A Low Power, Fully Pipelined JPEG-LS Encoder for Lossless Image Compression

Xiaowen Li[1], Xinkai Chen[1], Xiang Xie[1], Guolin Li[1], Li Zhang[1], Chun Zhang[2], Zhihua Wang[2]

[1]Dept. of Electronic Engineering, Tsinghua University, Beijing, 100084, P. R. China
[2]Institute of Microelectronics, Tsinghua University, Beijing, 100084, P. R. China

ABSTRACT

By analyzing the features unfit for parallel computation and low power implementation, a VLSI architecture of JPEG-LS encoder for lossless image compression is proposed in this paper. It functionally consists of four parts: Mode decision module, clock controller, three linear parallel pipelines, and a two-tier data packer. Computations are organized in a fully pipelined style in these modules, so that real time data processing can be achieved. The clock management scheme with four interlaced clock domains and a dedicated clock controller is applied to ensure the bottleneck calculation, reduce the clock frequency on non-critical paths, and shut off the working clocks of idle modules, which reduces 15.7% of overall power consumption. The proposed JPEG-LS encoder with the features of low power and high processing speed, has been applied in a wireless endoscopy system.

Index terms: JPEG-LS, fully pipelined architecture, clock management scheme, real time data processing, low power application.

I. INTRODUCTION

Lossless image compression is mainly used in areas that require high quality image processing, such as high speed scanning, medical image processing, remote sensing image processing, etc. Due to the rapid development of image sensor technology, large volumes of data need to be processed in near real time, which puts a great challenge to the storage capacity, bandwidth, and transmitting power of the entire system. Hence, a high speed, low power lossless image encoder with high compression efficiency is crucial to solve these problems.

Among various existing lossless compression schemes, JPEG-LS is the newly ITU/ISO standard for lossless image compression. The relatively low complexity algorithm, low storage requirement, while efficient compression capability of JPEG-LS makes it ideal for hardware implementation [1].

In this paper, a high data processing rate, low power consumption JPEG-LS encoder is proposed, in which a parallel fully pipelined architecture with dedicated clock management scheme is applied. This paper is organized as follows. In Section 2, we give an overview of JPEG-LS algorithm, the features unfit for parallel computation and low power application when implemented in hardware is also analyzed. In Section 3, the architecture of the encoder and the clock management scheme is elaborated. In Section 4, the implementation details and power analysis are described. Finally, conclusions are given in Section 5.

II. OVERVIEW OF JPEG-LS

JPEG-LS is based on modeling and coding to achieve compression. Its data processing follows a raster scan sequence, which is just consistent with the way most image sensors release image data. The causal template used in JPEG-LS is depicted in Figure 1, where *x* denotes the current pixel which to be compressed, *a*, *b*, *c* and *d* are its neighboring pixels, the relative positions of which are shown in the figure. *Ix*, *Ra*, *Rb*, *Rc* and *Rd* denote the intensities of *x*, *a*, *b*, *c* and *d* respectively. The compression of a single pixel includes mode decision, prediction, context modeling, and entropy coding.

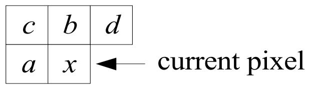

Figure 1: A causal template used in JPEG-LS

A. Mode decision

When *Ra*=*Rb*=*Rc*=*Rd* is detected, the encoder enters Run mode, otherwise it goes into Regular mode. In Run mode, the number of continuous pixels which equal to *Rb*, i.e. run length, is counted until it comes across an unmatched pixel or the end of a line, when the encoder goes into Interrupt mode.

B. Prediction and context modeling

In Regular mode, prediction consists of fixed prediction and adaptive prediction. MED predictor [2] is applied for fix prediction. Adaptive prediction value depends on results of context modeling. The vector {*Rd*-*Rb*, *Rb*-*Rc*, *Rc*-*Ra*} is quantized and mapped into a smaller subset of 365 contexts, we use an integer *q* which ranges from 0~364 to indicate these contexts. Each *q* indexes a context look up table, where the statistics of previous coding errors are collected. The adaptive prediction value is calculated according to the context statistics, and the context statistics are updated with the prediction error subsequently.

In Interrupt mode, the interrupt pixel is also predicted and context modeled similarly, but with only two contexts.

C. Entropy Coding

The prediction errors in Regular mode and Interrupt mode, as well as the run length in Run mode are encoded by Golomb-Rice coder [3][4] respectively, in which a parameter *k* is also dependent upon the context statistics.

In terms of hardware implementation, an encoder with a pipelined architecture has to be realized for real time data processing, so as to avoid the large storage consumption for buffering image data between image sensor and compression circuit. However, due to the poor parallelizability in the algorithm itself, the implementation of JPEG-LS in a parallel architecture is difficult. Once two continuous pixels get a same context, the computation for the second pixel must wait for the completion of the first one, for the updating of the context history. In [5], a limited parallelism is obtained only when the computations do not depend on previous ones. In [6] and [7], more on-chip memory and logic gates are sacrificed to exchange for higher processing rates. However, none of the three methods can realize real time data processing, and the methods in both [6] and [7] suffer from degradation of compression rates.

Furthermore, a single pixel is to be in an exclusive mode only, Regular, Run or Interrupt. If the pipeline is arranged in a normal style, the three modes would be working simultaneously, which results in a high power consumption of the encoder.

III. ARCHITECTURE OF ENCODER

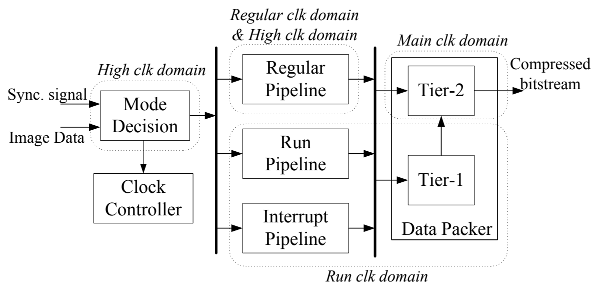

Figure 2: Proposed architecture for JPEG-LS encoder

For the purpose of real time data processing and low power application, a fully pipelined architecture with a clock management scheme is proposed for the JPEG-LS encoder, which is shown in Figure 2. The encoder consists of four parts: 1) mode decision module, 2) clock controller, 3) three parallel pipelines, including regular pipeline, run pipeline and interrupt pipeline, 4) two-tier data packer. These modules work in four different interlaced clock domains, the generations of which are under the control of the dedicated clock controller. The clock management scheme ensures the performance of bottleneck calculations. It also helps to reduce the clock frequency on non-critical paths, as well as control the shutting down of the working clock for each module, so as to reduce the overall power consumption.

A. Mode Decision module

The mode decision module works in the High clk domain. It consists of a single port SRAM for buffering **one row** of a frame, and a FSM including four states. The SRAM is used to buffer data in the current row, which would be used as neighboring pixels of data in the next row. The fours states of the FSM are designed to realize the following operations:

State 1: Receive data from image sensor (or the previous stage), and the received data is the current pixel to be compressed. Update *Ra*, *Rb* and *Rc*, and read out the value of *Rd* from SRAM.

State 2: Write the current *Ix* to the corresponding address. The original data is no more contributing for the subsequent processing, and thus gets overwritten. Determine which mode the current pixel should enter according to the values of *Ix* and its neighboring pixels. Run length is to be counted if in Run mode.

State 3: Change the address and the WE/RD line of the SRAM to get prepared for reading the next *Rd*.

State 4: Wait for the synchronization signal from the image sensor. Go back to State 1 if valid.

B. Clock Controller

This design contains four different interlaced clock domains given by dashed lines as shown in Figure 2. Mode decision module and the bottleneck of regular pipeline work in High clk domain. The other parts of regular pipeline work in Regular clk domain. Run pipeline, interrupt pipeline and tier-1 of data packer work in Run clk domain. And tier-2 of data packer works in the Main clk domain. The task of the clock controller is to control the generation and working states of the three clocks, that is Main clk, Regular clk and Run clk. The architecture of the clock controller is depicted in Figure 3.

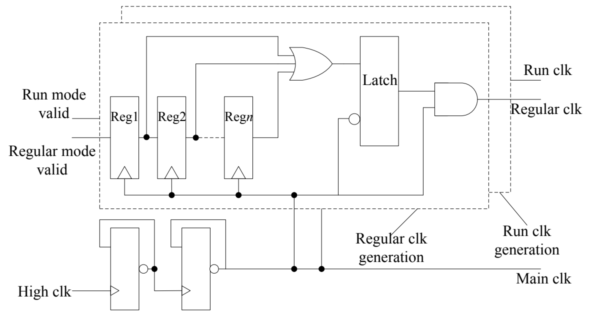

Figure 3: Architecture of the clock controller

The Main clk is generated by four division of High clk through two T flip-flops, and then sent to Regular clk /Run clk generation circuit for processing respectively. The Regular clk generation circuit is shown in the dashed block diagram in figure 3. If a pixel in Regular mode occurs, the port ‘Regular mode’ valid would appear to be ‘1’, otherwise ‘0’ appeared. **Reg 1** to **Reg n** compose a *n*-bit right shift register. The information about the validness of Regular mode is passed down through the right shift register and keeps valid for *n* clock cycles because of the OR gate. The latch and AND gate compose a typical clock gated circuit. *n* denotes the number of the stages in regular pipeline. Thus, the clock generation circuit can provide adequate clock cycles for all the computations in regular pipeline, as well as shut down the clock immediately unless another pixel in Regular mode occurs. Run clk generation circuit works in the same way as the Regular clk generation circuit, but it has to provide clock for three modules, and the value of *n* is the overall latency of the three modules.

C. Parallel pipelines

In this design, regular pipeline, run pipeline and interrupt pipeline which are parallel to each other, can work simultaneously for accessing a high data processing rate. The three pipelines work in different clock domain. The parallelized structure ensures that the idle pipeline can be shut down immediately for power saving, while not affecting other pipeline's normal working state.

The regular pipeline is arranged for three stages and thus has a latency of three clock cycles (see Figure 4). In the first stage, fixed prediction value and context of the current pixel are calculated, the second stage is used to look up the context table according to the context *q*, calculate the parameters for Golomb-Rice encoder, and update the context table. A Golomb-Rice encoder is implemented in the third stage. After the three steps, the input pixel has been transformed into a variable length code.

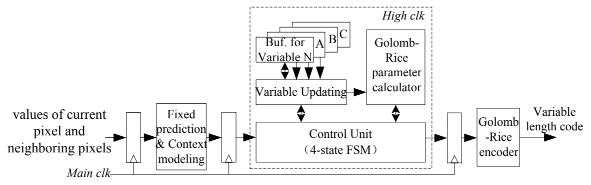
High clk
Buf. for Variable N
A
B
C
Golomb-Rice parameter calculator
Variable Updating
values of current pixel and neighboring pixels
Fixed prediction & Context modeling
Control Unit (4-state FSM)
Golomb -Rice encoder
Variable length code
Main clk

Figure 4: Block diagram of regular pipeline

Regular pipeline works in the Main clk domain mostly. The computations in the second stage involves reading and writing to context table. Therefore, High clk is used as an assistant clock. Since the frequency of High clk is four times of Main clk, the computations can be partitioned into four intervals. The second stage comprises of four blocks of single port synchronous SRAM for buffering parameters *A*, *B*, *C* and *N*, a 4-state FSM as control unit, and logic circuits for updating parameters and calculation of Golomb-Rice encoding parameters. In the four states of the FSM, the following computations are performed under High clk sequentially: Read out the values of parameters *A*, *B*, *C* and *N* according to the context *q*; Calculate the final residual according to *N* and the result of fixed prediction, calculate Golomb-Rice encoding parameter *k* according to *N* and *A*, and then update *N* and *A*; Calculate Golomb-Rice encoding parameter *MErrval* according to *k*, *B* and the final residual, then update *B* and *C*; Write the updated *A*, *B*, *C* and *N* to their corresponding storage cell of the SRAM.

The architecture of Golomb-Rice encoder is given in Figure 5. The encoder has three inputs, *n* is the data to be encoded, *k* is a parameter used in encoding procedure, and *limit* is a parameter to restrict the maximum length of the output code, which is set to 32 in JPEG-LS. *n* and *k* are corresponding to *MErrval* and *k* mentioned in the second stage respectively. The basic design principle of the Golomb-Rice encoder is to left shift 32'h0001 as high bits, and combine it with low bits. After combination, the result is stored in a 32-bit register **CODE**, and the length of the valid bits in **CODE** is indicated by register **LENGTH**. The comparison result of *value1* and *value3* denotes whether the length of the code would exceed 32 bits, and the three MUXs choose (*n,k*) or (*n,limit*) to encode with according to the comparison result.

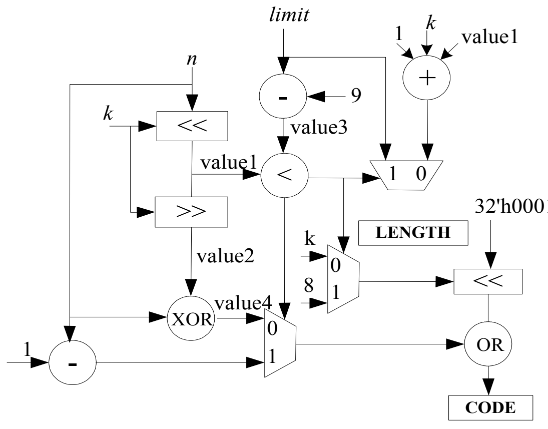

Figure 5: Architecture of Golomb-Rice encoder

The latency of run pipeline is one clock cycle, during which run length is encoded with two outputs generated, indicating the variable length code and the length of valid bits respectively. Interrupt pipeline has a latency of three clock cycles, the tasks in each stage are similar to that in the regular pipeline. Since there are only two contexts existed, registers are used instead of SRAM in the second stage, and thus High clk is unnecessary. Interrupt mode always follows the Run mode, so both of the two pipelines are assigned to work in Run clk domain.

D. Two-tier data packer

The task of the data packer is to convert variable length compressed data into 32-bit fixed length data stream, for the convenience of storage and transportation. The data packer consists of two tiers. Tier-1 works in Run clk domain, Tier-2 in Main clk domain, and both with pipelined architectures.

Tier-1 of the data packer is used to combine the data from run pipeline with that from interrupt pipeline. The two pipelines have latencies of one clock cycle and three clock cycles respectively. The variable length code and the valid length generated by run pipeline are delayed for two clock cycles firstly, and then packed as high bits. This structure can avoid the confusion under the circumstances that a new packing occurs when the previous packing hasn't finished yet.

Tier-2 of the data packer combines the data from Tier-1 with that from regular pipeline, and output the packed data in 32-bit fixed length code stream. The two outputs from regular pipeline are delayed for one cycle, and then combined as low bits.

IV. IMPLEMENTATIONS AND POWER ANALYSIS

The proposed JPEG-LS encoder has been applied in the wireless endoscopy capsule system in [8]. In the system, OMINIVISION OV7648 is used as image sensor, which can provide an image with a maximum resolution of 640×480×8 bits. The digital circuits in the capsule work under a system clock of 40MHz, and the frequency of the sensor's output pixel clock is 10MHz. The function of the whole digital part has been verified by Xilinx xc2v1000-4fg256 [9] (FPGA).

Implemented in UMC 0.18 µm 1P6M standard CMOS technology, the total scale of the JPEG-LS encoder is 17.6k logic gates, plus 18k bits on-chip SRAM (640×8 bits for mode decision, 365×34 bits for regular pipeline). In Table 1, a comparison of the proposed implementation with other implementations available in academic is offered. It can be easily observed that our implementation outperforms in logic scale and on-chip MEM. usage as well. The processing speed in [7] is a little higher. However, the implementation in [7] is not compliant with the JPEG-LS algorithm, and two identical structures, which are used to access high processing speed, result in much more hardware overhead.

Under the High clk of 40MHz, the encoder can compress an image with a resolution of 640×480×8 bits in 31 ms. For a single pixel, the compression is in real time with a latency of 6 Main clk cycles or 24 High clk cycles. The value of *n* for Regular clk and Run clk generation circuit is set to 3 and 4 respectively.

Table 2: Power analysis of 3 different images

	Regular clk domain (mw)	Run clk domain (µw)	Overall (mw)	Saving
lenna	2.82/2.95	12.8/556.8	7.17/8.32	13.8%
baboon	2.82/2.96	13.7/558.1	7.19/8.35	13.9%
black	0.24/0.42	10.3/552.4	4.47/5.68	21.3%

The Power consumption results of post layout simulations using Synopsys VCS and PrimePower [10] is shown in Table 2, including the power consumption in Regular clk domain, Run clk domain and the overall power consumption. Three images with different smooth quality are tested under the 40MHz High clk with 1.8 V supply voltage. The data before and after '/' are the simulation results with and without clock management respectively. The average power consumption is reduced to 15.7 %.

V. CONCLUSIONS

By analyzing the features unfit for parallel computation and low power implementation, a low power, fully pipelined VLSI architecture of JPEG-LS encoder for lossless image compression is proposed in this paper. The parallel, fully pipelined structure ensures a real time data processing of the encoder. The dedicated clock management scheme ensures the bottleneck calculation, as well as reduces the clock frequency on non-critial paths, and shuts down the working clock of idle modules, which reduces 15.7% of overall power consumption. The proposed JPEG-LS encoder has been applied in a wireless endoscopy capsule system, the whole digital part of the capsule has passed the FPGA-based verification and been taped out.

REFERENCES

[1] M. J. Weinberger, G. Sapiro and G. Seroussi, "The LOCO-I Lossless image compression algorithm: Principle and standardization into JPEG-LS," *IEEE Trans. on Image Processing*, vol. 9, pp. 1309-1324, Aug. 2000.

[2] S.A.Martucci, "Reversible compression of HDTV images using median adaptive prediction and arithmetic coding," *Proc. IEEE intern'l Symp. On Circuits and Syst.*, pp. 1310-1313, 1990.

[3] Golomb S W. "Run-length encodings," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 399-401, July 1966.

[4] Rice R F. "Some practical universal noiseless coding techniques," *Tech. Rep. JPL- 79-22*, Jet Propulsion Laboratory, Pasadena, CA, Mar. 1979.

[5] A. Savakis and M. Pioriun, "Benchmarking and Hardware Implementation of JPEG-LS," *ICIP'02*, Rochester, NY, Sep. 2002.

[6] M. Klimesh, V. Stanton, and D. Watola, "Hardware Implementation of a Lossless Image Compression Algorithm Using a Field Programmable Gate Array," *NASA JPL TMO Progress Report* 42-144, 2001.

[7] M. Ferretti, M. Boffadossi, "A Parallel Pipelined Implementation of LOCO-I for JPEG-LS," *17th International Conference on Pattern Recognition (ICPR'04)*, vol. 1, pp. 769-772. 2004.

[8] Xiang Xie, GuoLin Li, and XinKai Chen, "A Low Power Digital IC Design Inside the Wireless Endoscopy Capsule," *Asian Solid-State Circuits Conference (A-SSCC 2005),* pp. 217-220, 2005.

[9] www.xilinx.com

[10] www.synopsys.com

Table1: Characteristics of proposed implementation and other implementations

Work	**Technology**	**Logic area (eq. gates)**	**MEM. Usage (bits)**		**Operating Frequency (MHz)**	**Processing speed (pixel/clock)**
			Context table	**Image buf.**		
Proposed	UMC 0.18µm	17.6k	365×34	1 row	10(Main clk) / 40(High clk)	1
[5]	-	49457	2k	2 rows	66	0.0364
[6]	Xilinx XCV50	-	-	2 rows	12	0.1108
[7]	STM 0.13µm	53096	2×368×38	2 rows	-	1.4