The X-MatchPRO 100 Mbytes/second FPGA-Based Lossless Data Compressor.

José Luis Núñez, Simon Jones
Electronic Systems Design Group, Loughborough University
Loughborough, Leicestershire. LE11 3TU. England.
J.L.Nunez-Yanez@lboro.ac.uk, S.Jones@lboro.ac.uk

Abstract

This paper presents the X-MatchPRO lossless data compression algorithm and its hardware realization that enable data independent throughputs in excess of 100 Mbytes/second compression and decompression using contemporary FPGA technology. A comparison between this device and other commercially available data compressors is made in terms of technology, compression ratio and throughput. X-MatchPRO shows how state-of-the-art FPGA technology can be used to shorten the design cycle and achieve better performance than existing ASIC compressors.

1. Introduction

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

One of the common factors for successful integration of lossless data compression in these applications is a high throughput so the compression/decompression processes do not slow the original system down. We have been researching high performance lossless data compression hardware as the means to achieve the high throughput target [1]-[5].

The recent arrival of high density FPGA devices [6] has allowed the migration of the X-Match concept [4] to programmable technology with only minor degradations in compression. The new generation of low sub-micron ProASIC FPGA's [7] has paved the way for extensive algorithm and VHDL redesign resulting in the X-MatchPRO compressor. This chip boasts a data independent throughput of 100 Mbytes/s, matching the performance of the original X-Match ASIC softcore and outperforming other currently available lossless data compressors.

The remainder of this paper is organized as follows: Section 2 describes the basic characteristics of the X-MatchPRO algorithm. Section 3 introduces some of the features present in the X-MatchPRO hardware. Section 4 compares the X-MatchPRO compressor with other commercially available data compressor devices. Finally section 5 concludes this paper.

2. The X-MatchPRO algorithm.

The X-MatchPRO algorithm uses a dictionary of previously seen data and attempts to 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.

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

- The match location. It uses the binary code associated to the matching location. Since the dictionary has 64 entries 6 bits are used to code each location.
- A match type. That indicates which bytes of the incoming tuple have matched. This is coded using a static Huffman code [9] based on the statistics obtained through extensive simulation.
- Any extra characters that did not match, transmitted in literal form.

A data tuple (= 4 bytes) is always 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 down to the location previous to the matching tuple move down one position, while the matching tuple is placed at the front of the dictionary. The algorithm is given as pseudo-code in Figure 1.

```
Set the dictionary to its initial state;
DO
    { read in tuple T from the data stream;
      search the dictionary for tuple T;
      IF (full or partial hit)
            { determine the best match location
              ML and the match type MT;
              output '0';
              output Binary code for ML;
              output Huffman code for MT;
              output any required literal
              characters of T; }
      ELSE
            { output '1';
              output tuple T; }
      IF (full hit)
            {move dictionary entries 0 to ML-1 by
             one location;}
      ELSE
            { move all dictionary entries down by
              one location;}
      copy tuple T to dictionary location 0; }
WHILE (more data is to be compressed);.
```

Figure 1. The X-MatchPRO algorithm

3. The X-MatchPRO hardware

The architecture is based around a block of CAM to realize the dictionary [1], [4], [5]. This is necessary since the search operation must be done in parallel in all the entries in the dictionary to allow high throughput. The size of the CAM is 64 words with 4 bytes/word and it has to be selectively shiftable to be able to reorder itself adapting to the incoming stream of data. The selectively shiftable characteristic implies that each word of the CAM maintains its data or loads the data of the previous word depending on the value of its associated bit in the vector produced by the dictionary maintenance functions. Initially the CAM is set to an initial state with a pattern of common data making sure that no two positions contain the same data. This initial dictionary state is hardwire in the FPGA using the clear and set control signals available in the storage cells. The re-programability of the FPGA allows for tuning of the initial dictionary state depending of the type of data that is expected in the system improving compression. The chip supports both compression and decompression but not simultaneously. The reason is that the same dictionary is shared between the compression and decompression logic. This means that it is not possible to store two different history windows, one for each of the two processes. Since the CAM-Based dictionary uses most of the logic in the system the device has no internal resources to enable dictionary duplication. Special attention has been paid during the redesign process to balance the compression and decompression critical path delays present in the chip so they have a similar value. This way we have avoided overoptimizing some parts of the design and using valuable logic and routing resources in vain. A new pipeline scheme has been implemented to help to achieve this objective resulting in a 3 level pipeline for compression and 2 level pipeline for decompression. The critical path has a value of 39 ns. This figure is based on post-layout simulation results under worst operating conditions and worst process using a 0.25 um Flash-CMOS A500K270 ProASIC [7] device manufactured by Actel Corporation.

4. Performance comparison.

Table 1 shows a comparison among commercially available lossless data compression devices. Except from X-MatchPRO all the other devices are ASIC compressors.

Table 1 shows that X-MatchPRO offers a competitive level of compression when comparing with the other algorithms. Compression has been defined as the ratio output_bits/input_bits and the experiments are based on the data set introduced in [4] compressing 4Kbytes blocks of data. Our device clocks at a lower frequency but it offers more throughput because it uses its internal parallelism to process 4 bytes per clock cycle while the rest process one byte.

5. Conclusions

X-MatchPRO offers unprecedented level of compression/decompression throughput in a FPGA implementation of a lossless data

compression algorithm for general application. 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 around 70% 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 throughputs in excess of 2 Gbit/second should be obtained. As future work we are now focusing on setting up a ProASIC verification system that will allow us to test the device at real time. We are also studying the possibility of improving the compression ratio by adding run length coding techniques to the algorithm.

Developers		**IBM** [10]	**AHA** [11]		**STAC** [12]	**DCP** [13]	**SDG** [14]
Chip		ALDC1-40S	AHA3521	AHA3231	Hi/fn 9610	DCP816	X-MatchPRO
Technology	**Process**	IBM CMOS 0.8 micron gate array/std cell	0.5 micron CMOS	0.5 micron CMOS	0.5 micron CMOS gate array/std cell	1.0 micron CMOS gate array	0.25 micron FLASH-CMOS FPGA
	Complexity	Not Stated	Not Stated	Not Stated	100 Kgates	15 Kgates	61 Kgates
	Clock Frequency	40 MHz	40 MHz	40 MHz	50 MHz	40 MHz	25 MHz
Throughput		40 Mbytes/s	20 Mbytes/s	20 Mbytes/s	50 Mbytes/s	210 Kbytes/s	100 Mbytes/s
Algorithm		ALDC	ALDC	DCZL	LZS	GCA	X-MatchPRO
External RAM		NO	NO	NO	NO	YES	NO
Compression ratio		0.4425	0.4425	0.5177	0.4398	0.4701	0.53

Table 1. Comparison summary

References

[1] S.Jones, '100Mbit/s Adaptive Data Compressor Design Using Selectively Shiftable Content-Addressable Memory', Proceedings of IEE (part G), vol.139, no.4, pp.498-502, 1992.

[2] M. Kjelsø, M. Gooch, U. Simm, S. Jones, ' Hardware Data Compression and Memory Management for Flash-Memory Disks', Proceedings ISIC-95, 6th International Symposium on IC Technology, Systems and Applications, IEEE Press, pp 161-165.

[6] GateField Corporation., Fremont, CA. The GF250F ProASIC Products DATAbook, 1997

[7] Information available from http://www.actel.com/products/proasic/

[8] J.L.Bentley at all, 'A Locally Adaptive Data Compression Scheme', Communications of the ACM, vol.29, no.4, pp.320-330, 1986.

[9] D.Huffman, 'A method for the construction of Minimum Redundancy Codes', Proceedings of the I.R.E, pp.1098-1101, 1958.

[10] Information available from http:// www.chips.ibm.com/products/aldc

[3] J. Jiang, S.Jones, 'Parallel Design of Arithmetic Coding', Proceedings IEE, PartE, Vol 141, pp 327-333, November 1994.
[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.Nuñez, C. Feregrino, S.Bateman, S.Jones, 'The X-MatchLITE FPGA-based Data Compressor', Proceedings 25th EuroMicro Conference, pp126-133, September 1999, Milan, Italy.
[11] Information available from http:// www. aha.com/home.asp?file=product
[12] Information available from http:// www. hifn.com/
[13] Information available from http:// www. datacompression.com/
[14] Information available from http:// www. lboro.ac.uk/departments/el/research/sys/