computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

International Scientific Journal of Computing

ISSN 1727-6209

X-MATCHPRO: A HIGH PERFORMANCE FULL-DUPLEX LOSSLESS DATA COMPRESSOR ON A PROASIC FPGA.

Josй Luis Nъcez, Simon Jones, Stephen Bateman*

Electronic Systems Design Group, Loughborough University,
Loughborough, Leicestershire LE11 3TU. England.
*BridgeWave Communications, Inc.3350 Thomas Rd, Santa Clara, CA 95954, USA.
J.L.Nunez-Yanez@lboro.ac.uk, S.R.Jones@lboro.ac.uk, SBateman@Bridgewave.com

Abstract. *This paper presents the full-duplex architecture of the X-MatchPRO lossless data compressor and its highly integrated implementation in a non-volatile reprogrammable ProASIC FPGA. The X-MatchPRO architecture offers a data independent throughput of 100 Mbytes/s and simultaneous compression/decompression for a combine full-duplex performance of 200 Mbytes/s clocking at 25 MHz. Both compression and decompression channels fit into a single A500K130 ProASIC FPGA with a typical compression ratio that halves the original uncompressed data. This device is specifically targeted to enhance the performance of Gbit/s data networks and storage applications where it can double the performance of the original system.*

Keywords. *lossless, compression, network, storage, FPGA.*

1. INTRODUCTION

Lossless data compression [1], where the original data is reconstructed exactly after decompression is accepted as a tool that can bring important benefits to an electronic system. Its applications have been increasing over the past years thanks to the arrival of compression standards and a combination of pressure for more bandwidth and storage capacity while still reducing power consumption. Lossless data compression has been successfully applied to storage systems (tapes, hard disk drives, solid state storage, file servers) and communication networks (LAN, WAN, wireless).

The remainder of this paper is organized as follows: Section 2 establishes the motivation of our work. Section 3 describes the basic characteristics of the X-MatchPRO algorithm. Section 4 depicts the X-MatchPRO full-duplex architecture. Section 5 introduces our low-cost device verification methodology. Section 6 compares our device with other high-performance lossless data compressors. Finally section 7 concludes this paper.

2. MOTIVATION

The significant requirements for bandwidth and data capacity generated by applications such as real-time video conferencing, 3D animation modeling, Internet telephony, virtual reality, video on demand, etc have pushed forward networking and storage technology to operate at speeds in excess of 1 Gbit/s. Gigabit networking [2] has been made possible thanks to fiber optic signaling equipment able to transmit at a bandwidth of Gigabit/s over long distances with low error rates. Storage equipment has benefit from technology such as RAID [3] (Redundant Array of Inexpensive Disks) to achieve over 1 Gbit/s bandwidth performance.

Data compression technology is not currently being used to its full advantage in these applications due to performance limitations encountered in current data compression hardware. It has the potential of doubling the performance of a storage/communication system by increasing the available transmission bandwidth and data capacity with minimum investment.

3. THE X-MATCHPRO ALGORITHM.

The X-MatchPRO algorithm [4-6] uses a 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 wide and several types of matches are possible where all or some of the bytes at different positions inside the tuple match. Those bytes that do not match are transmitted literally. 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 to the literal. The dictionary is maintained using a move

to front (MTF) strategy [8] 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. This Run-Length-Internal (RLI) technique is used to efficiently code any 32-bit repeating pattern.

The coding function for a match is required to code several fields as follows:

A zero followed by:

If normal code:

1. The match location: It uses the binary code associated to the matching location. Since the dictionary has 16 entries 4 bits are used to code each location.

2. A match type: That indicates which bytes of the incoming tuple have matched. This is codified using a static Huffman code [7] based on the statistics obtained through extensive simulation.

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

If RLI code:

1. The RLI location: The last address in the dictionary is reserved to code RLI events.

2. A run length: 8 bits are used to indicate how many 32-bit repeating patterns have been observed. The maximum run length that it is possible to process in a single code is therefore 255.

The coding function for a miss has two fields as follows:

A one followed by:

1. The four bytes in literal form.

A data tuple (4 bytes) is added to the front of the dictionary while the rest move one position down if a full match has not occurred. The move-to-front technique is only applied when dealing with full matches. In this case the tuples from the first location until the location previous to the matching tuple move down one location, while the matching tuple is placed at the front of the dictionary. Additionally an Out-of-Date-Adaptation (ODA) policy is used in X-MatchPRO for throughput purposes. This means that adaptation at time t+2 takes place using the adaptation vector generated at time t.

4. THE X-MATCHPRO HARDWARE

The full-duplex architecture has 5 major components, namely: the Model, the Coder, the Decoder, the Packer and the Unpacker. The architecture is depicted in Figure 1.

The model is the section of the compressor whose function is to identify where the redundancy is located in a block of data and signal repetitive data sequences to the coder.

The model is composed of:

1. Compression dictionary: CAM-based [9] dictionary with 16 words. A bigger dictionary improves compression but increases complexity. The dictionary size is always one location smaller because one codeword is kept to signal RLI events.

2. Decompression dictionary: RAM-based dictionary that stores the history data during a decompression operation. The contents of the RAM dictionary during decompression must be same as the contents of the CAM dictionary during the compression in each cycle. Adaptation must take place

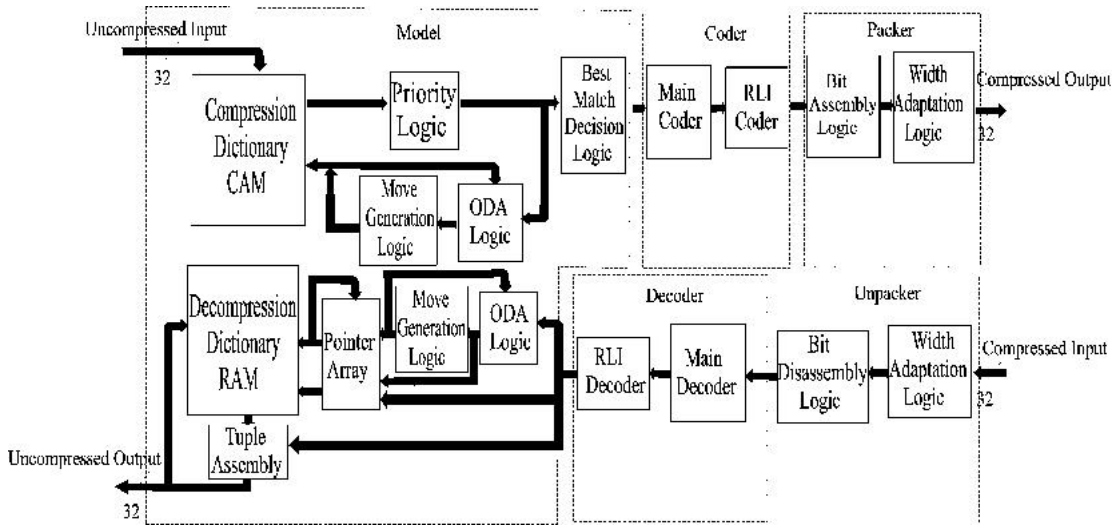

Fig. 1. ***X-MatchPRO full-duplex architecture.***

in exactly the same way to enable correct decompression of the compressed block.

3. Pointer Array. The pointer array logic performs an indirection function over the read and write addresses that accessed the RAM dictionary. It models the MTF maintenance policy of the CAM dictionary moving pointers instead of data. The pointer array enables mapping the CAM dictionary to RAM for decompression. Since RAM is plentiful in ProASIC FPGA's and the pointer array is much smaller than the CAM dictionary the savings in complexity allow having the full-duplex architecture in a single device.

4. Priority logic: Logic that assigns a different priority to each of the possible matches in the dictionary. Those matches that improve compression are assign a higher priority.

5. Best match decision logic: Logic that selects one of the matches as the best for compression using the priority information.

6. Tuple assembly: Module that assembles a decompressed tuple using dictionary information and any literal characters present in the code.

7. Move generation logic: Generation of the move vector depending of the match type and match location. The move vector adapts the CAM dictionary in compression and the pointer array in decompression.

8. ODA Logic: Out of Date Adaptation logic that forces the dictionary to adapt with previous match information and breaks in 2 the critical path to improve speed. The ODA logic and movement generation logic are replicated for the compression and decompression channels. They have exactly the same functionality.

The coder function is to use the information provided by the model to produce a minimum output of bits and obtain compression. It is composed of:

1. Main coder: Main X-MatchPRO coder assigns a uniform binary code to the matching location and static Huffman code to the match type and concatenates any necessary bytes in literal form.

2. RLI coder: RLI coder that detects the existence of runs of full matches at location zero. If a RLI becomes active the pipeline is empty from the previous code and the output of the chip is frozen while the run length is taken place.

The decoder function is to decode the compressed input stream and provided the model with a combination of dictionary address plus literal data so the model can reproduce the original uncompressed data. It is composed of:

1. RLI decoder: RLI decoder that when a RLI code is detected in the compressed input outputs match location zero and match type zero as many times as the number of repetitions indicated in the RLI code.

2. Main decoder: The match location and match types are decoded here together with any needed literal characters.

The packer function is to pack the variable length codewords output from the coder into fixed-length codewords of 32 bits. It is composed of:

1. Bit assembly: Logic that writes a new 64-bit compressed output to the output buffers whenever more than 64 bits of compressed data are valid in the internal buffer.

2. Width adaptation logic. This logic reads in 64-bit compressed words from the coder and writes out 32-bit compressed words to the compressed output bus. It performs a buffering function smoothing the data flow out of the chip to the compressed port.

The unpacker function is to break the fixed-length codewords input from the compressed bus into variable length codewords to be processed in the decoder. It is composed of:

1. Bit disassembly: Logic that reads a new 64-bit compressed vector from the internal buffer whenever less than 66 bits are left valid in the internal decompression register after a decompression operation.

2. Width adaptation logic: This logic performs the equivalent but opposite function as its counterpart in the compression channel. It reads in 32-bit of compressed data from the input compressed bus and it writes out 64-bit of compressed data to the bit disassembly logic when it requires more data. It performs a buffering function smoothing the data flow in the chip from the compressed port.

The initialization of the compression CAM sets all words to zero. This means that a possible input word formed by zeros will generate multiple full matches in different locations. The algorithm simply selects the full match closer to the top. This operational mode initializes the dictionary to a state where all the words with location address bigger than zero are declared invalid without the need for extra logic. The reason is that location x can never generate a match until the data contents of location x-1 are different from zero because locations closer to the top have higher priority generating matches. The MTF adaptation mechanism shifts down the dictionary when full matches are not detected and, therefore, ensures that the last word from this initial state to be deleted from the dictionary is always the word located at location zero at time zero. This operational mode in compression enables the decompression RAM to have only lo-

cation zero loaded with value zero during the initialization phase. This technique avoids having a long overhead equal to dictionary size cycles to initialize the RAM before each decompression operation.

The compression CAM dictionary can have the same location read and written simultaneously because it is legal to read a value at time t and replaced it with a new value for time t+1. Special attention has to be paid to avoid providing the same location for reading and writing to the decompression RAM. The decompression RAM reads its contents in asynchronous mode and writes its contents in synchronous mode to model the behavior of the CAM.

If the same address appears during cycle t for reading and writing it means that the algorithm needs to asynchronously read the contents of the RAM at time t and synchronously replaced their contents for time t+1. The RAM fails to work under these conditions. To avoid this situation special logic monitors the read and write addresses. If both addresses are the same the match type is used to determine which bytes are matching and missing in the 4-byte tuple. Those bytes that match are writing in location x a value that it is already present in location x. Therefore writing can be skipped. Those bytes that miss are reading a value from location x that can not be used to reconstruct the original tuple so reading is unnecessary and only writing must done in location x. This address adjustment technique avoids accessing the same location for simultaneous reading and writing in the decompression RAM but maintains the same operational mode as the compression CAM.

X-MatchPRO uses a simple coprocessor style interface to communicate with the rest of the system. Compression and decompression commands are issued through a common 16 bit control data port. A 3-bit address is used to access the internal registers that store the commands plus information related to compressed and uncompressed block sizes for reading or writing. A total of 6 registers form the register bank. 3 registers are used to control the compression channel and the other 3 for the decompression channel. The first bit in the address line indicates if the read/write operation accesses compression or decompression registers. The chip is designed to compress any block size ranging from 8 bytes to 32 Kbytes. A decompression operation can be requested in the middle of a compression operation and vice versa. The full-duplex architecture using a 16-word dictionary has been implemented in a A500K130 ProASIC FPGA [9].

5. TEST METHODOLOGY

Our functional test of the device, once post-layout back-annotation is completed successfully, uses a low cost PC-based test methodology and the JTAG port available in the FPGA. A text file is written automatically by a PERL script translating the original test vectors to the standard JAM [10] programming and test language. JAM is a vendor-and-platform-independent interpreted language for programming and testing devices via the IEEE standard 1149.1 TAP controller, commonly known as JTAG. This file contains the test vectors and JAM instructions ready to be executed by the Gatefield ProASIC JAM player [11] that controls the JTAG port shifting in the input test vectors clocking the device and shifting out the output test vectors. These vectors are compared with the expected output and fail or pass is reported. The same test vectors used during the simulation phased are now used in this verification phase to maintain consistency during the whole testing process.

Each original test vector is decomposed in two vectors corresponds to clock cycle low and the other to clock cycle high. After some propagation time the output of the circuit is ready to be strobed and scan out. This test allows us to verify the correct functionality of the silicon.

6. RESULTS.

Table 1 shows a comparison of the FPGA-Based X-MatchPRO implementation against several popular high-performance ASIC compressors. The selection includes:

1. The the ALDC1-40S [12] (IBM) and the AHA3521 [13] (AHA) that implement the ALDC [14] (Adaptive Lossless Data Compression) algorithm. This algorithm is a LZ1 derivative developed by IBM.

2. The AHA3211 [15] that implements the DCLZ [16] (Data Compression Lempel Ziv) algorithm. This algorithm is a LZ2 derivative developed by Hewlett/packer and AHA.

3. The Hi/fn 9600 [17] that implements the LZS [18] (Lempel-Ziv Stac) algorithm . This algorithm is another LZ1 derivative developed by STAC/Hifn.

Table 1 reports the complexity of the X-MatchPRO design in ProASIC tile's. Tile is the basic logic unit in the architecture of the ProASIC technology. Actel ProASIC tiles are simple blocks that can implement a logic function with 3 inputs and 1 output such as an AND gate or a flip-flop. Each tile can be configured to implement one of these simple functions using the internal non-volatile FLASH-based switches. Actel ProASIC archi-

Table 1. X-MatchPRO comparison.

DEVELOPERS		IBM	Advance Hardware Architectures (AHA)		STAC Electronics	System Design Group Loughborough University
CHIP*		ALDC1-40S	AHA3521	AHA3231	Hi/fn 9600	X-MatchPRO
TECHNOLOGY DETAILS	PROCESS	IBM CMOS 0.8 micron triple-level gate array/std cell	0.5 micron CMOS	0.5 micron CMOS	0.35 micron gate array/std cell	0.25 micron FLASH-CMOS FPGA Actel A500K ProASIC
	COMPLEXITY	70 Kgates	Not Stated	Not Stated	100 Kgates	9039 TILE's 70% of a A500K130-BG456
	CLOCK SPEED	40 MHZ	40 MHZ	40 MHZ	80 MHZ	25 MHz
THROUGHPUT		40 Mbytes/s	20 Mbytes/s	20 Mbytes/s	80 Mbytes/s	100 Mbytes/s
FULL-DUPLEX PERFORMANCE		N/A	N/A	N/A	160 Mbytes/s	200 Mbytes/s
ALGORITHM		ALDC	ALDC	DCZL	LZS	X-MatchPRO
EXTERNAL RAM REQUIRED		NO	NO	NO	NO	NO
COMPRESSION RATIO		0.44	0.44	0.52	0.44	0.58

tecture is fine-granularity and flat so the simple tiles are repeated across the device forming a matrix of identical logic elements. Dedicated memory blocks are group in the north side of the device. There are a total of 20 memory blocks in a A500K130 and each of them can implement 2304 bits of fully-synchronous dual port RAM. The design uses 70% of the device logic that is approximately equivalent to 30 Kgates and the 20 blocks of embedded RAM available (5 Kbytes). The total gate count equivalent of logic plus memory is 210 K gates.

Table 1 shows that X-MatchPRO can achieve higher performance throughput than the ASIC compressors with a lower clock ratio and this is due to its optimal parallel architecture. The compression ratio figure in the last row is a ratio of output bits to input bits (output_bits/input_bits) and it is based on a data set formed by 100 Mbytes of data found in the main memory of a UNIX workstation compressing data in 4 Kbytes blocks. The FPGA-based X-MatchPRO uses a very small dictionary of only 16 locations and that limits its compression performance. The ASIC compressors use dictionary sizes from 512 to 2048 positions.

7.CONCLUSIONS

X-MatchPRO offers unprecedented level of compression/decompression throughput in a FPGA implementation of a lossless data compression algorithm for general application. The full-duplex implementation effectively uses the resources available in the FPGA to simultaneously handle a compressed and uncompressed data stream. The use of a fine granularity device like the ProASIC where each block defines a very simple logic function, has proven to be well suited to implement the CAM-based dictionary that represents most of the logic present in the device. Other FPGA architectures where the building blocks implement mixed combinatorial and sequential functions offer poorer utilization ratios. The architecture is easily scalable so it can be adapted to new FPGA's with higher gate count with little effort and since the ProASIC architecture is ASIC-style the same RTL can be used to migrate towards ASIC's where higher throughputs should be obtained. As future work we are now focusing on improving compression ratios with the use of bigger dictionary sizes targeted to higher density FPGA devices.

Acknowledgements: We acknowledge with gratitude the support donated by Actel/Gatefield corporation.

REFERENCES.

[1] M. Nelson, 'The Data Compression Book', Prentice Hall, 1991.

[2] S. Djumin, 'Gigabit Networking: High Speed Routing and Switching', www2.cis.ohio-state.edu/~jain/cis788-97/gigabit_nets/index.htm , 1997.

[3] 'RAID Technology', White Paper, Dell Computer Corporation, March, 1999.

[4] M.Kjelso, M.Gooch, S.Jones, 'Design & Performance of a Main Memory Hardware Data Compressor', Proceedings 22nd EuroMicro Conference, pp. 423-430, September 1996, Prague, Czech Republic.

[5] J.Nucez, C. Feregrino, S.Bateman, S.Jones, 'The X-MatchLITE FPGA-based Data Compressor', Proceedings 25th EuroMicro Conference, pp126-133, September 1999.

[6] Josŭ Luis Nъcez,, Simon Jones,'The X-MatchPRO 100 Mbytes/second FPGA-Based Lossless Data Compressor', proceedings of Design, Automation and Test in Europe, DATE Conference 2000, pp. 139-142, March, 2000.

[7] J. L. Bentley, D. D. Sleator, R. E. Tarjan, V. K. Wei, 'A Locally Adaptive Data Compression Scheme', Communications of the ACM, Vol. 29, No. 4, pp. 320-330, April 1986.

[8] S.Jones, '100Mbit/s Adaptive Data Com-

pressor Design Using Selectively Shiftable Content-Addressable Memory', Proceedings of IEE (part G), vol.139, no.4, pp.498-502, 1992.

[9] 'ProASIC™ 500K Family', Data sheet, Actel corporation, 955 East Arques Avenue, Sunnyvale, CA, 2000.

[10] 'JAM Programming & Test Language Specification Version 2.0', Altera Corporation, Altera corporation, 101 Innovation Drive, San Jose, CA 1998.

[11] 'Using the Gatefield JAM Player', Gatefield corporation, 47436 Fremont Blvd, Fremont, CA, 1999.

[12] 'ALDC1-40S-M', Data sheet, IBM Microelectronics Division, 15080 Route 52, Bldg 504 Hopewell Junction, NY, 1994.

[13] 'AHA3521 40 Mbytes/s ALDC Data Compression Coprocessor IC', Product Brief, Advanced Hardware Architectures Inc, 2635 Hopkins Court, Pullman, WA, 1997.

[14] 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.

[15] 'AHA3211 20 Mbytes/s DCLZ Data Compression Coprocessor IC', Product Brief, Advanced Hardware Architectures Inc, 2635 Hopkins Court, Pullman, WA, 1997.

[16] 'Primer: Data Compression (DCLZ)', Application Note, Advanced Hardware Architectures Inc, 2635 Hopkins Court, Pullman, WA, 1996.

[17] '9600 Data Compression Processor', Data Sheet, Hi/fn Inc, 750 University Avenue, Los Gatos, CA, 1999.

[18] 'How LZS Data Compression Works', Application Note, Hi/fn Inc, 750 University Avenue, Los Gatos, CA, 1996.

Josŭ Luis Nъcez received the B.S degree in Industrial Engineering from Universidad de la Coruca (La Coruca, Spain) in 1995 and the M.S degree in Electronics Engineering from Universidad Politŭcnica de Cataluca (Barcelona, Spain) in 1997 designing neural networks for pattern recognition systems. He received his PhD degree at Loughborough University (Loughborough, England) in summer 2001 working in the area of hardware architectures for high-speed data compression. He is currently working as a Senior Research Fellow with special interest on the areas of data compression, computer architecture, VLSI design, FPGA-based design and high-speed data networks.

Simon Jones holds the ARM/Royal Academy of Engineering Research Chair in Embedded Microelectronic Systems at Loughborough University. He leads a large research group in application-specific processor design. He is Chairman of the IEEE for the UK and Ireland and Chairman of the IEE's Professional Network on System-on-Chip