Viabilidad de acelerar la transmisión de datos entre una CPU y una FPGA a través del puerto PCIe (Peripheral Component Interconnect Express) usando compresión de datos.

IVÁN FELIPE OBREGÓN CARREÑO

JULIÁN GONZALO MANTILLA ARIAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
Bucaramanga
2014

Viabilidad de acelerar la transmisión de datos entre una CPU y una FPGA a través del puerto PCIe (Peripheral Component Interconnect Express) usando compresión de datos.

IVÁN FELIPE OBREGÓN CARREÑO

JULIÁN GONZALO MANTILLA ARIAS

Trabajo de Grado para optar al título de

Ingeniero Electrónico

Director

Phdc. Carlos Augusto Fajardo Ariza

Co-director

ing. Carlos A. Angulo Julio

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
Bucaramanga
2014

CONTENT

LIST OF FIGURES

LIST OF TABLES

RESUMEN

Título:

VIABILIDAD DE ACELERAR LA TRANSMISIÓN DE DATOS ENTRE UNA CPU Y UNA FPGA A TRAVÉS DEL PUERTO PCIE (PERIPHERAL COMPONENT INTERCONNECT EXPRESS) USANDO COMPRESIÓN DE DATOS.[1]

Autores: Iván Felipe Obregón Carreño[2], Julián Gonzalo Mantilla Arias[2]

Palabras Claves: PCIe, FPGA, descompresión, Huffman.

Las FPGA's han sido muy efectivas en el área de la computación de alto rendimiento, ya que estas permite a los diseñadores aplicar técnicas de co-diseño para distribuir la carga computacional entre software y hardware. Sin embargo, para maximizar el rendimiento de las aplicaciones intensivas de datos – como son los algoritmos sísmicos – la velocidad de envío desde el *host* (CPU) al dispositivo (FPGA) debe ser muy similar con la velocidad de procesamiento del dispositivo.

En la actualidad este es una brecha entre las velocidades de transferencia y procesado, porque esta última son millones de veces más grandes que la primera. Este inconveniente no permite a los diseñadores usar los recursos de la FPGA de manera eficiente. Nosotros hemos implementado una estrategia de compresión para reducir l tamaños de los datos que van a ser transferidos. De este modo el tiempo de transferencia se ve reducido. El algoritmo incluye una cuantización uniforme y un esquema de codificación Huffman.

Nuestra implementación muestra una reducción en el tiempo de transferencia a pesar de los tiempos de procesado de compresión y descompresión. Los resultados experimentales de la transferencia de los datos sísmicos muestra como la estrategia propuesta mejora hasta 8 veces más rápido que el método tradicional de transferencia.

[1] Trabajo de Grado modalidad en investigación

[2] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: PhDc. Carlos A. Fajardo. Codirector: M.Sc Carlos Angulo Julio.

ABSTRACT

TITLE:

VIABILITY OF SPEED UP DATA TRANSMISSION BETWEEN A CPU AND FPGA THROUGH THE PCIE (PERIPHERAL COMPONENT INTERCONNECT EXPRESS) PORT USING DATA COMPRESSION[3]

TITLE: Iván Felipe Obregón Carreño[4], Julián Gonzalo Mantilla Arias[4]

KEYWORDS: Co-design, Huffman coding, PCI Express, Seismic data transmission.

The FPGAs have been effectively used in high performance computing because they allow designers to apply co-design techniques to distribute the computational load between software and hardware. However, in order to maximize the performance of data-intensive applications - such as seismic algorithms - the transfer speed from the host (CPU) to the device (FPGA) must be similar to the device processing speed. Currently there is a gap between the transfer speed and the processing speed, because the last one is hundreds of times greater than the first one.

This gap does not allow designers to use the FPGA resources efficiently. We have implemented a compression strategy that reduces the amount of data to be transferred, and in this way the required transfer time is lower. The algorithm includes a uniform quantization and Huffman coding scheme. Our implementation shows a reduction in the transfer time despite the time used in the compression and decompression processes. Experimental results transfering seismic data show how the proposed strategy is up to 8 times faster than the traditional transfer method.

[3] Degree Project

[4] Faculty of Physics Mechanics Engineering. Electrical, Electronics Engineering and Telecommunications School. Director: PhDc. Carlos A Fajardo. Codirector: M.Sc. Carlos Angulo Julio.

INTRODUCTION

One of the main challenges in a computer system is to overcome the growing gap between bandwidth and computational throughput, because computation speed is hundreds of times greater than transfer speed. The FPGAs have been used successfully to speed up seismic applications [1]. However, these implementations also are limited by PCIe (Peripheral Component Interconnect express) port bandwidth [2].

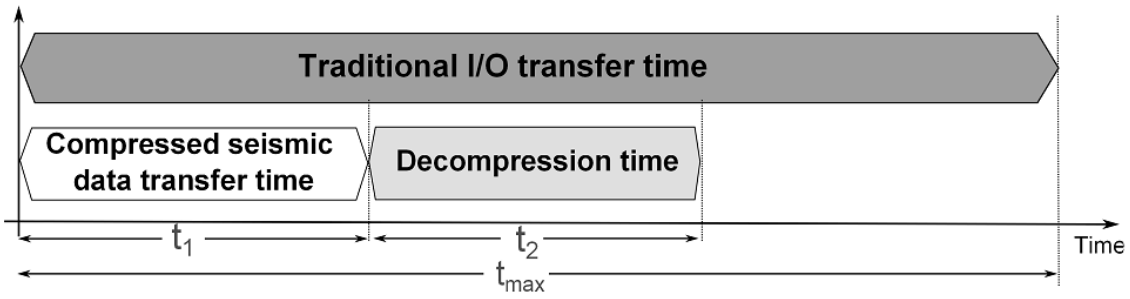
The principal strategy to overcome the bandwidth limitations in FPGA implementations is to exploit the memory hierarchy [3]–[5]. Another strategy includes the use of fixedpoint instead of float-point to reduce the number of bits to be sent [6].

As far as we know, the only published research that uses seismic data compression to reduce overall I/O and memory access times was developed by Aqrawi and Elster in 2011 [7]. The work aimed to overcome memory bandwidth limitations in GPU implementations. A multithreaded compression of seismic images was implemented, achieving a bandwidth speed up to 6x by using a compression scheme that consist of transformation (DCT 3D), uniform quantization and coding (Huffman and RLE).

This work is part of a project that seeks to use the FPGAs to accelerate seismic applications, however, is required to overcome the bandwidth limitations [1] in order to use effectively this technology. We aim to use data compression to improve the seismic data transfer between an FPGA and a CPU through the PCIe port. The idea is that the seismic data could be compressed at the acquisition time and then transmitted to the computing center where the seismic data will be decompressed and processed into the FPGAs.

The time requirements for this strategy are shown in Figure 1 and the Equation 1. The compressed data transfer time (t1) plus the decompression time (t2) has to be less than the I/O traditional time (tmax). Otherwise, the strategy will not improve the transfer speed.

Figure 1 Time requirements (Adapted from [7])



$$t_1 + t_2 < t_{max} \qquad (1)$$

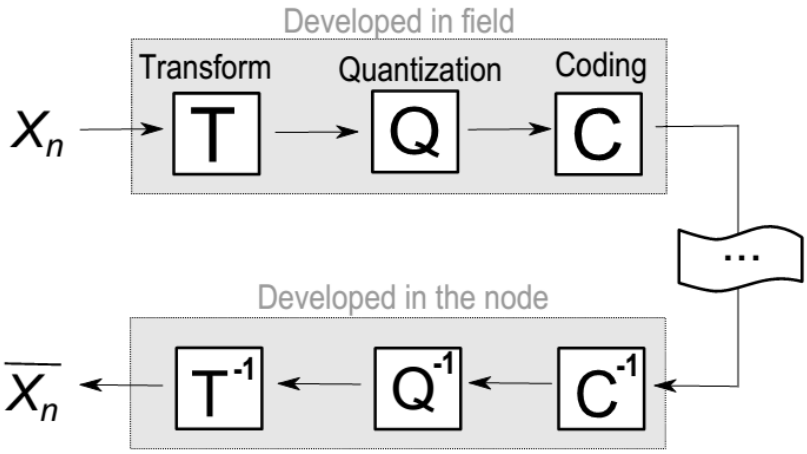
We decided to take into account only the decompression time, because the compression process could be realized during the seismic survey. We assume that there is enough time to compress the data while the seismic survey is developed [1]. Typically, seismic data compression has been used to make the storage more efficient and to reduce time and costs related to network and satellite transmission. Several works have been developed, which aim to reach a high compression ratio above of 40 dB in SNR [8]–[10]. However, our goal is not only to compress as much as possible the seismic data – to reduce *t1* – but also to achieve a good FPGA implementation – to reduce *t2* – so as to accomplish the equation 1.

This paper is organized as follow: The second section presents the algorithm used to perform the data compression in order to reduce the data transmission time. In the third section are shown the hardware and software components used in this project. Section four presents the results obtained in the tests and finally in the section five are exposed the conclusions obtained with the strategy proposed.

1. COMPRESSION ALGORITHM USED

Seismic data is correlated and its energy is generally concentrated at the lower frequencies. This data can be considered as a combination of three types of components: geophysical information, redundancy of this information, and uncorrelated and broadband noise [8], [11]. In the literature, some works make use of different strategies to compress seismic data decreasing redundancy. Commonly, these strategies use a transform first, then a uniform quantization and finally a coding scheme. (Figure 2). On the other hand, decompression algorithms perform the inverse process of each one of these steps [8], [12], [13].

Figure 2 Traditional compression algorithm



The transformation stage allows the decorrelation of the seismic data, that is, the representation of the seismic data in terms of coefficients is more compact than the original representation, meaning that the transformed data has better chance to reach an improved compression ratio. The quantization process consists in an approximation of the floating-point transform coefficients by a finite set of values. Finally, the coding stage is a lossless compression strategy which seeks a new represention of the data in such a way the average number of bits per symbol is lower than the number of bits per symbol in the original data. In order to reduce the

decompression time, we implement an algorithm which consists of a uniform quantization and a Huffman coding scheme [14], in other words, we did not use the transform stage (Figure 3).

Figure 3 Compression algorithm implemented

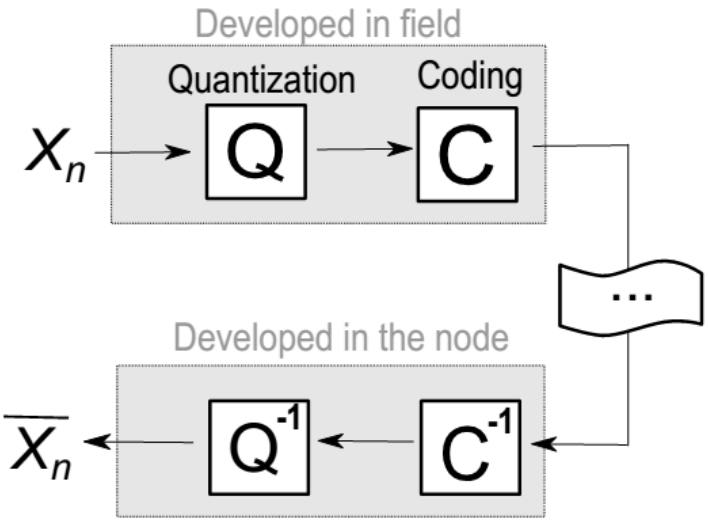
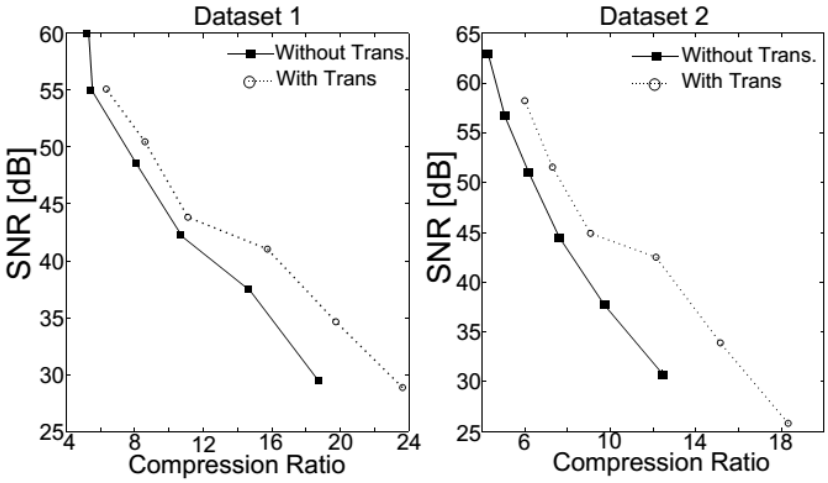


Figure 4 shows the difference, in terms of compression ratio, between the traditional algorithm and the proposed algorithm. The algorithms were applied to two different seismic data (Dataset 1 and Dataset 2).

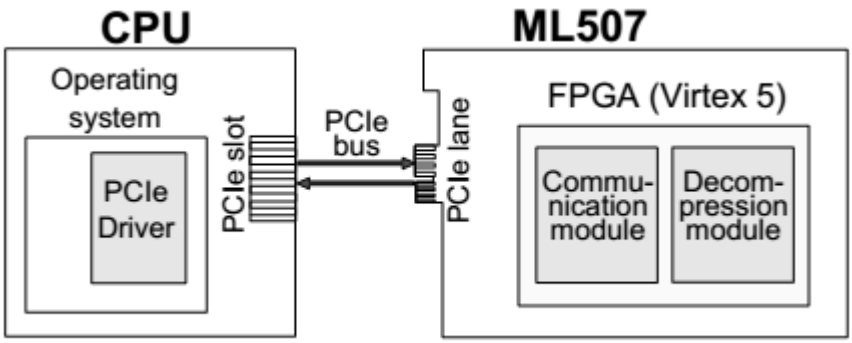
Figure 4 Two methods of compression



2. PROPOSED ARCHITECTURE

The platform used to test our strategy is shown in Figure 5. The system consists of a 3 GHz Pentium 4 CPU and a LXT ML507 Development Board which includes a Virtex 5 XC5VFX70T FPGA.

Figure 5 Co-processing platform



2.1. PCIE DRIVER (CPU)

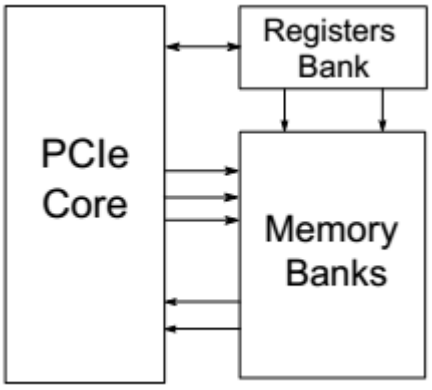
The first step to achieve the CPU-FPGA communication was the implementation of a driver which is provided by Xilinx [15]. Some modifications in this driver were required because the original version only allows write operations in the whole memory, not in an specific position. We implemented an offset mechanism, called llseek [16], that permitted us to write data in a particular memory position.

2.2. COMMUNICATION MODULE (FPGA)

We used the Endpoint Block for PCI Express provided by Xilinx [17] as start point. This module consist of a PCIe core and four memories.

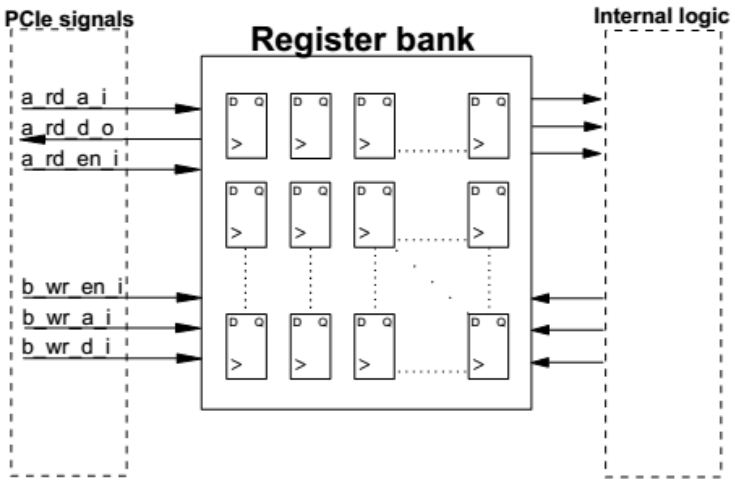
1. *PCIe core*: This module offers the hardware description required by the communication protocol to connect and to transfer data via the PCIe port.
2. *Memories*: Originally the Endpoint Block has four memories to store the data sent through PCIe. These memories have a capacity of 512 × 32 bits each one. So as to increase the memory capacity and to have direct access to the PCIe protocol signals, we eliminated these memories and designed our custom memory system (Figure 6).

Figure 6 Communication module



- Registers Bank: In order to have an easy access to some parameters needed during the decompression process, a 32×32 register bank was implemented. As is shown in Figure 7, the bank is connected to PCIe signals. This connection allowed us to control these parameters from software in CPU.

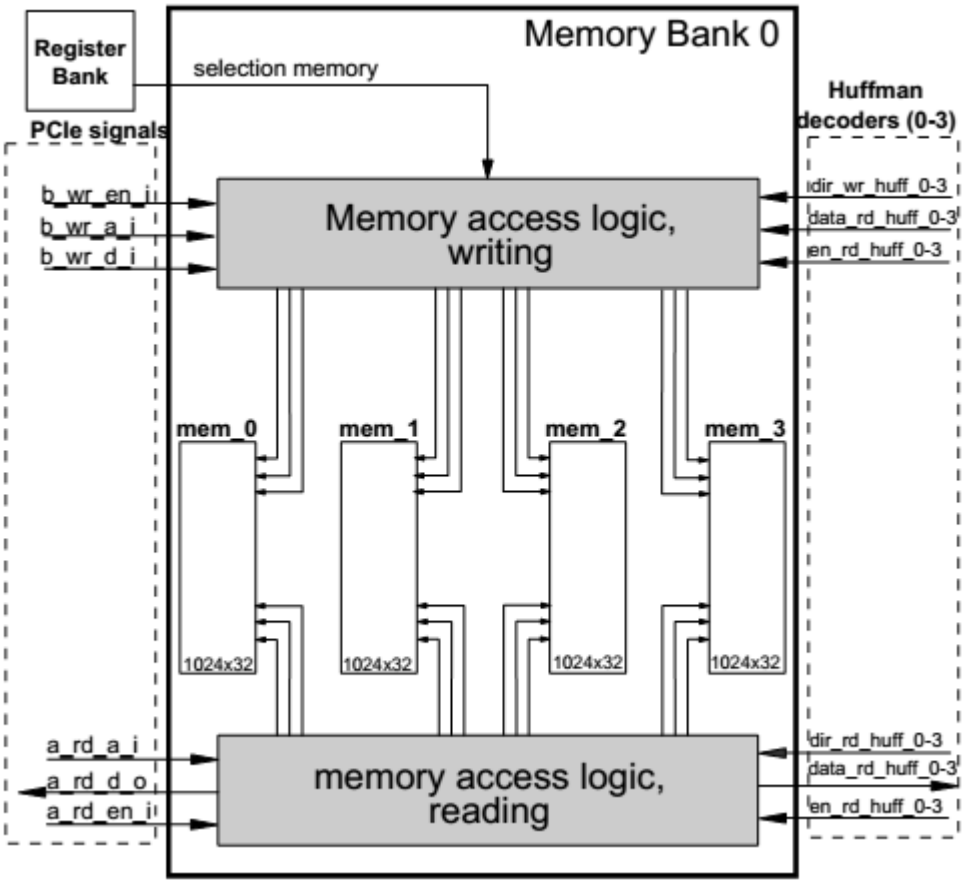
Figure 7 Register Bank



- Memory Banks: The memories were arranged in three memory banks. The first memory bank was used to store the Huffman dictionary necesary to decompress the data, the second bank was used to store the compressed data, while the third bank was used to store the decompressed data. Figure 8 hows one of the memory banks implemented to store the transferred data through PCIe port. These memories were implemented using the blocks RAM that are available in the FPGA [17].

  Each memory bank has four dual port memories with 1024 × 32 bits each one, which are connected in parallel and share the same signals from communication module. The memory ports inside each bank were multiplexed and controlled by the select memory in the internal logic, in this way it is possible to access to each one of the memories from the hardware in FPGA and the software in CPU. However, access priority was given to the hardware to ensure that the memory will not have modifications during the decompression process.

Figure 8 Memory Bank



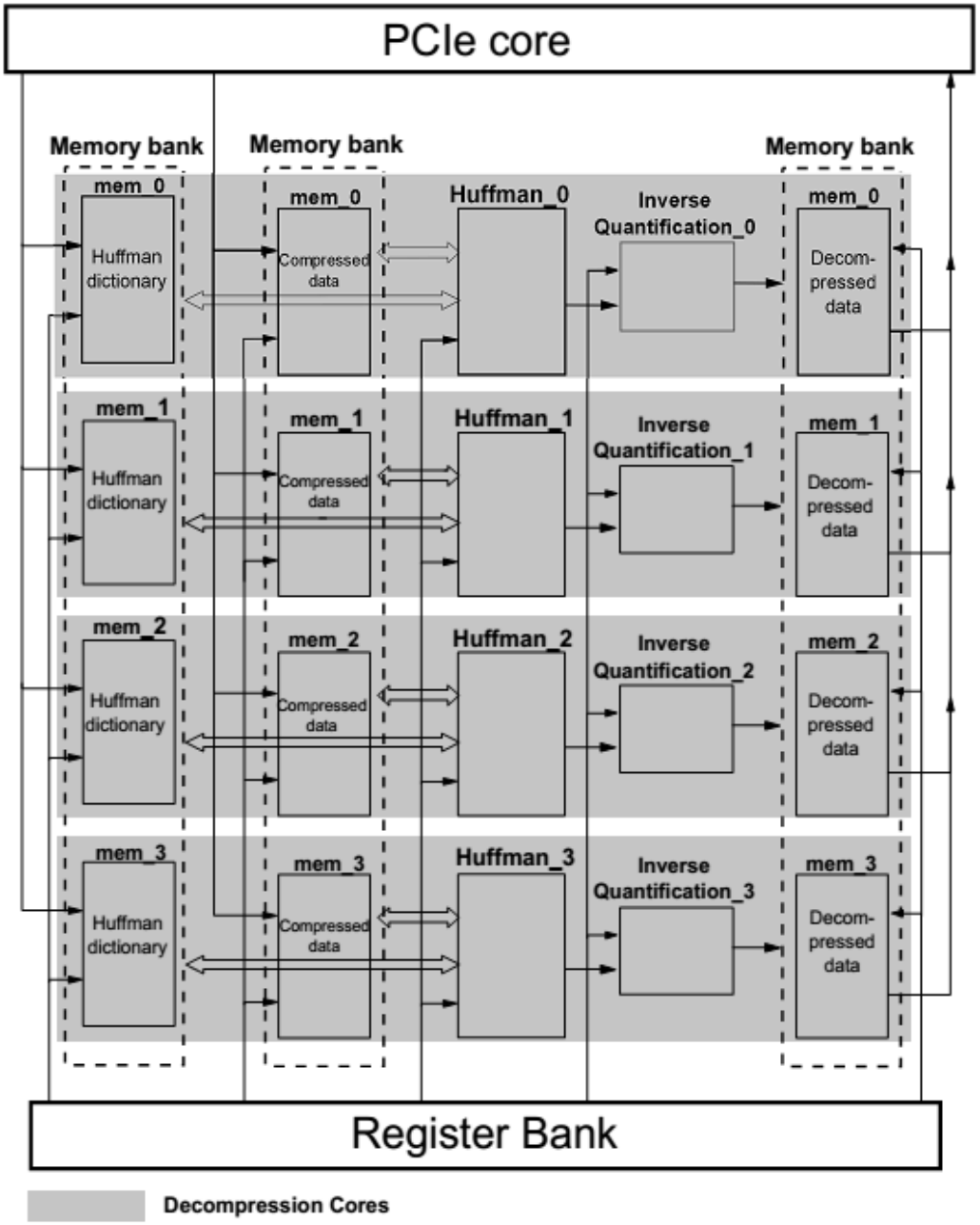
2.3. COMMUNICATION MODULE (FPGA)

Our implementation has four decompression cores working in parallel. As is shown in Figure 9, each core has a Huffman decoder, an inverse quantization module and three memories. Additionally, each core has access to the PCIe core and the *Registers Bank*.

1. *Huffman Decoder*: To implement the decoding process we used the decoder developed by Angulo et al. [18] which is an FPGA implementation of a Huffman decoder. This decoder (Figure 10) includes RAM memories to store the dictionary, the address generator for those memories, shift

registers, comparators and the control unit which controls the whole Figure 9 Parallel Implementation.

Figure 9 Parallel Implementation



2. *Inverse Quantization*: The inverse quantization was done by a multiplication and an addittion. Figure 11 shows the implementation of this module. The flip flop in the middle of both operations, allow us to developed this operations (multiply and add) in parallel.

Figure 10 Huffman Decoder.

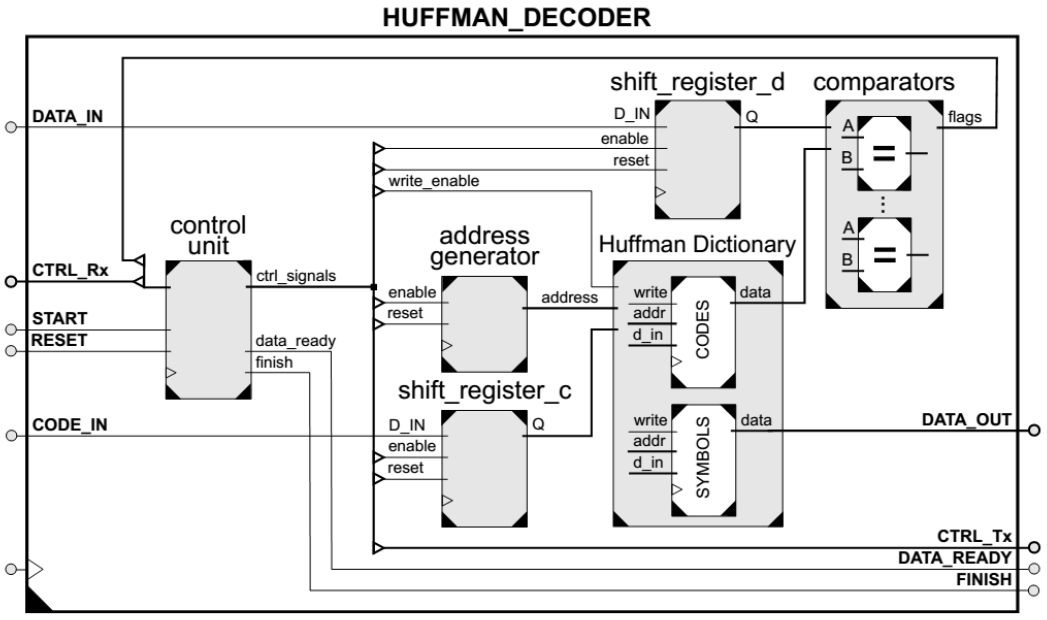
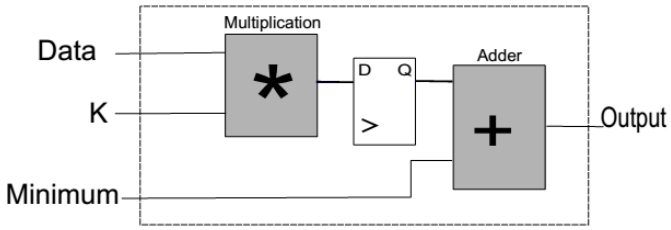


Figure 11 Inverse Quantization



3. RESULTS

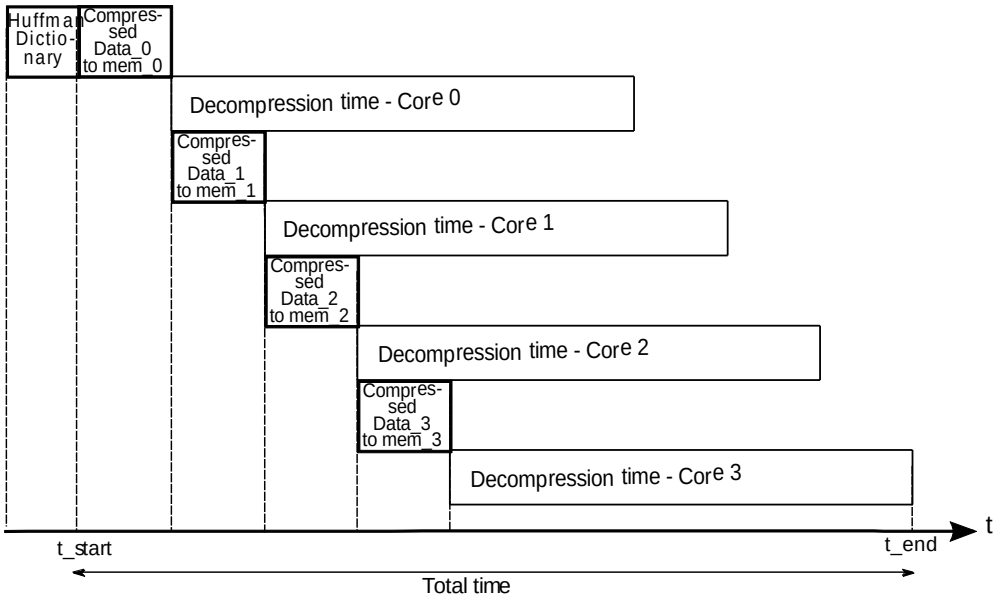
For testing purposes, we compressed different seismic traces, which belong to a marine acquisition provided by the project sponsor. We were not interested in to assure 40 dB in the decompressed data. Instead, we focused on to obtain a wide range of compression ratios so as to test the strategy. Table I summarizes the datasets used, in terms of compression ratio (CR) and SNR.

Table 1 Datasets Used.

Number of the Traces	CR	SNR
28	11.84	42.34
35	12.86	41.45
50	16.82	32.13
30	18.04	30.84
64	22.31	29.5
69	24.37	29.52

Figure 12 shows the time schedule that was carried out in our tests. First, we sent the Huffman dictionary, which was saved in each memory of the memory bank # 1. After that, we sent the first section of the compressed data, wich was saved in the first memory of the memory bank # 2, then we sent the second section of compressed data and so on. As is shown in Figure 12, once the first data section has been sent, the first decompression core begins to operate, and so on.

Figure 12 Parallel Time.

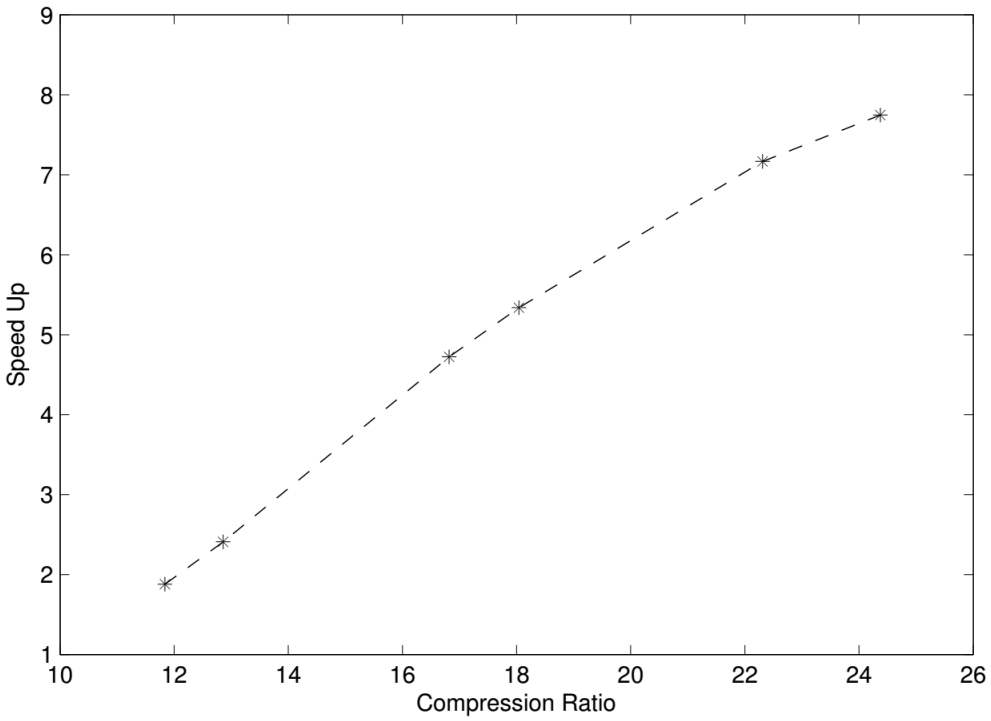


So as to test the compression strategy, first, we measured the time required to transfer the seismic traces in the traditional way, i.e. without compression (t_T). Then, we measured the time required to transfer the compressed data (t_{CD}) and the time required to decompress them (t_D). The speed up was calculated by the equation 2.

$$Speed\ Up = \frac{t_T}{Total\ time} \qquad (2)$$

As is shown in Figure 13, as the compression ratio is increased, the speed up reached is higher.

Figure 13 Speedup vs Compression Ratio.



4. CONCLUSIONS

We tested the possibility to speed up transfer process of seismic data between a CPU and an FPGA, using a compression data strategy. Our results suggest that is possible to overcome this bandwidth limitation using the proposed strategy.

The speed up in transfer process is related to the compression factor and the performance of the FPGA implementation. The bottleneck in the decompression process is the Huffman decoding, because this process has a high computational cost owing to of the variable length of its codes.

A future work could focus on developing a parallel version of the Huffman decoding. Additionally, we suggest to test the strategy using a compression scheme that contains the transform stage.

ACKNOWLEDGMENTS

This work is supported by Colombian Oil Company ECOPETROL and COLCIENCIAS as a part of the research project grants No. 531/2011 and 511/2010. The authors gratefully acknowledge CPS research group at Industrial University of Santander for their constant support.

REFERENCES

[1] C. Fajardo, J. Castillo, and C. Pedraza, “Reduction of computation time of Seismic Migration using FPGAs and GPGPUs: A review article.” Ingenería y Ciencia, vol. 9, no. 7, pp. 261–293, 2013.

[2] H. Fu, W. Osborne, R. G. Clapp, and O. Pell, “Accelerating seismic computations on FPGAs–from the perspective of number representations,” in 70th EAGE Conference & Exhibition, 2008.

[3] H. Jiang and V. Owall, “FPGA implementation of real-time image convolutions with three level of memory hierarchy,” in FieldProgrammable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on, Dec 2003, pp. 424–427.

[4] H. Morishita, Y. Osana, N. Fujita, and H. Amano, “Exploiting memory hierarchy for a computational fluid dynamics accelerator on FPGAs,” in ICECE Technology, 2008. FPT 2008. International Conference on, Dec 2008, pp. 193–200.

[5] C. He, W. Zhao, and M. Lu, “Time Domain Numerical Simulation for Transient Waves on Reconfigurable Coprocessor Platform,” 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’05), pp. 127–136, 2005.

[6] O. Pell, R. G. Clapp et al., “Accelerating subsurface offset gathers for 3d seismic applications using FPGAs,” in 2007 SEG Annual Meeting. Society of Exploration Geophysicists, 2007. [7] A. Aqrawi and A. Elster, “Bandwidth Reduction through Multithreaded Compression of Seismic Images,” 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, pp. 1730–1739, May 2011.

[8] M. Al-Moohimeed, “Towards an Efficient Compression Algorithm for Seismic Data,” Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific, pp. 550–553, Aug 2004.

[9] W. Wu, Z. Yang, Q. Qin, and F. Hu, “Adaptive Seismic Data Compression Using Wavelet Packets,” 2006 IEEE International Symposium on Geoscience and Remote Sensing, no. 3, pp. 787–789, Jul. 2006.

[10] W. Xi-zhen, T. Yun-tina, G. Meng-tan, and J. Hui, “Seismic data compression based on integer wavelet transform,” ACTA SEISMOLOGICA SINICA, vol. 17, no. 04, pp. 123–128, 2004.

[11] L. Duval and T. Rosten, “Filter bank decomposition of seismic data with application to compression and denoising,” SEG Annual International Meeting Soc. Expl. Geophysicists, pp. 2055–2058, 2000. [12] P. Aparna and S. David, “Adaptive Local Cosine transform for Seismic Image Compression,” 2006 International Conference on Advanced Computing and Communications, no. x, pp. 254–257, Dec. 2006.

[13] F. Zheng and S. Liu, “A fast compression algorithm for seismic data from non-cable seismographs,” 2012 World Congress on Information and Communication Technologies, pp. 1215–1219, Oct. 2012.

[14] D. Huffman, “A method for the construction of minimum-redundancy codes,” Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, 1952. [15] J. Ayer, “Using the Memory Endpoint Test Driver (MET) with the Programmed Input/Output Example Design for PCI Express Endpoint Cores,” 2010. [Online]. Available: http://www.xilinx.com/ support/documentation/application notes/xapp1022.pdf

[16] J. Corbet, A. Rubini, and G. Kroah-Hartman, Linux Device Drivers, 3rd Edition. O’Reilly Media, Inc., 2005.

[17] Xilinx, “LogiCORE IP Endpoint Block Plus v1.14 for PCI Express,” 2010. [Online]. Available: http://www.xilinx.com/ [18] C. Angulo, C. Fajardo, O. Reyes, and J. Castillo, “FPGA implementation of a Huffman decoder for high speed seismic data decompression,” in 2014 Data Compression Conference, no. 0266. Salt Lake, United States.: IEEE Comput. Soc, 2014, p. 396.

BIBLIOGRAPHY

C. Fajardo, J. Castillo, and C. Pedraza, “Reduction of computation time of Seismic Migration using FPGAs and GPGPUs: A review article.” Ingenería y Ciencia, vol. 9, no. 7, pp. 261–293, 2013.

C. He, W. Zhao, and M. Lu, “Time Domain Numerical Simulation for Transient Waves on Reconfigurable Coprocessor Platform,” 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’05), pp. 127–136, 2005.

D. Huffman, “A method for the construction of minimum-redundancy codes,” Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, 1952. [15] J. Ayer, “Using the Memory Endpoint Test Driver (MET) with the Programmed Input/Output Example Design for PCI Express Endpoint Cores,” 2010. [Online]. Available: http://www.xilinx.com/ support/documentation/application notes/xapp1022.pdf

F. Zheng and S. Liu, “A fast compression algorithm for seismic data from non-cable seismographs,” 2012 World Congress on Information and Communication Technologies, pp. 1215–1219, Oct. 2012.

H. Fu, W. Osborne, R. G. Clapp, and O. Pell, “Accelerating seismic computations on FPGAs–from the perspective of number representations,” in 70th EAGE Conference & Exhibition, 2008.

H. Jiang and V. Owall, “FPGA implementation of real-time image convolutions with three level of memory hierarchy,” in FieldProgrammable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on, Dec 2003, pp. 424–427.

H. Morishita, Y. Osana, N. Fujita, and H. Amano, “Exploiting memory hierarchy for a computational fluid dynamics accelerator on FPGAs,” in ICECE Technology, 2008. FPT 2008. International Conference on, Dec 2008, pp. 193–200.

J. Corbet, A. Rubini, and G. Kroah-Hartman, Linux Device Drivers, 3rd Edition. O’Reilly Media, Inc., 2005.

L. Duval and T. Rosten, “Filter bank decomposition of seismic data with application to compression and denoising,” SEG Annual International Meeting Soc. Expl. Geophysicists, pp. 2055–2058, 2000. [12] P. Aparna and S. David, “Adaptive Local Cosine transform for Seismic Image Compression,” 2006 International Conference on Advanced Computing and Communications, no. x, pp. 254–257, Dec. 2006.

M. Al-Moohimeed, “Towards an Efficient Compression Algorithm for Seismic Data,” Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific, pp. 550–553, Aug 2004.

O. Pell, R. G. Clapp et al., “Accelerating subsurface offset gathers for 3d seismic applications using FPGAs,” in 2007 SEG Annual Meeting. Society of Exploration Geophysicists, 2007. [7] A. Aqrawi and A. Elster, “Bandwidth Reduction through Multithreaded Compression of Seismic Images,” 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, pp. 1730–1739, May 2011.

W. Wu, Z. Yang, Q. Qin, and F. Hu, “Adaptive Seismic Data Compression Using Wavelet Packets,” 2006 IEEE International Symposium on Geoscience and Remote Sensing, no. 3, pp. 787–789, Jul. 2006.

W. Xi-zhen, T. Yun-tina, G. Meng-tan, and J. Hui, “Seismic data compression based on integer wavelet transform,” ACTA SEISMOLOGICA SINICA, vol. 17, no. 04, pp. 123–128, 2004.

Xilinx, “LogiCORE IP Endpoint Block Plus v1.14 for PCI Express,” 2010. [Online]. Available: http://www.xilinx.com/ [18] C. Angulo, C. Fajardo, O. Reyes, and J. Castillo, “FPGA implementation of a Huffman decoder for high speed seismic data decompression,” in 2014 Data Compression Conference, no. 0266. Salt Lake, United States.: IEEE Comput. Soc, 2014, p. 396.