Lossless Data Compression Programmable Hardware for High-speed Data Networks

José Luis Núñez, Simon Jones
Department of Electronic and Electrical Engineering
Loughborough University, Loughborough LE11 3TU, UK
j.l.nunez-yanez@lboro.ac.uk, s.r.jones@lboro.ac.uk

Abstract

This paper presents a high-performance application specific architecture for real time lossless data compression, which enables data throughputs over 1.5 Gbits/s compression and decompression using contemporary low-cost re-programmable FPGA technology. The inplementation is embedded into a PCI-based system and tested at speed using a PC as the host computer. A single FPGA is used to map all the functions in the system including the compression and decompressor cores, DMA logic, control logic and Master/Target PCI core. The independent compression and decompression channels enable a combined compression and decompression performance over 3 Gbits/s and robust self-checking hardware where each compress block can be automatically decompress to detect hardware failures or errors introduced by the communication channel.

Index Terms-- data compression, lossless circuits, field-programmable-gate-arrays.

1. Introduction

Lossless data compression is a key technology for enabling higher speeds and lower cost in data transmission and storage applications. It achieves this by reducing the number of bits that must be transmitted or stored while maintaining the exact information content of the data source. The significant requirements for bandwidth and capacity generated for data intensive applications have demanded storage/communication equipment that operate at speeds over the Gigabit/second range. Storage area networks (SAN) using fibre channel to communicate a set of high-capacity and high-speed disk arrays (RAID) in a server farm environment or a Gigabit Ethernet backbone connecting a group of Ethernet LAN's running at lower speeds are 2 examples where Terabytes of data must be stored and transmitted at high speeds. This paper presents a solution to the bandwidth problem by describing the X-MatchPRO hardware amenable algorithm and its hardware architecture that enable the usage of compression at over the 1 Gbit/second while maintaining good compression ratios.

The remainder of this paper is organized as follows: Section 2 introduces the X-MatchPRO algorithm. Section 3 depicts the PCI-based system architecture paying special attention to the compression/decompression cores. Section 4 describes the hardware implementation based on programmable technology. Section 5 compares its performance against other lossless data compressor devices. Finally, section 6 concludes this paper.

2. Algorithm Description

The X-MatchPRO algorithm uses a fixed-width dictionary of previously seen data and attempts to match or partially match the current data element with an entry in the dictionary. Each entry is 4 bytes (tuple) wide and several types of matches are possible where all or some of the bytes at different positions within the tuple match. Those bytes that do not match are transmitted as literals. This partial match concept gives the name to the procedure- the X referring to 'don't care'. At least 2 bytes have to match and when no valid match is generated a miss is codified adding a single bit set to 1 to the 4-byte tuple. The coding function for a match is required to code 4 separate fields as follows:

·A single bit set to 0 indicating a match.

·The match location. It uses the phased binary code (PBC) associated to the matching location. Phased binary coding [1] is a technique used to code the locations of a dictionary that starts empty and then grows, as new data is processed. The advantage is that a smaller dictionary uses fewer bits to code its positions so there is a compression gain during the growing stage.

·A match type. That indicates which bytes of the incoming tuple have matched. This is coded using a static Huffman code [2] based on the statistics obtained through extensive simulation.

·Any extra characters that did not match, transmitted in literal form.

The dictionary is maintained using a Move To Front (MTF) strategy whereby a new tuple is placed at the front of the dictionary while the rest move down one position. When the dictionary becomes full the tuple placed in the last position is discarded leaving space for a new one. X-MatchPRO reserves one location in the dictionary to code internal runs of full matches at location zero. Since the MTF strategy forces anything that repeats to be stored at location zero (top of dictionary), this Run-Length-Internal (RLI) technique is used to efficiently code any 32-bit repeating pattern. Additionally an Out of Date Adaptation (ODA) policy is used in X-MatchPRO for throughput purposes. This means that dictionary adaptation at time t+2 vector takes place using the adaptation vector generated at time t. This ODA technique does not affect compression because it is designed to maintain dictionary efficiency avoiding the danger of duplicating the same data tuple in several dictionary positions.

3. System Architecture

The system architecture based on the PCI bus is illustrated in Figure 2. The PCI core itself is obtained as a standard piece of IP from a third party vendor whilst the interface logic to the PCI core is designed to generate the appropriate signals to both pieces of IP when the device is mastering the PCI bus of being accessed through it. It includes master control logic, target control logic and DMA controller. The X-MatchPRO core has 2 independent compression and decompression engines that can work simultaneously in full-duplex mode. Each of these engines has 2 32-bit ports that are used to move data to and from the PCI interface logic. Each of these ports has its own buffering scheme formed by dual-port SRAM memory blocks plus control logic that enable a smooth flow of data during compression and decompression operations. A hand-shaking protocol formed by bus request, bus acknowledge and wait signals is used by each of these 4 ports to input or output data to the PCI core. Compression and decompression commands are issued through a common 32-bit control data port. A 4-bit address is used to access the internal registers that store the commands, information related to compressed and uncompressed block sizes and CRC (Cyclic Redundancy Check) codes to verify the compression operations. Each channel includes a CRC unit that calculates a 32-bit CRC code using all the data that accesses the compression engine or that leaves the decompression engine. A total of 10 registers form the register bank. 5 registers are used to control the compression channel and the other 5 for the decompression channel.

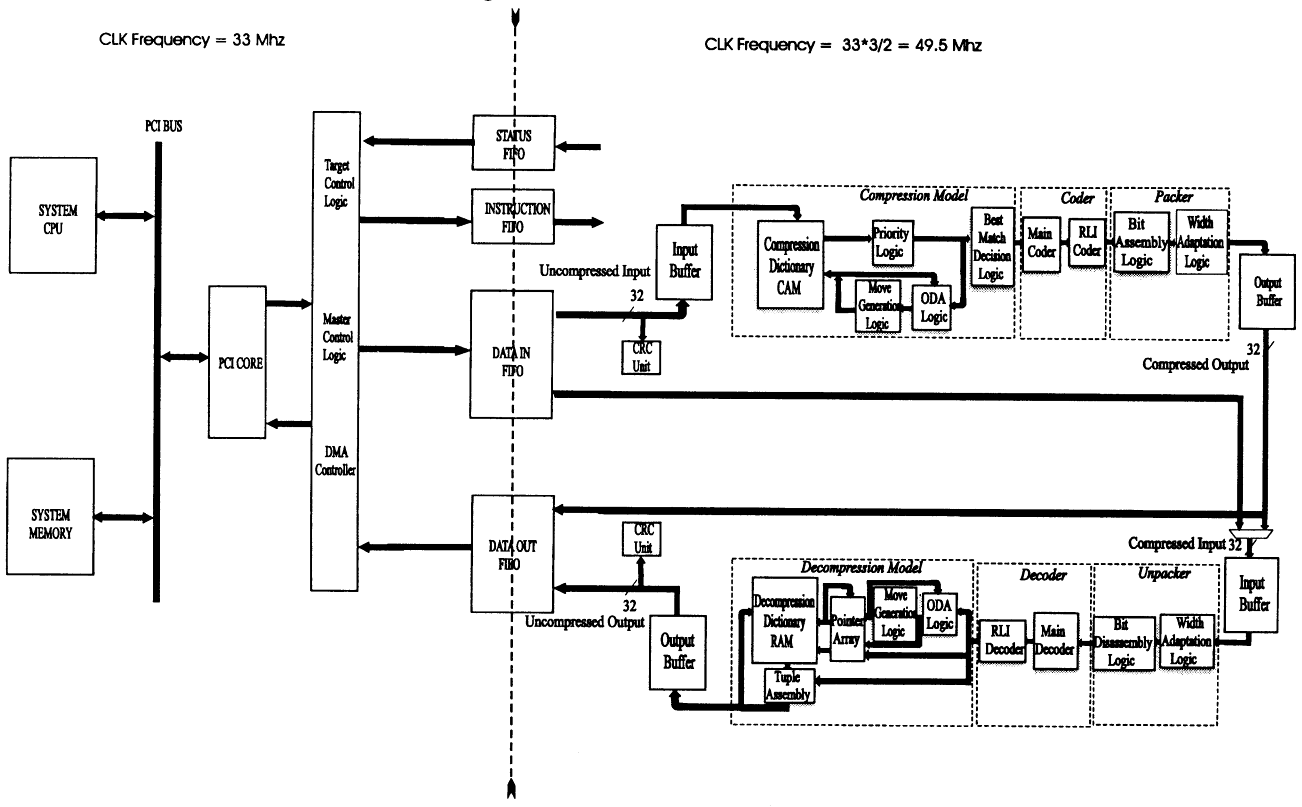

Figure 1. PCI-Based Hardware Architecture.

The first bit in the address line indicates if the read/write operation accesses compression or decompression registers. The device includes a test mode that simultaneously decompresses the block being compressed and reports any mismatches in the CRC codes using an interrupt. It is designed to compress any block size ranging from 8 bytes to 4 Gbytes. A decompression operation can be requested in the middle of a compression operation and vice versa.The architecture is based around a block of CAM to realize the dictionary. This is necessary since the search operation must be done in parallel in all the entries in the dictionary to allow high and data independent throughput. The length of the CAM varies with values ranging from 16 to 1024 tuples (4-byte locations) trading complexity for compression. Typically, the devices complexity increases by a factor of 1.5 each time the dictionary doubles. Dictionary size is variable to be able to adapt algorithm complexity to the resources available in the selected FPGA. The number of different locations present in the dictionary has an important effect on compression. In principle, the larger the dictionary the higher the probability of having a match and improving compression. On the other hand, our experimentation shows that a diminishing return rule applies and when a maximum size of around 1024 is reached further gains in compression are negible.

5. Hardware Implementation

The hardware implementation is based on the Altera APEX20KE PCI board that includes an APEX20K400E device as the user programmable FPGA. This device is used to implement the compression/decompression core, the PCI core plus DMA controller and control logic functions. The PCI core is available through the IP Altera Megastore. Table 1 shows a summary of the implementation details based on a dictionary of 16 locations. The system gates and typical gates figures are gate equivalent values available from the manufacture. Typically, 6 typical gates are equivalent to 1 ASIC gate. The complexity of X-MatchPRO is measured in logic elements as used by the FPGA. The figure in X-MatchPRO with a dictionary of 16 locations is approximately equivalent to 40K ASIC gates increasing to 600K ASIC gates for a 1024-location dictionary. The FPGA implements a 33-MHz PCI core so in order to be able to verify our compression/ decompression technology at maximum throughput we have included a PLL to generate a second clock at 49.5 MHz plus FIFO memories to communicate the 2 clock domains at 33 and 49.5 MHz.

Table 1. Technology Details

Technology Details									
Full-duplex X-MatchPRO					PCI Core/FIFO's/DMA controller				
Logic Units	% Logic Elements	Memory bits (K)	% Memory	PLL Clock (MHz)	Logic Units	%Logic Elements	Memory bits (k)	% Memory	Clock (MHz)
7948	48%	83	40%	49.5	3020	18%	124	58%	33

A 32-bit interface maps well with our compressor able to process 32-bit of data per clock cycle. Although the PCI bus becomes a bottleneck because not enough bandwidth is available to keep the core busy the core can run at its maximum throughput of 49.5 Mhz * 4 Bytes = 198 Mbytes/second by reading and writing data to the internal FIFO's and this setup achieves the objective of demonstrating the technology. We have used 2 BAR (Base Address Register) out of a maximum of 6 to map the registers of the design in the memory address space. Bar 0 maps the compressor and decompressor register file. Bar 1 is used to map the DMA registers. There are 2 registers that can be accessed in the DMA function and they are used to store the source address in main memory where the data block to be compressed or decompressed can be found and the destination address where the results of the operation should be written. These 2 registers are 32-bit wide so a full 4 Gbyte of memory space can be addressed. Additionally, a reset command has been mapped to address 0 in BAR 1 so writing to this location resets the DMA and control logic to a known initial state. The operational mode is simple: the host CPU starts writing the internal registers of the system with data indicating the memory address where the data block to be processed starts, its size in bytes and the memory address where the processed data block should be stored. Then it writes the instruction FIFO with a command indicating the operation to be executed compression or decompression plus some additional control information. After the command has been written the device takes control of the PCI bus in master mode and transfers data from main memory to the input FIFO, the core processes this data and the output is written in the output FIFO. Finally, the device writes the contents of the output FIFO to main memory. When this process terminates an interrupt is issued by the device so the host CPU can read the internal status registers using the status FIFO and verified that an abnormal termination has not taken place.

6. Performance Benchmarking

Figures 2 shows the compression performance obtained by the X-MatchPRO algorithm. The X axis varies the block size that is compressed independently resetting the dictionary between blocks. This means that no history information is kept from the compression of one block to be used in the compression of the next block. The Y axis measures the compression performance as a ratio of output bits to input bits so the smaller the value the better the compression. We have selected the 3 hardware amenable algorithms for the performance benchmarking. The ALDC (Adaptive Lossless Data Compression) [3] developed by IBM, the DCLZ (Data Compression Lempel-ziv) [4] developed by Hewlett-Packer and the LZS (Lempel-Ziv Stac) [5] developed by HiFn. These methods are representative of the fastest and better compressing technology available today. We have increased dictionary size to the maximum allowed in each algorithm to have a fair comparison of compression performance. This value is up to 2048 locations for the LZ algorithms and 1024 locations for X-MatchPRO.

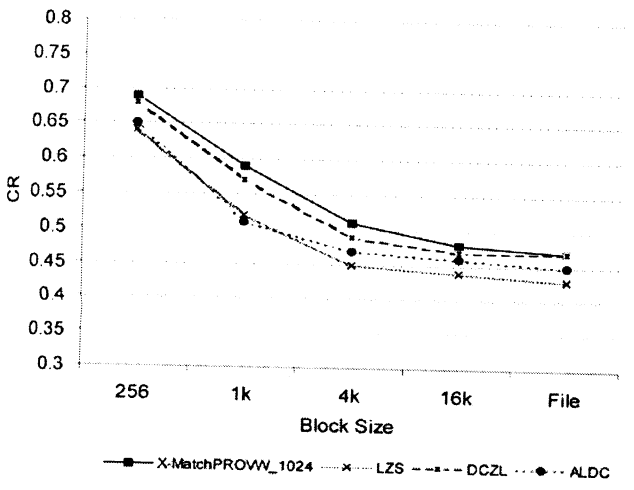

Figure 2. Canterbury data set

The Canterbury data set [6] includes representative data found in modern computer systems. It is biased to data textual in nature such as book chapters, poetry, C and Lisp source code or html web pages but it also includes application data, excel files and fax images. From the figure it can be observed that the performance of the 4 algorithms is similar with compression slightly better than 0.5. Compression improves with block size but it is also noticeable a saturation effect after a block size of 4 Kbytes

7. Conclusions

This paper has presented a new method of performing lossless compression based on parallel dictionary architectures where each dictionary entry stores a multiplicity of bytes instead of the classical approach of storing a single byte. The algorithm uses an array of techniques to compensate for the smaller number of matches that are generated in a wider dictionary and obtain competitive compression ratios. The hardware architecture has been demonstrated using an Altera PCI board that includes an APEX20KE FPGA device as its programmable logic. Our test methodology has replaced the classical test vector approach for self-checking hardware where each compression operation is deemed to be correct by simultaneously decompression of the data being compressed. This has been made possible thanks to the full-duplex architecture used in X-MatchPRO and the usage of CRC codes for automatic verification. The device has been successfully operated at 49.5 MHz generating a throughput of 1584 Mbits/second since up to 4-bytes can be processed per clock cycle. The full-duplex architecture operating in non-test mode allows simultaneous compression and decompression of data doubling the performance of the half-duplex device up to 3168 Mbits/second.

Acknowledgements: We acknowledge with gratitude the equipment donation obtained from Altera corporation for this project.

[1] T. C. Bell, J. G. Cleary and I. H. Witten, 'Text Compression', ,Prentice-Hall, NJ, 1990.

[2] D. A. Huffman, 'A Method for the Construction of Minimum Redundancy Codes', Proceedings of IRE, Vol. 40, pp. 1098-1101, 1951.

[3] J.M.Cheng and L.M.Duyanovich, 'Fast and Highly Reliable IBMLZ1 Compression Chip and Algorithm for Storage',Hot Chips VII Symposium, August 14-15, pp. 155-165, 1995.

[4] AHA3211 40 Mbytes/s DCLZ Data Compression Coprocessor IC', Product brief, Advanced Hardware Architectures Inc, 2635 Hopkins Court, Pullman, WA, 1997.

[5] '9600 Data Compression Processor', Data Sheet, Hi/fn Inc, 750 University Avenue, Los Gatos, CA, 1999.

[6] R. Arnold, T.Bell, 'A Corpus for the Evaluation of Lossless Compression Algorithms', Data Compression Conference, pp. 201-210, 1997.