Design of an Area-Efficient ASIC Architecture for Context-Based Binary Arithmetic Coding

Chu Yu and *Hwai-Tsu Hu*
Department of Electronic Engineering
National I-Lan University, I-Lan, Taiwan, R.O.C.
{chu, hthu}@niu.edu.tw

ABSTRACT

This paper presents a novel pipelined ASIC architecture for the context-based binary arithmetic encoder in JPEG2000, which is compatible with the arithmetic encoder defined in ISO/IEC 155444-1. By making use of a particularly 3-stage pipelined architecture, the proposed encoding architecture is able to process every input symbol within a single clock cycle. This architecture not only overcomes the problem of unfixed pipeline stages emerging from the uncertain times of renormalizations during the encoding phase, but also reduces the number of registers necessitated by the pipeline structure.

1: INTRODUCTION

Arithmetic coding in JPEG2000 employs an adaptive binary arithmetic coder, called MQ coder. This coder is a kind of statistical coding technique, of which the efficiency is better than that of the classical Huffman coding method. To achieve source data compression, the arithmetic coder adopts both modeling and coding schemes to eliminate the redundancies among input data sequences. The modeling scheme is primarily used to estimate the probability of each input symbol and subsequently update such a probability. Once the probability is predicted by the modeling procedure, the arithmetic coder can produce the resultant coded bit-stream accordingly. The applications of the arithmetic coder are generally addressed in data compression, image coding, and video coding, etc. It can also combine with other facilities, such as the discrete wavelet transform, scalar quantizer, and embedded block coding with optimized truncation (EBCOT), to form a JPEG2000 image coder. Such an image coder may serve as the core of digital cameras to substitute for the conventional JPEG image coder so that better image quality and advanced functions can be provided.

Generally, entropy coding for data compression can be done either by arithmetic coding or by Huffman coding. In particular, the entropy coding of JPEG2000 [1] is composed of both EBCOT and arithmetic coding. As the EBCOT is manipulated in a bit-wise manner, a binary arithmetic coding technique is designed to meet the specification. In order to enhance the coding performance, a context-based binary arithmetic coder, the same as the MQ-coder of JBIG2, has been chosen. However, the amount of coding computation is huge, thus requiring dedicated hardware realizations for such an arithmetic coder.

Several ASIC architectures [2]-[8] for the arithmetic coder have been proposed. Chen et al. [2] presented an implementation of a 4-stage pipeline arithmetic encoder for JPEG2000. This architecture took advantage of the 4-stage pipelined facility to improve the throughput rate. Kuang et al. [3] proposed a dynamically pipelined design of an adaptive binary arithmetic coder with run-time determined latency. Ong et al. [4] demonstrated an efficient architecture for context-based adaptive arithmetic coding. Based on parallel leading-zeros detection and bit-stuffing handling, each of input symbols was encoded and decoded within one clock time. Recently, Chiang et al. [5] devised a pass-switch version of the 4-stage pipelined architecture. The architecture could be shared by the 3-pass outputs of ECBOT. Dyer et al. [6] also proposed a 4-stage pipelined architecture. Especially, this architecture consumed two (*CX*, *D*) pairs every clock cycle. It achieved a 1.9 times improvement in throughput at a cost of 1.7 times hardware increase. More recently, Zhu et al. [7] proposed a 3-stage pipelined architecture. Based on a flexible MQ coder, another 4-stage pipelined architecture was presented by Zhang and Xu [8]. The architecture was able to process a dual symbol within a single clock cycle as well.

Although the above-mentioned architectures were mainly devoted to the reduction of coding complexity, they might either take a lot of clock cycles or require more hardware resources to encode an input message. To overcome the drawbacks of these existing architectures, we attempt to develop a special pipelined architecture for the context-based arithmetic encoder. The proposed architecture employs a 3-stage pipelined architecture to get around the unfixed pipeline stages resulting from the uncertain times of renormalizations during the encoding phase. This proposed design differs from the commonly pipelined system mainly in combinational logics between two pipeline registers. As for our design, a finite-state-machine (FSM) circuit is employed instead of conventional combinational logics. In addition, some useful cost-down techniques are used to improve the efficiency in area. By exploiting these foregoing strategies, the hardware cost used in the proposed architecture can be effectively decreased.

2: ARITHMETIC ENCODING SCHEME

Arithmetic coding is a kind of lossless data compression technique for either bi-level or multilevel symbols. The concept of arithmetic coding originates from the recursive probability interval subdivision, by which each input symbol can be mapped to a real-number interval. In the JPEG2000 standard, the real-number interval is defined in between 0.75 and 1.5. Thus, by means of persistent interval subdivision, an input message will be ultimately mapped to a high-precision real value in the range from 0.75 to 1.5. For the binary (or bi-level) arithmetic coding procedure, the current probability interval is subdivided into two unequal sub-intervals. The more probable symbol (MPS) is mapped to a larger interval subdivision, whereas the less probable symbol (LPS) is allotted to the smaller one. In other words, the precision for the interval subdivision needs to be high whenever the divided interval is small. On the other hand, as the MPS appears more frequently, the interval subdivision is expressed based on a coarser precision, i.e., fewer bits are enough to encode the interval subdivision. This just coincides with the object of data compression.

The manipulation of context-based arithmetic coder in JPEG2000 is conducted as follows. First, a pair of inputs (i.e., decision, *D*, along with context, *CX*) comes from the output of the previous EBCOT stage. Both of them are processed in parallel to generate the wanted compressed data (*CD*). Context, *CX*, is then selected from the estimated probability during the encoding stage of *D*. Three working registers together with some ROM's and RAM's are required to accomplish the encoding procedure. The main functions with respect to these three registers are summarized in Table 1.

Table 1: Function of registers used in the encoder

Register Name	Bit Width	Function
Register *C*	28	Record the current interval base
Register *A*	16	Record the current interval size
Register *B*	8	Record the compressed output data

As for the encoding procedure shown in Fig. 1, the size of the current interval is kept in the register *A*. The register *C* is always maintained to point to the bottom of the current interval after completing each coding phase. Once every input symbol is fed into the encoder, the current interval *A* is subdivided into two new sub-intervals with length *A*-*Qe* and *Qe*, where *Qe* denotes the probability estimation of an LPS. Depending on whether an MPS or LPS is encoded, one of these two sub-intervals is chosen as the newest interval of the register *A*. To avoid the underflow of register *A* during encoding the LPS and MPS, a renormalization procedure ought to initiate here. Moreover, a terminal procedure (or called the flush procedure) is brought in at the end of every input message. The purpose of this procedure is to flush and output all the coded data which remain in the register *C*.

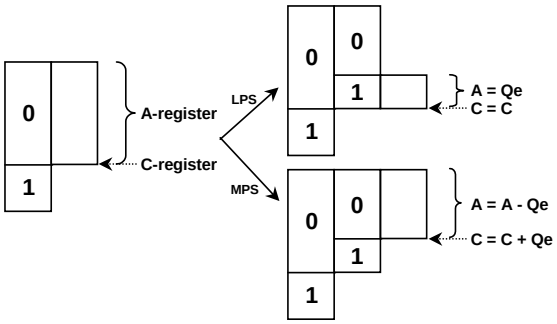

Fig. 1 Encoding process of the arithmetic coder.

3: PROPOSED ARCHITECTURE

The existing architectures for the MQ-coder can be categorized into FSM (also known as non-pipeline) and pipelined design methodologies. The FSM needs at least two cycles to process each input sample. Although this design style easily solves the renormalization problem, the throughput rate is low. Hence the alternative design style, namely pipelined, is often adopted to support real-time applications.

3.1: Proposed Architecture

In general, it is difficult to design the MQ coder based on a regular-flow pipelined architecture due to the uncertain number of compressed output bytes. Another concern against the pipelined structure is dataflow dependency. For instance, because the compressed data formation will maintain a new content in the coding register *C* after each computation, extra waiting cycle time is required for the new low-bound update so as to deal with the next computation. This, in turn, makes the design of these two modules very difficult, especially for the part of pipelining. Apparently, separating the computations of these two modules will facilitate our design. In light of such an idea, we present a novel solution that employs a different pipeline structure. Based on this proposed structure, all the modules used in the proposed architecture are expected to carry out within a single clock cycle. More specifically, we let the interval-size probability subdivision and compressed data formation modules computed in parallel. These two modules perform individually regardless of their mutual dependency. The foregoing mechanism makes every input sample manageable within one clock cycle as a pipelined design, thus providing a high-throughput rate. This is important for real-time image processing. In

particular, we can always accommodate such an MQ coder to high-performance embedded block coding (EBCOT) to construct a high-throughput and high-performance Tier-1 coder in JPEG2000.

According to the above discussion, the proposed arithmetic encoding architecture, like a 3-stage pipelined design style, is illustrated in Fig. 2. This proposed architecture is composed of the context model update, interval-size probability subdivision, and compressed data formation modules. These modules, which correspond to the dash boxes shown in Fig. 2, are described below.

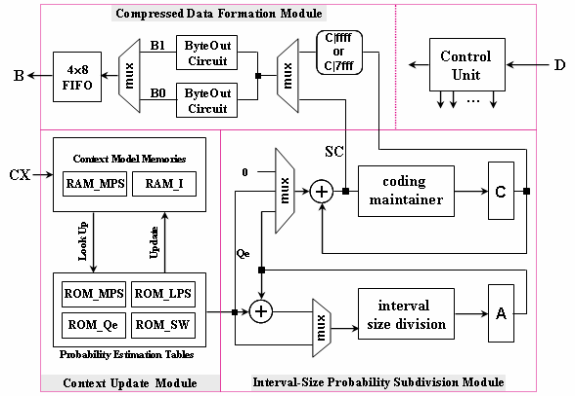

Fig. 2. Proposed ASIC architecture for the MQ-coder.

The context-model module consists of four ROM's and two RAM's. The context model stored in RAM's is updated after the comparison between input decision value (*D*) and MPS symbol under a context value.

The interval-size probability subdivision module is responsible for computing the probable interval size stored in the register *A* and maintaining the low bound of the interval stored in the coding register *C*. The coding maintainer contains a barrel shifter and a new low-bound update. The interval-size division facility is also equipped with a barrel shifter to ease the generation of a new interval-size subdivision. The number of shifts equals the quantity of leading zeros of the new interval size *A*. This means that the whole renormalization used in our design runs at one clock cycle time, regardless of the number of compressed data outputs. Nevertheless, the compressed data formation can be independently extracted from the intermediate outcome of the interval-size probability subdivision module, as shown in Fig. 2, without affecting the subsequent interval-size probability subdivision. This scheme elegantly gets over the problem of unfixed pipeline stages caused by the uncertain number of renormalizations during the encoding phase. Note that this interval-size probability subdivision module employs a finite state machine (FSM) placed between two pipeline registers. Our design differs from the traditional pipelined structure in that the circuit is often a simple combinational logic rather than an FSM circuit.

The compressed data formation module mainly contains two byte-out submodules, a flush hardware, and a four-size FIFO buffer. The proper size of FIFO's has been discussed in [4]. In general, this four-size FIFO shall be enough because a byte-out formation requires at least one input sample. Thus, there is time enough to gracefully send the byte-out data into the FIFO. The byte-out submodule will be described later in more detail. Also, a flush procedure is adopted in the proposed architecture to make the design fully compatible with the arithmetic encoder defined in ISO/IEC 155444-1.

3.2: Byte-out Circuit Submodule

The compressed data output of the MQ encoder is produced by the BYTEOUT procedure shown in Fig. 3, which is invoked from the previous RENORME procedure. As seen from this procedure, it is apparent that the position of the expression $B = B+1$ must be placed after the second logic expression $B = 0xFF$ so as to make all the decisions of this procedure executable within one clock cycle time. However, this will cause an error because the execution sequence of the expression $B=B+1$ is changed. To resolve this dilemma, the original logic expression should be modified as $B = 0xFE$.

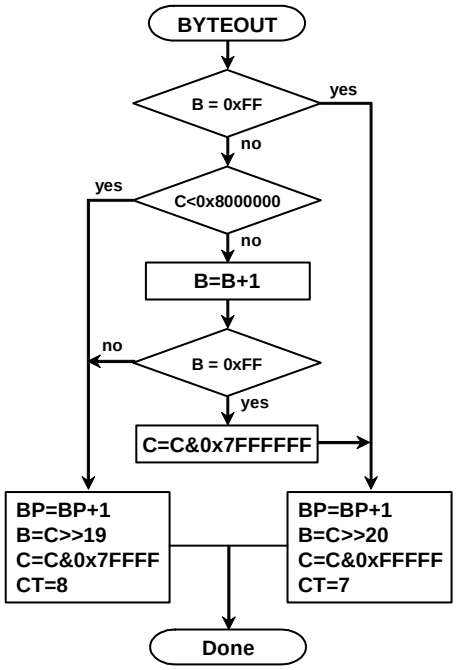

Fig. 3. Byte-out procedure.

The block diagram of the proposed byte-out circuit is shown in Fig. 4. This circuit merely contains simple logic circuits deduced from the foregoing discussion. In order to further simplify the proposed circuit, we convert many logic expressions to some simple logic operations. For instance, the expression $B=B+1$ with a

condition C<0x8000000 is merged into expression $B=B+C_{27}$, where C_{27} is the MSB of C register. Thus, the original decision logic circuit is skipped. Such a manner is often used in the whole proposed architecture to reduce the hardware cost.

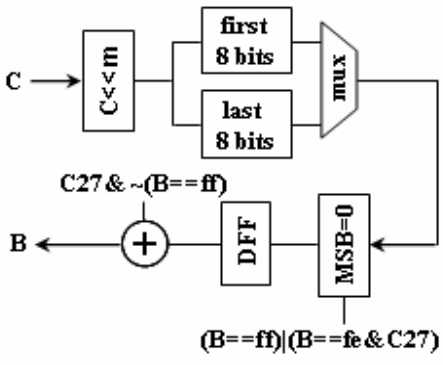

Fig. 4. Proposed byte-out circuit block diagram.

4: PERFORMANCE EVALUATION

To provide a proper performance evaluation, the properties of three other designs along with ours are listed in Table 2. The throughput of these four architectures is the same as the clock rate. As seen from this table, the 4-stage pipelined arithmetic coding architecture proposed by [2] requires 11k logic gates, while another two 3-stage pipelined architectures proposed by [4] and [7] consume about 7k logic gates. In contrast, our design uses merely about 6k logic gates. To shorten the critical path, a carry-select adder scheme has been connected to the adder in our design. Thus, our architecture is allowed to run at a working frequency up to about 250 MHz synthesized by using TSMC 0.18 μm CMOS technology. In addition, the control unit used in our design is very simple, which only demands approximately 150 logic gates. Obviously, our proposed architecture has lower complexity with a high-throughput rate, thus providing a good choice in terms of hardware cost.

Table 2: Comparison of various MQ-coder architectures

Designs	Pipeline Stages	Gates	Speed	Process
Chen [2]	4	11K	100 MHz	0.35 μm
Ong [4]	3	6.9K	200 MHz	0.35 μm
Zhu [7]	3	7K	206 MHz	0.18 μm
ours	3	6K	250 MHz	0.18 μm

5: CONCLUSION

An area-efficient and high-throughput ASIC architecture for the context-based arithmetic encoder has been described in this paper. The architecture employs an innovative pipelined architecture to efficiently solve the uncertain number of renormalizations. Compared with the existing pipelined arithmetic coder ([2], [4], and [7]), our design uses fewer pipeline registers, thus resulting in a decrease of hardware cost. In addition, our design has employed merging and/or simplifying encoding procedures, which significantly reduce the proposed hardware cost. As the proposed architecture demonstrates its advantages in the hardware cost and control complexity, it is suited for the advanced ASIC design, especially for an FPGA implementation. Moreover, the throughput of our design is equal to the working clock rate. Therefore, it can be accommodated into the EBCOT to provide a high-throughput and high-performance Tier-1 coder in JPEG2000 for real-time image processing.

REFERENCES

[1] ISO/IEC JTC1/SC29 WG1, *JPEG* 2000 *part I final committee draft version 1.0*, May. 2000.

[2] H. H. Chen, C. J. Lian, K. F. Chen, and L. G. Chen, "Context-based adaptive arithmetic encoder design for JPEG2000," in *Proc. 12th VLSI Design/CAD Symposium*, Aug. 2001.

[3] S. R. Kuang, J. M. Jou, R. D. Chen, and Y. H. Shiau, "Dynamic pipeline design of an adaptive binary arithmetic coder," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 9, pp. 813-825, Sep. 2001.

[4] Keng-Khai Ong, Wei-Hsin Chang, Yi-Chen Tseng, and Chen-Yi Lee, "A high throughput context-based adaptive arithmetic codec for JPEG2000," in *Proc. IEEE Int. Symp. Circuits Syst*., vol. 4, pp. 133-136, 2002.

[5] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-You Hsieh, and Chih-Hsieh Hsia, "High-speed EBCOT with dual context-modeling coding architecture for JPEG2000," in *Proc*. 2004 *Int*. *Symp*. *Circuits and Systems*, vol. 3, pp. 865-868, May 2004.

[6] M. Dyer, D. Taubman, and S. Nooshabadi, "Improved throughput arithmetic coder for JPEG2000," in *Proc*. *Int*. *Conf*. *Image Processing*, vol. 4, pp. 2817-2820, Oct. 2004.

[7] Ke Zhu, Fang Wang, Xiaofang Zhou, and Qianling Zhang, "An efficient accelerating architecture for tier-1 coding in JPEG2000," in *Proc.7th Int. Conf. Solid-State and Integrated Circuits Technology*, vol. 3, pp. 1653-1656, Oct. 2004.

[8] Yi-Zhen Zhang and Chao Xu, "Analysis and high performance parallel architecture design for EBCOT in JPEG2000," in *Proc*. *Int*. *Conf*. *Image Processing*, vol. 3, pp. 996-999, Sept. 2005.