Volume 4 Issue III, March 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

# High throughput Architecture of Arithmetic Coder Used in SPIHT

S.T. Mrudula
*Assistant Professor, Dept Of ECE, S.K.D.E.C*

***Abstract--In this paper we propose a high-throughput memory-efficient arithmetic coder architecture for the set partitioning in hierarchical trees(SPIHT) image compression is proposed based on a simple context model in this paper. The architecture benefits from various optimizations performed at different levels of arithmetic coding from higher algorithm abstraction to lower circuits' implementations. First, the complex context model used by software is mitigated by designing a simple context model, which just uses the brother nodes' states in the coding zero tree of SPIHT to form context symbols for the arithmetic coding. The simple context model results in a regular access pattern during reading the wavelet transform coefficients, which is convenient to the hardware implementation, but at a cost of slight performance loss. Second, in order to avoid rescanning the wavelet transform coefficients, a breadth first search SPIHT without lists algorithm is used instead of SPIHT with lists algorithm. Especially, the coding bit-planes of each zero tree are processed in parallel. Third, an out-of-order execution mechanism for different types of context is proposed that can allocate the context symbol to the idle arithmetic coding core witha different order that of the input. For the balance of the input rate of the wavelet coefficients, eight arithmetic coders are replicated in the compression system. And in one arithmetic coder, there exists four cores to process different contexts. Fourth, several dedicated circuits are designed to further improve the throughput of the architecture.***

***Index Terms—Arithmetic coding circuit, context model, out-of-order execution, set partitioning in hierarchical trees (SPIHT), VLSI arithmetic coder architecture.***

## I. INTRODUCTION

AS arithmetic coding (AC) method can obtain optimal performance for its ability to generate codes with fractional bits, it is widely used by various image compression algorithms, such as the QM in JPEG, the MQ in JPEG2000, and the context-based adaptive binary arithmetic coder (CABAC) in H.264. Especially, the set partitioning in hierarchical trees (SPIHT) [9] uses an AC method to improveits peak signal-to-noise ratio (PSNR) about 0.5 dB. Although the theory and program code of AC are mature, the complicated internal operations of AC limit its application for some real time fields, such as satellite image and high speed camera image compressions. In order to achieve performance gains, high speed architecture of AC in compression scenarios must be designed to meet the throughput requirement. Thus both industrial and academic research groups have put their efforts to AC hardware architectures for various image compression systems. However, there are two main challenges in hardware architecture design for high speed applications. One is data dependencies in AC which require the result of iteration before next run can commence during the adaptive model up-date and internal loops. The other one is that AC requires increasingly greater precision as more data arrive.

## II. OVERVIEW OF THE SYSTEM

The detailed architecture of SPIHT encoding is shown. The original images are transformed by the line based lifting wavelet engine at first. The transformed coefficients are written into the wavelet coefficients buffer. The processor dispatcher receives coefficients in the breadth first way from the wavelet coefficients buffer and allocates these coefficients to one of arithmetic coders (ACs) from eight processors array through the internal bus. In order to adapt a wide precision and compression ratio range, eight arithmetic coders are symmetric and work in parallel. The output of each arithmetic coder is sent to the internal bus and is distributed to the corresponding code FIFO by the code FIFO dispatcher. The Read FIFO and Truncate module are responsible for the final code stream formation,which reads each code FIFO from top to bottom and truncates the code stream according to the bit rate requirement. Besides the main parts in the architecture, there are some auxiliary modules. The power management part will stop the clock input for the unused bit-planes of each arithmetic coder based on the maximal bit-plane register file for power reduction. The configuration and control part is responsible for the parameters setting such as image resolution, wavelet type, decomposition array includes twelve FIFOs, which store the context values of twelve bit-planes. The size of each FIFO is 256 5 bits because each code tree has 256 nodes and 16 different contexts will use 4 bits, the last bit is used for the binary context symbol.

Volume 4 Issue III, March 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

The coding core part reads these context first-input–first-outputs (FIFOs) sequentially and calculates the corresponding context to form code bytes. The control signals for the whole architecture are also asserted by this part. The arithmetic coder in the overall architecture plays an important role during the coding process. The arithmetic coder consists of three main parts, i.e., the tree construction noted as Tree Con, the bit plane context FIFOs array and the coding core. The tree construction part visits the wavelet coefficients by the breadth first search order. During the reading process, the context values of each bit-plane are formed based on the context model mentioned. For speedup, all valid bit planes are scanned in parallel. The invalid bit-planes are idle by stopping the corresponding clock. The bit plane context FIFOs array includes twelve FIFOs, which store the context values of twelve bit-planes. The size of each FIFO is 256 5 bits because each code tree has 256 nodes and 16 different contexts will use 4 bits, the last bit is used for the binary context symbol. The coding core part reads these context first-input–first-outputs (FIFOs) sequentially and calculates the corresponding context to form code bytes.

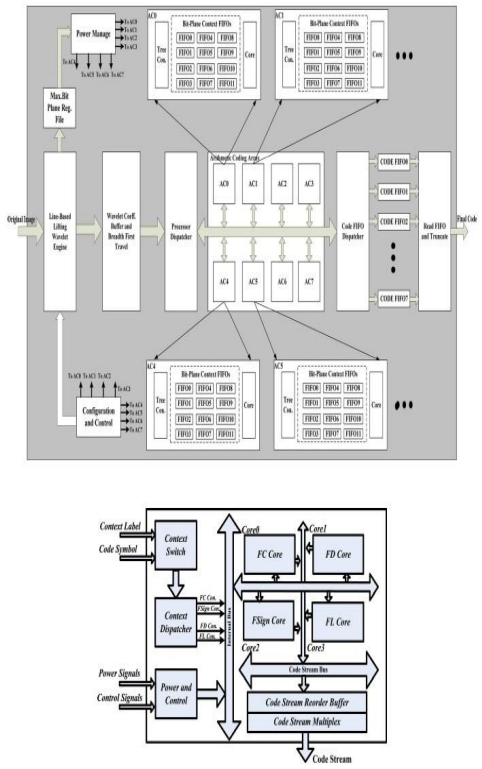



Figure 1architecture for spiht

## III. ARCHITECTURE OF AC CORE

The structure of the core part is illustrated in fig1. The input signals can be divided into two categories, i.e., the context related and the control related. When the context label and binary connected to the common bit detector (CBD) part which unrolls the internal loop and records the same bits from the MSB to the LSB between two registers. At last the same bits are collected to form byte-align code stream in the Bit Assembly. These bytes are supplied directly to the Code Stream Output part for emitting bytes to the external bus. The Coding Control part is responsible for the whole code core's running and sending the various commands and control signals. There are only one adder and one multiplier in the unit which is suitable for high speed implementation with FPGA devices. For speedup purpose, a CLA and a fast multiplier array are employed to reduce the delay of critical path. The new bound values are then registered and connected to the common bit detector (CBD) part which unrolls the internal loop and records the same bits from the MSB to the LSB between two registers. At last the same bits are collected to form byte-align code stream in the Bit Assembly. These bytes are supplied directly to the Code Stream Output part for emitting bytes to the external bus. The Coding Control part is responsible for the whole code core's running and sending the various commands and control signals.

For AC, the code bits are generated by an internal loop, which is essential to the architecture design. In Fig. 7, a CBD module is used to unroll the internal loop. The inputs of CBD are low and high values after calculation of formula (1). The bit_valid_count signals the output bit count from the MSB to the LSB of bit_value register. The bit_value is just a concatenation of common bits and

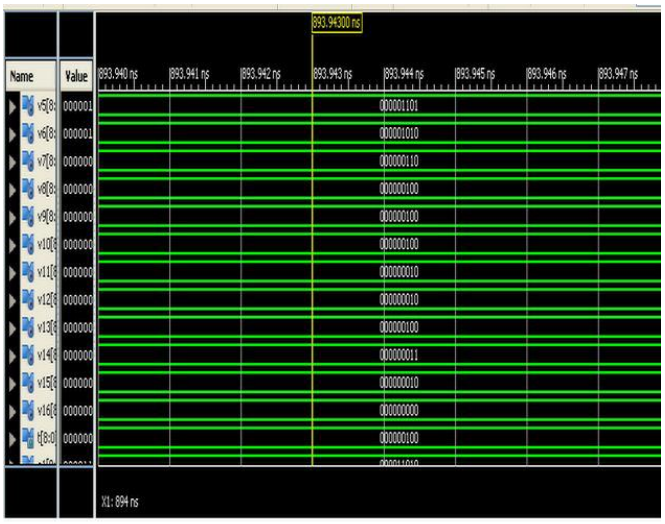
Volume 4 Issue III, March 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

bits_to_follow which is used for underflow. The low_update and high_update registers are shifted values for two bounds used for the next run. Fig. 8 illustrates the detailed structure inside the CBD module.

## IV. EXPERIMENTAL RESULTS

In CBD, the leading bit check (LBC) detects the common bits between low and high registers which consists of a 16-XOR gate and a leading zero detector for 16 bits (LZD16) circuit . The bit follow check (BFC) module is an array for checking the mode of underflow, which is explained by Fig. 9. For speedup purpose, we check 15 possible cases for the bit follow check, which means that the pos_selof LBC selects one of the bit follow values from 15 cases based on the proper common bit position. the internal structure for the BFC. The BFC denotes the bit follow check of two vectors, which can be implemented by a simple logic gate and an LZD circuit . The corresponding bit follow value is then registered in the Bit Follow Register File, which is used for the output and the shift of two bounds. Three register files are employed for the output, low and high registers. The low and high registers are shifted left according to the values of possel and the bit_to_follow register. And the output register emits the proper common bits and the underflow bits according to these registers.

Simulation results for threshold calculator used for finding significant one using the formula is as in the following figure




Simulation for spiht module where spiht lists output only for values greater than threshold is as in the figure

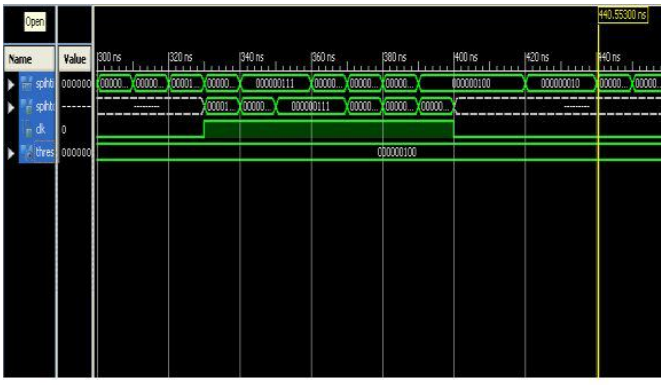



Figure 2 internal structure for arithmetic coder

## V. CONCLUSION

Arithmetic coding makes itself a standard technique for its high efficiency. However, as far as hardware implementation is concerned, the complexity of calculation limits AC in the filed of high speed real-time coding. For improvement of throughput purpose, we propose a high speed architecture of AC used in SPIHT without lists algorithm. In the architecture, a simple context scheme is used first to reduce the memory size. Then high speed calculation units are employed for speedup purpose. Especially, a

power control module can reduce the power dissipation efficiently. It is a high parallelism and calculation device that makes the speed of context processing fast. From the simulation results, our AC architecture can meet many high speed image compression requirements. And the degradation of performance incurred by the fixed point calculation is slight.

## REFERENCES

[1] J. Rissanen, "Generalized kraft inequality and arithmetic coding," IBM J. Res. Developm., vol. 20, no. 3, pp. 198–203, May 1976.
[2] J. Rissanen and G. G. Langdon, "Arithmetic coding," IBM J. Res. De- velopm., vol. 23, no. 2, pp. 149–162, Mar. 1979.
[3] ISO/IEC JTC1 Information Technology-Digital Compression and Coding of Continuous-Tone Still Images-Part 1: Requirements and Guidelines, ISO/IEC International Standard 10918-1, ITU-T Rec.
[4] JPEG2000 Part I Final Draft International Standard, ISO/IEC JTC1/ SC29/WG1 N1890, Sep. 2000.
[5] D. Taubman, "High performance scalable image compression with EBCOT," IEEE Trans. Image Process., vol. 9, no. 7, pp. 1158–1170, Jul. 2000.
[6] B. Cao, Y.-S. Li, and K. Liu, "VLSI architecture of MQ encoder in JPEG2000," J. Xidian Xuebao, vol. 31, no. 5, pp. 714–718, Oct. 2004.
[7] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC), JVT-G050, Joint Video Team of ITU-T and ISO/IEC JTC 1, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2003.
[8] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression stan-dard," IEEE Trans. Circuits Syst. for Video Technol., vol. 13, no. 7, pp. 620–636, Jul. 2003.
[9] A. Said and W. A. Pearlman, "A new ,fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. for Video Technol., vol. 6, no. 3, pp. 243–249, Mar. 1996.
[10] Y. Wiseman, "A pipeline chip for quasi arithmetic coding,"