters, and G. J. M. Smit, "Towards software defined radios using coarse-grained reconfigurable hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 1, pp. 3–13, Jan. 2008.
- [12] G. J. M. Smit, A. B. J. Kokkeler, P. T. Wolkotte, P. K. F. Hölzenspies, M. D. van de Burgwal, and P. M. Heysters, "The Chameleon architecture for streaming DSP applications," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 78082, Jan. 2007.
- [13] K. C. H. Blom, "DVB-S signal tracking techniques for mobile phased arrays," Master's thesis, University of Twente, Enschede, The Netherlands, Dec. 2009.
- [14] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," *The Journal of Supercomputing*, vol. 26, no. 3, pp. 283–308, Nov. 2003.
- [15] J. Volder, "The CORDIC computing technique," in *Proc. AFIPS Joint Computer Conferences*. New York, NY, USA: ACM, Mar. 1959, pp. 257–261.
- [16] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, Feb. 1998, pp. 191–200.
- [17] S. Hitotumatu, "Complex arithmetic through CORDIC," *Kōdai Mathematical Seminar Reports*, vol. 26, no. 2-3, pp. 176–186, Mar. 1975.
- [18] J. Ou and V. K. Prasanna, "Parameterized and energy efficient adaptive beamforming on FPGAs using MATLAB/Simulink," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 5. Los Alamitos, CA, USA: IEEE Signal Processing Society, May 2004, pp. V – 181–184.
- [19] D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "A reconfigurable beamformer for audio applications," in *IEEE 7th Symposium on Application Specific Processors (SASP '09)*. Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2009, pp. 80–87.
- [20] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an OFDM receiver on the RaPiD reconfigurable architecture," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1436–1448, Nov. 2004.
- [21] D. K. Hwang, T. L. Wernimont, and W. K. Fuchs, "Evaluation of a reconfigurable architecture for digital beamforming using the OODRA workbench," in *Proc. 26th ACM/IEEE Design Automation Conference (DAC '89)*. New York, NY, USA: ACM, 1989, pp. 614–617.
- [22] K. Sarrigeorgidis and J. Rabaey, "Massively parallel wireless reconfigurable processor architecture and programming," in *Proc. 17th IEEE International Parallel and Distributed Processing Symposium (IPDPS '03)*. Washington, DC, USA: IEEE Computer Society, Apr. 2003, p. 170.2.
- [23] S. Scalera, M. Falco, and B. E. Nelson, "A reconfigurable computing architecture for microsensors," in *Proc. 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2000, pp. 59–67.
- [24] 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, and C. Channell, "MONARCH: A first generation polymorphic computing processor," in *10th High Performance Embedded Computing Workshop*, Sep. 2006.