Implementation of LZMA compression algorithm on FPGA

Bing Li, Lin Zhang, Zhuangzhuang Shang and Qian Dong

In the era of big data, compression techniques are needed to improve data processing bandwidth and data storage efficiency. Compared with software-based compression techniques, hardware-based compression techniques can improve speed and reduce power consumption. LZMA (Lempel–Ziv–Markov chain algorithm) is a lossless compression technology, and its hardware implementation has broad application prospects. A novel high-performance implementation of the LZMA compression algorithm capable of processing up to 125 Mbit/s on a Virtex-6 field programmable gate array (FPGA) chip is proposed. A typical application and its compression performance for a specific data sample are then presented.

Introduction: With rapid development of information and communication technology, the increasing quantity of data makes data processing more complex. Therefore, it is necessary to compress data effectively, so as to reduce data storage space and improve the use of limited communication bandwidth. Data compression can be divided into two main types. One is lossless compression, the other is lossy compression. Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data [1]. Many lossless compression algorithms have been applied to servers, workstations and other large data processing systems to achieve different goals.

LZMA is a lossless compression algorithm invented by Igor Pavlov in 1998, and it is the default algorithm in 7zip [2]. Fig. 1 shows the throughput test results of LZMA-SDK_4.26 [3] on a server; the server has an Intel Corei3-2100 central processing unit (CPU) at 3.10 GHz and 4 GB memory, and the average compression rate is only 10–20 Mbit/s. LZMA can provide a high compression ratio, but it consumes a large amount of random access memory (RAM) and CPU resources. As a result, it is difficult for a computer to do other operations while processing massive data.

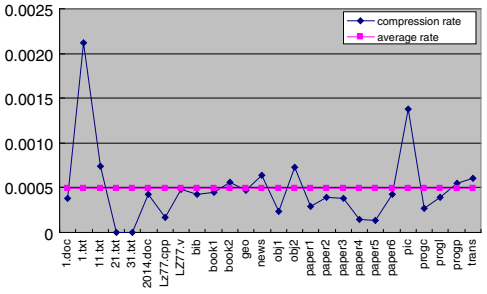

Fig. 1 *Throughput test results of LZMA*

Hardware design architecture: According to the compression process of LZMA, the design in this Letter is divided into three parts, as shown in Fig. 2. LZ77_COMPRESSOR compresses the input file with the LZ77 algorithm and outputs the stream into the cache of RANGE_ENCODER. RANGE_ENCODER further compresses data in the cache with established methods. Finally, SEND_OUT splices outputs into a more reasonable data format, so as to adapt to the high-speed bus outside of the design.

As shown in Fig. 3, LZ77_COMPRESSOR includes three further parts. DATA_RAM_ARRY is a temporary buffer to store the data that will be compressed. Data are read into the buffer by the ping-pong operation. When a data block is being compressed, another data block will be read into the other storage area, so it can ensure that there is no need for LZ77 to wait for data. Through this way, LZ77 can process the data without being interrupted. In our design, a signal 'DMA_in_start' is used to inform the external bus when the data could be read into DATA_RAM_ARRY. The encoded string's information is stored in HASH_RAM_ARRY. This Letter uses four cascade read-first RAMs to make up this module [4]. The four RAMs can be enabled or disabled by configuring the search-depth register. LZ77 is designed to generate the control signals associated with the two modules above. The data stream is compressed in accordance with the LZ77 algorithm in this module.

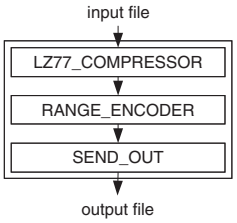

Fig. 2 *Structure of proposed design*

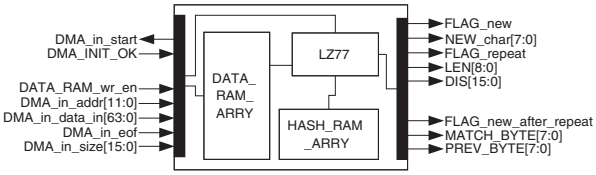

Fig. 3 *LZ77_COMPRESSOR*

RANGE_ENCODER is used to further compress the stream which is outputted by the LZ77_COMPRESSOR [5]. Fig. 4 is the block diagram of RANGE_ENCODER. The DIS_to_posSlot module is designed to transform DIS into posSlot [6]. The RANGE_RAM is used to store the encoded probability. RANGE_RAM is divided into several aspects as described in Table 1.

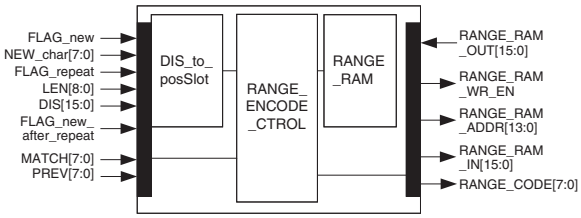

Fig. 4 *Structure of RANGE_ENCODER*

Table 1: Distribution of RANGE_RAM

Address	Name	Size
0x_xxxx_xxxx_xxxx	litprobs	6144
10_0000_xxxx_xxxx	isMatch	[0:10][0:3]
10_0001_xxxx_xxxx	isRep	[0:10]
10_0010_xxxx_xxxx	p_low	[0:127]
10_0011_xxxx_xxxx	p_mid	[0:127]
10_0100_xxxx_xxxx	p_high	[0:255]
10_0101_xxxx_xxxx	posSlotEncoder	[0:3][0:63]
10_0110_xxxx_xxxx	posEncoders	[0:113]
10_0111_xxxx_xxxx	posAlignEncoder	[0:15]
10_1000_0000_0000	choice1	–
10_1000_0000_0001	choice2	–

The output of LZMA is in the format of a byte. To suit an external bus, it is necessary to further process the data. As shown in Fig. 5, it is a controller that can group byte packets into 64 bits. Two buffers are input into this module in the same way as DATA_RAM_ARRY. When a compressed data block in a buffer can be read, the compressed stream from RANGE_ENCODER will be written into the other buffer. Only when it meets the transmission conditions, SEND_OUT will it output a signal 'DMA_out_start' to inform the external bus that the compressed data can be read out. In this way, it can improve the utilisation of the external bus without always occupying it. The transmission conditions and related operations are as follows: (a) a buffer is already full, output the signal 'DMA_out_start', meanwhile output the size of the compressed data block, and (b) the compression process is completed, output the signal 'DMA_out_start', output the size of the compressed data block and set the signal 'DMA_out_eof' to inform the external bus that the compression is finished.

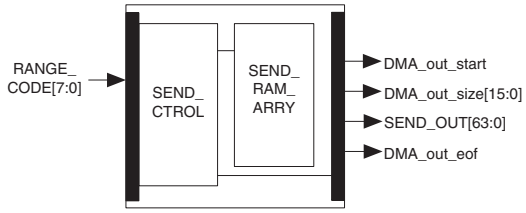

Fig. 5 *Structure of SEND_OUT*

As shown in Fig. 6, the three submodules above make up the hardware implementation of the LZMA algorithm; here it is called LZMA-IP. All the interface signals are marked in Fig. 6. The definition and description of the signals are listed in Table 2. All these modules are developed with Verilog. In this Letter, a Virtex-6 field programmable gate array (FPGA) ML605 [7] is chosen as the experimental platform. After being synthesised and routed, LZMA-IP (internet protocol) can run at the maximum frequency of 159 MHz.

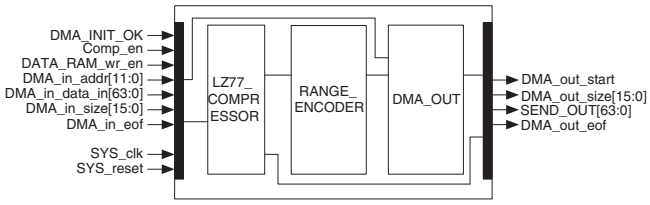

Fig. 6 *Structure of LZMA-IP*

Table 2: Input/output of LZMA-IP

Signal name	Width	Active	Description
SYS_clk	1	–	system clock
SYS_reset	1	positive	system reset
DMA_INIT_OK	1	positive	initialisation finish
Comp_en	1	high	compress enable
DMA_RAM_wr_en	1	high	input enable
DMA_in_addr	[11:0]	–	input address
DMA_in_data_in	[63:0]	–	input data
DMA_in_size	[15:0]	–	input size
DMA_in_eof	1	high	the last block
DMA_out_start	1	positive	output start
DMA_out_size	[15:0]	–	output size
SEND_OUT	[63:0]	–	output data
DMA_out_eof	1	high	compression finish

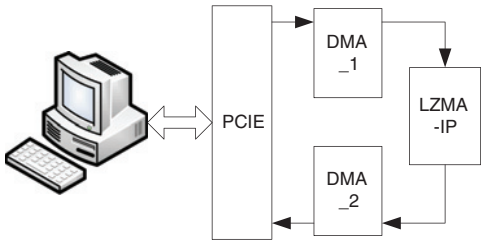

Fig. 7 *Typical application of LZMA-IP*

Typical application and results: A typical application of LZMA-IP is shown in Fig. 7. In the system, LZMA-IP is used as a co-processor. When there is a compression task, the personal computer (PC) transfers operation codes and data to LZMA-IP through PCIE (peripheral component internet express) and DMA_1. When the data are compressed, LZMA-IP writes back the compressed data into hard disk through DMA_2 and PCIE. During the compression process, PC is only required to specify the configurations of the source file and target file at the beginning, so the compression task will not take up too many CPU resources [8]. In this system, the data bus is requested when necessary, so the compression process will not affect other applications.

To evaluate the performance of LZMA-IP, a PCIE bus and two DMAs are integrated with it .The working frequency of the system is set at 125 MHz. In the experiment, two data sets, standard test files from Calgary Corpus and some other common text files, are used. Table 3 compares a software implementation running on an Intel Corei3-2100 CPU at 3.10 GHz with 4 GB memory with the hardware implementation. The results show that the hardware implementation is about 10 times faster than the software implementation with the same compression ratio.

Table 3: Performance evaluation of LZMA-IP

File name	Throughput (Mbit/s)		File name	Throughput (Mbit/s)	
	Soft	Hard		Soft	Hard
1.doc	9.548	101.424	obj1	5.728	57.348
1.txt	52.501	601.578	obj2	17.955	74.424
11.txt	18.426	604.513	paper1	7.266	63.054
21.txt	0	329.628	paper2	9.597	58.213
31.txt	0	21.333	paper3	9.523	56.130
4.doc	10.614	52.906	paper4	3.620	57.273
Lz77.c	4.166	79.061	paper5	3.273	59.550
LZ77.v	11.978	169.536	paper6	10.440	64.403
bib	10.440	66.123	pic	34.224	170.327
book1	10.986	51.274	progc	6.572	70.932
book2	13.962	60.465	progl	9.845	92.035
geo	11.705	37.720	progp	13.689	101.155
news	15.872	59.873	trans	14.979	92.456
Average				12.189	125.105

Conclusion: In this Letter, a hardware implementation of the LZMA algorithm is presented. By using the dual-port RAM reasonable and pipeline state machine, the hardware implementation of the LZMA is nearly 10 times faster than the software-based algorithm. Moreover the compressed file can be decompressed by 7zip. Most importantly, the design proposed in this Letter can be easily integrated into other systems. Although all the experiments in this Letter are implemented on an FPGA, there are still many differences with ASIC (application specific integrated circuit). The next step will be to select an appropriate technology library to implement our design. The proposed RANGE_ENCODER only encodes the stream bit-by-bit, encoding efficiency is relatively low, so the performance of the LZ77_COMPRESSOR is limited, so we will further enhance the performance of the RANGE_ENCODER [9], so as to obtain higher throughput.

12 May 2014
doi: 10.1049/el.2014.1734
One or more of the Figures in this Letter are available in colour online.

Bing Li, Lin Zhang, Zhuangzhuang Shang and Qian Dong (*School of Integrated Circuits, Southeast University, Nanjing, People's Republic of China*)

E-mail: 807322637@qq.com

Bing Li: Also with Advanced Cloud-Systems Research Center, Southeast University, Nanjing, People's Republic of China

References

1 Salomon, D.: 'Data compression: the complete reference [M]' (Springer-Verlag London Limited, London, UK, 2007, 4th edn)
2 LZMA SDK [Online]. Available at http://www.7zip.org/sdk.html
3 Pavlov, I.: '7z format', [Online] Available at http://www.7-zip.org/7z.html
4 Sheng, S.: 'A research and implementation of deflate compression algorithm on FPGA [D]' (Guilin University of technology, Guilin, 2010) (in Chinese)
5 Nigel, G., and Martin, N.: 'Range encoding: an algorithm for removing redundancy from a digitized message [J]'. Video & Data Recording Conf., Southampton, UK, 24–27 July 1979
6 Pavlov, I.: '7z Format [S]'. Retrieved 2013-06-16, 2013, [Online] Available at http://www.7-zip.org/7z.html
7 (2011, Dec) Xilinx FPGAs, [Online] Available at http://www.xilinx.com/products/silicon-devices/fpga/index.htm
8 Hardware Implementation of LZMA Data Compression Algorithm [J], International Journal of Applied Information Systems (IJAIS)–ISSN: 2249-0868,Volume 5– No.4, March 2013
9 Shcherbakov, I., Weis, C., and When, N.: 'A parallel adaptive range coding compressor: algorithm, FPGA prototype, evaluation [J]'. Data Compression Conf. (DCC), April 2012, pp. 119–128, Snowbird, UT, USA