A High Performance FPGA-GPU-CPU Platform for a Real-Time Locating System

Mohammad Alawieh, Maximilian Kasparek, Norbert Franke and Jochen Hupfer
Locating and Communiction Systems Deptartment
Fraunhofer Institute for Integrated Circuits IIS , Germany
{Mohammad.Alawieh}@iis.fraunhofer.de

Abstract. A heterogeneous Software Defined Radio (SDR) cluster platform that handles highly demanding processing algorithms in real-time is proposed. The solution based on a combination of FPGA, GPU and CPU offers the best balance between performance, cost, and flexibility. The key feature of our heterogeneous platform is achieving the required performance by assigning the tasks according to the technology characteristics. The FPGA in the proposed system does not only acquire external data but perform initial acquisition. This process aids in* facilitating *parallelism on the GPU side and optimizing the data transfer.

The performance of the platform is demonstrated for an intensive real-time localization application. The overall costs are kept extremely low when compared to other solutions that can provide similar performance.

Keywords—Wirelss locating systems; FPGA; GPU; CPU; heterogeneous platform; SDR

I. INTRODUCTION

Wireless locating systems providing position information for objects and individuals are demanded by diverse applications. The accuracy, update rate and the number of localized objects are parameters that define the complexity of such a system. An example for a demanding locating system is RedFIR [1]; a high-accuracy and high-rate tracking system designed for real-time sports analysis.

The challenge to track up to 50,000 positions per second in real-time, with an accuracy of a few centimeters, would traditionally require designing a custom hardware platform. This in turn limits the flexibility and scalability of the platform, e.g. variation of the number of transmitters to be located or the support of variable burst length; favor an adaptive Software Defined Radio (SDR) approach.

Due to the required combination of computationally intensive signal processing, adaptive and flexibility, receivers are based on a programmable heterogeneous platform, consisting of Field Programmable Gate Array (FPGA), Central Processing Unit (CPU) and Graphic Processing Units (GPU).

II. SYSTEM OVERVIEW

In RedFIR, multiple receivers are used to triangulate the positions of multiple, miniaturized transmitters. Each receiver calculates the Time Of Arrival (TOA) of the signals sent by the transmitters. The Time Difference Of Arrival (TDOA) method is applied to the TOA measurements, which aids for a later position detection without the need to know the exact time of transmission. The unambiguous TDOA-based triangulation requires at least four receivers. However, in favor of accuracy and redundancy, more receivers are used in an average system setup.

Figure 1 shows the system's layout: Beside the free-running transmitters on the pitch, the system mainly consists of the receiving infrastructure around the locating area. Thereby the antennas are distributed, while the digital processing receivers are located in a central computing cluster. The synchronization of the receivers, as well as the data exchange between antenna units and computing cluster, involves a fiber-optical network.

The computing cluster is the computational core of the system. It comprises all signal processing required for the computation of TOAs, TDOAs and transmitter positions.

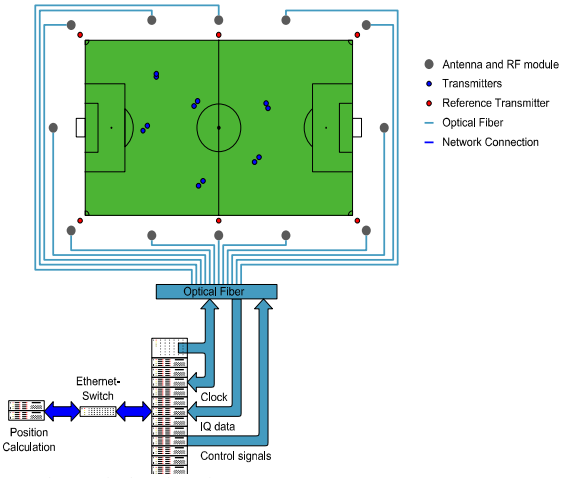

Figure 1. RedFIR wireless locating system.

A. Complexity Factors

The system complexity is concentrated at the receiver side and the central computing unit while are transmitters designed for low power and small size. The complexity of such a system depends on a number of factors, most relevant are:

1) Real- time processing:

Real-time refers to the online processing of the received data from each antenna within few milliseconds. Unlike in hard real-time systems (ex. heart pacemaker, flight control) missing deadline is not fatal but average time must be met.

2) Signal bandwidth:

RedFIR operates in the 2.4GHz ISM band which allows the usage of a bandwidth close to 80MHz. Direct Sequence Spread Spectrum (DSSS) has been widely used in different

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

ranging and navigation applications[2]. The essence of spread spectrum is to expand the narrow-band signal over the available bandwidth, which in turns enables a better performance in a noisy signal environment and leads to a better resolution. According to Shannon-Nyquist sampling theorem, this requires that the signal has to be sampled with at least twice its signal bandwidth. Each of the analog down-converted in-phase and quadrature components (I, Q) are sampled with two 100MSPS ADCs (Analog to Digital Convertor).

3) Update rate:

The communication system operates in burst mode. The update rate represents the rate, at which the transmitter sends its bursts. A higher rate requires more processing power, and the receivers' peak performance should be at least 50,000 bursts/sec.

4) Burst length:

The burst length is the duration of the transmitted signal. The length of the burst depends on the required processing gain and the number of users. A longer burst results in a larger processing gain at the cost of higher processing complexity.

5) Number of users:

The system identifies different codes for each transmitter. This implies that even at the same frequency and the same time, the receivers can detect all the transmitters. The system supports 144 different transmitters, six times more than a GPS receiver.

6) Adaptive Parametrization:

Based on the application at hand, the number of transmitters and the burst rate vary. Hence the overall system capacity has to be flexibly partitionable.

B. Motivation and Related Work

The first platform designed to support the requirements of RedFIR is a FPGA design based on Xilinx Virtex-II [3]. The high input data rate and the complexity of the algorithms requires around 450,000 Virtex-II Pro logic cells in addition to six PPC405. Despite the fact that the FPGA based receiver platform achieves the required performance, however the cost of a platform: the number of FPGAs, added to the number of receivers needed results in extremely high material cost.

Designers nowadays have plenty of options to bring their algorithms to life. As NVIDIA introduced the Computer Unified Device Architecture (CUDA), a new device is available that dramatically increase parallel computing performance by exploiting the power of the Graphic GPUs. On another aspect, modern FPGAs not only offer a large number of logic cells, but also dedicated blocks for mathematical operations such as DSP48 added to other hard cores available.

A number of recent studies have been conducted in this field, mostly comparing the performance of different computing architecture. For instance, in many applications GPU performance surpasses that of the FPGA, while FPGA provides a better solution for other applications [4].

In this paper we introduce a hardware architecture based on FPGA, CPU and GPU implemented with off-the-shelf components. This platform takes the advantage of the FPGA flexibility to create custom hardware along with GPU parallel processing power to accelerate CPU performance. Unlike similar recently published architectures [5], we introduce a concept for a balanced operation of these three components.
Although the proposed heterogonous platform is designed for RedFIR locating system, however the principle of operation can be beneficial for a wide range of applications (e.g. astronomy [6] and communication [2][7]).

III. Signal Processing Algorithm

In order to increase the battery life time of the transmitters and reduce the channel usage, signals are sent in a burst mode. Every transmitter sends burst spread-spectrum signals (tracking burst); it waits then for a defined time and sends again. At a much lower rate the transmitter introduces also a narrowband signal (acquisition burst). Unlike wideband, narrowband signals suffer from a lack in resolution however their detection processing is computationally less demanding. Hence this signal is needed by the system in the acquisition phase to acquire initial acquisition and keep the receiver synchronized with the transmitter.

The utilization of a heterogeneous platform is only profitable if the algorithms to be deployed are capable of exploiting the platforms heterogeneity. Accordingly the platform selection is based on the characteristics of the acquisition and tracking algorithms.

A. Acquistion

Every transmitter sends a narrowband signal in one of the available frequency channels. Bursts belonging to transmitters in the same frequency channel use different codes, with good cross-correlation properties in order to identify the different transmitters.

Figure 2 shows the acquisition operation on the receiver side. The IQ output of the ADC is down converted to the desired frequency channels which are later down sampled and passed through a root raised cosine filter. Afterwards each filter output from each of the N channels reaches the correlation filter. The correlation filter correlates the filtered signal with M reference sequences. Hence, N(channels)×M(correlations) is the total number of transmitters the receiver can identify. The final step is to identify the transmitter related to the correlation outputs and calculate the time the corresponding acquisition burst arrived [8].

The design of the digital down convertor can be achieved easily implemented using the method presented in [9]. However the implementation of a real-time correlator, or more precisely partial-correlator, is a complex component in term of resource utilization. The terminology partial-correlation comes from breaking of the burst signal into parts. Correlation is then performed on each of the partitions to minimize the effect of frequency offsets [10].

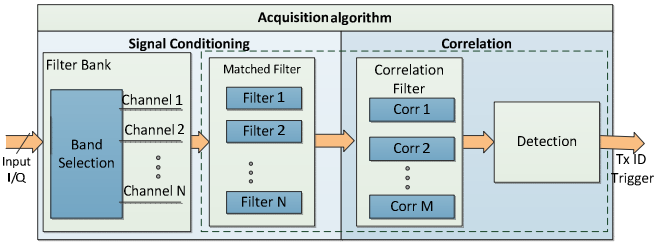

Figure 2. Acquisition algorithm data flow.

B. *Tracking*

Because of the predetermined time interval separating the acquisition and tracking bursts, once the acquisition burst has been successfully detected, the receiver predicts the window where the next tracking burst is expected. Across this window samples, correlation is performed allowing to precisely calculate the Time Of Arrival value of the tracking burst.

The tracking burst is spread over the complete available bandwidth, where each transmitter is again differentiated by a different code. Over the pre-determined window, a complex correlation for the received signals with the reference signal operates at the input data rate. The correlation filter is followed by frequency offset correction and then outputs the outcome for a precise detector. The function of the precise detector is to disregard multipath signals and predict the precise time of arrival of the spotted tracking burst, as visualized in Figure 3.

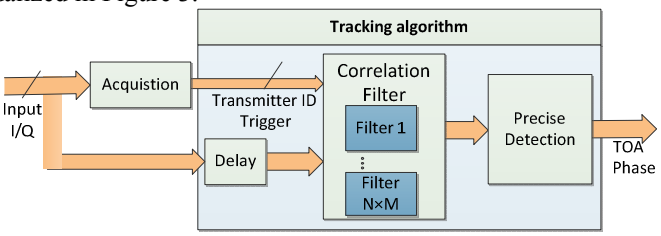

Figure 3. Tracking algorithm data flow.

Once the correlation filter identifies the tracking burst, the receiver proceeds in the tracking mode and hops for the next burst. In case the synchronization is lost, the correlation filter waits for the next trigger from the acquisition.

IV. System Design and Architecture

In the previous section, the receiver algorithm was introduced. The available receiver platform, based on Virtex-II, provides a guideline for the proposed implementation and the efficiency of FPGA resource utilization for the acquisition and tracking algorithims.

It is noteworthy that neither the choosen FPGA, nor the GPU or CPU would be capable to perform the signal procecesing sufficiently on its own. While it would be possible to use a homogeneous platform, consisting of several FPGAs, GPUs or CPUs, to meet the performance requirements, our heterogeneous approach offers the best realization by making use from the peculiarity for each of these technoligies.

Down-conversion, matched filtering and correlation operations involve serial processing on the input data stream. GPU implementation for the acquisition leads to reduced performance and higher latency in comparison with an FPGA implementation. On the other hand, tracking processing is performed on a GPU due to parallel realization of the signal tracking algorithms. As a result, a FPGA was choosen for the implementation of the Acquisition, and the Tracking is performed by a GPU. Further more some parts of the signal processing are performed by the CPU, leading to a balanced load on all system components.

A. *Acquisition Implementation*

In the acquisition algorithm, the first block is a digital down converter. One of the main features for the designed down-mixer in [9] is that no actual mixers or multipliers are needed to acquire the desired spectra from the received IQ components. Down conversion is performed at each stage with shifts $f_s/4$, 0 and - f_s /4, where f_s represents the sampling frequency (100MHz). The transmitter sends the acquisition signal at one of N dedicated channels. With N=9, acquiring these nine channels is performed by using a two stage mixer. The mixed signal is then passed through FIR (finite impulse response) low-pass filter with a polyphase structure which simultaneously performs down sampling.

The input data rate is decimated after the matched filter output by $f_d = f_s/16$ (i.e. 6.25MHz) on each channel. The filtered output (Figure 5 a) is correlated with all the M possible sequences. In consideration of the number of correlation filter needed ($N \times M$), a highly efficient implementation is vital for this design. Correlation is performed when a code sequence, of length L, is compared with the input data stream.

Since the code sequence has a logic representation of 1 and -1, the multiplication in the correlation can be encoded into the FPGA solely by adder or subtraction operations. The IQ correlation filter in Figure 4 operates at a rate 32 faster than input filtered data. This allows the correlation filter to perform several multiply and accumulate operations per data cycle and take advantage of the SRL32 available on the Virtex6 FPGAs [11].

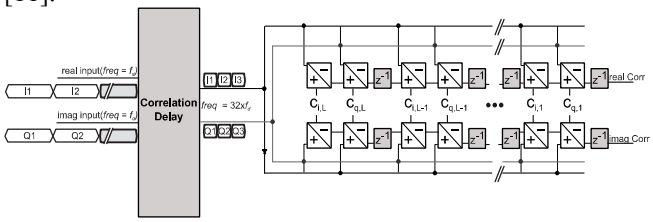

Figure 4. FPGA implementation for a complex correlation filter.

The magnitude of the complex correlation indicates how the received signal resembles the code sequence (Figure 5 b). The output of every correlation result enters a detection filter that detects the presence of the transmitter and identifies the point in time the signal arrived at receiver antenna.

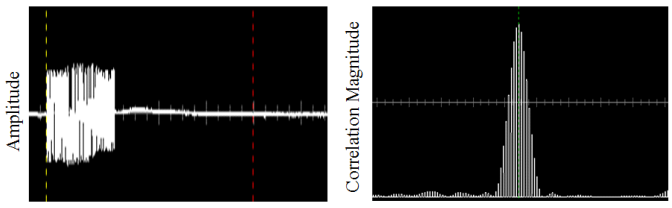

Figure 5. FPGA captured data: a) acquisition signal after filtering and down-conversion; b) correlation filter output.

B. *Tracking Implementation*

In contrast to FPGA acquisition algorithm implementation, the GPU does not perform the tracking algorithm discretely. The GPU is rather a co-processor, controlled by the host system's CPU. The overall tracking correlation process, comprising three distinct steps, is therefore heterogeneous: The window prediction and the post processing of the correlation result are performed by the CPU, while the tracking correlation is outsourced to the GPU.

The window prediction is triggered either from the acquisition signal in acquisition mode or from previously correlated tracking burst in tracking mode. The ADC IQ streamed data, processed by the GPU, is partitioned into 2.5ms segments, and for each segment the window prediction determines the contained tracking bursts. This prediction is used to fill an index buffer, containing the transmitter IDs and the predicted bursts' locations with respect to the segment's origin. The index buffer is transferred to the GPU's memory in parallel with the IQ data stream for further processing. Afterwards the correlation kernel is executed by the GPU for every signal segment containing the tracking bursts. The execution itself is initialized by the CPU application after performing the window prediction for the signal segment. Two distinct CUDA streams are used alternately for this processing, in order to overlap data transfers, CPU processing and GPU operations.

The CUDA kernel's execution configuration is based on the number of bursts that has to be processed per kernel execution: Each tracking burst is processed by one thread block consisting of 15 warps (= 32 threads). The maximum system load, of 50.400 Bursts/second, accounts to an average of 126 thread blocks per execution. On the contrary, a system load of about 11% is sufficient to occupy the maximum of the 15 SMs (Streaming Microprocessors) available on the GPU. While a high number of blocks are beneficial for the GPU's occupancy, a low number does not require the same efficiency as a highly loaded system.

Each block determines the data that has to be processed by the index buffer and its block ID: Block n reads the n-th entry of the index buffer. Based on the retrieved offset, the start of the correlation window is determined. The retrieved transmitter ID is used to determine the reference sequence used for the correlation. After the kernel execution, the corresponding result data is stored in the GPU's main memory and transferred to the host system's main memory for post-processing.

The partitioning between CPU and GPU does not only reflect the CUDA processing flow, but also accounts for the exploitable degree of parallelism: Furthermore the post-processing involves a high number of complex operations, favoring the CPU's or GPU's Floating-Point Unit.

The tracking algorithm part running on the CPU requires only about 70% of one of the cores, which is insignificant compared to the post processing of the correlation results (e.g. position calculation). For the measurement of the GPU utilization two different Nvidia GPUs have been used: A consumer grade Geforce 780 and a Quadro K4000. The execution time for both GPUs is shown in Figure 6.

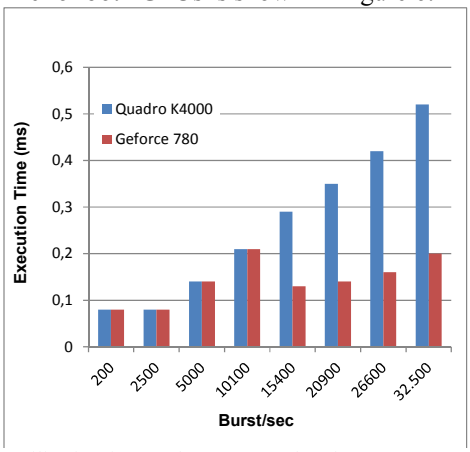

Figure 6. GPU utilization for Quadro K4000 and Geforce 780.

V. INTERFACES

External interfaces define the inputs and outputs to the platform. Meanwhile the internal interface denote the communication between the different component. Figure 7 provides an overview of the platform main interfaces.

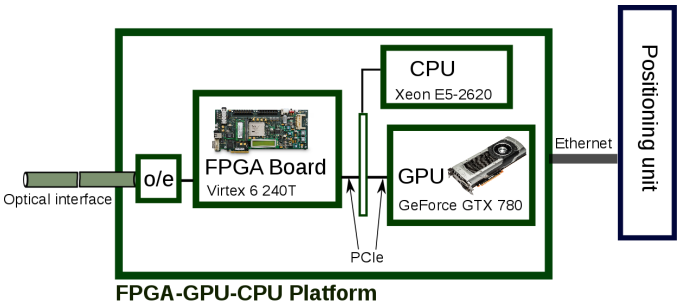

Figure 7. Internal and external main interfaces of the platform.

A. *External interface*

Added to the flexibility in implementing the acquisition algorithm, the FPGA plays a central role in acquiring the data to the platform. This can be achieved through connecting the input (e.g. ADC) directly to the FPGA input pins or by using the Gigabit transceivers. For this platform the FPGA acquires the IQ data through fast serial transmission over long distances by means of optical communication. The FPGA can also function with multiple inputs, for example the used Xilinx 2XC6VLX240T supports 24 GTX transceivers. The processed data are transferred using a standard Gigabit Ethernet to the position computing unit.

B. Internal interface: GPU/FPGA Communication Via PCIe

Data transfer between the FPGA and the GPU is performed in two steps: At first, the data is transferred by the FPGA's DMA controller to the host system's main memory via the 8 Lane PCIe 1.0 interface. Afterwards the GPU's DMA controller reads the data from the host memory.

Basically the tracking algorithm performed on the GPU requires only the data stream window where the tracking burst correlation is executed. However, in typical conditions the bursts from different transmitters occupy more than 75% of the radio channel. Hence the extraction of single correlation windows would provide only a minor reduction of the data volume. Furthermore the extraction would require additional communication between FPGA and CPU: Results of the tracking correlation had to be fed back from the CPU to the FPGA with very low latency. Accordingly the IQ data stream is transferred entirety from the FPGA to host system's RAM, which omits time critical communication between CPU and FPGA at the cost of slightly increased data transfer volume. Likewise the data stream is transferred to the GPU in order to avoid costly in-memory copying.

1) DMA transfer

One critical element of the DMA transfer between FPGA and the host memory is the buffering of the IQ data stream:

Ideally the CPU program continuously initializes DMA transfers, and the stream data is continuously fed from the FPGA to the CPU's RAM. In practice however, latency between DMA transfers has to be considered. This latency has to be bridged by buffering the streamed data prior to the DMA transfer. Based on the data volume, about 0.5KB has to be buffered per microsecond; as a consequence even a buffer of 128KB is not sufficient to prevent buffer overflows reliably.

Instead of increasing the buffer size, e.g. by using DDR memory for buffering, the DMA core was extended with a queuing mechanism and a ring buffer in the host memory. By queuing the transfers up to 12 data chunks, a latency of about 30ms between transfer requests, issued by the CPU, can be bypassed. This allows the continuous streaming of data with a buffer of only 64KB. Bypassing a latency of 30ms without this queuing mechanism would require a buffer of 24MB on the FPGA board.

2) Heterogeneous data streams

Beside the IQ ADC data stream, additional data has to be transferred from the FPGA to the host memory, in order to perform the window prediction. This additional data consists of timestamps associated with the data stream and transmitter IDs and timestamps associated with acquisition trigger. Furthermore data erroneous, originating from buffer overflows, has to be detected.

A valid approach consists of transferring the data along with the IQ input stream by additional DMA channels and/or register based data transfer. This however requires synchronization between the different data streams, for example to relate a register provided timestamp with IQ data segment.

A more efficient approach, with respect to the software's complexity, as well as the utilization of the PCIe bus, consists of keying this additional data in the data stream. This way also the alleged drawback of transferring data between FPGA and GPU via the host system's RAM can be exploited: While data is transferred from the host memory to the GPU memory, the CPU program can in parallel readout the additional data.

VI. Conclusion

In this paper we introduced a novel platform that takes into account the characteristics of FPGAs, GPUs and CPUs. High performance is achieved by optimizing the communication between the FPGA and GPU and making a clever map of the algorithms. The FPGA is required to connect between the external data input and the platform. In addition the FPGA is in charge of performing signal acquisition. This facilitates the parallelism of the GPU-CPU approach for signal tracking and time estimation.

By using standard components (except for the FPGA board) the overall cost is extremely low when compared to other solutions. The performance is demonstrated for a real-time localization system.

References :

[1] T. von der Grün, N. Franke, D. Wolf, N. Witt, and A. Eidloth, "A real-time tracking system for football match and training analysis," in *Microelectronic Systems*, Springer, 2011, pp. 199–212.

[2] R. C. Dixon, *Spread spectrum systems: with commercial applications*. John Wiley & Sons, Inc., 1994, pp. 319–360.

[3] P. Specification, "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet," *DS083 v4*, vol. 5, 2002.

[4] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 126–131.

[5] B. HUANG, Z. YAO, F. GUO, S. DENG, X. CUI, and M. LU, "Reaching for the STARx A Software-Defined All-GNSS Solution," *InsideGNSS*, vol. 9, no. 1, pp. 50–60, 2014.

[6] L. de Souza, J. D. Bunton, D. Campbell-Wilson, R. J. Cappallo, and B. Kincaid, "A Radio Astronomy Correlator Optimized for the XILINX VIRTEX-4 SX FPGA.," in *FPL*, 2007, pp. 62–67.

[7] B. Haberland, F. Derakhshan, H. Grob-Lipski, R. Klotsche, W. Rehm, P. Schefczik, and M. Soellner, "Radio base stations in the cloud," *Bell Labs Tech. J.*, vol. 18, no. 1, pp. 129–152, 2013.

[8] H. Stadali and C. Wagner, "Device and method for determining a correlation maximum," 76499362010.

[9] H. Stadali and C. Wagner, "Mixer for mixing a signal and method for mixing a signal," 7,602,8562009.

[10] M. Breiling, G. Hofmann, H. Stadali, and C. Wagner, "Device and method for determining a correlation value," 7,643,5412010.

[11] XILINX, "Virtex-6 FPGA Configurable Logic Block," *User Guid.*, no. 1.2, pp. 24–26, 2012.