Architecture for Simultaneous Coding and Encryption Using Chaotic Maps

Amit Pande
Department of Computer Science
University of California
Davis, CA, USA
Email: amit@cs.ucdavis.edu

Joseph Zambreno
Electrical and Computer Engineering
Iowa State University
Ames, IA, USA
Email: zambreno@iastate.edu

Prasant Mohapatra
Department of Computer Science
University of California
Davis, CA, USA
Email: prasant@cs.ucdavis.edu

***Abstract*—In this work, we discuss an interpretation of arithmetic coding using chaotic maps. We present a hardware implementation using 64 bit fixed point arithmetic on Virtex-6 FPGA (with and without using DSP slices). The encoder resources are slightly higher than a traditional AC encoder, but there are savings in decoder performance. The architectures achieve clock frequency of 400-500 MHz on Virtex-6 xc6vlx75 device.**

I. Introduction

Arithmetic coding is a data compression technique that encodes data by creating a code string which represents a fractional value on the interval [0, 1). When a string is compressed using arithmetic coder, frequently-used characters are stored with fewer bits and not-so-frequently occurring characters are stored with more bits, resulting in fewer bits used in total [1]

This paper discusses arithmetic coding from a slightly different perspective. Recent work has established how arithmetic coding can be viewed as an iteration on piece-wise linear chaotic maps [2], [3]. Further, many researchers have studied the use of arithmetic coding for joint encryption and compression [4], [5], [6]. In this paper, we extend this discussion to hardware community - to study the hardware optimizations in design of such schemes. Particularly, we study the implementation of arithmetic coding using piece-wise chaotic maps [2], [3]. As we shall study, this implementation has lower decoder requirements than the commercial implementations. The reduced decoding efficiency of arithmetic coding allows it to trend towards the low computational complexity of Huffman coders, allowing BAC to enter embedded systems market. The aspects of context-modeling and adaptation and renormalization, as done in CABAC coder are beyond the scope of this work, where we focus on architectural optimizations on encoder and decoder processes. It allows simultaneous encryption of multimedia content without computational overheard.

This research is supported by the National Science Foundation under Grant #1019343 to the Computing Research Association for the CIFellows Project.

Scope of the work

In the regular coding mode, prior to the actual arithmetic coding process the given binary data enters the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements. Then, after the assignment of a context model, the bin value along with its associated model is passed to the regular coding engine, where the final stage of arithmetic encoding together with a subsequent model updating takes place. We shall restrict the focus of further discussions on the final arithmetic encoding (and decoding) stages of CABAC coder.

Arithmetic coding based encryption schemes have been proposed in research literature for joint compression and encryption purposes [6], [**?**]. It would be interesting to integrate both coding and encryption using chaotic maps at a computational complexity lower than existing implementations. This motivates the need of coding and encryption architecture using chaotic maps. A description of equivalence between binary arithmetic coding and chaotic maps is given in earlier works [3], [6].

II. Hardware architecture

In this section, we discuss the hardware architecture for arithmetic coding using chaotic maps, and N-ary chaotic arithmetic encryption.

The chaotic encoder operation inverse inverse mapping of interval [0,1) on the chaotic map according to input symbol. For binary arithmetic coder, we have a fixed map to be iterated in each cycle.

First, the control unit receives the input bit stream, which is passed on to the chaotic map Iterator (CMI). The control unit passes the bitstream, one symbol per cycle (unless in the case of multiple symbol encoding, which will be discussed later). For encoding, the initial interval passed to CMI is [0,1), which is transmitted as the beginning (B_n) and end (E_n) interval values. Both the intervals are then iterated over CMI (using two instances of CMI), and then the output is sorted so that $B_n < E_n$. If the difference ($D_n = E_n - D_n$) is lower than a threshold, we need to renormalize the encoder. The renormalization procedure for arithmetic coding has been discussed in [7]. A similar

extension of renormalization procedure may be possible for chaotic maps. But, for the evaluation designs considered in this work, we have considered 64 bit encoder without any renormalization procedure.

In case of decoding, Control Unit (CU) transmits the coded symbol into CMI, which is then iterated over Piecewise linear map and reported back to CU. The CU makes a comparison with chaotic map indicated by the key and outputs a single bit output.

CMI has a multiplier and an adder to perform chaotic iteration. The multiplication and addition coefficients are obtained from a look-up table/ RAM collating the input symbol, key value and probability value as the input address. The Look-ed up value or a word is demultiplexed to obtain the multiplication and addition coefficients. This option can work fine for at most binary case, and for the case where p value is limited to fixed precision, say 8 bits. Such fixed precision approximations have been introduced in CABAC [8], however it leads to approximation of results. Alternatively, we can use a multiplexer which can implement look-up using physical circuits to compute the return values. The second approach has been implemented in this work, as it allows more flexibility in design and accuracy in computation.

For implementation, the input and output intervals to the Chaotic Map Iterator are represented in 64 fixed point (0 bits integer and 64 bits fraction, shortly I.F 0.64) arithmetic. The symbol probability has been quantized to 8 bits (I.F 0.8).

In BCAC architecture, the choice of chaotic map is made based on a key value, and is not precomputed. For this implementation, the CMI has 1 bit symbol input, 8 bit symbol probability and 3 bits for choice of chaotic map (for binary case $N = 2$, hence number of different chaotic maps is $N2^N = 8$. The 12 bit lookup can be implemented using a 512 words RAM or Look-up Table, with 16 bits word. Alternatively, we used 8-to-1 multiplexer to obtain the coefficients corresponding to a key, each coefficient being generated based on value in Table 1 in [6]. The implementation on target FPGA gave a clock frequency of 500 MHz, utilizing 321 slices and 10 DSP48E1 slices (which have optimized multiplier and accumulator operation implemented in VLSI). Mapping these multiplication to FPGA logic increased the slice usage to 1474, without any change in achievable clock frequency.

The BCAC decoder hardware utilization was 173 slice LUT with 5 DSP slices (806 slice LUTs with LUT multiplier) with a clock frequency of 510 MHz (500 MHz). The 64x8 bit multiplier is implemented by ISE into 5 DSP slices. However, the same multiplier can be optimized and implemented without hardware multipliers using other multiplier such as square root multiplier, reconfigurable constant multipliers etc. The hardware requirements are basically dependent on size of Look-up logic which increases exponentially with increase of N. The throughput of this implementation is 1 bit per cycle with a 510 MHz clock, i.e. 510 Mbps. To consider the area effectiveness of this design, we consider throughput per slice, with the second implementation where we implement multiplication in LUTs rather than using DSP48E1 slices present in device. The throughput/ slice for this design is obtained as 322 Kb/slice.

III. Conclusion

In this paper, we presented architecture for simultaneous coding and encryption using chaotic maps. After presenting the hardware requirements and computations involved in chaotic maps, we mapped these designs into a Virtex-6 FPGA to obtain a performance analysis on real hardware. This work is one of the earliest hardware implementation of chaotic maps, first reported implementation of chaotic maps for simultaneous coding and encryption.

We are looking for, and encourage other readers also for future work in incorporating re-normalization and context to this encoder, so that this mode can be added to CABAC or other encoders.

References

[1] G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Communications*, vol. 29, no. 6, pp. 858–867, Jun 1981.

[2] M. Luca, A. Serbanescu, S. Azou, and G. Burel, "A new compression method using a chaotic symbolic approach," in *Proc. IEEE Commun. Conf.* Citeseer, 2004, pp. 3–5.

[3] N. Nagaraj, P. Vaidya, and K. Bhat, "Arithmetic coding as a non-linear dynamical system," *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 4, pp. 1013–1020, 2009.

[4] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 905–917, Oct. 2006.

[5] H. Kim, J. Wen, and J. Villasenor, "Secure arithmetic coding," *IEEE Trans. Signal Processing*, vol. 55, no. 5, pp. 2263–2272, May 2007.

[6] A. Pande, J. Zambreno, and P. Mohapatra, "Joint video compression and encryption using arithmetic coding and chaos," in *IEEE International Conference on Internet Multimedia Systems Architecture and Application*, 2010.

[7] A. Moffat, R. Neal, and I. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems (TOIS)*, vol. 16, no. 3, pp. 256–294, 1998.

[8] D. Marpe, H. Schwarz, G. Blttermann, G. Heising, and T. Wieg, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, pp. 620–636, 2003.