2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines

Online Bandwidth Reduction Using Dynamic Partial Reconfiguration

Seyyed Mahdi Najmabadi, Zhe Wang, Yousef Baroud, Sven Simon
Department of Parallel Systems
University of Stuttgart
70569 Stuttgart, Germany
Email: mahdi.najmabadi@ipvs.uni-stuttgart.de

***Abstract*—Online compression of I/O-data streams in Custom Computing Machines will enhance the effective network bandwidth of computing systems as well as storage bandwidth and capacity.**

In this paper a self-adaptive dynamic partial reconfigurable architecture for online compression is proposed. The proposed architecture will bring new possibilities in online compression due to its adaptivity to dynamic environments. In this study, network traffic, and input data distribution are considered as two dynamic behaviors. The degree of improvement provided by the architecture depends on data distribution, bandwidth, and available resources. Our analysis shows an improvement of up to 20% in compression ratios in comparison to non-adaptive approaches.

I. Introduction

I/O bandwidth is one of the major performance bottlenecks in general purpose computing systems that reduces the efficiency and scaling of such systems. Data compression is one technique that can be used to increase I/O performance and minimize storage utilization.

Abali et al. proposed data compression to reduce storage requirements, bus bandwidth, and energy consumption by using a hardware compressor [1]. Finally, partial dynamic reconfiguration technology dramatically increases the functionality of embedded systems and creates an opportunity for intelligent resource management.

Online compression faces several design challenges due to various dynamic parameters. For example, the performance of the compression is dependent on the available computation resources, compression techniques, data characteristics and available bandwidths. Such challenges can potentially be successfully addressed by an adaptive design of the encoder. However, state-of-the-art hardware-based online compression architectures are not adaptive and are designed based on the worst case scenario. In order to be able to compress online, the architecture must handle the maximum possible input data rate (D_{max}). Under this constraint, the compression algorithm that can process the highest data rate is selected. However, the input data rate usually varies over time, for various reasons, such as variable network traffic. As it is shown in the results section, the drawback of the traditional non-adaptive approach is that compression efficiency is degraded when the input data rate is less than D_{max}. To tackle this problem, we introduce a self-adaptive dynamic partial reconfigurable architecture. In this paper, dynamic adaptation is based on data characteristics and network throughput that influence the input data rate.

II. Related Work

In recent years there have been several of adaptive software-based designs, e.g. [2]–[5]. Hovestadt et al. proposed an algorithm that chose the compression method most suitable for the current execution environment [2]. The compression methods are qualitatively ranked through profiling and real-time system resource status monitoring. The compression level is selected by a decision model that constantly estimates the performance gain based on system metrics like the current CPU load, available I/O bandwidth, or the compressibility of the data [2].

Several adaptive compression algorithms have been proposed that target software-based implementation. However, the increasing growth of embedded applications has created a demand for high-performance online hardware-based compression. It has been shown that FPGA-based compressor implementation performs better than CPU implementation [6]. Additionally, using a hardware compressor allows running high level tasks e.g simulation on other platform in parallel with the compression.

Recently, there have been several hardware-based compressors, e.g. [7]–[9]. Shcherbakov et al. proposed a high-performance implementation of the LZSS compression algorithm capable of processing up to 50 MB/s [7]. They have shown that there are various trade-offs between compression efficiency, resources, and throughput. For example, by increasing the windows size and hash bit count, which leads to more resource usage, the compression ratio will be increased. It is mentioned that one way of improving the compression ratio is adjusting the algorithm parameters (i.e. amount of matching attempts before giving up). This can improve the compression ratio by 20% at a cost of an 82% performance decrease.

III. Proposed Architecture

Figure 1 shows the high-level system diagram of self-adaptive online compression that consists of dynamic and static regions. The *Entropy encoder*, the *entropy decoder*, the *encoding table* and the *decoding table* are located in the dynamic regions and the other units are located in the static regions. The *Ethernet controller* is responsible for network

978-1-5090-2356-1/16 $31.00 © 2016 IEEE
DOI 10.1109/FCCM.2016.49

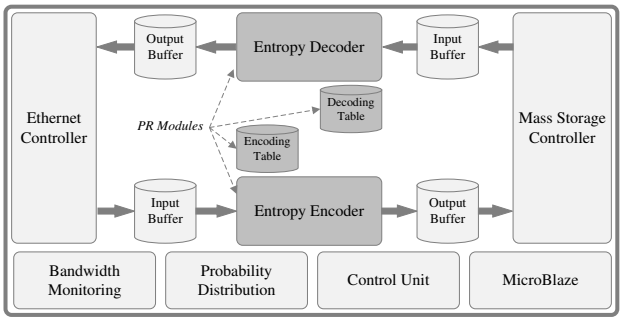

Fig. 1. High-level system diagram of the proposed hardware architecture for self-adaptive online compression

communication, and transfers data to/from buffers. The *SSD controller* is responsible for transferring data to/from SSD. The *Bandwidth monitoring* unit determines the actual data rate of the input data stream. The *Probability distribution* unit determines the probability distribution of the input data stream. The *Entropy encoder* unit runs a lossless data compression algorithm. In the proposed architecture the algorithm might change during the run time. This decision is made by the *control* unit. The *Entropy decoder* module runs a decompression algorithm, and the decompression algorithm is selected based on the entropy encoder algorithm that earlier encoded the data. The *Encoding table* and the *decoding table* are used for the algorithms that require a table for the encoding/decoding process. The *encoding table* is reconfigured during run time based on the probability distribution of the data stream.

Figure 2(a) shows the main components of the online compression architecture. The encoder and decoder are dynamic and may change during run time but the other components are static. The decoder is changed based on the encoder that encoded the data stream, and the encoder is dependent on factors such as network throughput, encoder input throughput, encoder compression ratio and decoder throughput. Each component has several attributes that are shown in the Figure 2(a). The component's attributes are highly dependent on each other. Figure 2(b) shows the dependency between attributes via a directed graph. For instance, the encoder reconfiguration time depends on the amount of the resources, the mass storage throughput depends on the encoder output throughput, decoder input throughput, data source data rate, and network channel throughput. We have used these dependencies and proposed a self-adaptive dynamic partial architecture.

The architecture of the *control* unit is shown in Figure 3. The *control* unit receives all required variables from the other units. The attributes of each entropy encoder are saved in to the *Encoder Attribute Table* before the run time. Control unit selects the proper entropy encoder based on the several priority levels defined by the user. The priority levels may be changed during the run time according to the user requirements. Table I shows two examples of the distribution of priority levels. When there are multiple encoders at the same level of priority, the decision is made according to the next priority level. If

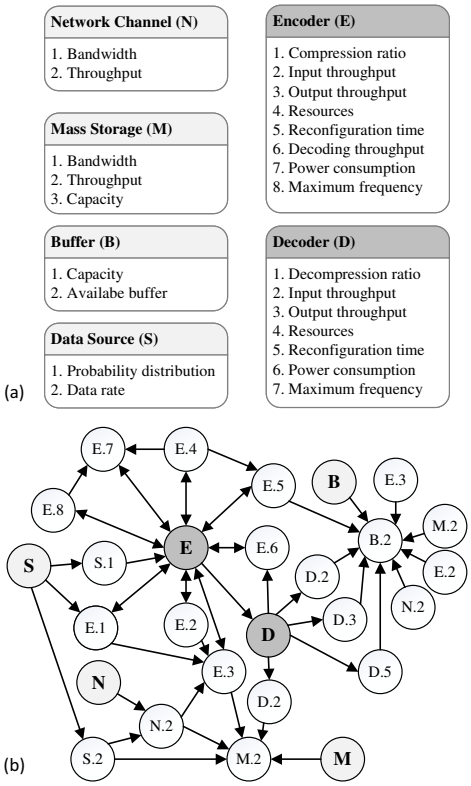

Fig. 2. (a) Main system components and their attributes. The encoder and decoder are dynamic components. (b) Representation of the dependency between the components and the attributes.

reconfiguration is required, the maximum required buffer will be calculated and will be compared with the available input buffer. If there is not enough space, the *empty buffer* signal will be asserted, and the data inside the input buffer will be transferred with maximum speed to the output stream without being compressed. Due to the dynamic partial reconfiguration capability this transfer is done in parallel with the reconfiguration process. During the reconfiguration process the *PLL* unit is also adjusted to the maximum operating frequency mentioned in the *Encoder Attribute Table*.

TABLE I
TWO EXAMPLES OF USER DEFINED PRIORITY TABLE

Priority	Example 1	Example 2
P. L1	Encoder input throughput	Decoder output throughput
P. L2	Encoder compression ratio	Encoder input throughput
P. L3	Decoder output throughput	Encoder compression ratio
P. L4	Encoder reconfiguration time	Encoder Power consumption
P. L5	Encoder Power consumption	Encoder Reconfiguration time

IV. Results and Discussion

In this paper, we focus on compressing scientific simulation data. These scientific simulation data are generated on a simulation platform and transmitted to an FPGA. The data are compressed and stored to a storage unit. simultaneously,

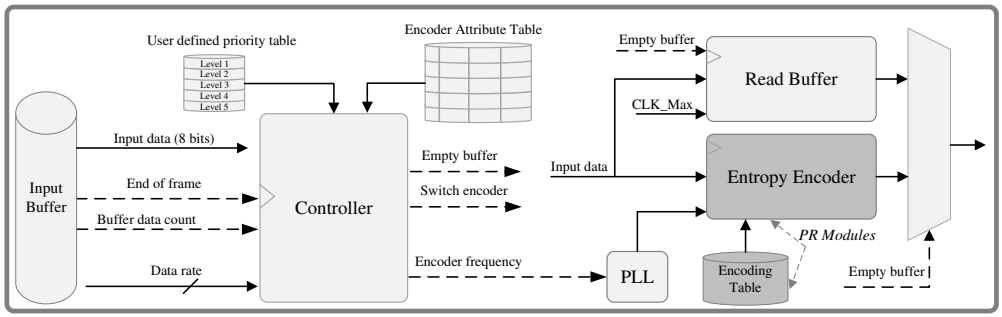

Fig. 3. Hardware architecture of the proposed controller and adaptive entropy encoder

a user might ask for specific simulation data to be analyzed or visualized at the post-processing platform. In such a case, the data are decompressed and sent to the post-processing platform. In this application scenario the network throughput may vary because of two main reasons: (1) variable network bandwidth, and (2) variable size of generated data by the simulator.

In order to evaluate the performance and behavior of the proposed architecture, four entropy encoders, namely: Huffman, Golomb, Binary Arithmetic coder (BAC), and tabled Asymmetric Numeral System (tANS) [10] are selected. All the entropy encoders are synthesized individually for Xilinx Virtex 6 FPGA. Table II shows the required resources for each entropy encoder. The bitstream size and reconfiguration time are estimated based on the resource utilization, and according to the cost models proposed by [11], [12]. It is assumed that the bitstreams are stored in DDR, and the memory bus clock rate is 200 MHz. The user defined priority table is the same as Example 1 in Table I.

TABLE II
COMPARATIVE RESULTS IN RESOURCE UTILIZATION.

	#Slices	#Flip-Flops	#BRAMs	Max Freq. (MHz)
BAC [13]	251	177	0	400
tANS [14]	210	208	3	125
Golomb	233	243	0	150
Huffman	152	171	1	500

The synthesized reports also indicate that each of the entropy encoders operates at different maximum frequencies and therefore provides different throughputs. Table III shows the *Encoder Attribute Table* which stores the parameter of each entropy encoder. It can be seen that the Huffman encoder has the highest throughput and BAC has the lowest throughput among the other entropy encoders. The Huffman and tANS entropy encoders have a higher reconfiguration time due to the additional block RAMs.

In order to compare the compression ratio of the four entropy encoders, three data sets were applied. The first data set is the prediction residual of a large-scale molecular dynamic (MD) simulation. The second data set is the prediction residual of a Computational Fluid Dynamic (CFD) simulation, and the third data set called Normal consists of 256 symbols

TABLE III
ENCODER ATTRIBUTE TABLE

Index	Type	Throughput (Mbps)	Reconfiguration Time (ms)	Max Freq. (MHz)
1	BAC	400	1	400
2	tANS	1000	5	125
3	Golomb	1200	1	150
4	Huffman	4000	5	500

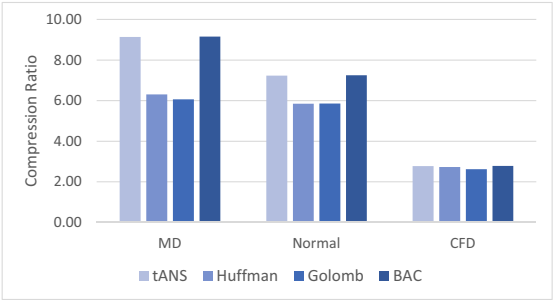

Fig. 4. Compression ratio comparison

and was generated randomly under the condition that the probability of the occurrence of one symbol is more than 50%. Figure 4 demonstrates that when the probability of occurrence of symbols is skewed, tANS and BAC provide a higher compression ratio than Huffman and Golomb. If the probability is normally distributed and the probability of all symbols are less than 50%, i.e., the CFD data set, all the entropy encoders provide the same compression ratio.

Figure 5 shows the compression ratio comparison between the self-adaptive dynamic partial reconfiguration (SADPR) approach and a static approach at different network throughputs for the MD and Normal data sets. It can be seen that when the input data rate is less than 400 Mbps the Binary Arithmetic Coder is selected and the compression ratio is up to 20% higher than Huffman coding. In the case of the static approach, the Huffman encoder, which is the entropy encoder with the highest throughput, is considered as a fixed entropy encoder. It's worth mentioning that a similar performance can be achieved by implementing all entropy encoders statically.

Although with this approach no reconfiguration is required, the main drawback is that almost four times the resources are required.

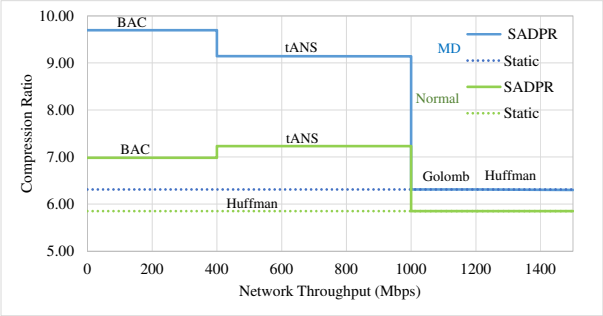

Fig. 5. Compression ratio comparison between the self-adaptive dynamic partial reconfiguration (SADPR) approach and the static approach at different network throughput.

Figure 6(a) shows one example of variable network throughput that is generated through an analysis of TeraByte-scale event datasets for particle physics. [15]. Accordingly, Figure 6(b) shows the adaptation of the proposed architecture to the variable data rate.

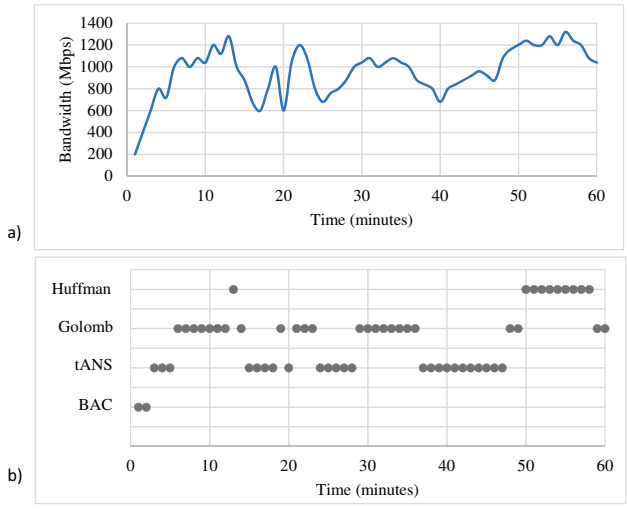

Fig. 6. (a) The input data rate that is varying over time [15]. The throughput is scaled to fit to this work. (b) The selected entropy encoders based on the input data rate over time.

V. Conclusion

In this paper we have presented a self-adaptive hardware architecture for online data stream compression in combination with dynamic partial reconfiguration. The proposed architecture improves the compression efficiency of data streams that are transferred through network or data streams, which have dynamic data rates. This improvement is achieved by adapting the compression algorithm to the input data rate. Additionally, the encoding table for the table-based algorithm, i.e. Huffman, is reconfigured after each frame based on the data distribution of the previous frame. The proposed architecture was evaluated with four different lossless compression algorithms and scientific data sets. Our analysis shows an improvement of up to 20% in the compression ratio in comparison to static approaches. This improvement is highly dependent on the data distribution and data rate of the input data stream.

References

[1] B. Abali, M. Banikazemi, X. Shen, H. Franke, D. E. Poff, and T. B. Smith, "Hardware compressed main memory: Operating system support and performance evaluation," *IEEE Trans. Computers*, vol. 50, no. 11, pp. 1219–1233, 2001.

[2] M. Hovestadt, O. Kao, A. Kliem, and D. Warneke, "Evaluating adaptive compression to mitigate the effects of shared i/o in clouds," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, May 2011, pp. 1042–1051.

[3] C. Pu and L. Singaravelu, "Fine-grain adaptive compression in dynamically variable networks," in *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, June 2005, pp. 685–694.

[4] B. Gedik, "Discriminative fine-grained mixing for adaptive compression of data streams," *Computers, IEEE Transactions on*, vol. 63, no. 9, pp. 2228–2244, Sept 2014.

[5] H. Zou, Y. Yu, W. Tang, and H. M. Chen, "Improving i/o performance with adaptive data compression for big data applications," in *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, ser. IPDPSW '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1228–1237.

[6] M. S. Abdelfattah, A. Hagiescu, and D. Singh, "Gzip on a chip: High performance lossless data compression on fpgas using opencl," in *Proceedings of the International Workshop on OpenCL 2013 & 2014*, ser. IWOCL '14. New York, NY, USA: ACM, 2014, pp. 4:1–4:9.

[7] I. Shcherbakov, C. Weis, and N. Wehn, "A high-performance fpga-based implementation of the lzss compression algorithm," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 449–453.

[8] T. Ueno, R. Ito, K. Sano, and S. Yamamoto, "Bandwidth compression of multiple numerical data streams for high performance custom computing," in *Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference on*, June 2014, pp. 190–191.

[9] S. Lee, J. Park, K. Fleming, Arvind, and J. Kim, "Improving performance and lifetime of solid-state drives using hardware-accelerated compression," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 4, pp. 1732–1739, November 2011.

[10] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, "The use of asymmetric numeral systems as an accurate replacement for huffman coding," *31st Picture Coding Symposium*, May 2015.

[11] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in fpga systems: A survey and a cost model," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 36:1–36:24, Dec. 2011.

[12] A. Morales-Villanueva and A. Gordon-Ross, "Partial region and bitstream cost models for hardware multitasking on partially reconfigurable fpgas," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, May 2015, pp. 90–96.

[13] L. Horrigue, T. Saidani, R. Ghodhbani, and M. Atri, "An efficient, high speed architecture for jpeg2000 mq-coder," in *Image Processing, Applications and Systems Conference (IPAS), 2014 First International*, Nov 2014, pp. 1–6.

[14] S. Najmabadi, Z. Wang, Y. Baroud, and S. Simon, "High throughput hardware architectures for asymmetric numeral systems entropy coding," in *Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on*, Sept 2015, pp. 256–259.

[15] J. J. Bunn, H. B. Newman, S. McKee, D. G. Foster, R. Cavanaugh, and R. Hughes-Jones, "Bandwidth challenge - high speed data gathering, distribution and analysis for physics discoveries at the large hadron collider," in *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing, November 11-17, 2006, Tampa, FL, USA*, 2006, p. 241.