Low-Power Hardware Design of Binary Arithmetic Encoder in H.264

Ben Hamida Asma[1], Nedra Jarray[1]
[1] Laboratory of Electronic and Micro-Electronic (LAB-IT06)
Faculty of Sciences of Monastir
5019, Tunisia

Zitouni Abdelkrim[2]
[2]College of Education in Jubail,
University of Dammam,
Saudi Arabia

***Abstract*—Context-Based Adaptive Binary Arithmetic Coding (CABAC) is a well-known bottleneck in H.264/AVC, owing to the highly serialized calculation and high data dependency of the binary arithmetic encoder. This work presents a hardware architecture for the sub-module binary arithmetic encoder of the CABAC. Moreover, a clock gating technique is inserted into our design for power saving. An FPGA design of the proposed architecture can work at a frequency up to 268 MHz on Virtex 5. The suggested design can achieve 17% of power consumption saving, which allows it to be applied for low power video coding applications.**

Keywords—H.264; Binary Arithmetic Encoder (BAE); Context-based Adaptive Binary Arithmetic Coding (CABAC); clock gating

I. INTRODUCTION

In the H.264/AVC standard, two entropy encoders are defined: Context-based Adaptive Variable Length Coding (CAVLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC). The CAVLC is a low-complexity entropy coding technique based on the use of switched context-adaptively sets of variable-length codes. Compared to CABAC, The compression efficiency improvement is obtained at the cost of an inevitable complexity overhead. Software-based complexity analysis results show that switching from CAVLC to CABAC usually leads to complexity increasing by 25–30% for encoding and 12% for decoding. As an average, 30–40 cycles are required to encode a single bit on digital signal processors, so it takes thousands of cycles to encode one macroblock, which is unacceptable for real-time video coding applications [2]. Therefore, a hardware implementation of CABAC encoder is always required. However, the bit-serial nature of the CABAC algorithm and the strong data dependency between contiguous bits make it hard to improve the throughput and to parallelize the encoding process.

Hence, a lot of work has been proposed to improve the throughput of the CABAC by processing more than one bin in a single cycle. Yuan Li et al. put forward in [3] a high-throughput low-latency arithmetic encoder (AE) design suitable for HD real-time applications, utilizing a macroblock level pipeline. This design could achieve a throughput of 2~4 bins per cycle sufficient for real-time encoding. In [4], a software-hardware codesign for a whole entropy coder was suggested, which took Binary Arithmetic Encoder (BAE) module for the H.264/AVC CABAC entropy encoder as a hardware accelerator. Vagner Rosa et al. presented in [5] a hardware proposal of BAE. The throughput was improved by developing three different architectures of the renormalization step, presenting a processing rate from 0.68 to 1 bin per clock cycle. An RDO-support CABAC encoder was given by [6] and [7] to achieve the bit-rate saving of around 20 percent. In [6], an FPGA-based RISC CPU extension was proposed to accelerate the CABAC in a rate-distortion framework. This design achieved a coding speed of 1 bin per cycle and a clock frequency of 100 MHz. In [7], an efficient memory access was suggested to reduce the access frequency of the context RAM.

Most studies have mainly focused on ameliorating the throughput, but limited attention has been paid to reduce power consumption. Therefore, this paper aims to design BAE including a low-power technique. The main contributions of this paper are outlined as follows:

1) We implement a hardware of the BAE, which is the bottleneck of CABAC.

2) We further insert a low-power technique into the BAE architecture. In fact,a clock-gating technique is added into the design of a BAE sub-module, achieving reduced power consumption at a minor implementation effort.

The rest of this paper is organized as follows. Section II presents the CABAC encoding algorithm. Section III shows both encoding processes of the binary arithmetic coder and their corresponding proposed architecture .Section IV provides the FPGA synthesis results, and section V concludes the paper.

II. CABAC ENCODING ALGORITHM IN H.264

As presented in Fig. 1, CABAC encoding consists of three main functions: binarization, context modeling, and binary arithmetic coding. The binarization part permits mapping the non-binary valued syntax elements into binary symbols, also known as *bins* or a *bin string*. Then each *bin* is arithmetically coded by a regular coding engine or a bypass coding engine. In the regular coding engine, a context model is used to encode each *bin*. In the bypass encoding engine, the context is not needed due to the equivalent probability of the appearance of these bins.

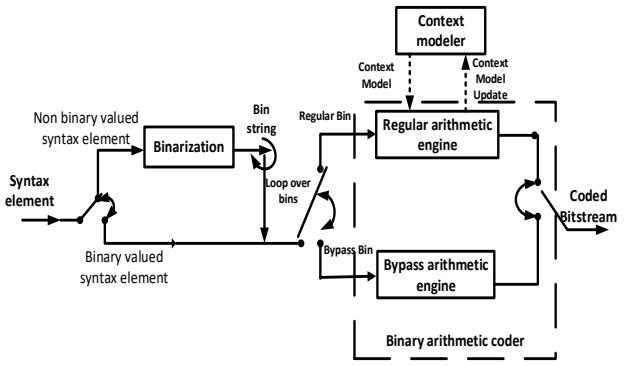

Fig. 1. Diagram of the CABAC encoder.

A. Binarization

In the binarization process, each syntax element is converted into a bin string. This step is done with different schemes: unary, truncated unary, fixed length and parameterized exp-Golomb. Each task is dedicated to some types of syntax elements, as given in Table 1. The input and output of the binarization process are the mapped syntax elements and the Context Index (CtxIdx) information. The next step is to use the CtxIdx information to fetch the context model from the context table.

TABLE. I. SYNTAX ELEMENTS AND ASSOCIATED TYPES OF BINARIZATION [1]

Syntax element	Binarization Method
mb_type	*Table mapping*
mb_skip_flag	*Fixed length*
Sub_mb	*Table mapping*
Ref_indx_10	*Unary*
Ref_indx_11	*Unary*
mvd_10	*Truncated unary and exp-Golomb with =3 ,truncated value9*
mvd_11	*Truncated unary and exp-Golomb with =3 ,truncated value9*
Intra4x4_pre_mode	*Fixed_length*
rem_intra_4x4_pre-mode	*Fixed_length*
Chroma_pre_mode	*Fixed_length*
Coded_block_pattern	*Fixed length and truncated unary,truncated value 2*
Mb_qp_delta	*Unary and table mapiing*
Coded_block_flag	*Fixed_length*
Significant_coefficient_flag	*Fixed length*
Last_significant_flag	*Fixed length*
Coeff_abs_level_minus1	*Truncated unary , exp-Golomb with =3 and truncated value 14*
Coeff_sig_flag	*Fixed length*
End_slice_length	*Fixed_length*

B. Context modeling

A context model is a probabilistic model with a statistical occurrence rate for each symbol, such that each type of syntax elements has a set of 399 context models as defined by the H.264 standard documentation [1]. Each context model comprises 6-bits representing the Probability State Indices (pStateIdx) and a 7th bit representing the value of the Most Probable Symbol (MPS).

C. Arithmetic coding

The aim of the arithmetic encoding process is to generate a bit stream from reading the bins and their context models, if the latter exist. Its principle is based on the division of an initial interval into two sub-intervals according to the context model (Fig. 2). One of two sub-intervals corresponds to the MPS, and the other refers to the Less Probable Symbol (LPS). After that, one of the two intervals is selected asa new one according to the bin value (MPS or LPS). Each interval is defined by two values: range (the length of the interval) and low (the bottom of the interval). These rules determine the updated value of the interval as follows:

If bin =LPS
New range= rLPS (range of LPS)
New low= low
If bin= MPS
New range = range – rLPS=rMPS (range of MPS)
New low = low + rLPS

Where, the value of *rLPS* is indexed by *pStateIdx*, read from context modeling.

For the bins that have the same probability, no context model is needed; and the bins are coded by a simpler bypass coding engine within a CABAC module.

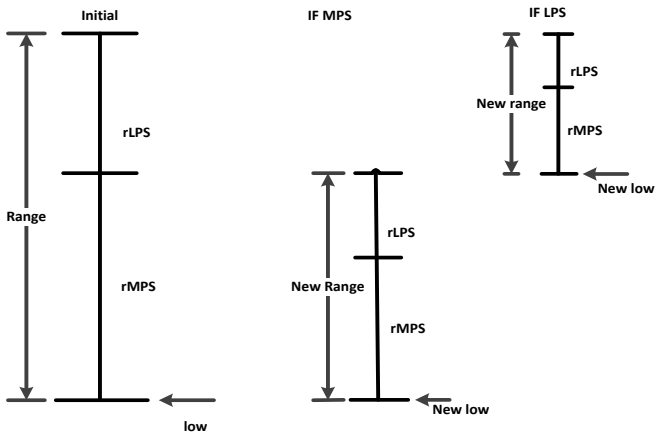

Fig. 2. Interval subdivision process of CABAC.

III. PROPOSED HARDWARE ARCHITECTURE OF BINARY ARITHMETIC CODER

At the binarization process, the syntax element of each MB can be treated in parallel. However, at the binary arithmetic coding process, all bin strings should be encoded sequentially. Thus, the binary arithmetic coder is the critical block that affects the throughput. This section firstly presents the processing flow of both regular and bypass BAE modes, and then its provides their corresponding hardware architectures. The clock gating technique is also presented in this section.

A. Regular BAE process and its proposed architecture

1) Regular BAE process

The regular BAE process is illustrated in Fig. 3. The chart consists of three steps: interval subdivision, probability-model updating and regular renormalization.

In the interval subdivision, the interval value is updated according to whether the current input bin (binval) is an MPS or not. The probability model pStateIdx is updated through two tables: TransIdxLPS and TransIdxMPS. The TransIdxMPS is selected when the bin value is equal to an MPS. Otherwise, the TransIdxLPS is used. The final update for low and range values is done by the regular renormalization process, which is needed to keep the interval range between 256 and 512. Fig. 4 shows the flowchart of the regular renormalization.

2) Regular arithmetic coder architecture

The hardware design of a regular BAE is depected in Fig. 5. It consists of three main modules: probability-model updating, interval subdivision module, and regular renormalization.

The module of probability-model updating is constituted by three principal steps: context model read, context model update, and context read. When the context is read out, the context model will be updated according to bin value through the ROM of either TransIdxLPS or TransIdxMPS. Next, the new context model will be written back to the context table.

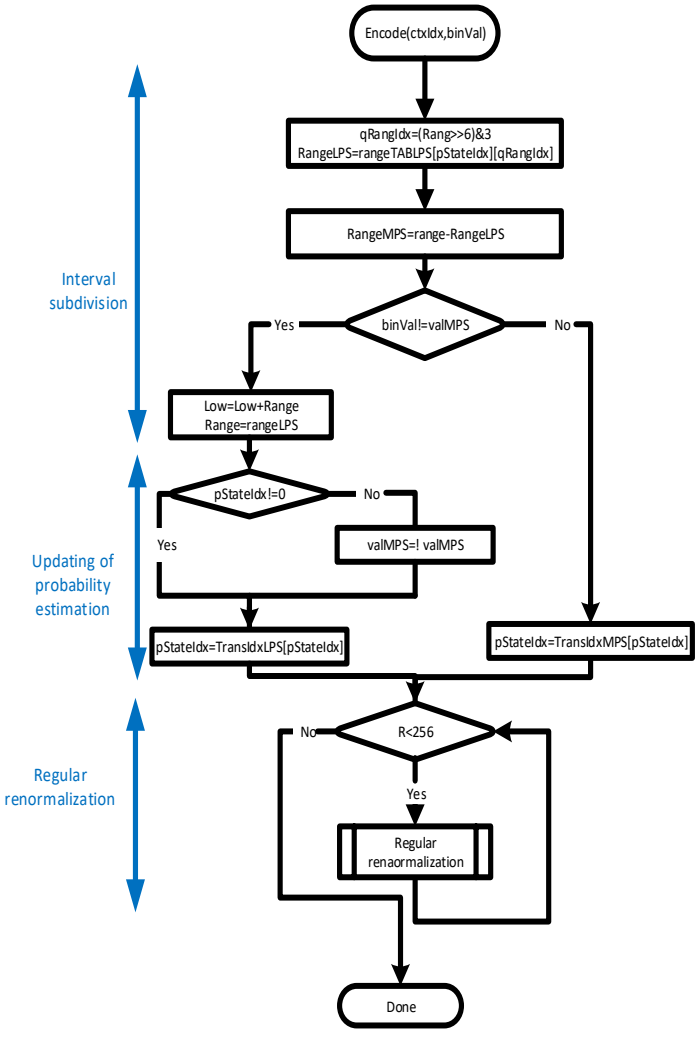

Fig. 3. Regular arithmetic encoding flowchart (from [1] with some modifications).

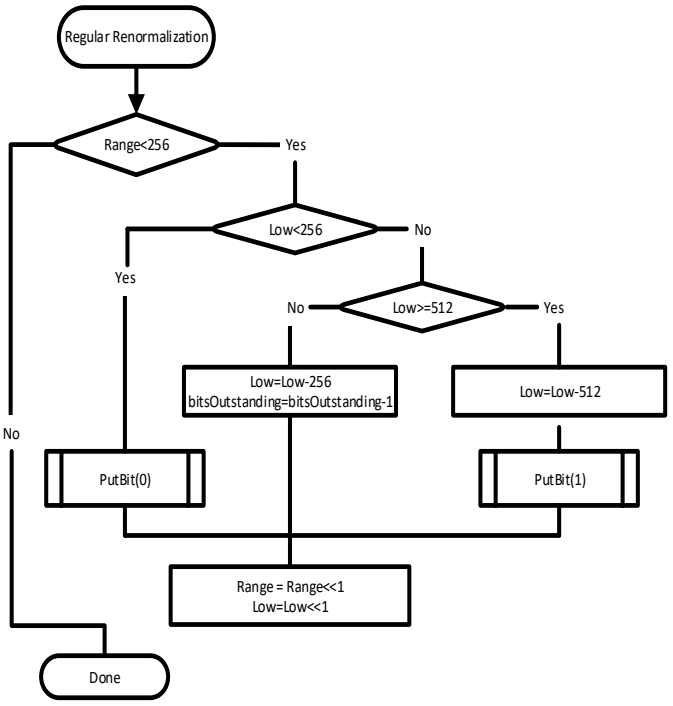

Fig. 4. Regular renormalization flowchart [1].

The module of interval subdivision will be performed when the context model *pStateIdx* is read from the RAM of the context table. Both *pStateIdx*[5:0] and *range*[7:6] are used to index the *rLPS* value from the *rLPStable*. After that, the interval values (*range* and *low*) are calculated by using a ten-bit adder and ten-bit subtractor. According to the bin value, the two top and low multiplexers will select the appropriate value of *low* and *range*, respectively.

The module of regular renormalization will be carried after encoding each bin, when the range value is decreased to less than 256. This module is implemented by a finite state machine.

IV. Bypass BAE Process and its Proposed Architecture

A. Bypass BAE process

For the bypass mode, the bin is coded using a coding decision process. The context modeling is skipped as the bins show almost an equiprobable behavior. This encoding mode is a much simpler encoding process compared to the regular mode. Fig. 6 illustrates the bypass process, including the interval subdivision stage and the renormalization stage. There is no iteration loop in the renormalization process in the bypass mode unlike the renormalization in the regular coding mode.

B. Bypass BAE architecture

A hardware design of the bypass BAE is depicted in Fig. 7. Bypass coding generates valid coding states that conform to equations shown in the flowchart of Fig. 6. This mode is faster than the regular mode since there is no context modeling process. In addition, it is to note that the there is no loop presented in bypass renormalization module. This latter is implemented by a simple finite state machine.

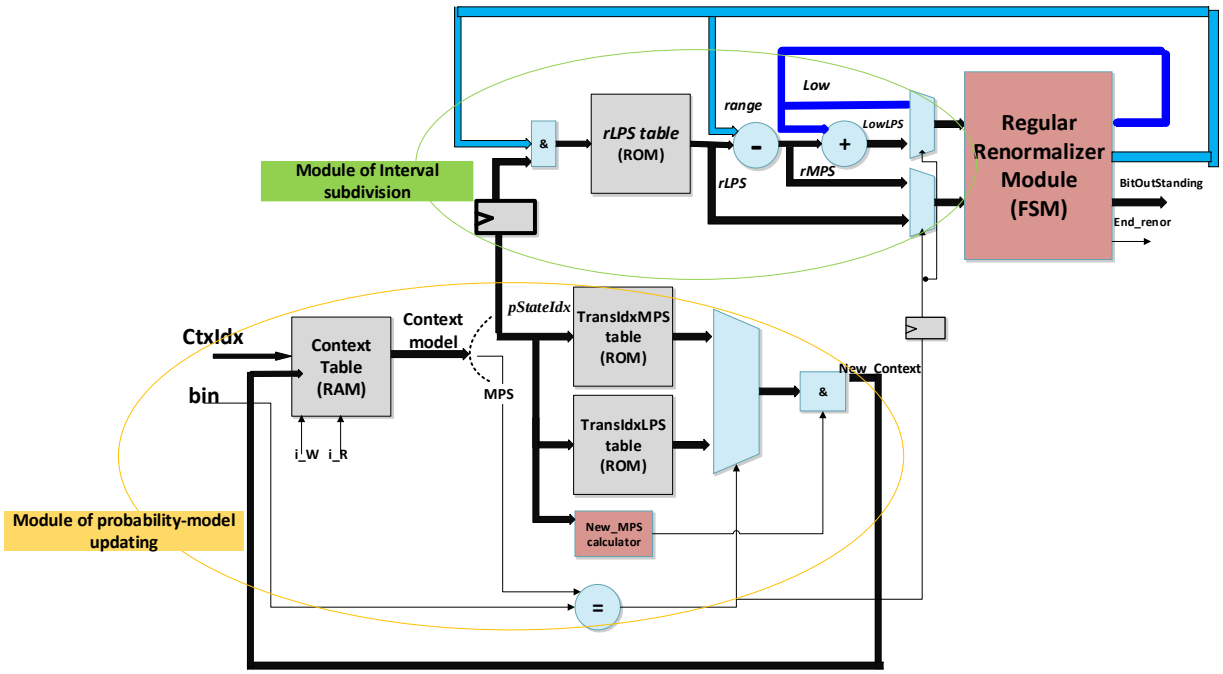

Fig. 5. Architecture representation of regular BAE.

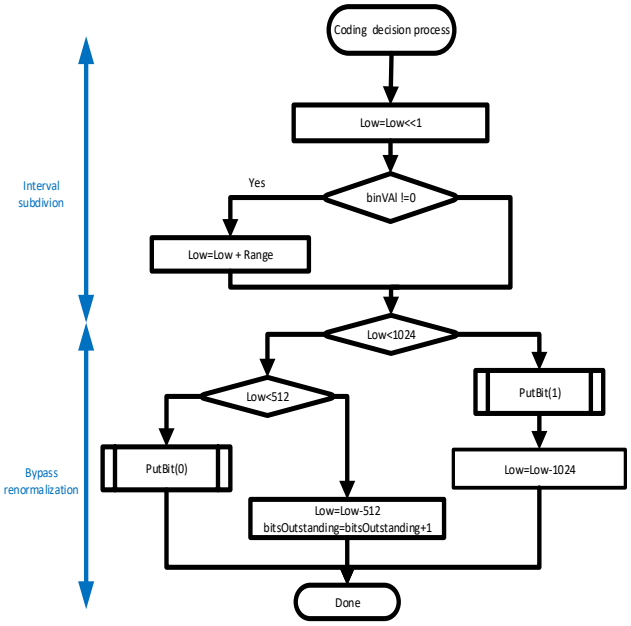

Fig. 6. Bypass BAE flowchart [1].

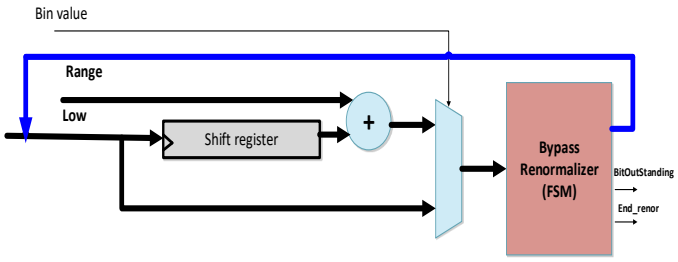

Fig. 7. Architecture representation of bypass BAE.

V. Clock-Gating Technique

Clock gating is among the techniques that are used for reducing dynamic power dissipation. This technique saves power by taking the enable conditions attached to registers and uses them to gate the clock.

At each input bin coming from the binarizer, one of the two coding modes (regular or bypass) is selected. The clock gating technique is inserted to prune the clock either for a regular arithmetic engine or for a bypass coder (i.e. by disabling the flip-flop registers in them).

The practical approach to insert the clock-gating technique in our proposed arithmetic coder is shown in Fig. 8. To avoid the glitch problem caused by clock switching, we use a latch-based clock-gating style.

VI. Implementation Results

Our design is synthesized and simulated by using the *XILINX ISE* and *ModelSim* tools, respectively. The synthesized circuit area of each component of the encoder is listed in Table 2. Synthesis results demonstrate that the BAE can work properly at a clock frequency of 268.5 MHz.

The design occupies 300 slices of which a regular BAE unit occupied 82%. It is to clear that the bypass BAE operates at a higher clock frequency compared to the regular mode.

Table 3 presents a comparison with previous work. Our design uses a higher frequency compared to the work [5], which was implemented in the same FPGA technology. Moreover, it is evident that the proposed architecture will achieve the lowest power consumption relative to power consumption of [7] when it is designed on ASIC-based technology. Indeed, as explained in [9], [10] and [11], the power consumption of ASIC designs was observed as being between 3 to 10 times greater than FPGA designs.

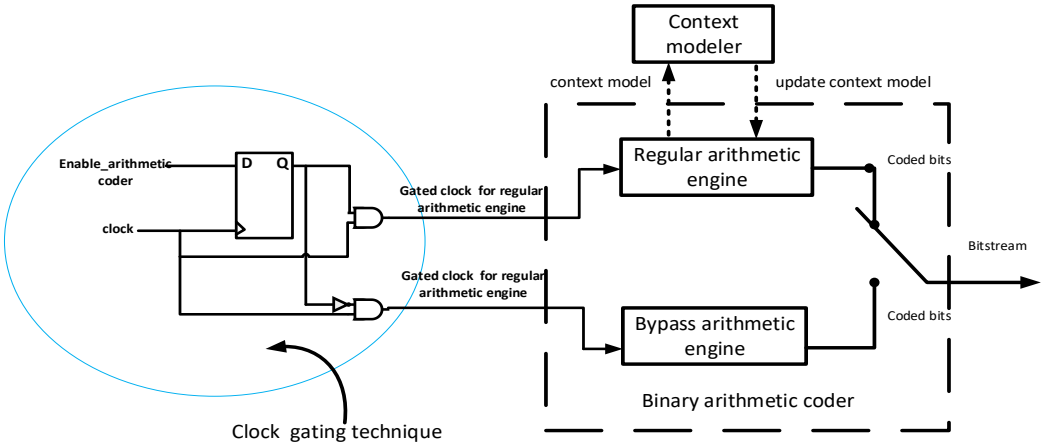

Fig. 8. Diagram of BAE with clock-gating technique.

TABLE. II. SYNTHESIS RESULTS OF EACH BAE UNITS ON VIRTEX 5

Unit Name	Area (slices)	Frequency (MHz)
Regular BAE	247	268.516
Bypass BAE	51	417.81
Total BAE (without clock gating)	298	268.516
Total BAE(with clock gating)	300	268.516

TABLE. III. COMPARISON OF PERFORMANCE RESULTS

	Process technology	Clock frequency (MHz)	Circuit Area (LUT slices)	Total power (mW)	Design parts
[5]	Virtex 5	189	436	Na	BAE
[6]	Startix II	130	603	Na	Total CABAC
[7]	Virtex4 FPGA	145	2559	Na	Total CABAC
	ASIC 0.13 µm	200	Na	26.6	
[8]	ASIC 0.15 µm	333	13.3K gates	Na	Total CABAC
Proposed	Virtex4 FPGA	219.479	298	43	BAE
	Virtex5 FPGA	268.516	300	17.77	

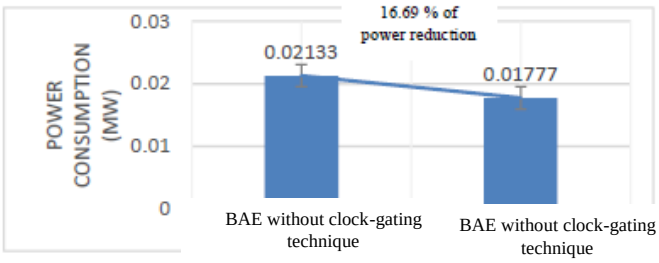

Fig. 9. Diagram of dynamic power consumption of our proposed BAE

Fig. 9 shows the power consumption for both designs (BAE without clock gating and BAE with clock gating). With the insertion of a clock-gating technique, there is about 17% of dynamic power consumption reduction.

VII. CONCLUSION

In this paper, our design has focused on the BAE that presents the critical sub-block of the CABAC. Furthermore, a clock-gating technique has been employed to reduce the power consumption. As a result, power consumption can be reduced by about 17%. Therefore, our design can be suitable for low power video coding applications. The synthesis results on Virtex 5 have indicated that the design is capable of operating at 268.516 MHz. Finally, it is important to mention that our BAE can fit both H.264/AVC and HEVC formats.

REFERENCES

[1] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Draft ITU-T Recommendation and final draft in- ternational standard of joint video specification ITU-T Rec. H.264/ISO/IEC 14496-10 AVC, May 2003.

[2] Yu, Wei, and Yun He. "A high performance CABAC decoding architecture." IEEE Transactions on Consumer Electronics 51.4 (2005): 1352-1359.

[3] Y. Li, S. Zhang, H. Jia, X. Xie, and W. Gao, "A high-throughput low-latency arithmetic encoder design for HDTV," in IEEE International Symposium on Circuits and Systems, 2013, pp. 998-1001.

[4] R. R. Osorio and J. D. Bruguera,"A New Architecture for Fast Arithmetic Coding in H.264 Advanced Video Coder", 8th Euromicro Conference on Digital System Design (DSD'05)

[5] V. Rosa, L. Max, S. Bampi," High Performance Architectures for the Arithmetic Encoder of the H.264/AVC CABAC Entropy Coder", Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on

[6] Y.L. Nunez-Yanez, V.A. Chouliaras, et al., "Hardware-assisted rate distortion optimization with embedded CABAC accelerator for the H.264 advanced video codec", IEEE Trans. CE, vol. 52, no. 2, pp. 590-597, May 2006.

[7] X.H. Tian, T.M. Le SM-IEEE, X. Jiang, Y. Lian SM-IEEE ," A HW CABAC Encoder with Efficient Context Access Scheme for H.264/AVC", 2008 IEEE International Symposium on Circuits and Systems

[8] C. Jian-long, L. Yu-kun and C. Tian-sheuan, member, ieee,"a low cost context adaptive arithmetic coder for h.264/mpeg-4 avc video coding",ieee international conference on acoustics, speech and signal processing, 2007. icassp 2007

[9] A. Amara, F. Amiel, T. Ea, "FPGA vs. ASIC for low power applications", Microelectronics Journal, Vol. 37, p. 669–677, January 2006.

[10] A. Chang and W. J. Dally. Explaining the gap between ASIC and custom power: a custom perspective. In DAC '05, pages 281{284, new york, ny, usa, 2005. acm press.

[11] D. G. Chinnery and K. Keutzer. Closing the power gap between ASIC and custom: an ASIC perspective. In DAC '05, pages 275, 280.