High Speed Adaptive Binary Arithmetic Coder used in SPIHT

Smitha Mariam Chacko

Abstract-The CABAC design is based on the key elements of binarization, context modeling, and binary arithmetic coding. Binarization makes efficient binary arithmetic coding via a unique mapping of nonbinary syntax elements to a sequence of bits, which are called bins. Each bin can either be processed in the regular coding mode or the bypass mode. The latter is chosen for selected bins in order to allow a speedup of the whole encoding process means of simplified non adaptive coding benefit, where a bin may be context modeled and subsequently arithmetic encoded. A pipeline register is inserted between context memory and the binary arithmetic decoder to reduce the critical path of the loop. The speed of the SPIHT algorithm is increased by using CABAC then using Arithmetic Coder. The Coding is done in VHDL language and synthesized using Xilinx ISE 13.2 and simulated using ISim. As a design decision, the speed of the SPIHT algorithm is increased by using CABAC than using Arithmetic Coder.

Index Terms- Arithmetic coding, context model, Set Partitioning In Hierarchical Trees, CABAC, syntax elements, binarization, bin.

1 INTRODUCTION

Image compression is an application of image processing performed on digital images. The main objective of image compression is to reduce the redundancy of the image data in order to be able to store or transmit data in an efficient form. Generally a compression system consists of encoder and decoder stages. An arithmetic coder could be used as symbol encoders. Arithmetic coding (AC) method can obtain optimal performance for its ability to generate codes with fractional bits and it is widely used by various image compression algorithms. Arithmetic coding makes itself a standard technique for its high efficiency.

In recent times, the SPIHT with list algorithm has become more popular in image compression techniques. SPIHT with lists algorithm uses three different lists to store significant information of wavelet coefficients for image coding purpose. Three lists are the list of Insignificant Pixels (LIP), and list of Significant Pixels (LSP). At first, SPIHT combines nodes of a coefficient tree in wavelet domain and its successor nodes into one set which is denoted as insignificant. With travelling each tree node, sets in the LIS are partitioned into four different subsets and these are tested for significant state.

2 LITERATURE REVIEW

For the QM coder in JPEG, Andra's [1] gave a new architecture which decreases operation for the more probable symbol (MPS) and used a non-overlap window style for the speedup purpose. In [1], the probability interval partition's accelerated by exchange of the less probable symbol (LPS) interval with the MPS interval. Thus the amount of operations is reduced by 60% to 70% compared with other coders. Another highlight of [1] is the nonoverlap window that is applied to the continuous MPS in order to simplify renormalization operation. Due to simple operations in Andra's order the performance is slightly lowered by 1% - 3%.

- *Smitha Mariam Chacko is currently pursuing masters degree program in embedded systems in Sree Buddha College of Engineering,Kerala, India,E-mail: smitha.rene@gmail.com*

Wheeler [2] proposed a modified SPIHT algorithm which does not use list. Because of no insert and search operations for list, the speed of algorithm can be improved greatly.

Wiseman [3] proposed systolic hardware architecture for a quasi AC which is a simple version of AC. In [3] the AC uses a pipeline processing to compute each stage, which eliminates an internal high frequency clock and utilizes fast lookup table for state transitions. Although the architecture can improve the speed of the internal operations, such as the probability internal update and cumulative calculations, it cannot offer supports for multi-context processing in image compression fields.

Marpe [4] proposed that by combining adaptive binary arithmetic coding technique with context modeling, a high degree of adaptation and redundancy reduction is achieved. The CABAC framework also includes a novel low- complexity method for binary arithmetic coding and probability estimation that is suited for efficient hardware and software implementations.

Taubman[5] proposed a new image compression algorithm, based on independent Embedded Block Coding with Optimized Truncation of the embedded bit streams (EBCOT). The algorithm exhibits state-of-the-art compression performance while producing a bit-stream with a set of features, including resolution and SNR scalability. The algorithm has modest complexity and is suitable for applications involving remote browsing of large compressed images. The algorithm lends itself to optimization with respect to MSE as well as more realistic metrics, capable of modeling the spatially varying visual masking phenomenon.

Jyotheshwar[6] proposed an implementation of the image compression technique of SPIHT in programmable hardware. The lifting based DWT architecture has been selected for exploiting the correlation among the image pixels. In addition, he provided a study on what storage elements are required for the wavelet coefficients.

In SPIHT algorithm aspect, many researchers proposed various modifications to improve performance of

SPIHT. Some algorithms aim for better PSNR values. Kassim[7] introduced a method for selecting an optimal wavelet packet transform basis for SPIHT, which efficiently compacts the high frequency sub band energy into a few trees as possible and avoids parental conflicts. Their proposed SPIHT-WPT coder achieved improved coding gains for highly textured images.

Inorder to reduce memory and speed up SPIHT software, Akter[8] used one list to store the coordinates of wavelet coefficients instead of three lists of SPIHT and merged the sorting pass and the refinement pass together as one scan pass.

Ansari[9], proposed Context-Based SPIHT(CSPIHT) method, which used a segmentation and interactive method for selecting the contextual region of interest mask to achieve a better performance results in medical images.

Mohanty[10], presented a novel extension technique to SPIHT base image compression with spatial scalability. The preprocessing techniques provide significantly better quality reconstruction at the decoder with little computational complexity. There are two proposals for this paper. Firstly, a preprocessing scheme called Zero-Shifting, that brings the spatial values in signed integer range without changing the dynamic ranges, so that transformed coefficients calculation becomes more consistent. Secondly, the idea to facilitate resolution scalable decoding by rearranging the order of encoded output bit stream has been incorporated. During the sorting pass of the SPIHT algorithm, he had modeled the transformed coefficient based on the probability of significance, at a fixed threshold of the offspring.

A high throughput memory efficient arithmetic coder architecture for the SPIHT image compression based on a simple context model had been proposed by Kai Liu[11]. The architecture benefits from various optimizations performed at different levels of arithmetic coding. The simple context model results in a regular access pattern during reading the wavelet transform coefficients. In order to avoid rescanning the wavelet transform coefficients a BFS-SPIHT without lists algorithm is used.

3 PROPOSED METHOD

3.1 Wavelet Transform

Two dimensional wavelet transform decomposes airspace domain image to frequency domain through multi-resolution decomposition, and can continue the multi resolution decomposition of the low frequency data after decomposition. A two dimensional wavelet transform transforms results decompose the image into four sub hands respectively named LL1, LH1, HL1 and HH1. The second change only further decomposes the low frequency sub image LL1 to four sub images: LL2, LH2, HL2 and HH2.

There is a strong geometrical similarity among the sub images, especially among the sub images of the same direction. The sub image is similar to the original image in geometry, so further coding can be proceeded according to its characteristic. It can continually restore the superior low frequency sub-band data by the decomposed low frequency sub-band data and the high frequency sub-band data, it can also restore the original image through the reconstruction process.

In the usual wavelet transform, the wavelet transform coefficient is floating point type, so when coding the image of wavelet transform domain, the first step is to change the wavelet transform coefficient into integer and then quantify it. The above process will cause the data distortion and make the data quantified not suitable for the nondestructive compression of image. It can be solved by the reversible wavelet transform through proper lifting scheme based on integer number-integer number reversible transformation.

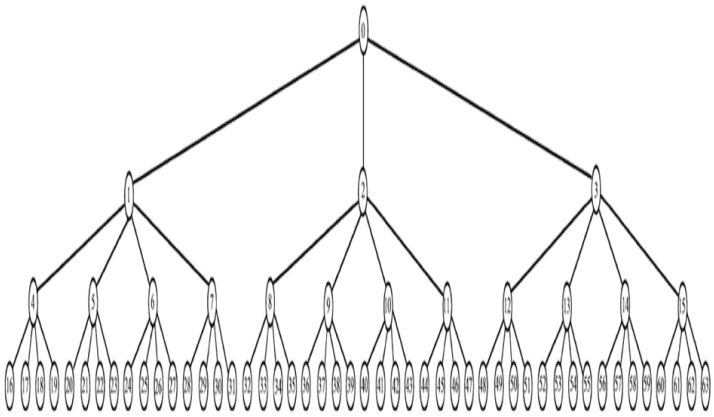

Fig1: Travel order of BFS in SPIHT [11]

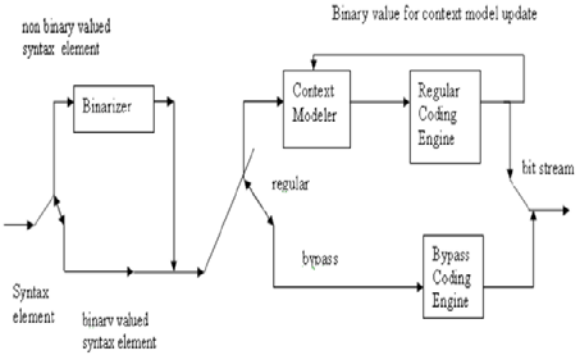

Fig2: CABAC Framework

3.2 SPIHT Algorithm

Wavelet Transform make most of the energy of image transform domain coefficient focuses on the low frequency sub-band, combining with the data in high frequency sub-band to form a series of zerotree. SPIHT algorithm shows the relationship between different sub-band through the data quad tree structure, using the characteristics of the wavelet coefficient and position relations. It continually plots the set of the important coefficients in the offspring and divides the children nodes into important pixels and not important pixels through scanning and form a new division to set with the offspring of

the non-children nodes. When there is no important pixels in offspring of some nodes in unimportant pixel set, use a single bit of data 0 to express the offspring.

As shown in figure 1, single bit data can be used to express some pixel set, and the quadtree structure reflects the position relationship among pixel, therefore, image can get higher compression ratio through SPIHT algorithm. SPIHT algorithm finally causes a series of binary coding sequence expressing the sign of the pixels and whether the pixels are important or not. It shows that wavelet transform makes energy focuses on the low frequency sub-band, which makes the probability increase greatly that the quad tree father node coefficient is bigger than its offspring. It is more advantageous to the decomposition of the unimportant pixel set in SPIHT algorithm, so as to enhance the efficiency of the SPIHT algorithm.

3.3 CABAC Algorithm

After transformation, quantitative processing and some other process, usually last step is entropy coding. Entropy coding is a kind of lossless information to compress data. Figure 2 shows the CABAC framework. CABAC realizes lossless compression by establishing corresponding relationship between source symbols and code word according to the distribution character of the probability of the source symbols. Based on the analysis of the coding process of CABAC algorithm, the first step is the process the input syntax elements to binary digit sequence, mapping the input symbols as the binary code sequence consisting of "0" and "1" second is the introduction of probability conditionally to encode symbols as the binary code sequence consisting of "0" and "1". Second is the introduction of probability conditionally to encode symbols through context modeling. Then adaptively update the probability distribution of the symbols once again and eventually arithmetic coding the generated binary coding sequence according to the probability distribution.

Here the kth order Exp-Golomb Binarization Scheme is used. Exponential Golomb codes were first proposed by Teuhola in the context of run-length coding schemes. This parameterized family of codes is a derivative of Golomb codes, which have been proven to be optimal prefix-free codes for geometrically distributed sources. Exp-Golomb codes are constructed by a concatenation of a prefix and a suffix code word. The prefix part of the EGk code word consists of a unary code corresponding to the value of $l(x) = [\log2 (x/2k + 1)]$. The EGk suffix part is computed as the binary representation of $x + 2k(1 -2l(x))$ using $k + l(x)$ significant bits. Consequently, for the EGk binarization, the number of symbols having the same code length of is geometrically growing.

The input signals can be divided into two categories i.e., the context related and the control related. When the context label and binary code symbol arrive, the context switch differentiates the input context and sends the context value to the context dispatcher by different paths. The main task of the context dispatcher is to schedule the order of the input contexts, which are sent to different calculation cores. In order for speed up, the context dispatcher can emit the context values to each core by a disorder, which means that execution order can be different from that of input. A small buffer for context value is set in the context dispatcher to implement reorganizing the processing order. Table 1 shows an example of execution using twelve different contexts. Each of four coding cores has its state register to indicate whether the coding core can receive new context. When there is no context in the buffer, the state is set to be idle. If a context symbol arrives, the state of core is set to the context label to block any new context. The dispatcher checks the states to find if there is an idle core. Then the dispatcher combines several contexts and emits them to the corresponding cores. The context and its binary symbol are emitted to the corresponding calculation cores i.e., FC core, FSign core, FD core and FL core through the internal bus. If the incoming context is blocked, it will be delayed in the dispatcher and wait for the next clock cycle to be emitted. At the beginning, four cores are ready for processing the contexts. Then in the first clock cycle, four contexts are emitted simultaneously. In the second clock cycle, two new contexts arrive. As (FD0, D0) context pair is not finished, context pair (FD1, D1) is blocked. Only (FC1, D1) context pair can be emitted to the FC core because (FC0,D0) has been done by the FC core. But in the third clock cycle, (FD1,D1) context pair can be emitted to the FD core because the FD core is ready to process new pair.

Binary Arithmetic Coding performs arithmetic coding of each bin based on bin value, type, and the corresponding context model of the bin. BAC is a recursive procedure of coding interval subdivision and selection. Because Range and Low of coding interval are represented by finite number of bits (9 bits for Range).

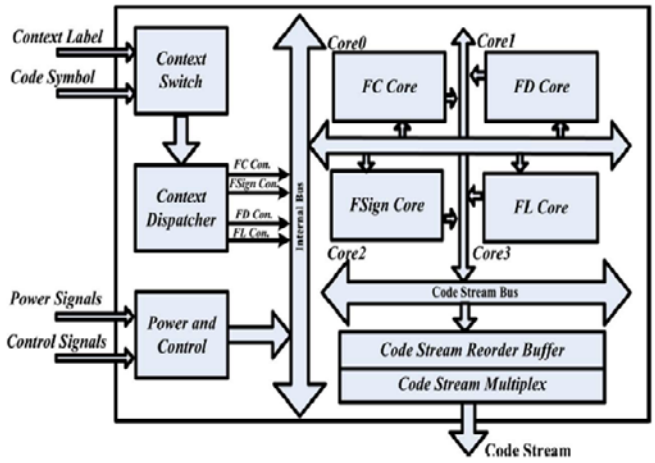

Fig3: Context Modeling [11]

It is necessary to renormalize (scale up) the interval to prevent precision degradation, and the upper bits of Low are output as coded bits during renormalization. The coding interval of (Low, Low + Range) is renormalized when Range is smaller than the threshold value 256 (0x100), which is 1/4 of the maximum range of coding interval renormalization of Range and Low is an iterative procedure, and the maximum number of iterations is 6, as the smallest possible value of Range is 6. For the processing of carry propagation and output of coding bits, the coded bits of CABAC are not output until it is confirmed that further carry propagation will not influence bit values.

When interval length (Range) is smaller than the threshold 0x100, one bit can be output if the interval is located within the top half [0x200, 0x400) or bottom half [0, 0x200) of the maximum coding range.

Arithmetic coding consists of the iterative division of an interval according to the probability of the different symbols. CABAC implements a binary coder, a particular case that allows substantial complexity reduction with high compression efficiency. More specifically, CABAC's arithmetic coder is related to the Q-coder family. By calling low and range to the lower point and the length of the current interval, the encoding equations are: MPS:

lownew = low

rangenew = range − rLPS

LPS:

lownew = low + range − rLPS

rangenew = rLPS

Fig5 shows the flow of CABAC. Firstly, encoding iteration needs the value of rLPS, which is read from memory. rLPS memory address depends on the value of range. It is possible, however, to remove this dependence as only 2 bits from range are used for addressing rLPS. Thus rLPS can be addressed using 6 bits from the state of current context. As a result, four different values of rLPS are obtained. These values are latched. Then the right value of rLPS is selected based on the value of range.

Table1: Example of Out of Order Execution

	Context dispatcher part				State of coding cores			
Cycle	#	Context	Send	Finish	FC core	FD core	FL core	FSign core
1	1	(FC0,D0)	1		1	2	3	4
	2	(FD0,D0)	2					
	3	(FL0,D0)	3					
	4	(FS0,D0)	4					
2	5	(FC1,D1)	5	1	5	2	3	4
	6	(FD1,D1)	–					
3	7	(FC2,D2)	7	5	7	6	–	8
	8	(FS1,D1)	8	4				
			6	2				
4	9	(FC3,D3)	9	7	9	10	11	8
	10	(FD2,D2)	10	6				
	11	(FL1,D1)	11					
	12	(FS2,D2)	–					
5				9	–	10	11	12
				8				
6				10	–	–	–	12
				11				
7				12	–	–	–	–

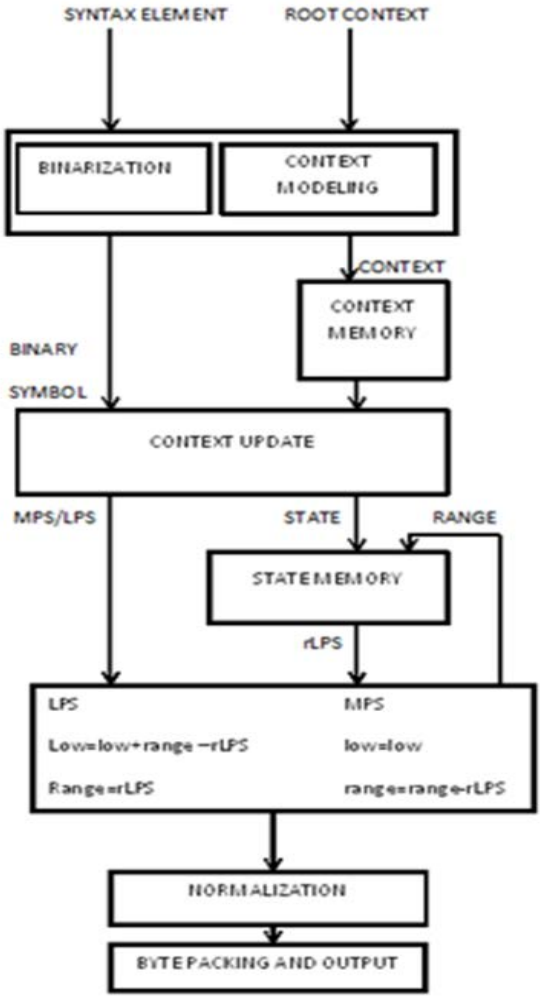

Fig4: Design flow of CABAC

Encoding is different depending on whether the symbol is the Most or the Least Probable Symbol for the current encoding context. The conversion to MPS/LPS is performed during context managing. For equally-probable symbols this conversion is not necessary. Then, interval normalization is applied. As a result, a variable amount of bits is produced every cycle ranging from 0 to 8. At this point, data dependence ends. The new values of low and range are now normalized and a new iteration can start. Normalization is performed in a single step by using Leading-Zero Detection and barrel-shifters.

3.4 CABAC USED IN SPIHT

After coding by SPIHT algorithm based on wavelet transform, image data changes to a bunch of binary codes which is very fit to perform arithmetic coding. At the same time CABAC is one of the high powered arithmetic coding algorithms. The essential problem is to binary transform source data. Therefore, it could be realized further compression by continually carrying out CABAC algorithm to the coding sequence coming from SPIHT algorithm based on wavelet transform (fig5). And it will save the cost of binary transformation to source data in CABAC algorithm.

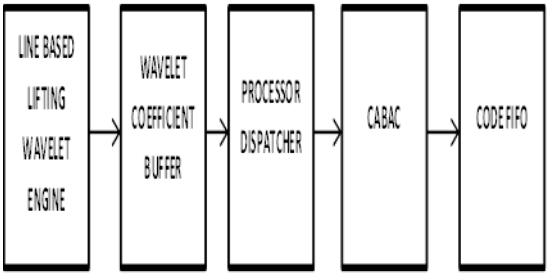

Fig5: Block diagram of CABAC in SPIHT

In fig6 the CABAC decoding process is divided into two stages. Context selection and memory read is done in stage 1 while binary arithmetic decoding is done in stage 2. When the binary arithmetic decoder decodes the current bin, the context modeler fetches the probability the models used for the next bin. Here an iteration of bin decode is done in one cycle, where the CABAC performs context selection and binary arithmetic coding. A pipeline register is inserted between the context memory and the binary arithmetic coder to reduce the critical path of the loop.

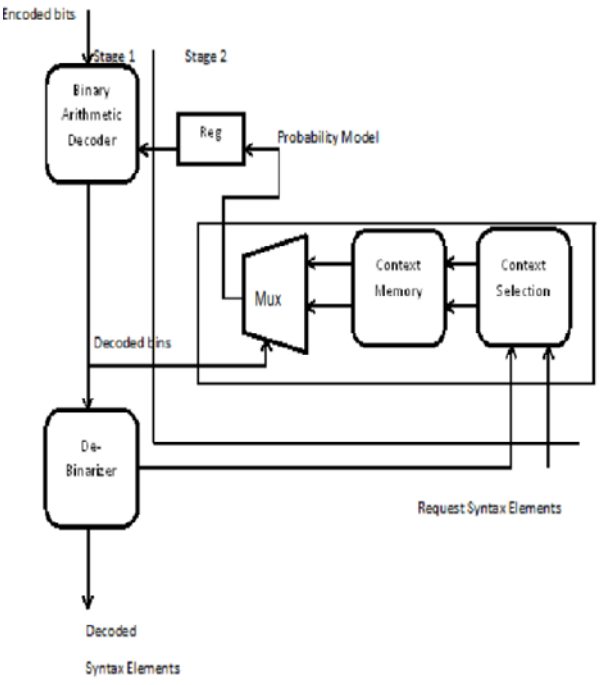

Fig6: Block Diagram of pipelined CABAC

4 SIMULATION RESULTS

In fig7, the input is inData .And the coded output is obtained from bits_to_go.

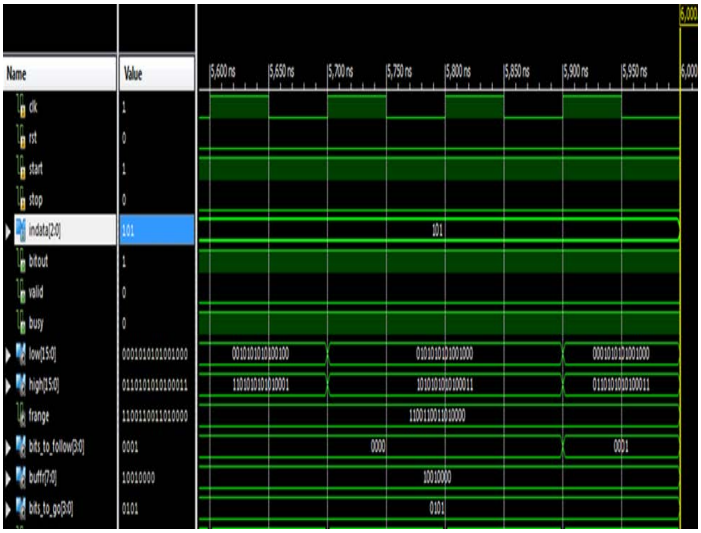

Fig7: Waveform of Arithmetic Coder

In Figure 8, the indata is the input and the coded output is obtained from codeword

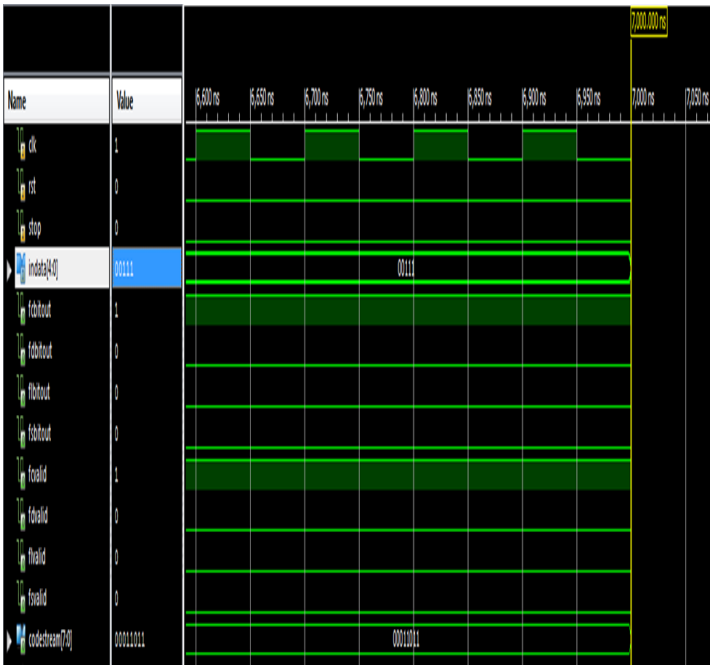

Fig8: Waveform of Arithmetic Coder in SPIHT

The fig9 and fig10 shows the simulation results of CABAC and CABAC used in SPIHT respectively. The inputs are given and the coded outputs are obtained.

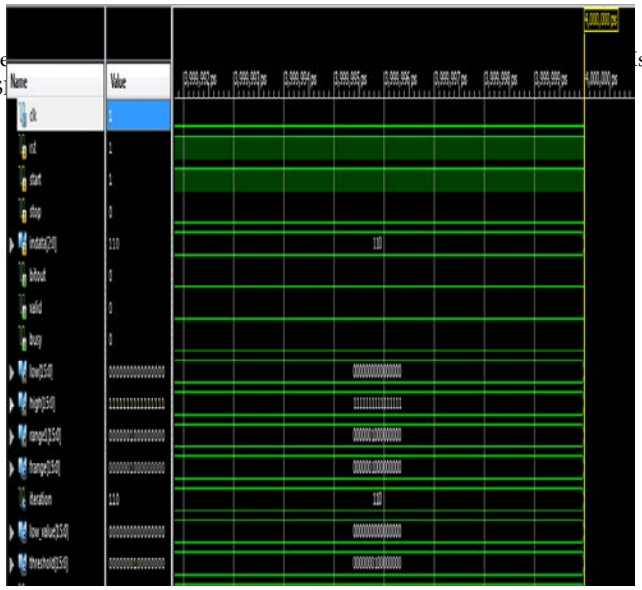

Fig9: Waveform of CABAC

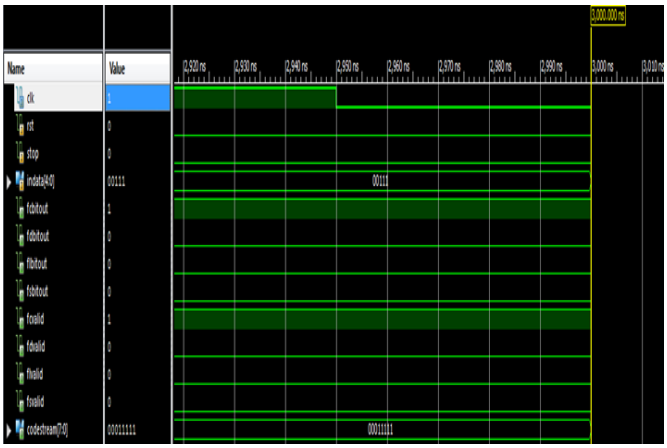

Fig10: Waveform of CABAC in SPIHT

Table 2: Comparison Table of AC and CABAC in SPIHT

PARAMETERS	AC IN SPIHT	CABAC IN SPIHT
SLICES	1446 (23% Utilization)	1215 (22 % Utilization)
4 INPUT LUTs	2712 (22% Utilization)	1904 (17 % Utilization)
NO. OF BONDED IOBs	86 (23% Utilization)	46 (19% Utilization)
CLOCK FREQUENCY(MHz)	56.404	31.395
DELAY (ns)	42.561	37.073

The selected device is 4vfx12sf363-12. Both CABAC and Arithmetic Coder had been synthesized with Vertex 4. From the table 2, the delay of CABAC is less than the delay of Arithmetic Coder. Therefore, the speed of CABAC is greater than that of Arithmetic Coder.

5 CONCLUSION

Arithmetic coding makes itself a standard technique for its high efficiency. For improvement of throughput purpose, high speed architecture of AC used in SPIHT without lists algorithm is proposed. In the architecture, a simple context scheme is used first to reduce the memory size.

The CABAC design is based on the key elements of binarization, context modeling, and binary arithmetic coding. Binarization enables efficient binary arithmetic coding via a unique mapping of nonbinary syntax elements to a sequence of bits, which are called bins. Each bin can either be processed in the regular coding mode or the bypass mode. The latter is chosen for selected bins in order to allow a speedup of the whole encoding (and decoding) process by means of a simplified nonadaptive coding engine without the usage of probability estimation. The regular coding mode provides the actual coding benefit, where a bin may be context modeled and subsequently arithmetic encoded. CABAC is one of the high powered arithmetic coding algorithm, which the essential problem is to binary transform source data. Therefore, further compression can be achieved by continually carrying out CABAC algorithm to the coding sequence coming from SPIHT algorithm based on wavelet transform. A pipeline register is inserted between context memory and the binary arithmetic decoder to reduce the critical path of the loop. The speed of the SPIHT algorithm is increased by using CABAC than using Arithmetic Coder. The coding is done in VHDL language and synthesized using Xilinx ISE 13.2 and simulated using ISim. The CABAC has a very sequential algorithm to decode the incoming bitstream. But because every slice is independently decoded, this can be done in a massive parallel way. Multicore parallelism can be used to speedup the decoding.

References

[1] K. Andra, T. Acharya, and C. Chakrabarti, "*A multi-bit binary arithmetic coding technique*," in Proc. Int. Conf. Image Process., Vancouver, BC, Canada, vol. 1, pp. 928–931 , Sep2000.

[2] F. W. Wheeler and W. A. Pearlman, "*SPIHT image compression without lists*," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Istanbul, Turkey, , pp. 2047–2050, Jun. 2000.

[3] Y. Wiseman, "*A pipeline chip for quasi arithmetic coding*," IEICE Trans. Fundamentals, vol. E84-A, no. 4, pp. 1034–1041, Apr. 2001.

[4] Detlev Marpe,, Heiko Schwarz, and Thomas Wiegand, "*Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard*," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, July 2003.

[5] M. Dyer, D. Taubman, and S. Nooshabadi, "*Concurrency techniques for arithmetic coding in JPEG2000*," IEEE Trans. Circuits Systems I, Reg. Papers, vol. 53, no. 6, pp. 1203–1213, Jun. 2006.

[6] J. Jyotheswar and S. Mahapatra, "*Efficient FPGA implementation of DWT and modified SPIHT for lossless image compression*," J. Syst. Arch., vol. 53, no. 7, pp. 369–378, Jul. 2007.

[7] A. A. Kassim, N. Yan, and D. Zonoobi, "*Wavelet packet transform basis selection method for set partitioning in hierarchical trees*," J. Electron. Imag., vol. 17, no. 3, p. 033007, Jul. 2008.

[8] M. Akter, M. B. I. Reaz, F. Mohd-Yasin, and F. Choong, "*A modified-set partitioning in hierarchical trees algorithm for real-time image compression*," J. Commun. Technol. Electron., vol. 53, no. 6, pp. 642–650, Jun.

2008.

[9] M. A. Ansari and R. S. Ananda, "*Context based medical image compression for ultrasound images with contextual set partitioning in hierarchical trees algorithm*," Adv. Eng. Softw., vol. 40, no. 7, pp. 487–496, Jul. 2009.

[10] Bibhuprasad Mohanty, Abhishek Singh & Sudipta Mahapatra, "*A High Performance Modified SPIHT for Scalable Image Compression*," International Journal of Image processing (IJIP), Volume (5) : Issue (4) : pp.390-402 , 2011.

[11] Kai Liu Xi''an, Belyaev, E. Jie Guo,"*VLSI Architecture of Arithmetic Coder Used in SPIHT*", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume:20 , Page(s): 697- 710, 2012 .