Widefield Science and Technology for the SKA
SKADS Conference 2009
S.A. Torchinsky, A. van Ardenne, T. van den Brink-Havinga, A.J.J. van Es, A.J. Faulkner (eds.)
4-6 November 2009, Château de Limelette, Belgium

A Real-time Software Broadband Beamformer on the IBM Cyclops Multiprocessor System ⋆

A. AhmedSaid

Jodrell Bank Centre for Astrophysics, The University of Manchester, Jodrell Bank Observatory,
Macclesfield, Cheshire, SK11 9DL, UK

Abstract. Digitally beamforming broadband signals of several hundred MHz of bandwidth in real time, is computationally highly expensive and very demanding. Performing this task in software is even more challenging. However, with the emergence of massively parallel multiprocessor chips, it is becoming realistically feasible. The aim of this work is to investigate the implementation effort and the performance achievable by a digital broadband beamformer implemented in software using a state-of-the-art multiprocessor chip. Using the IBM Cyclops processor, a complete software beamformer has been designed, implemented and tested. The software has been particularly optimized for the application and the Cyclops system architecture. It includes a kernel and libraries designed exclusively to maximise performance for this application. The obtained results show that several hundred MHz of bandwidth can be beamformed in real time.

1. Introduction

Beamforming is a technique used to separate signals arriving from different direction at an array of receivers by means of spatial filtering (Veen & Buckley 1988; Krim & Viberg 1996). This technique is used in many applications such as in Telecommunications, Sonar, Radar, Medical imaging, Geophysics, Radio Astronomy...etc (Veen & Buckley 1988). It can be wide-band, if the signals envelope changes noticeably as the wave front passes through the array or narrow-band if the changes are negligible.

It is common to implement wide-band beamformers using narrow-band beamforming techniques. This is made possible by splitting the wide-band signals, using filter banks, into narrow band signals which can be processed using a narrow-band beamformer (Veen & Buckley 1988).

Narrowband beamforming is simply the operation of summing the weighted outputs of the receivers in order to achieve spatial filtering. By properly choosing the weighting coefficients, it is possible to obtain the signals coming from a particular direction while attenuating, ideally completely removing, the signals coming from other directions.

The objective of this work is to investigate, implement and test a broadband, real-rime, software beamformer. The target bandwidth is in the order of several hundred MHz, up to 8 simultaneous beams are desired and 2 polarisations are required. The state-of-the-art IBM multi-processor Cyclops system has been chosen to perform this study.

⋆ This work was supported by the Science & Technology Facilities Council, and by the European Commission Framework Program 6, Project SKADS, Square Kilometre Array Design Studies (SKADS), contract no 011938.

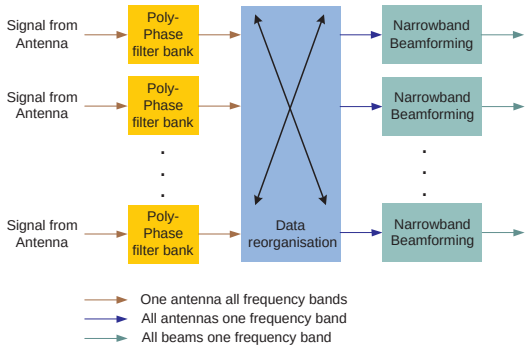

Fig. 1: Typical broadband beamformer

The rest of this paper is organized as follows:
In section 2, a brief description of the beamforming scheme is given and in section 3, the architecture of the Cyclops system is presented. Section 4 gives details of the software development undertaken to achieve the objectives of this work and section 5 presents the simulation results. Finally, conclusions and future work are given in the last section.

2. Digital Broadband Beamforming

A typical broadband beamformer is represented in Figure 1. It consists of three main operations:

- Channelisation using Poly-phase filters;
- Data redistribution/reorganisation;
- Narrowband Beamforming.

Data redistribution/reorganisation involves gathering data, from all antennas, of each frequency band in order to be processed by a narrowband beamformer. This process requires a communication bandwidth proportional to the product of the number of antennas by the sampling frequency. Therefore it constitutes a bottleneck that reduces the maximum signal

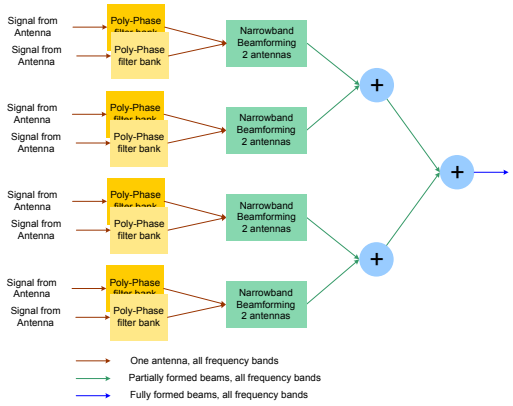

Fig. 2: Scalable and efficient beamformer

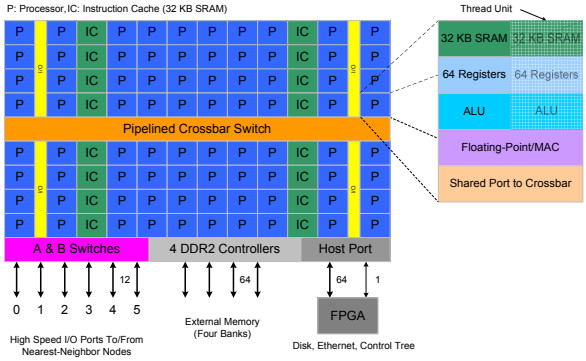

Fig. 3: The Cyclops Processor

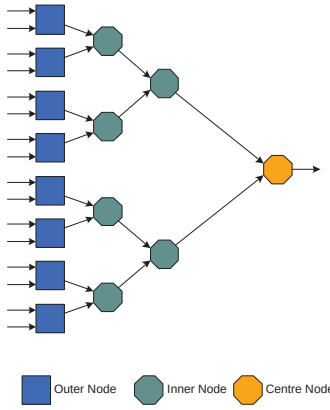

Fig. 4: Logical mapping of a 16 antennas beamformer on a Cyclops system

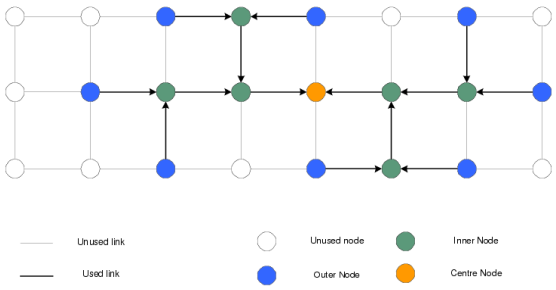

Fig. 5: Physical mapping of a 16 antennas beamformer on a Cyclops system

bandwidth that can be processed and the architecture's scalability. For this reason an alternative beamforming scheme has been developed. This beamforming scheme uses a different partitioning which consists of reducing the problem size antenna wise rather than frequency wise. In this scheme all frequency bands are processed together, however, only a small number of antennas (a minimum of two) are beamformed at once. Figure 2 illustrates this strategy.

After an initial beamforming stage, the resulting data streams are all added together to form the final beam(s). This beamforming method is just a reorganisation of the beamforming operations, it uses the beamforming coefficients calculated for the whole array and is, therefore, not sub-optimal. It is a mathematically equivalent beamforming method.

3. The Cyclops system Architecture

3.1. The Cyclops Processor

Cyclops is a highly parallel IBM chip with 80 processors clocked at 500 MHz and high speed communication links (IBM 2007a,b). Each processor consists of 2 thread units and one floating-point arithmetic unit. The thread units have 32K of RAM, 64 registers (64 bits wide) and an ALU. A Cyclops system is made of nodes arranged in a 3D grid configuration and containing one Cyclops chip each. The nodes communicate with each other using a device called the ASwitch. The ASwitch has 6 bidirectional high speed ports to allow communication in the three orthogonal directions (x,y, and z axis). The beamformer implementation presented in this document has been designed and optimised for this system.

3.2. Beamformer mapping on a Cyclops system

The architecture illustrated in Figure 2 is mapped logically into a Cyclops system as shown in Figure 4. In this mapping, there are three types of nodes: The Outer nodes perform the beamforming of two antenna signals, the Inner nodes perform the addition of two signals and the Centre node performs the addition of two signals and outputs the result to 'a host computer'.

A Cyclops system consists of $8 \times 3 \times 2$ shelves of Cyclops blades (nodes). The best physical mapping fit is presented in Figure 5.

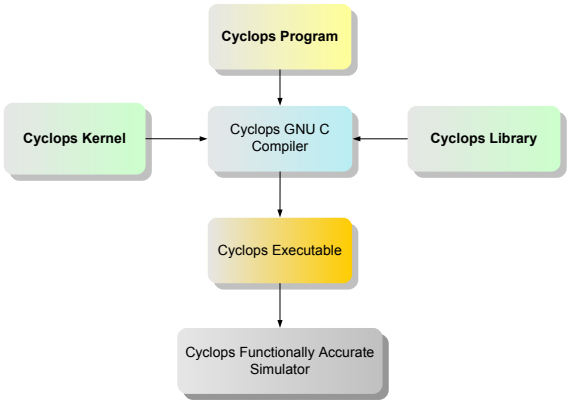

Fig. 6: Cyclops Work Flow

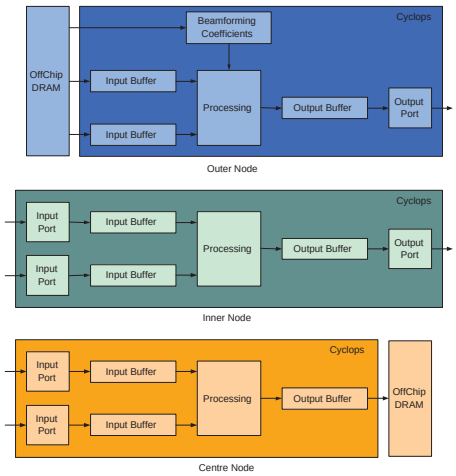

Fig. 7: Data path for each node type

4. Beamformer Software Design and implementation

The Cyclops software development environment used is based on the C GNU tools (Free Software Foundation 2009). In addition, a Kernel and libraries are provided to support the design of high level C applications. The evaluation of the provided libraries revealed that the communication functions are not suitable for our application because of large overheads which make them too slow. Therefore, it was necessary to develop communication functions that are simpler and more efficient. However, some restrictions with the Kernel provided do not allow full control of the Cyclops chip making it necessary to develop a new kernel as well. As a result, the kernel and libraries provided were not used and new ones have been developed. The design methodology adopted for the software, favours speed and efficiency over flexibility, simplicity and compile time parameterisation over complexity and runtime variability. The reason for these choices is that the target application is real-time signal processing and not general purpose computing.

4.1. High performance kernel

The design of the kernel was guided by the need for efficiency, speed, flexibility and compatibility with the available C compiler. The kernel has been written in assembly, it is very minimalistic and it performs the following tasks:

- Setup interrupt handlers;
- Initialise the thread units;
- Start the main program.

All the thread units execute the same initialisation code and run the same main C program. Thread units can be instructed to perform different tasks by using thread IDs, node IDs and conditional statements (ex if(ID==1) do this;).

4.2. High performance library

The library contains very low latency functions that can be grouped into the following categories:

- Printing and File I/O functions that can be used within the simulator environment only;
- System control functions;
- Communication functions.

The main performance limiting factor in our application is the communication bandwidth. In order to maximise it, a number of restrictions have been adopted:

- Communications occur between neighbouring nodes only;
- Packet sizes are fixed and predefined at compile time;
- Headers are fixed and predefined at compile time;
- Input and Output buffers are predefined at compile time;

In order to modify the system parameters to suit an application's need, a number of constants (such as the packet size, the buffers size, the headers) are defined in a C header file. These compile time constants offer a good compromise between flexibility and efficiency.

4.3. Beamformer design

The data path is quite similar for all node types and is illustrated in Figure 7. Data is received into 2 input buffers, it then passes through a processing stage and the result is stored in an output buffer. The difference between the three node types is the source and destination of the data. The Outer nodes receive the beamformer input data which is read from the Off-chip memory and the result transmitted via an ASwitch port. For the Inner nodes, data is received and transmitted via an ASwitch port, and for a Centre node, data is received via an ASwitch port but is stored in the Off-chip memory from where it can be read by a host computer.

The other difference between the node types is the processing. The Outer nodes perform the beamforming of two input signals whereas the Inner and Centre nodes perform the addition if two data streams (partial beams). Figure 8 shows the chronological progression of data through the pipeline

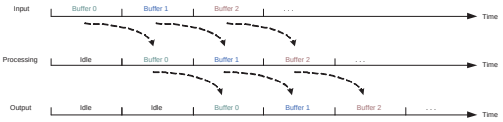

Fig. 8: Pipeline stages

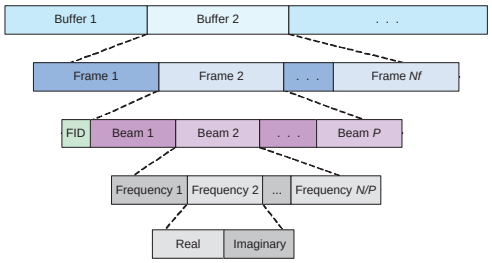

Fig. 9: Data streams structure

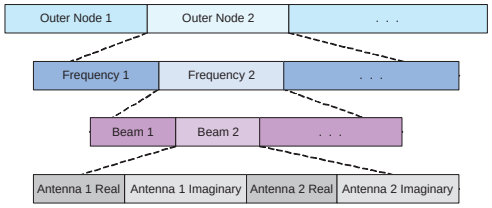

Fig. 10: Beamforming coefficients structure

stages inside a node.

The output data stream structure is presented in Figure 9. The output data is generated in a buffered fashion with each buffer containing a number of frames. This is due to the fact that communications between nodes is performed in a buffered and packetised way (a packet encapsulates one frame of data). Each frame is divided into P sub-frames to hold data for P beams ($P = 1..8$). Each sub-frame, holds N/P complex samples (where N is the total number of samples per frame). The frames are preceded by an 8 bits Frame ID (FID). The FID is simply a counter incremented from 0 to 255. It is used to make sure that the frames are processed in the right order.

The beamforming coefficients are used by the Outer nodes only. They are stored as an $(N/P) \times P \times 2$ matrix of floating-point double precision complex data. Their structure is presented in Figure 10.

The coefficients are stored in three buffers as illustrated in Figure 11. Two of them are reserved in the SRAM (Onchip RAM) and are used as a double buffer to facilitate run-time updating without stopping the beamforming program. When one of them is being updated the other is used for beamforming then the second buffer is updated while the first one is used for beamforming. These buffers are updated by copying the content of the third buffer which is reserved in the DRAM (Offchip RAM). The updating is performed when a flag (a variable in the shared memory space) is set to 1. After the

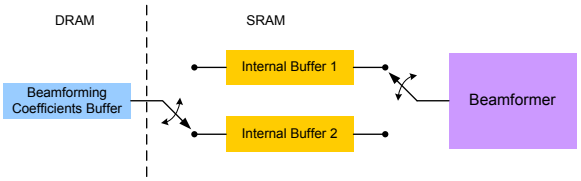

Fig. 11: The beamforming coefficients buffers

update, the flag is reset to zero.

The DRAM coefficients buffer is updated by receiving new coefficients from the host system. Its content can be changed at any time without interference with the beamforming operation. When new coefficients are uploaded a flag is set to 1 to signal that new coefficients are available.

4.4. Implementation results

The beamformer has been implemented and tested on the Cyclops functionally accurate simulator (Cuvillo et al. 2005) (version 4.0). It features the following characteristics:

- 2 Polarisations;
- 1,2,4 or 8 beams;
- 1×8 bits input data and 2×8 bits output data;
- Mixed arithmetic precision: Coefficients are applied in double floating-point precision and partial-beams are accumulated in 8 bits integer format;
- On-the-fly updatable beamforming coefficients;
- Independent from array geometry and antenna choice;
- Can be used with any advanced beamforming coefficients computation algorithm;
- Can trade-off between the bandwidth and the number of beams.

The beamformer has been designed to provide the highest bandwidth possible for any given number of beams. Therefore, it obeys the following formula:

$$2 \times NumBeams \times NumPolarisations \times Bandwidth = K$$

Where K is the total data rate achievable by our code on the Cyclops architecture.

This formula means that bandwidth can be traded for beams and vice-versa.

Simulator tests give an estimate of $K \approx 1.28$ GB/s. Therefore the formula can be simplified to:

$$NumBeamsXBandwidth = 320$$

For example for 8 beams, 2 polarisations, max 40MHz per beam. For 1 beam it is 320MHz.

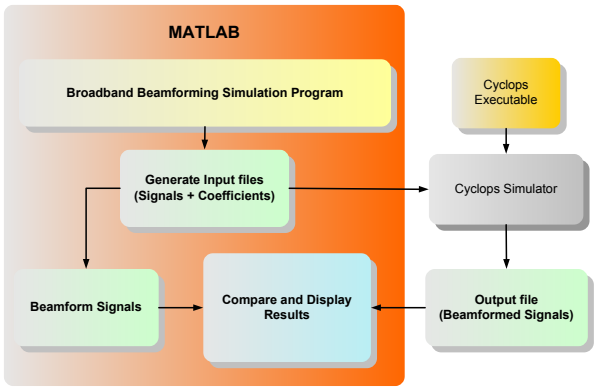

Fig. 12: The simulation setup

5. Numerical Test, Simulation and Evaluation

An advanced test and simulation program has been written in MATLAB. This program tests the Cyclops beamformer by generating input data and beamforming coefficients which are passed to the Cyclops simulator to use them as an input to the beamformer. The same data is also beamformed in MATLAB and the results are compared. The operation of the test program is illustrated in Figure 12.

Data generation is performed by modelling the signals received by an antenna array. The model uses Gaussian noise as the waveform for the incoming signals whose number is set in the MATLAB program and directions of arrival chosen randomly. The program also models the coupling between antennas and it takes into consideration the fact that the array can be in the middle of a bigger array with the outside antennas contributing to the mutual coupling but their received signals are unavailable.

The results of this algorithmic and numerical testing are presented in Figure 13. In this figure, 2 frames of samples are visualised. In red is the beamformer output from the Cyclops program and in blue is the MATLAB output. The green plot is the difference between the two. It can be seen that the Cyclops beamformer performs well and produces the expected output. The small differences between the two beamformers are due the integer arithmetic operations used in the Cyclops beamformer.

6. Conclusions and future work

The feasibility of a real-time software broadband beamformer has been demonstrated. However, a significant software development was required to achieve a good performance. The test results revealed that the I/O bandwidth is the main performance limiting factor.

Scalability is also an issue with the predefined node interconnect configurations. For example, no matter how big a 3D grid can be, it is impossible to grow the tree architecture

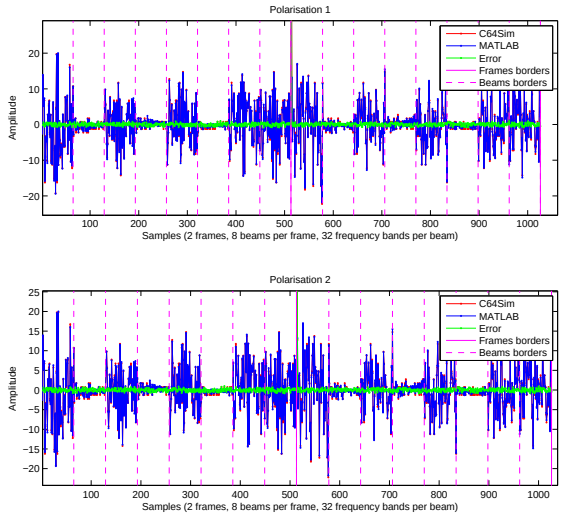

Fig. 13: The Cyclops Processor

(Figure 4) indefinitely (estimate: max 64 input nodes).

The future work will be to confirm the findings by testing a 2×8 inputs beamformer on a real $3 \times 3 \times 3$ nodes Cyclops system.

References

Cuvillo, J. D., Zhu, W., Hu, Z., & Gao, G. R. 2005, in 2nd Annual International Symposium on Computer Architecture

Free Software Foundation. 2009, GCC, the GNU Compiler Collection, http://gcc.gnu.org/

IBM. 2007a, 64-Bit Cyclops Principles of Operation Part I,

IBM. 2007b, 64-Bit Cyclops Principles of Operation Part II,

Krim, H. & Viberg, M. 1996, Two decades of array signal processing research, IEEE Signal Processing Magazine

Veen, B. D. V. & Buckley, K. M. 1988, Beamforming: A Versatile Approach to Spatial Filtering, IEEE ASSP Magazine