Implementation of a Multi-User Detector for Satellite Return Links on a GPU Platform

Francisco Lázaro Blasco, *Student Member, IEEE*, Chen Tang,

***Abstract*—Due to the scarcity and high cost of satellite frequency spectrum, it is very important to utilize the available spectrum as efficiently as possible. The efficient usage of the spectrum in the satellite return link is a challenging task, especially if multiple users are present. In previous works Multi-User Detection (MUD) techniques have been widely studied to increase the spectral efficiency of the satellite return link. However, due to the high computational complexity and its sensitivity to synchronization and channel estimation errors, only few implementations of MUD for satellite communications exist. In this paper a novel Graphics Processing Unit (GPU)-based Software Defined Radio (SDR) implementation of a MUD receiver for transparent satellite return link is presented, which uses iterative channel estimation and decoding. The implementation can cope with the presence of strong phase noise. In addition to its high flexibility and low cost, with the GPU acceleration our SDR MUD receiver implementation achieves a decoding throughput of 270 Kbps using a single GPU card.**

***Keywords*—muti-user detection, satellite return link, successive interference cancellation, software defined radio, GPU**

I. Introduction

Satellite communication systems are a very good solution when it comes to delivering service to a broad area, since they do not require the deployment of costly terrestrial infrastructures. Traditionally, satellite systems have been used to provide television and radio broadcasting services, which are unidirectional services. However, in the past decade new applications have emerged, which require a bidirectional communication. The most prominent example is internet access for rural areas, where using the satellite service is commonly the only possibility. Moreover, there is a trend towards interactive applications in satellite television services. Hence, it is necessary to provide the user terminals with transmission capabilities through satellite return links, while still maintaining a low cost. Such cheap user terminals have a very limited transmission power and can only transmit with low data rates (at most in the order of 100 kbps). Moreover, the scarcity of frequency spectrum drives satellite operators to use higher frequency bands, where phase noise becomes more problematic for cheap user terminals with low stability oscillators. Therefore, it is challenging to provide an efficient spectrum access in satellite return link in the presence of strong phase noise.

Francisco Lázaro Blasco and Chen Tang are with the Institute of Communications and Navigation, German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany. Email:{Francisco.LazaroBlasco,Chen.Tang}@dlr.de

The satellite return link can be considered as a multi-user communication problem. In most of today's systems, the multiple access issue is solved by orthogonalization, such as the case of GSM, in which TDMA is used, or Digital Video Broadcasting - Return Channel via Satellite (DVB-RCS), where Multi Frequency TDMA (MF-TDMA) is used. This means that users are assigned with different resources so that they do not interfere with each other. In our implementation we employ MUD techniques to boost the spectral efficiency in a satellite return link, in which users are assigned with the same resources and their transmissions interfere. Although there is an enormous amount of literature in the field of MUD, there are actually only a few practical implementations for satellite systems [1] [2]. In [1] an implementation of MUD for the return link of a satellite system is presented, which is not designed for the presence of strong phase noise. In [2] a MUD receiver for spread spectrum Aloha is presented. In contrast to [1] and [2], our system is designed to operate in the presence of strong phase noise without employing spreading techniques.

The reasons for the low adoption rate of MUD techniques are twofold. The first reason is the high computational complexity. The second reason is the sensitivity of MUD techniques to synchronization and channel estimation errors. The latter is even more problematic, when the channel is time variant because of phase noise.

Due to the high computational complexity of MUD techniques, implementations are commonly restricted to use programmable hardware devices, such as Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC). However, the development of such hardware devices is very costly and implementations are not flexible to accommodate changes in the communication standards. From the perspective of economic cost and flexibility, SDR implementations of the MUD receiver have big advantages compared to conventional hardware implementations. The software of the SDR implementation can be easily modified to support new radio protocols or system requirements. Another advantage of SDR is its low implementation cost. However, the weak point of SDR implementations is their low throughput, especially when complex MUD algorithms have to be implemented completely in software.

In this paper we present a novel SDR implementation of the MUD receiver, which uses Successive Interference Cancellation (SIC), iterative channel estimation and Low Density Parity Check (LDPC) channel decoding. Although the algorithms used in the receiver are very complex, the

implementation achieves a decoding throughput of 270 kbps running completely on software on a single GPU. For a two user uplink where both users employ BPSK modulation and in the presence of strong phase noise the receiver achieves an spectral efficiency of 1.1 b/s/Hz.

The article is structured as follows. Section II motivates why MUD can increase the throughput of a satellite uplink. Section III describes our system model and MUD receiver design. Section IV gives a general introduction from the implementation perspective of our MUD receiver. Then, more details of the GPU-based implementation of our MUD receiver are introduced in V. Afterwards, the simulation results of the Packet Error Rate (PER) and the processing time of our GPU-based SDR MUD implementation are presented in VI. Finally Section VII draws the conclusions.

II. A case for employing MUD in Satellite Uplinks

In this section we give a short overview of MUD and explain why it can increase the spectral efficiency of a satellite uplink compared to TDMA. Fig. 1 shows the achievable rate region for the uplink AWGN channel with two users [3], where it has been assumed that the signal to noise ratio of users 1 and 2 are $P_1/N = 2.51$ and $P_2/N = 1$ (corresponds to a power imbalance of 4 dB). For a given scheme, the achievable rate region is that which is within the boundary depicted in Fig. 1. In other words, it is possible that user 1 and 2 communicate at rates R_1 and R_2 respectively, if the point (R_1, R_2) lies in the achievable region.

Let us first look at the achievable rate region of MUD whose boundary is the solid line in Fig. 1. When MUD is used, one would like to operate in the points on the solid line joining the corner points A and B, since these points maximize the sum rate $(R_1 + R_2)$. With MUD any point of that line is achievable. If one wants to be *fair* the best choice would be operating at point A where user 2 gets its maximum possible rate (1 b/s/Hz in this case) and user 1 still gets a substantial rate (1.585 b/s/Hz). Let us now look at the dashed line which gives the achievable rate region for TDMA with a constraint on the mean transmit power. Using TDMA it is also possible to operate a point which gives the maximum rate since the dashed touches the solid line at one point. At this point the rate of user 1 is close to its maximum rate but the rate of user 2 is considerable lower with respect to its maximum rate. Hence, although TDMA can theoretically achieve the sum rate it does so by assigning the channel to the strongest user most of the time and the weak users get a very low rate. In other words, TDMA cannot be *fair* and provide a high sum rate at the same time. Moreover, this rate region assumes that the total transmit power stays the same and hence there is no constraint on the maximum transmit power. The dashed line with square markers delimits the achievable rate region of TDMA when the maximum transmit power is limited to $P_1 = 4$ and $P_1 = 1$, which is the case of satellite uplinks. If we consider this constraint, the achievable rates with MUD are much higher and provide a much better fairness compared to TDMA. These gains are obtained mainly due to the fact that

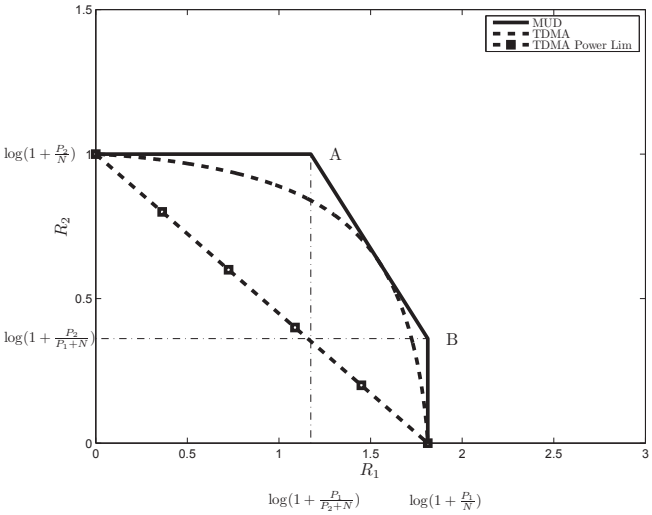

Fig. 1. Achievable rate region for the uplink AWGN channel with two users.

with MUD we can let users transmit always with maximum power and in the same frequency. In TDMA users need to be silent when other users are transmitting.

If we would consider FDMA, the achievable rate region would be delimited by the dashed line without markers, even in presence of a constraint on the maximum transmit power. However, using FDMA may not be advisable in a real system for several reasons. First, one needs to introduce guardbands in order to prevent interference among neighbouring carriers. Second, in the presence of strong phase noise decreasing the symbol rate makes dealing with phase noise more difficult. If we consider a random walk model for phase noise, the phase increments between consecutive symbols follow a Gaussian distribution. For example, if we reduce the symbol rate by a factor 2 the variance of the phase increments among consecutive symbols doubles. A stronger phase makes synchronization more difficult.

In the following sections of this paper we give details about our MUD which allows the system to operate in a point close to A in Fig. 1, where the weak user gets its maximum possible rate and the strong user can still communicate at a considerable rate. We remark that Fig. 1 assumes an AWGN channel without phase noise. In our implementation, as we will show in later sections of this paper, BPSK modulation was used and a strong phase noise was present in the system. The achievable rates in the presence of phase noise and using BPSK modulation would actually be lower as shown in Fig. 1. However the aim remains unchanged, the weak user shall be able to communicate a rate very close to the maximum it would achieve in the absence of the strong user.

III. System Model and Receiver Design

In this paper we focus on a single carrier of a satellite return link. This could be the case of DVB-RCS which uses MF-TDMA and divides the bandwidth of the return link into several carriers and uses TDMA within each carrier.

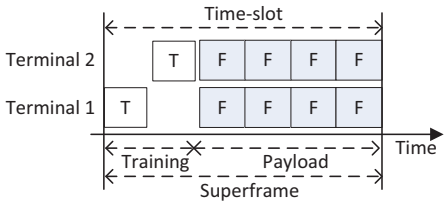

Fig. 2. Transmission inside one time-slot. The blocks denoted by T represent the training part and the blocks denoted with F represent payload frames.

We assume the scheduler assigns the same frequency and time-slot to K user terminals, which are assumed to have limited transmission power and low transmission data rate, as introduced in the previous section. Terminals are assumed to be roughly synchronized. Fig. 2 shows how the transmission of the two terminals is organized inside one time-slot. In each time-slot each terminal transmits a super-frame, which consists of a training part and a payload part. In the training part the first terminal transmits while the second is silent, then the second terminal transmits and the first terminal is silent. This interference free part will be used by the burst demodulator for detection and synchronization purposes. In the second part terminals transmit a number of data frames that collide, which requires MUD to decode.

If only the data part of the time-slot is considered, the received signal $y(t)$ at time instant t can be expressed as:

$$y(t) = \sum_{k=0}^{K-1} \sum_{l=0}^{L} h_k(t - lT_s) x_k(l) p_k(t - lT_s - \tau_k) + n(t) \quad (1)$$

where K is the number of users, L is the number of symbols in a frame, $h_k(t)$ is the complex channel coefficient for user k at time instant t, $x_k(l)$ is the l-th transmitted symbol from user k, $p_k(t)$ is the transmit pulse shape filter of user k, T_s is the symbol duration, τ_k represents the delay of user k with respect to the start of the frame and $n(t)$ is the complex valued additive white Gaussian noise. In our case the channel coefficient is induced by phase noise, $h_k(t) = |h_k| e^{j\phi_k(t)}$. $|h_k|$ is assumed to be constant, while the variation of $\phi(t)$ can be modelled as a random-walk (Wiener) process [4]. Under this assumption the evolution of the channel phase for discrete time instants can be expressed as:

$$\theta_k(t - lT_s) = \theta_k(l) = \theta_k(l-1) + \Delta_k(l) \quad (2)$$

where $\Delta_k(l)$ are i.i.d zero mean real Gaussian random variables with variance ρ_p^2. We will assume that the transmitted symbols $x_k(l)$ belong to an M-PSK alphabet. Bit interleaved coded modulation will be used together with a Soft-Input Soft-Output (SISO) channel decoder.

The MUD receiver employed is based on SIC [5], as shown in Fig 3. The reason of choosing SIC is that although it is not optimal, its computational complexity increases linearly with the number of system users. SIC is based on a simple idea: User 1 is decoded treating the interference from the user 2 as Gaussian noise. After decoding user 1, its waveform is reconstructed and removed from the received signal. If this

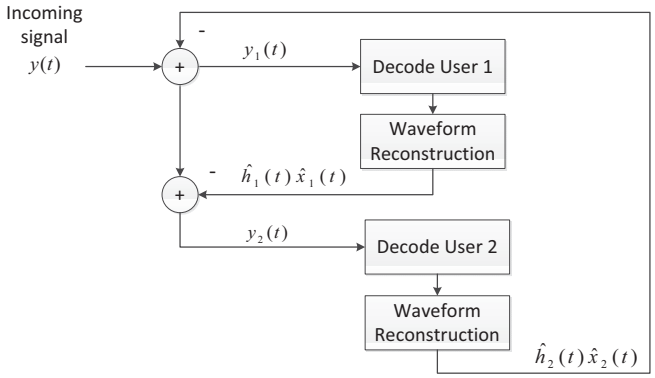

Fig. 3. SIC decoding scheme

reconstruction is perfect, user 2 is now free of interference and can be decoded. In order to reduce the PER, a multi-stage SIC can be used [5], where the SIC process is repeated for several times and each execution of SIC is denoted as a stage. If the user 1 signal cannot be recovered error free after the first stage, a second stage is carried out by reconstructing the signal from user 2 and cancelling it from the received signal on user 1. User 1 will now have a lower interference and can hopefully be decoded correctly. This process is repeated until the maximum number of stages is achieved, or until both users are decoded correctly. Based on SIC, our simplified case with $K = 2$ number of users can be easily extended to support more user terminals.

SIC is very sensitive to channel estimation errors. In order to improve the performance of the SIC receiver, an iterative channel estimation and decoding algorithm is used. The basic idea of such an algorithm is to use not only pilot symbols but also data symbols to estimate the channel. In our receiver the Expectation-Maximization (EM) algorithm is applied as described in [6], where the E-step corresponds with the channel decoding and the M-step with the channel estimation. A simplified block diagram of the EM algorithm is shown in Fig. 4. Initially a channel estimation is performed using only pilot symbols. This channel estimation is used to perform soft demodulation and soft channel decoding. The soft output of the channel decoder is then fed back to the channel estimator which performs a channel estimation using pilot symbols and the loglikelihood values obtained from the channel decoder. The equations used for the M-step are the following [5]:

$$|\hat{h}_k| = \frac{\sum_{t_1=0}^{L-1} Re\left(\hat{x}_k^*(t_1)\, y_k(t_1)\right)}{\sum_{t_1=0}^{L-1} |\hat{x}_k^2(t_1)|} \quad (3)$$

$$\hat{\theta}_k(t_1) = \text{Arg} \sum_{t_1=t-W}^{t+W} \hat{x}_k^*(l)\, y_k(l) \quad (4)$$

$$\hat{h}_k(t_1) = |\hat{h}|_k e^{j\hat{\theta}_k(t_1)} \quad (5)$$

$$\hat{\sigma}_k^2 = \frac{1}{2L} \sum_{t_1=0}^{L-1} |y_k(t_1) - \hat{h}_k(t_1)\, \hat{x}_k(t_1)|^2 \quad (6)$$

where y_k is the input signal of the single user receiver of user

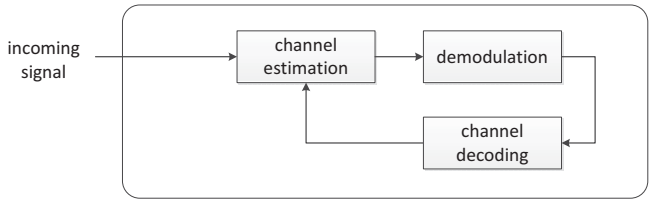

Fig. 4. EM algorithm used for channel estimation

k, which is the received signal minus the reconstructed signal from the other users. $\hat{x}_k$ is the output of the SISO channel decoder. In order to estimate the channel amplitude $|\hat{h}_k|$ for each user and the noise level $\hat{\sigma}_k^2$, the assumption is made that these magnitudes stay constant over the duration of the whole frame L. For the estimation of the channel phase $\hat{\theta}_k(t_1)$ at discrete time instant t_1, the channel phase is assumed to be constant over a window of $2W+1$ samples centered on t_1.

Prior to SIC decoding it is necessary to perform frequency, frame and time synchronization. In order to make synchronization easier, a training part is introduced in each superframe, which is free of multiuser interference, see Fig. 2. Using this training part, it is possible to perform frequency, frame and timing estimation using single user synchronization algorithms. In particular, in our implementation the Oerder & Meyr algorithm [7] is used for time synchronization. Timing synchronization is not affected by phase noise. However phase noise does have a negative impact in frequency and frame synchronization. For the frame synchronization the Villanti *et al.* algorithm [8] is used, which was specially developed to perform frame estimation in the presence of phase uncertainty due to doppler or phase noise. This algorithm breaks the synchronization sequence into smaller subsequences and performs correlation of the incoming signal with these subsequences. Then the metrics obtained are summed up with the correlation. Frequency estimation is done using the Rife and Boorstyn algorithm [9]. This algorithm estimates the carrier frequency by performing a FFT of the received signal. Its performance is affected by the presence of phase noise, but it achieves the Cramer-Rao lower bound.

IV. SDR-based MUD Receiver

Conventionally programmable hardware devices, such as Digital Signal Processors (DSPs) and FPGAs together with Radio Frequency (RF) transceiver are used to implement a SDR system. For instance, the real time implementations of MUD receiver by using dedicated DSPs and FPGAs were presented in [10] [11] [12]. A number of dedicated hardware implementations of LDPC decoders have also been proposed in the past few years [13] [14]. However, the drawbacks of high cost, considerable development effort and lack of flexibility of FPGAs and DSPs restrict the deployment of SDR technologies [15]. As an example, dedicated hardware-based LDPC implementations have to start a new design and development process to adapt to a different coding design, since the structure of the LDPC decoder changes according to the parity check matrix.

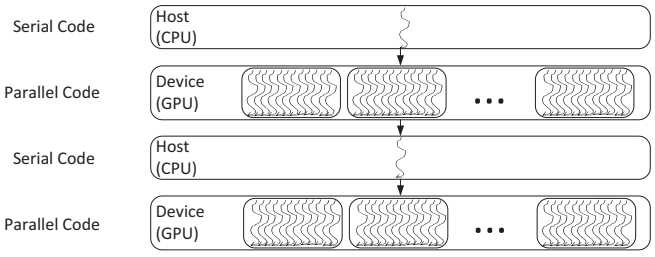

Fig. 5. GPGPU implementation structure

In recent years owing to the tremendous parallel computational power and very high memory bandwidth, general-purpose computing on graphics processing unit (GPGPU) has become an attractive alternative to FPGA and DSP to achieve high-performance SDR systems. GPUs are originally designed to optimize the floating-point calculations for image and video rendering. GPGPU technique utilizes a GPU to perform general computations in applications traditionally handled by the CPU. The implementation of a GPGPU in a conventional CPU-based system is illustrated in Fig. 5. The parts of the application that need to be executed serially are implemented on CPU. The parts that are very computationally intensive and can be parallelized are deployed on the GPU.

State of the art high-end GPUs consist of thousands of processing cores, which can provide over 3 tera-floating operation-per-second (TFLOP) computing capability [16]. Moreover, with novel GPGPU architectures, such as Compute Unified Device Architecture (CUDA) introduced by Nvidia [16], applications running on a GPUs can be programmed with high-level C-like languages, which makes it easier to implement and modify a GPU-driven SDR system. Therefore, due to its high computing capability, low cost and high flexibility, GPGPU has become a very promising solution for modern SDR systems.

For instance, many SDR blocks with high computational complexity and inherent parallelism, such as channel estimation, channel decoding and digital filters can be implemented on GPU. In [17] a high throughput SDR-based multiple-input and multiple-output (MIMO) detector is implemented on GPU. GPU-accelerated LDPC decoder has also been widely studied [18] [19] [20] [21].

In this work we adopt CUDA to exploit GPU to accelerate the SDR implementation of our MUD system. The implementation details of our GPU-accelerated MUD receiver will be given in the next section.

V. Implementation of GPU-accelerated MUD receiver

Based on the receiver design introduced in Sec. III, all components of our MUD receiver are implemented in software, except for a reconfigurable hardware for sampling and analog-to-digital conversion. As some of the algorithms of the MUD receiver used in our work have very high computational complexity, we exploit the CUDA programming model to accelerate our SDR implementation on GPU.

As the first step of a GPU implementation using CUDA, it is important to assess which algorithms require the most processing time and also wether these algorithms have a parallel structure. The processing bottlenecks of our SDR were the following:

- Channel decoding: In our system, irregular LDPC channel codes are used. The optimal iterative decoding algorithm, Sum-Product algorithm (SPA) is computationally slow since it requires computing a hyperbolic tangent. In order to speed up the decoding the Min-Sum algorithm [22] was used, which is faster at the cost of a small Es/N_0 degradation (in the order of 0.1 to 0.2 dB). Nevertheless, the Min-Sum algorithm is still heavy to be computed in software since it at every iteration one has to compute the output message at all variable and check nodes. Luckily, iterative LDPC decoding is inherently parallel, since the computation of the output message at one variable node is independent from the computation of the messages at all other variable nodes, and the same holds for check nodes. As we will show in Section VI a parallel implementation of the LDPC decoder running on a GPU was around 10 times faster than an implementation running on a CPU.
- Channel estimation: The EM channel estimation we deployed consists of a filtering operation in the time domain, as given in Eq. (3), (4) and (5). In our case channel estimation took a substantial amount of time, mainly because it was being performed several times inside the EM loop. Filtering can be implemented in the frequency domain using FFT which runs very fast on parallel platforms such as GPU.
- Downsampling and interference cancellation: in our implementation oversampling factor 4 is used. Channel decoding and estimation at 1 sample per symbols, however, other operations such as downsampling and interference cancellation run on the oversampled waveform which implies performing more operations. Both downsampling and interference cancellation can be parallelized.

Apart from the computing time spent on these algorithms one needs to take into account that data transfer between host (CPU) and device (GPU) memory takes also a substantial amount of time. The reason behind is that the peak theoretical GPU memory bandwidth is much higher than the peak bandwidth between host (CPU) memory and device (GPU) memory. For example, the EM algorithm employed iterates channel estimation and channel decoding several times, 4 times in our case. Thus, if there is any memory copy between host (CPU) and device (GPU) during the EM iterations, such a data transfer has to be repeated for 4 times. In order to achieve better processing performance, it is important to minimize data transfer between host and device [23].

Based on this assessment of our MUD receiver, we decided to implement all of the components of each single user receiver together with the interference cancellation on GPU, as illustrated in Fig. 6. Although some components of the single user receiver demonstrate little speedup from the GPU implementation, the benefit of running all components on GPU is to minimize the latency of intermediate data transfer between the host and device memory.

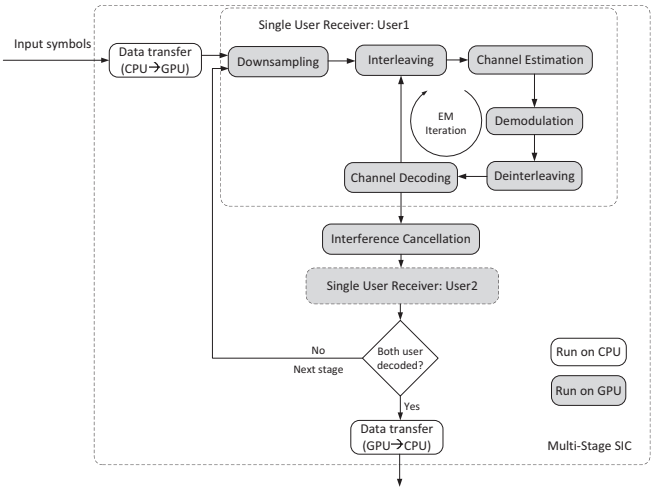

Fig. 6. Multi-Stage SIC MUD receiver on GPU

VI. Performance Measurements

In this section the simulation results in terms of PER and the processing performance of our GPU-based implementation of the MUD receiver are presented. These results are compared with the CPU-based implementation. The specifications of the devices are given in Tab. I. As introduced before, 2 satellite terminals are assumed to transmit using the same return link carrier. We will assume that carriers support a symbol rate of 62.5 ksymbols/s. The 2 terminals use BPSK modulation. The results presented in this section correspond to a scenario in which the power of user 1 is approximately 4 dB higher than the power of user 2, which is a realistic setting for a satellite return link in which user 1 could be assumed to be in the center of a beam and user 2 close to the beam edge. Both users employ irregular LDPC codes with a blocklength of 4800 bits, user 1 with rate $2/3$ and user 2 with rate $1/2$. A different 63 symbols long Gold sequence is employed as training sequence by each user.The number of SIC stages is fixed to 3. The terminals have a phase noise of -60 dBc/Hz at 10 kHz, which assuming a random-walk Wiener process [4] and a symbol rate of 62.5 Ksps results in Gaussian i.i.d phase increments between consecutive symbols with zero mean and standard deviation $2.29°$. This amount of phase noise can be regarded as a worst case scenario when very cheap oscillators are used.

For simplicity the satellite channel was assumed to be AWGN. In our case this approximation is reasonable for fixed terminals because the strong phase noise would dominate over other channel impairments. The performance measurements were carried using a real transmission and a channel simulator which was injecting Gaussian noise in the analogue domain.

Fig. 7 shows the PER vs E_s/N_0 for both user terminals. We remark that the results presented in this section include the detection and estimation algorithms. If we look at the PER of

TABLE I. IMPLEMENTATION DEVICES

	CPU	GPU
Platform	Intel Xeon(R) E5620	Nvidia Tesla C2070
Cores	4 (used only one core)	448
Clock rate	2.4GHz	1.15GHz

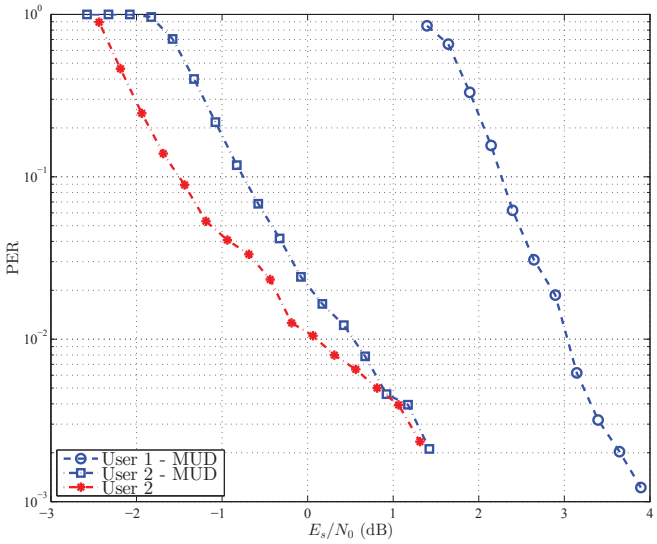

Fig. 7. PER vs E_s/N_0. The curves with circle and square markers represent the PER of user 1 and user 2 in the MUD setting. The red curve with asterisk markers represents the PER of user 2 for the single user setting (user 1 is not present).

user 2 we can see how for target PER of 1%, the performance with and without the presence of user 1 is very similar. When user 1 is present, user 2 needs to increase its E_s/N_0 around 0.5 dB compared to the case in which user 1 is absent. Hence, by increasing the transmit power of user 2 by 0.5 dB we can let also user 1 communicate at a rate of $2/3$. This allows us to increase the throughput in the satellite uplink using the same bandwidth.

Let us now look at decoding throughput and processing time of our implementation. The total processing time of the MUD implementation is defined as:

$$T_{\text{proc}} = T_{\text{DS}} + T_{\text{CE}} + T_{\text{CD}} + T_{\text{IC}} + T_{\text{misc}} \qquad (7)$$

which consists of the time for downsampling T_{DS}, channel estimation T_{CE}, channel decoding T_{CD}, interference cancellation T_{IC} and T_{misc} that includes memory operations and the remaining miscellaneous components of the MUD given in Fig. 6.

Tab. II shows a comparison of the processing time between the CPU-based and GPU-based implementations for both user terminals to decode one frame. The total processing time decreases as the E_s/N_0 increases. For high E_s/N_0, due to the early termination scheme used at the LDPC decoder, the channel decoder can skip the remaining decoding iterations and finish the decoding process as soon as it converges to a valid codeword. Moreover, for high E_s/N_0 values, there is a high probability that both users can already be decoded properly

TABLE II. COMPARISON OF TOTAL PROCESSING TIME OF THE MUD IMPLEMENTATION.

E_s/N_0 (dB)		$T_{\text{DS}}(\%)$	$T_{\text{CE}}(\%)$	$T_{\text{CD}}(\%)$	$T_{\text{IC}}(\%)$	$T_{\text{misc}}(\%)$	$T_{\text{proc}}(ms)$
User 1: 2.14 User 2: -1.83	CPU	4.5	7.8	77.9	7.6	2.2	525
	GPU	4.7	15.8	46.2	10.31	23.0	82.81
	Speedup factor	5.22	2.6	9.3	4	0.5	5.38
User 1: 3.39 User 2: -0.58	CPU	7.8	11.8	62.0	14.3	4.0	136
	GPU	7.2	19.8	21.4	16.7	34.9	27.1
	Speedup factor	5.6	3.1	14.9	4.4	0.6	5.0
User 1: 5.14 User 2: 1.17	CPU	16.8	15.4	28.9	25.9	13.0	65.96
	GPU	8.5	19.8	11.0	18.2	42.5	21.4
	Speedup factor	6.0	2.4	8.1	4.3	0.9	3.0

in the first SIC stage which allows to skip the successive SIC stages. The processing time consumed by the downsampling and the interference cancellation stays almost constant with the E_s/N_0.

Compared to the CPU-based solution, with the GPU acceleration, the processing time of the MUD receiver can be significantly reduced. As one can observe in Table II the channel decoding can be sped up by use of the GPU by a factor 8.1 to 14.9 depending on the E_s/N_0. Concretely, a channel decoding throughput of 2.0 Mbps is achieved for terminal 1 (code rate $2/3$) with $E_s/N_0 = 3.39$ dB and 1.9 Mbps for terminal 2 (code rate $1/2$) with $E_s/N_0 = -0.58$ dB. Channel estimation is also faster on the GPU as on a CPU by a factor 2.6 to 3.1 depending on the E_s/N_0. Downsampling and interference cancellation are also faster on GPU. However, if we look at the time spent on *miscellaneous* operations, it is higher when using a GPU. The reason is that using a GPU requires initializing the memory and memory transfer between CPU and GPU. If the total processing time is considered, the GPU implementation runs around 5 times faster at an $E_s/N_0 = 3.39$ dB for User 1.

Fig. 8 shows the throughput of the MUD receiver running on a GPU and on a CPU. It can be observed how the throughput on a GPU is higher than on a CPU for all the E_s/N_0 values. The throughput increases with the E_s/N_0. This increase in throughput is mainly due to the fact that the decoding throughput of the channel decoder increases with the E_s/N_0 due to the early termination scheme used. It can also be observed how for the GPU the throughput seems to saturate at about 270 kbps.

We would like to remark that the GPU used to obtain this results was a Nvidia Tesla C2070 which was no longer the fastest available GPU on the market at the time of writing this article. For instance, A NVidia Geforce GTX Titan Black, can perform single precision floating point operations approximately 5 times faster than a Nvidia Tesla C2070. Hence, we speculate that the decoding throughput of our system would increase substantially by simply running the SDR on a faster GPU.

VII. CONCLUSION

In this article, we described the system design and the implementation of a MUD receiver using SIC for a satellite return link based on MF-TDMA. The receiver is able to cope with strong phase noise and increase the spectral efficiency of satellite return links. Concretely by increasing the transmit power of the weakest user by 0.5 dB it allows to increase the spectral efficiency from 0.5 b/s/Hz to 1.16 b/s/Hz (without

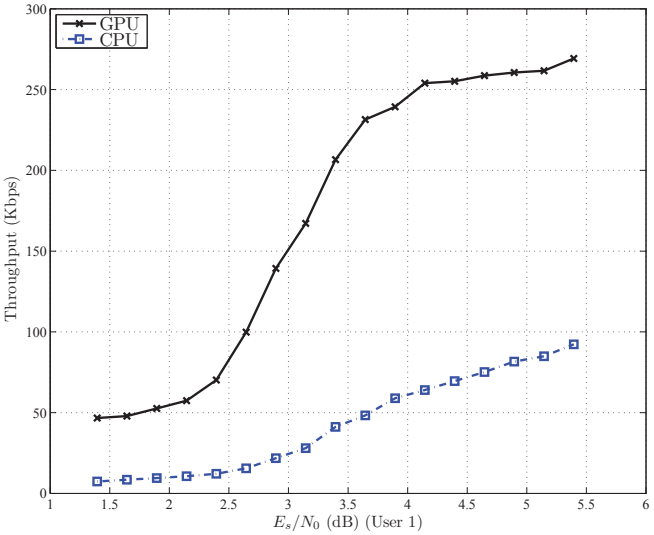

Fig. 8. Throughput vs E_s/N_0 of the MUD Receiver. The black curve with x markers represents the throughput of the MUD receiver running on a GPU and the dashed blue curve represents the throughput of the receiver running on a CPU.

considering protocol overhead). With the acceleration of an affordable commercial GPU, our SDR implementation of the MUD receiver has a decoding throughput of 270 kbps using one single GPU. This throughput would suffice to decode three 62.5 kbaud/s carriers with one single GPU unit. Hence, a receiver platform employing several GPUs could be used in a satellite hub to manage a bandwidth in order of MHz. Compared to a dedicated hardware implementation, our SDR-based solution has higher flexibility and lower development cost.

Performance measurements have only been provided for BPSK modulation because the strong phase noise prevented using higher order modulations. The receiver implementation actually supports MPSK modulation and can easily be extended to other modulation types.

VIII. Acknowledgements

The authors were supported by the Space Agency of the German Aerospace Center and the Federal Ministry of Economics and Technology based of on the agreement of the German Bundestag, under support code 50 YB 0905. The authors would like to acknowledge Dr. Hermann Bischl for the useful discussions.

References

[1] P. Fines, P. Febvre, and E. Christofylaki, "Turbo-code division multiple access: capacity enhancement of mobile satellite systems using narrow-band multiuser detection," in *Proc. of 10th International Workshop on Signal Processing for Space Communications*, Rhodes Island, Greece, Oct. 2008.

[2] F. Basile and G. Mendola, "Satellite hub communication system GPU based," in *Proc. of GPU Technology Conference*, San Jose, CA, USA, May 2012.

[3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge University Press, 2005.

[4] G. Colavolpe, A. Barbieri, and G. Caire, "Algorithms for iterative decoding in the presence of strong phase noise," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 9, pp. 1748–1757, 2005.

[5] F. Lázaro Blasco and F. Rossetto, "On the derivation of optimal partial successive interference cancellation," in *Proc. of IEEE Globecomm*, Houston, Texas, US, Nov. 2011.

[6] M. Kobayashi, J. Boutros, and G. Gaire, "Successive interference cancellation with SISO decoding and EM channel estimation," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 8, pp. 1450–1460, Aug. 2001.

[7] M. Oerder and H. Meyr, "Digital square and filter timing recovery," *IEEE Trans. Commun.*, vol. 36, pp. 605–612, May 1988.

[8] M. Villanti, P. Salmi, and G. E. Corazza, "Differential post detection integration techniques for robust code acquisition," *IEEE Trans. Commun.*, vol. 55, no. 11, pp. 2172–2184, Nov. 2007.

[9] D. Rife and R. Boorstyn, "Single tone parameter estimation from discrete-time observations," *IEEE Trans. Inf. Theory*, 1974.

[10] S. Rajagopal, S. Bhashyam, J. R. Cavallaro, and B. Aazhang, "Real-time algorithms and architectures for multiuser channel estimation and detection in wireless base-station receivers," *IEEE Trans. Wireless Commun.*, vol. 1, no. 3, pp. 468–479, Jul. 2002.

[11] J. Tranquilli, J. Farkas, J. Niedzwiecki, B. Pierce, L. Brothers, and J. Debardelaben, "Real time implementation of a multiuser detection enabled ad-hoc network," in *Proc. of Military Communications Conference*, San Diego, California, USA, Nov. 2008.

[12] I. Seskar and N. Mandayam, "A software radio architecture for linear multiuser detection," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 5, pp. 814–823, May 1999.

[13] G. Masera, F. Quaglio, and F. Vacca, "Implementation of a flexible LDPC decoder," *IEEE Trans. Circuits Syst. II*, vol. 54, no. 6, Jun. 2007.

[14] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Proc. of Design, Automation and Test in Europe*, Nice, France, Apr. 2009.

[15] J. Kim, S. Hyeon, S. Choi, and H. University, "Implementation of an sdr system using graphics processing unit," *IEEE Commun. Mag.*, pp. 156–162, Mar. 2010.

[16] Nvidia, "Nvidia CUDA Zone." [Online]. Available: https://developer.nvidia.com/category/zone/cuda-zone

[17] M. Wu, Y. Sun, S. Gupta, and J. Cavallaro, "Implementation of a high throughput soft MIMO detector on GPU," *Journal of Signal Processing Systems*, vol. 64, no. 1, pp. 123–136, Jul. 2011.

[18] G. Falcao, L. Sousa, and V. Silva, "Massively ldpc decoding on multicore architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 309–322, Feb. 2011.

[19] C. Chang, Y. Chang, M. Huang, and H. B., "Accelerating regular ldpc code decoders on GPUs," *IEEE J. Sel. Topics Appl. Earth Observ*, vol. 4, no. 3, pp. 653–659, 2011.

[20] G. Wang, M. Wu, Y. Sun, and C. J.R., "A massively parallel implementation of QC-LDPC decoder on GPU," in *Proc. of IEEE 9th Symposium on Application Specific Processors*, San Diego, California, USA, 2011, pp. 82–85.

[21] S. Kang and J. Moon, "Parallel LDPC decoder implementation on GPU based on unbalanced memory coalescing," in *Proc. of IEEE Int. Conf. on Commun.*, Ottawa, Canada, Jun. 2012.

[22] N. Wiberg, "Codes and decoding on general graphs," *Ph.D. dissertation, U.Linkping*, 1996.

[23] Nvidia, "CUDA C Best Practices Guide," Oct. 2012. [Online]. Available: {http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf}