Novel Multiple Bypass Bin Scheme and Low-power Approach for HEVC CABAC Binary Arithmetic Encoder

Fábio Luís Livi Ramos[1,3], Bruno Zatt[2], Marcelo Schiavon Porto[2], and Sergio Bampi[1]

[1] PPGC UFRGS, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
[2] UFPel, Universidade Federal de Pelotas, Pelotas, Brazil
[3] Unipampa, Universidade Federal do Pampa, Bagé, Brazil
e-mail: fabioramos@unipampa.edu.br

Abstract² HEVC is one of the most recent video coding standards, designed to face a new age of video processing challenges, such as higher video resolutions and limited traffic share bandwidth. The HEVC standard is divided into multiple steps, whereas the entropy encoding is the final stage before the coded bitstream generation. The CABAC (Context-Adaptive Binary Arithmetic Coding) is the sole algorithm used for the entropy encoding at HEVC, providing reduced final bitstream generation, at the cost of increased computational complexity and difficulties for parallelism opportunities. One of the novelties of the CABAC for the HEVC is the increase of a certain type of input data (called bins), which have smaller dependencies among them (i.e., bypass bins), thus leading to the possibility to process multiples of them in parallel at once. The present work introduces a novel scheme for multiple bypass bins processing, leading to increasing bins-per-cycle throughput compared to related works. Moreover, the new technique is suitable for achieving a BAE (Binary Arithmetic Encoder) architecture (the CABAC critical part) able to process real-time 8K UHD videos. Along with the multiple bypass bins technique, a low-power approach is presented, based on statistical analysis of the recommended test video sequences, accomplishing around 14% of power savings.

Index Terms² HEVC, CABAC, BAE, bypass-bins, low power.

I. Introduction

Video processing has been of increasing research interest, due to the current and upcoming demands related to the topic. Video streaming companies, such as Netflix, YouTube, HBO, among others, have already reached 50% of all internet traffic share [1], showing the motivation for constant upgrades into video coding tools and standards.

The HEVC (High-Efficiency Video Coding) [2] is one of the most recent video coding standard available, able to compress a video by twice the capability of its predecessor (the H.264/AVC), keeping the same visual quality [3].

The HEVC standard has a set of tools and steps that a video go throughout the encoding process, where the final step is the entropy encoding, using the CABAC (Context-Adaptive Binary Arithmetic Coding) [4] the algorithm used for that purpose. Many recent works have focused on CABAC as a whole, or solely in its critical step, the BAE (Binary Arithmetic Encoder) [5-8].

The increasing compression capabilities of CABAC come at the cost of difficulties in order to parallelize the incoming data, which receive the name bins, and are derived from the data generated at the previous encoding stages of the HEVC standard [2]. The bins are of two main types, which are regular and bypass (there are also the terminate bins, which occur rarely) [2]. The bypass bins have fewer data dependencies among them when they undergo processing, and thus one of the improvements of the HEVC is the increase in the total amount of bypass bins, along with the grouping of this type of bin [9].

The proposed work presents a novel scheme for multiple bypass bins processing, named Multiple Bypass Bins Scheme (MBBS). Moreover, the proposed method application into a baseline BAE architecture happens, showing an increasing bin per cycle throughput. A low-power approach introduction and insertion take place into the same BAE architecture, already with the MBBS, achieving significant power savings. Both proposals (MBBS and low-power approach) guidelines is a statistical analysis made prior to the conception of the techniques.

The text division is as follows: Section II introduces the main CABAC concepts required for the work and the related most recent CABAC and BAE architectures. Section III presents the statistical analysis made for the assessment of throughput increase and low-power opportunities into the BAE. Section IV shows in details the MBBS and the background for its inception. The low-power approach proposal and its insertion into a baseline BAE architecture, along with the MBBS, appears in Section V. The simulation and synthesis results, along with the relevant comparisons with related works of the proposed BAE architecture are shown in Section VI. Section VII concludes the paper.

II. CABAC Algorithm and Related Works

A. CABAC Algorithm

A CABAC is an algorithm based on subinterval recursive division [4]. It works based on certain variables that define an interval to be used, which are divided according to the input data, and then the new value of the variable is the chosen interval to be used for the next round of the algorithm.

The CABAC is divided into three main parts: Binarization (Binarizer), Context Modeling, and Binary Arithmetic Encoder (BAE), as depicted in Fig. 1. The Binarization

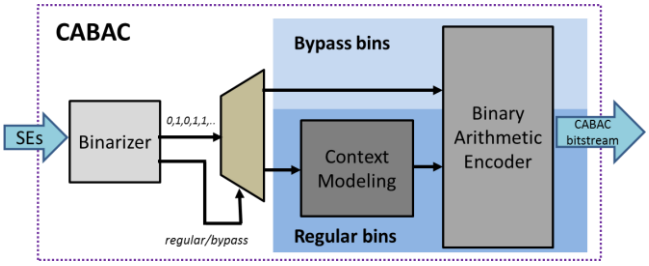

Fig. 1 CABAC sub-blocks

transforms the Syntax Elements (SE) (i.e., the data coming from the previous HEVC encoding steps) into a different binary representation, which from that point on will be called bins [2]. One may notice that a single bin value is either '0' or '1'.

The bins are of two main types: regular and bypass (there is also the terminate bins, seldom occurring during the processing), where only the regular bins shall undergo the Context Modeling (i.e., the probability of the future occurrence of a bin depends on the current bin processed). The regular bins are divided into two categories: the Most Probable Symbol (MPS) and the Least Probable Symbol (LPS), which values vary according to the type of SE from where they were generated [2]. Bypass bins are considered to be equiprobable (i.e., both bin values have the same probability to occur), and thus do not need the adjustment of the Context Modeling.

At the final step, the BAE processes all the types of bins, using the already mentioned subinterval recursive division, considering two main control variables: Range and Low. The Range definition for an LPS regular bin occurs via a Context Modeling variable called State, and it is called rLPS. The Range for an MPS is called rMPS and depends on the definition of the rLPS. The fashion, which all the aforementioned variables are updated, are according (1) for regular bins, and (2) for bypass bins. As one may notice, Range variable remains unchanged for bypass bins.

$$\begin{cases} rMPS \leftarrow Range - \text{rLPS} \\ Range \leftarrow rMPS \text{ and } Low \leftarrow Low, & \text{for bin = MPS} \\ Range \leftarrow rLPS \text{ and } Low \leftarrow Low + rMPS, & \text{for bin = LPS} \end{cases} \quad (1)$$

$$\begin{cases} Range \leftarrow Range \\ \text{Low} \leftarrow 2*Low, & \textit{for bin = '0'} \\ \text{Low} \leftarrow 2*Low + Range, & \textit{for bin = '1'} \end{cases} \quad (2)$$

A critical remark for the Range and Low is the fact that they may require renormalizations in case they fall below specific values, or for suitability to fit into the representation width of the variables (i.e., Range is a 9-bit variable, whereas Low is a 10-bit variable). Considering that bypass bins are the focus of the present work, Fig. 2 presents the pseudo-code of renormalization for the Low variable in case a bypass bin occurred [2], immediately following the update in (2). Another important remark is that the renormalization requires an extra special control variable called Outstanding Bits (OB) and that Low renormalization is what defines the bitstream generation for either regular and bypass bins. The symbol $\{1,0^{OB}\}$ represents a concatenation of a value '1', followed by the value '0' n times, whereas $\{0,1^{OB}\}$ corresponds to the concatenation of the value '0' followed by the value '1' n times. In either situation, n is the current value of the OB variable. When no bitstream generation happens, it appears as null in Fig. 2.

```
1.  if (Low >= 1024) then                    ┐
2.      bitstream ← {1,0^OB};                │ 1st condition
3.      OB        ← 0;                       │
4.      Low       ← Low – 1024;              ┘
5.  else if (Low < 512) then                 ┐
6.      bitstream ← {0,1^OB};                │ 2nd condition
7.      OB        ← 0;                       │
8.      Low       ← Low;                     ┘
9.  else                                     ┐
10.     bitstream ← null;                    │ 3rd condition
11.     OB        ← OB + 1                   │
12.     Low       ← Low – 512;               ┘
```

Fig. 2 Low renormalization conditions for bypass bins

For the bypass renormalization, Fig. 3, Fig. 4, Fig. 5, and Fig. 6 present some possible behavior the variable Low may follow, based on (2) and Fig. 2, where one may notice that the upper limit for the Low variable is 767 [10]. One important remark is that the Range value must always be above 256 after its renormalization [2] (what implies that the ninth most significant bit of this variable has the value '1' when it is used for Low renormalization). One may see that the Low variable always starts with value zero [2], and is incremented with the Range variable, as shown in (1) and (2). Therefore, at the end of a given Low renormalization, we cannot have both Low[9] and Low[8] with the value '1' (as already mentioned, the upper limit for Low is 767).

For instance, Fig. 3(a), Fig. 3(b), Fig. 4(a), and Fig. 4(b) start with different Low values, highlighting the four most-significant bits, and the posterior renormalization needed, considering only that a bypass bin with value '0'

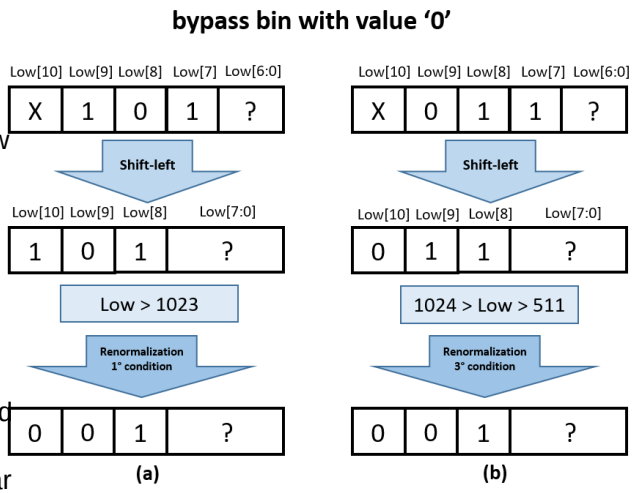

Fig. 3 Low renormalizations for bypass bin with value '0' – part 1

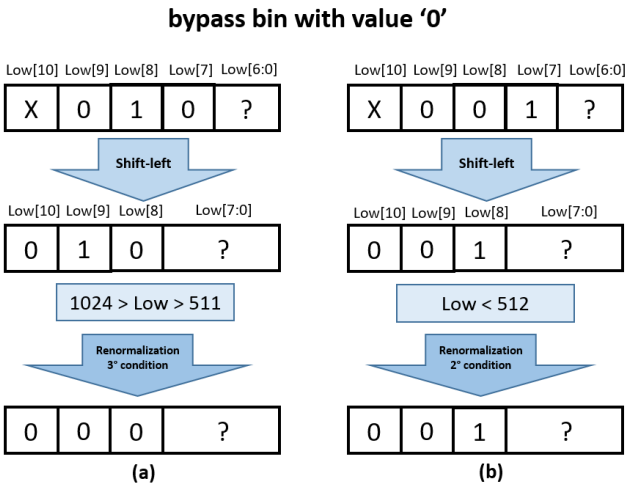

Fig. 4 Low renormalization for bypass bin with value '0' – part 2

has occurred. One may notice that a bypass bin with value '0' implies that a simple shift-left is required at the variable, i.e, multiplication by two. For all the cases shown, after the update of the Low, the Low[9] is already zero or must be zeroed (i.e., a subtraction by 512), leading to the results presented, wherein all cases, we do not have to set Low[9] and Low[8] with the value '1'.

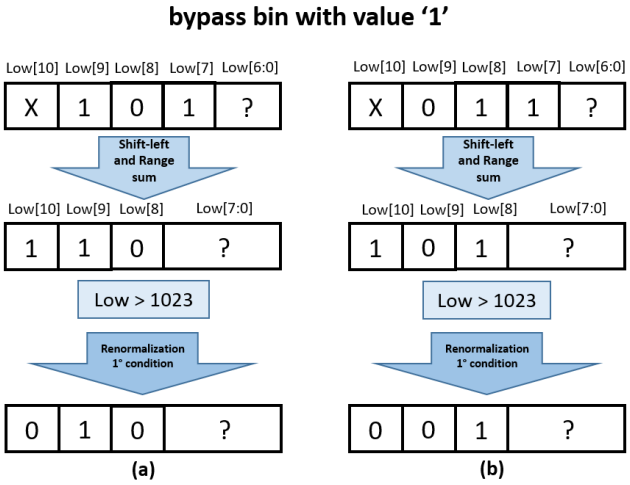

Fig. 5 Low renormalization for bypass bin with value '1' – part 1

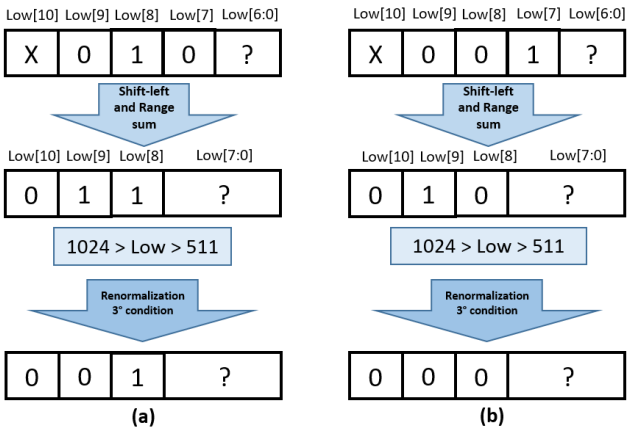

Fig. 6 Low renormalization for bypass bin with value '1' – part 2

When the current bypass bin has the value '1', Fig. 5(a), Fig. 5(b), Fig. 6(a) and Fig. 6(b) show the behavior of the Low variable. At first a shift-left is required (i.e., multiplication by two) and then a sum with the current renormalized Range (implying that the current Range value is at least 256, i.e., Range[8] has the value '1'). For all situations, we would end with the final renormalized Low with Low[9] equal to zero (either already having the value '0', or needing to be zeroed). The exception is the condition presented in Fig. 5(a), where the initial value may be the upper limit (i.e., Low[9] equal to '1', Low[8] equal to '0', and Low[7] equal to '1'), where the Low[9] must remain with value '1', even after the renormalization process.

B. Related CABAC works

There have been many works related to CABAC block in the recent years, either for H.264/AVC or HEVC standards, focusing on methods and tools to increase the throughput of the CABAC (especially on the BAE, since it is the critical step of the entropy encoding [8]). One-cycle renormalization method, increase of the number of bins processing per cycle using many pipelined BAE cores architecture, or completely splitting the flows for regular and bypass bins within the CABAC block, are some of the proposed methods.

The work of [5] was done considering the context of the H.264/AVC, where a 2-bin/cycle architecture is proposed, being the first to introduce a one-cycle renormalization for all the control variables (i.e., Range and Low) for all the main types of bins (regular and bypass), accomplishing a throughput of 634 Mbins/s.

The first work to achieved around 1 Gbin/s of throughput is [6], where the first proposing of a four-stage pipeline BAE core is presented, where the first stage is the LPS pre-selection, the second stage for Range calculation, the third for Low and OB calculation, and the fourth and final stage for final bitstream generation. Moreover, the authors proposed to append four of these BAE cores one another in a pure combinational fashion, to achieve the ~4 bins/cycle throughput, being the Range update stage the critical path of the architecture.

The work of [7] proposed some techniques in order to decrease the aforementioned critical path of [6], allowing asymmetric BAE cores that process only LPS bins, thus removing the requirements for some adders, as may be seen at (1), and pre-renormalization of rLPS (PN rLPS). Furthermore, the flow for regular and bypass bins is split until the Low update stage. Thus, the authors of [7] accomplished around 1.7 Gbins/s of throughput.

The follow-up of [7] is presented at [8], where along with the techniques aforementioned, a look-up table pre selection of the Range value when LPS bins occur (LH rLPS) is proposed, allowing an increase in the maximum frequency achieving around 1.8 Gbins/s of throughput, being the highest throughput up to date found for the CABAC block.

III. Statistical Analysis of HEVC Video Sequences

The occurrence of the types for the majority of the bins (regular or bypass) may vary according to the properties of a given video sequence. Moreover, regular bins are divided into LPS and MPS, which would imply that MPS bins tend to occur more often than the LPS. A statistical analysis of some recommended video sequences [11] running on the HEVC reference software [12] is a possible path to verify the statistical occurrence of the bins for better guidance of what possibilities we may follow for the hardware design of the CABAC/BAE.

The first thing to notice is that regular bins are expected to occur more often than bypass, which [8] already states. The percentage of MPS bins is higher than LPS, but bypass bins tend to happen more often than LPS [8].

For the current work, we are interested in finding if the bins also tend to occur grouped (i.e., if they occur in bursts), and what is the amount of bins grouped. The two reasons for that are:

- For bypass bins, to confirm the amount of grouped bin of these types and establish the possible efficiency of a multiple bypass bins scheme before its implementation.
- For low-power opportunities, to verify the amount of grouped bins of each type and thus turn-off possible parts of the architecture that are required only for the other kinds of bin beside the one occurring at a given moment.

In order to verify the demanded variables, five high-resolution recommended video sequences were used, applying two recommended configurations: Low Delay (LD) and Random Access (RA); and the top and bottom Quantization Parameters (QP) recommended (i.e., respectively 37 and 22). The sequences were run for the initial frames, due to limitations in the memory to save the results. Nevertheless, the results will point out valid indications of the statistical behavior for random videos.

The results for these sequences are presented in Fig. 7, Fig. 8, Fig. 9 and Fig. 10. Each figure displays the average amount of bins in a row for each of the types, discriminated into regular, bypass, regular MPS, and regular LPS, respectively.

The bursts of regular bins (see Fig. 7) tend to have around 11.87 bins in a row (i.e., when a regular bin occurs, the next eleven bins tend to be regular). Discriminating by QPs, by using the upper value (i.e., 37), we have around 9.4 regular bins occurring in a row, whereas for the bottom value (i.e., 22), about 13.6 regular bins happen in a row. Considering that there may be structures required only for bypass bins, they may be turned off in case a burst of regular bins is currently occurring.

The amount of consecutively bypass bins is smaller than for regular bins. Nevertheless, the bypass bins burst is also significant (Fig. 8). There is an average of 4.55 bypass bins occurring during a burst of this type of bin. The difference of the consecutive occurrence for low and high QPs tend to be almost negligible. The exceptions are sequence Kimono and PeopleOnStreet, were the burst for the low QP are considerably higher than for the high QP (around 8 and 6, respectively). These data confirm that (i) the bypass bins do occur grouped; (ii) Since bypass bins have smaller data dependency, there is a potential to increase throughput if the processing of multiple bypass bins happens at the same time. (iii) The parts of a given CABAC/BAE architecture that are required only for regular bins may be turned off when the burst of bypass bins is happening.

Finally, regular MPS bins also tend to occur grouped (Fig. 9) among the burst of the regular type (around 3.27

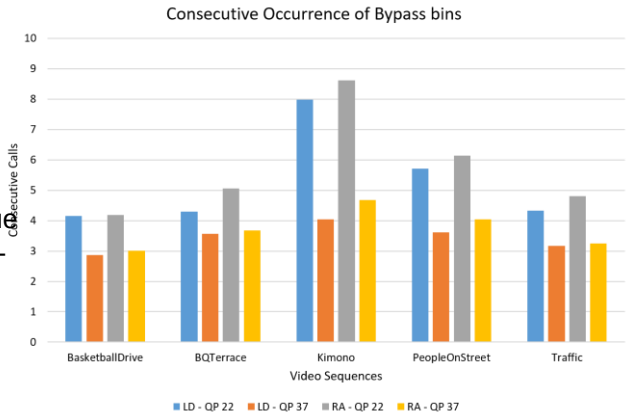

Fig. 8 Consecutive occurrence of bypass bins

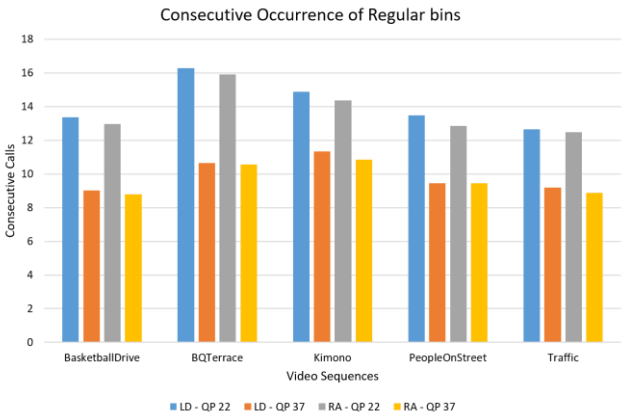

Fig. 7 Consecutive occurrence of regular bins

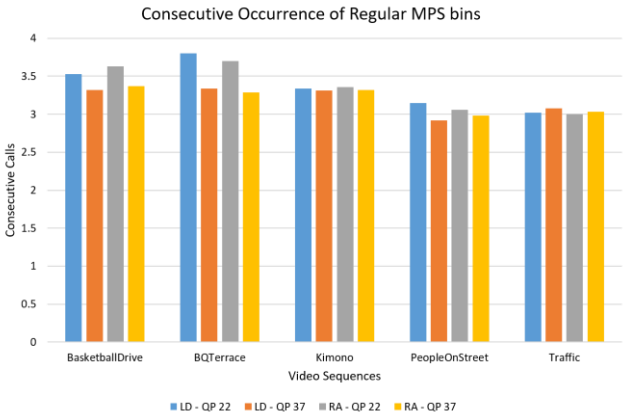

Fig. 9 Consecutive occurrence of MPS regular bins

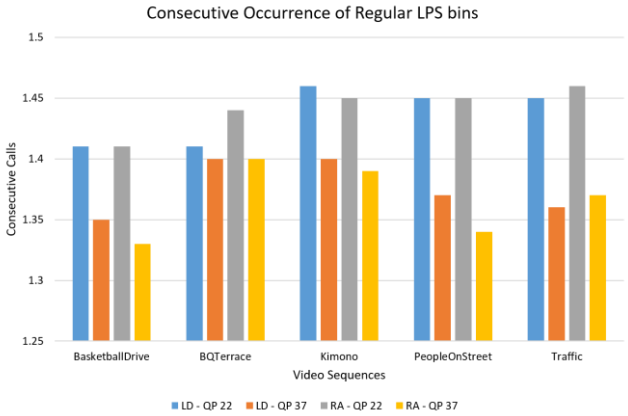

Fig. 10 Consecutive occurrence of LPS regular bins

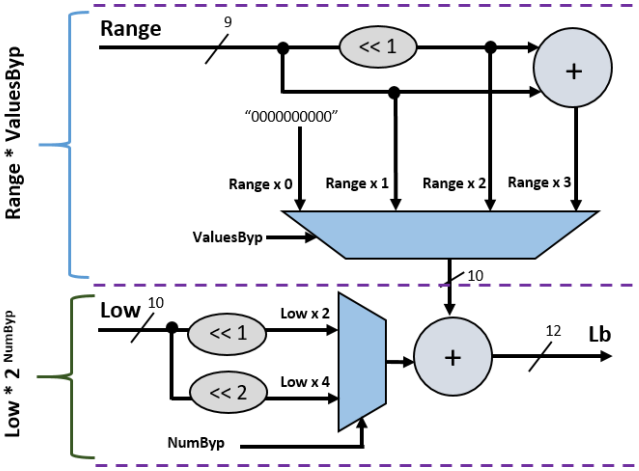

Fig. 11 Low update architecture for MBBS

regular MPS bins within the burst). As one may notice, the amount of consecutively occurring regular LPS bins is so small (approximately 1.4) that we may consider that they usually occur alone within a burst of regular bins (Fig. 10). Nevertheless, we may infer the same conclusion as for the aforementioned analysis: in case a part of the architecture is required only for LPS bins, they may be turned off during the regular MPS or bypass bursts

IV. Multiple Bypass Bin Scheme

One of the improvements of the HEVC standard compared to its predecessor is the increase in the total amount of bypass bins, since they have few data dependencies among them [9]. Furthermore, as depicted in (2), they do not update the Range variable, which is the current critical path in the most recent CABAC/BAE architectures [6][8].

One further look at (2) and Low variable update (the only variable updated by bypass bins) may be rewritten in a general fashion as (3), for one up to n bypass bins. The updated Low variable for multiple bypass bins receives the name Lb, whereas NumByp corresponds to the total amount of bypass bins we are processing at a given round, and ValuesByp corresponds to the value of the bypass bins being processed in reverse order of incoming. In case a single bypass bin is processed, (3) is collapsed back to (2).

$$Lb \leftarrow (\text{Low} * 2^{\text{NumByp}}) + (\text{Range} * \text{ValuesByp}) \quad (3)$$

The choice of the amount of bypass bins to be processed in parallel will determine the complexity of the chosen hardware implementation to be designed. We decided to process up to two bypass bins in a row in our proposal of Multiple Bypass Bin Scheme (MBBS), thus leading to the implementation of (3) as depicted in Fig. 11, in a multiplier-less fashion. Since we have at most two bypass bins, the possible ValuesByp are 0, 1, 2 or 3, and the possible NumByp are either one or two (considering that at least one bypass bin will be processed at a given moment).

Further explaining Fig. 11 related to (3), the Range multiplication by zero is a vector of zeros; the Range multiplication by one is the original Range value; a shift-left by one position achieves the Range multiplication by two; and the Range multiplication by three is the Range multiplied by two added with the original Range value. The Low multiplication will be either by two or by four (power of two values) and thus are accomplished by a shift by one and by two positions, respectively. By doing so, the multiplications at (3) are achieved by the used of adders and shift logic, as already mentioned. Moreover, as will be further explained, it avoids any increase in the critical path of the BAE logic for a multi-core BAE design.

As one may remember, the Low variable may undergo a renormalization, as already mentioned in Section II, along with the update of the auxiliary variable OB, and the generation of the bitstream for the given bypass bins. Therefore, we have to find an alternative to implement the behavior depicted in Fig. 2 for two bypass bins, according to the amount being processed at a given moment. For a single bypass bin, the proposal of [5] is the guideline followed.

For two bypass bins, the first thing to notice is that Lb is a 12-bit variable since it undergoes a multiplication by four (i.e., a shift-left by two positions), which will lead to two more positions on the original 10-bit Low variable. The nine less significant bits of Low will also remain the same (i.e., Low[8:0]). As already seen in Fig. 3, Fig. 4, Fig. 5, and Fig. 6, if we make any combination between two incoming bypass bins with any value in a row, the tenth bit (i.e., Lb[9]) will either have the value '0', or shall be zeroed. The exception occurs in case we have two bypass bin with '1' occurring in a row, and the Low initial value is above 640 (i.e., Low[9:7] is equal to '101'), requiring the 1st renormalization conditions twice. Furthermore, when a sequence of bypass bins with value '1' and '0' occurs, respectively, and the Low is again above 640, the 1st renormalization condition will be required twice. The difference is that, this time, Lb[9] will already have the value '0'. Thus, we can consider that, when the MBBS is processing two bypass bins, the final Low[9] always will be zeroed, except by the case that the 1st renormalization condition is required for both bypass bins processed, which is indicated by Lb[11] and Lb[10] having both the value '1'. The summary of MBBS Low renormalization is presented in

(4), considering two bypass bin processing only.

$$\begin{cases} Low[8:0] & \leftarrow Lb[8:0], & \\ Low[9] & \leftarrow Lb[9], & \text{if } Lb[11:10] = 11 \\ Low[9] & \leftarrow 0, & \text{otherwise.} \end{cases} \quad (4)$$

The OB variable also has to be updated, now considering that up to two bypass bins may be processed, which can be seen at (5). We have to look at the twelfth, eleventh, and tenth bit of the Lb variable to verify which of the situations presented at Fig. 2 have occurred for the first and the second incoming bypass bin to be processed in parallel (i.e., Lb [11], Lb[10], and Lb[9]). All possible situations are depicted in Fig. 12

$$\begin{cases} OB & \leftarrow 1, & \text{if } Lb[10:9] = 01 \\ OB & \leftarrow OB + 2, & \text{if } Lb[11:9] = 011 \\ OB & \leftarrow 0, & \text{otherwise.} \end{cases} \quad (5)$$

In case Lb[10:9] has the value '01', for the first incoming bypass bin, the value of the Low variable was either above 1024 or below 512, seen at Fig. 12 (a), meaning that the OB value has to be zeroed (1st or 2nd condition in Fig. 2). After the shift, the final value of Lb[10] is zero (i.e., for the second bypass bin). This means that the actual updated Lb has a value equal to or greater than 512 (but still smaller than 1024, since the Lb[10] has the value '0'), thus needing the sum of one to the current OB value (3rd condition at Fig. 2). Since it was zeroed for the first incoming bypass bin, the final updated OB value for this case is merely one.

When the twelfth, eleventh, and tenth bit of the Lb have the value '0', '1', and '1', respectively (i.e., Lb[11:9] = '011'), this means that the Lb should have had two values equal to or greater than 512, but smaller than 1024 for the first and second incoming bypass bin, as depicted in Fig. 12 (b). For the first incoming bypass bin, the value of the Lb is clearly between 1024 and 511, leading to the 3rd situation in Fig. 2. That would require a renormalization (subtraction by the value 512, i.e., zeroing the Lb[9] if only a single bypass bin has occurred). After the shift, the next value would be again between 1024 and 511 (because the new Lb[9] bit is '1', and the Lb[10] would have been zeroed if done separately for the two bypass bins). The situation leads to adding one twice to the original value of OB (i.e., the initial value of OB has to be added by two due to the 3rd condition of Fig. 2 having occurred twice).

When both values of the eleventh and tenth bit are '0', as presented in Fig. 12 (c), we cannot say whether, for the first bypass bin, the value of Lb was above 1023 or was below 512 (to discover which condition occurred, the reading of Lb[11] is required). Nevertheless, since the Lb[9] bit is '0', and Lb[10] was also '0', we can assure that the final Lb value is smaller than 512, which lead to OB to be updated with the value '0' (2nd condition of Fig. 2 for the second incoming bypass bin).

The OB variable also needs to be zero if Lb[11:9] has the value '010', depicted in Fig. 12 (d). We know that, for the first incoming bypass bin, Lb value is between 1024

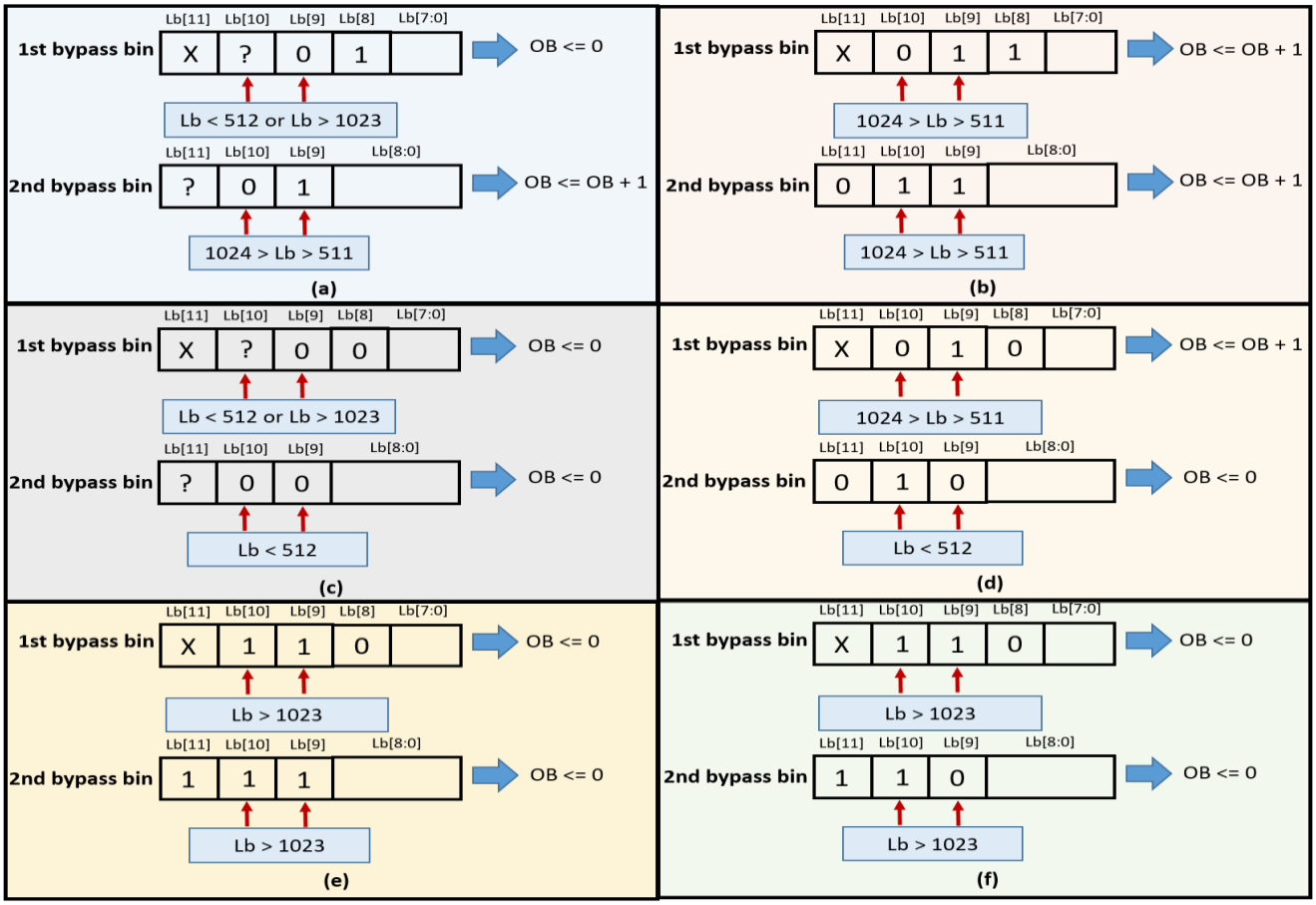

Fig. 12 OB and bitstream test conditions for BMS

and 511 (since the Lb[10] is zero). Moreover, we know for sure that its final value (i.e., for the second bypass bin) should be below 512, since the tenth bit should have been zeroed for the first bypass bin before the shift in case MBBS was not used (i.e., 3rd renormalization condition was required), leading to zeroing OB. Nevertheless, due to the usage of MBBS, the eleventh bit of Lb (i.e., Lb[10]) that should have been, otherwise, zeroed for the first bypass bin, is not, exactly because MBBS do not renormalize the values during the intermediary step of the process (i.e., in between the processing of the two bypass bins).

In Fig. 12(e) and Fig. 12(f), we have the situations where the 1st renormalization condition was required twice, which is indicated by Lb[11] and Lb[10] having both the value '1'. Therefore, the OB variable shall be zeroed for both bypass bin, receiving the value zero for these situations.

The bitstream when two bypass bins at the current round will follow what is depicted in the pseudo-code at Fig. 13. A similar notation as used in Fig. 2 is applied here: $\sim Lb[11]^{OB}$ represents the replication by OB times of the negated value of Lb[11], concatenated with the other values between the brackets.

When the tenth bit of Lb (i.e., Lb[9]) is zero, and Lb[11] is not equal to Lb[10], we know that the final renormalized Low will be below 512, for the same reasons presents for the OB update (refer to Fig. 12(c) and Fig. 12(d)). Thus, there will be bitstream generation at the current round (refer to 2nd condition at Fig. 2). If Lb[11:10] value is '01', the same situation presented in Fig. 12(a) has occurred. For the first incoming bypass bin, no bitstream generation happens; and for the second bypass bin, the bitstream will follow the 2nd condition of Fig. 2, added by one extra bit due to the previous incoming bypass bin (i.e., OB was incremented by one at the first bypass bin). If Lb[11:10] = '10', the situation presented at Fig. 12(b) has occurred (but considering that Lb[11] has the value '1') and we would have, at the beginning of the round, Low value above 1023, meaning that we would follow the bitstream behavior of the 1st condition at Fig. 2, with the original OB value before update. After that, the Low variable would end with a value smaller than 512, leading to the 2nd condition of Fig. 2. Nevertheless, since the OB was zeroed for the previous bin, a final bit '0' is put into the bitstream. The lines 1 and 2 of the pseudo-code at Fig. 13 contemplate both situations.

Considering again that Lb[9] is zero, we may have both Lb[11] and Lb[10] with the same value of Lb[9]. Therefore, this would always lead to the situation where, at the first bypass bin, the Lb value was below 512, leading to the related 2nd condition of Fig. 2 and zeroing the OB, generating a bitstream with the OB value before the update. Since when the next bypass bin occurred, the value of Lb remains below 512, we would follow again the same behavior of the previous bin (but now the OB variable was already zeroed at the previous round, generating a single final bit '0' at the bitstream). Furthermore, when Lb[11] and Lb[10] have both the value '1', as already mentioned, the 1st renormalization conditions was needed twice (i.e., the Low was equal or above to 1024 for the two bypass bins), and the same bitstream behavior presented before is applied. These situations are depicted in Fig. 12(c), Fig. 12(e) and Fig. 12 (f), and the lines 3 and 4 of Fig. 13 used for that purpose.

The other situation that leads to bitstream generation for two bypass bins happens when Lb[10:9] have the value '01' (refer to Fig. 12(a)). The first value of Lb could be either above 1023 or below 512 for the first bypass bin, leading to either 1st or 2nd condition of Fig. 2. This will lead to the generation of bitstream for that bin and zeroing the OB. After that, since for the second bypass, the value of Lb is between 1024 and 511 (3rd condition at Fig. 2), no bitstream is generated for that bin, and the OB variable receives the value one, the first condition at (5). The lines 5 and 6 of Fig. 13 contemplate that condition of bitstream generation.

The remaining situation is when Lb[9] and Lb[10] have both the value '1', and Lb[11] has the value '0' (refer to Fig. 12(b)). For that situation, we would have, for each of the two incoming bypass bins, an Lb with the value between 511 and 1024 (3rd condition in Fig. 2). Thus, no bitstream generation occurs at that round for the bypass bins, only the OB update, as according to the second condition presented at (5). The lines 7 and 8 of Fig. 13 shows how that situation is described.

For all others situations, i.e., the update of Range, Low, OB and bitstream generation for regular bins or for a single occurring bypass bin followed by a bin of another type, the behavior of the referred variables follows the same proposal of [5], as already mentioned.

```
1.  if (Lb[9] = 0) and (Lb[11] ≠ Lb[10]) then
2.        bitstream ← {Lb[11:10], ~Lb[11]^OB};
3.  else if (Lb[11:9] = 000) or (Lb[11:10] = 11) then
4.        bitstream ← {Lb[11], ~Lb[11]^OB, Lb[10]};
5.  else if Lb[10:9] = 01 then
6.        bitstream ← {Lb[11], ~Lb[11]^OB};
7.  else
8.        bitstream ← null;
9.  end if
```

Fig. 13 Pseudo-code for bitstream generation of MBBS

V. Low-power Approach for BAE architecture

Considering the statistics presents at [8], a low-power approach for the BAE is a desirable goal. Around 55% of the total bins for the related video sequences are regular MPS bins, leading to a total of 20% for regular LPS and 25% for bypass bins [8]. We know that regular and bypass bins tend to occur in bursts, according to the data presented in Section III.

One may notice looking at (1) and (2) the following: (i) Range update only occurs for regular bins; (ii) an extra

sum is required only for Low update of regular LPS bins (i.e., Low added with rMPS); (iii) Low update for bypass bins has a different logic compared to the Low update for regular bins.

Moreover, considering a baseline BAE architecture derived from [6], we have a four-BAE core structure (i.e., the paths between cores are purely combinational, where each core is composed of a four-stage pipeline structure. The pipeline stages are: (i) rLPS pre-selection with PN rLPS (ii) Range update; (iii) Low and OB update; (iv) bitstream generation. The sequential barriers between stages will have some registers, which carry the values required only for some of the situations highlighted in the previous paragraph.

The baseline single-BAE core appears in Fig. 14 (four of this core are required for the complete BAE solution), where we have highlighted each part of the architecture, which is necessary only for a particular type of bin (i.e., regular MPS/LPS bin; only regular LPS bin; or bypass bin). One significant modification compared to the original baseline work [6] is the insertion of the Low update circuit for MBBS on the design split between the 2nd stage (Range multiplication) and the 3rd stage (Low multiplication) of the architecture.

The chosen approach is to insert Clock-gating cells [13] into the pipeline registers required only for a particular bin situation, as already presented. Therefore, no clock switching will occur when the value of given registers is not necessary to be transmitted to the next stage (avoiding unnecessary short-circuit dynamic power consumption). The gated registers required only for the regular bin (either MPS or LPS) are rLPS pre-selection registers (1st to 2nd stage), the shift-left amount for Low update (2nd to 3rd stage), and updated Low value for regular bitstream generation (3rd to 4th stage). The gated registers, which are required only when a regular LPS occurs, are: rLPS pre-renormalization values, the Range will be updated with one of these values – refer to (1) (1st to 2nd stage), and the rMPS for Low update of an LPS bin – refer to (1) (2nd to 3rd stage). Finally, for bypass bins, the registers required are: the updated Range for the MBBS (2nd to 3rd stage), and the updated Low value for bypass bitstream generation (3rd to 4th stage). This technique will be effective considering that for many cycles, the chosen registers are not necessary, which confirmation comes from the analysis at Section III.

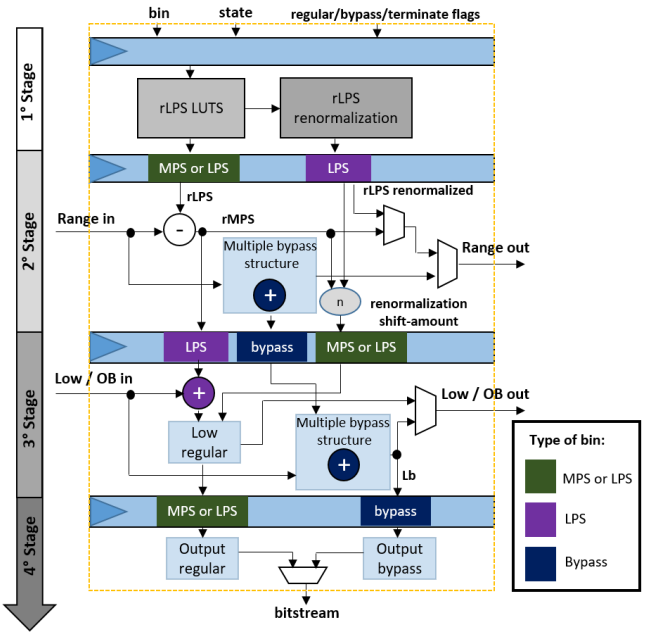

Fig. 14 Single BAE core with MBBS and low-power approach

The other low-power approach is to insert an Operand Isolation logic [14] at the combinational logic and paths (considering that we have combinational paths through the four BAE-core structure, where adders are within those paths). The Operand Isolation consists in keeping the inputs of a given logic with the values unchanged, thus leading to no switching from that point on (i.e., avoiding capacitance charging/discharging dynamic consumption). The adders are the primary target for that approach, where the adders required for the MBBS presented in Fig. 14 are needed only for bypass bins (at the 2nd and 3rd stage). Moreover, the adder required for the Low update of a regular LPS bin is also the other structure chosen (refer to (1)). The subtractor required for the Range update for both regular MPS and regular LPS bin was decided not to be isolated. The two main reasons are: (i) it is the critical path of the original architecture; (ii) when a bypass bin occurs, the Range variable is, by default, not updated at all. Thus, it will not switch for that situation, even without the application of the low-power technique.

One final remark is the fact that both Clock-gating and Operand Isolation techniques are applied for structures required for n-bits values, where the efficiency of the methods tends to be higher (e.g., Range is a 9-bit variable, whereas Low is a 10-bit variable, leading to the required registers and adders to have at least the same size).

VI. Results and Comparisons

Three BAE versions were described in VHDL and synthesized for ST 65 nm CMOS technology, using Cadence RTL Compiler tool. The baseline version named BA-BAE, which consists of the proposal of [6] with the PN rLPS proposal of [7]. The two modified baseline version with the PN rLPS, one with the insertion of the MBBS, named MB-BAE; and other with the insertion of MBBS and the low-power approach presented in Section V, named LPMB-BAE.

The first result is to verify if any degradation in frequency occurs in the case of the insertion of the MBBS and the low-power approach (i.e., the LPMB-BAE) in an otherwise architecture without the mentioned techniques (BA-BAE). Therefore, the use of the work [6] is a feasible start, due to its design simplicity compared to [7,8] and to avoid biasing any results due to the proposed novel techniques of the other works. The only improvement chosen was the PN-rLPS [7], due to its easiness of implementation as already mentioned. As a result, the maximum frequency achieved by BA-BAE was 268 MHZ, whereas for LPMB-

Table I. Throughput improvements due to MBBS insertion

Video	Configuration		Bins/Cycle ±BA-BAE	Bins/ Cycle ± MB-BAE and LPMB-BAE	Throughput Increase (%)
BasketballDrive (1920x1080)	LD	22	4	4.52	13.28
		37		4.27	7.02
	RA	22		4.46	11.78
		37		4.34	8.77
BQTerrace (1920x1080)	LD	22		4.67	17.04
		37		4.31	8.02
	RA	22		4.65	16.54
		37		4.37	9.52
Kimono (1920x1080)	LD	22		4.94	23.81
		37		4.34	8.77
	RA	22		4.96	24.31
		37		4.45	11.53
PeopleOnStreet (2560x1600)	LD	22		4.82	20.80
		37		4.52	13.28
	RA	22		4.80	20.30
		37		4.54	13.78
Traffic (2560x1600)	LD	22		4.76	19.30
		37		4.38	9.77
	RA	22		4.73	18.55
		37		4.43	11.03
Average			4	4.56	14.36

Table II. Power savings of low-power approach insertion

Video	Configuration		Power ±MB-BAE (mW)	Power ± LPMB-BAE (mW)	Power Savings (%)
BasketballDrive (1920x1080)	LD	22	18.82	16.41	14.69
		37	17.82	15.49	15.04
	RA	22	18.66	16.26	14.76
		37	17.66	15.37	14.90
BQTerrace (1920x1080)	LD	22	17.77	15.48	14.79
		37	18.24	15.94	14.43
	RA	22	17.79	15.60	14.04
		37	18.17	15.8	14.56
Kimono (1920x1080)	LD	22	16.39	14.30	14.62
		37	18.26	15.97	14.34
	RA	22	16.58	14.45	14.74
		37	18.29	16.02	14.17
PeopleOnStreet (2560x1600)	LD	22	18.21	15.82	15.11
		37	19.04	16.76	13.60
	RA	22	18.20	16.26	11.93
		37	19.09	16.83	13.43
Traffic (2560x1600)	LD	22	18.35	16.07	14.19
		37	19.08	16.73	14.05
	RA	22	18.71	16.41	14.02
		37	19.09	16.77	13.83
Average			18.21	15.93	14.26

BAE was 264 MHZ. Hence, the degradation in frequency is negligible (around 1.5%). That result was expected since the use of the MBBS and the low-power techniques does not directly insert any logic into the critical path of [6], which is the Range update (i.e., second stage). Nevertheless, since the Range variable is used for the flow update of MBBS, an increase into the capacitance of the critical path is possible, and that is the probable explanation for the minimal degradation in frequency.

The second result is shown in Table I, which is related to the increase in throughput of bins per cycle, comparing BA-BAE against MB-BAE/LPMB-BAE (the bins per cycle of both architectures using MBBS is the same, whereas for BA-BAE is always 4 bins/cycle [6]). The bins/cycle will depend on the intrinsic behavior of a given video sequence for the versions with MBBS (i.e., the number of bypass bins and the consecutive occurrence of them will increase the throughput by different amounts). Hence, the only way to discover that amount is through architecture simulation using recommended video sequences as stimuli. The same sequences, for the same configuration and same QPs values used at Section III, were run. The MBBS designs achieved an average of around 14.36% more bins/cycle compared to the BA-BAE. One may notice that the maximum frequency is almost the same for the versions with and without the MBBS, which again states the efficiency of the techniques in increasing throughput without prejudice to the frequency of the architecture. The video sequences with the higher amount of bypass bins as a whole [8] and with the more significant amount of consecutive bypass bins occurring tend to have better results when compared to the others (i.e., Kimono and PeopleOnStreet sequences). One critical remark is the fact that the throughput of bins/cycle depends only on the architecture description (i.e., RTL), and not on the synthesized version of the design.

The third result is the power consumption comparison between the MB-BAE and LPMB-BAE. The same reasoning of the second analysis is required to perform an accurate and realistic power analysis, real video sequences are needed, since the parts of the architecture that shall be turned-off depend on the statistics of a given video sequence. The same sequences, configurations, and QPs used by the bins/cycle throughput analysis were used as stimuli for the power consumption comparison, using the gate-level netlist of both versions. The results are presented in Table II. The use of the low-power techniques accomplishes an average of 14.26% of power savings.

Finally, a comparison among LPMB-BAE and some literature related works are presented in Table III. All works, besides [5], can process real-time 8K UHD video since they have a maximum throughput above 4 Gbin/s [15]. The proposed LPMB-BAE achieves the mentioned throughput constraint with the smaller clock frequency, thus showing the suitability of the MBBS for low-power design. Considering also low QP (i.e., only the QP with value 22), the minimum frequency is even smaller (one may notice that low QP refers to a higher video quality when compared to the top QP sequence, i.e., less degradation in quality). The maximum bin/s of LPMB-BAE is below the ones achieved by [7,8]. Nevertheless, some works do not strictly cite any of the PVT conditions for the synthesis, or at least do not mention the process used, which is a reasonable explanation for the value accomplished by our work compared to them (our synthesis was made at the worst corner conditions: worst process, 125°C of temperature, and 0.95 V). Moreover, since the comparison between BA-BAE (i.e., the work of [6] with the PNR LPS) and the LPMB-BAE presented negligible frequency degradation, the use of the MBBS and the Low-power approach cannot be considered

Table III. Comparisons with related works

Design	Liu [5] ICIP 2011	Fei [6] EUSIPCO 2011	Zhou [7] ICIP 2013	Zhou [8] TCSVT 2015	LPMB-BAE	
					Low QP	Average QP
#bins/cycle	2	4	4.4	4.37	4.73	4.56
Minimum Frequency for 8K UHD	500 MHz	250 MHz	227 MHz	229 MHz	211 MHz	219 MHz
Maximum frequency	238 MHz	279 MHz	402 MHz	420 MHz	264 MHz	
Technology	90 nm	90 nm	65 nm	90 nm	65 nm	
Gates Count	3.9 k	8.22 k	57.3 k*	64.1 k*	14.6 k	
Power	4.9 mW	-	-	-	15.93 mW	
Power Stimuli	User-defined	-	-	-	Real Sequences	

* Whole CABAC results without memory

the explanation for the overall smaller clock frequency. Regarding power consumption, only [5] also presents power values, which are lower than presented by LPMB-BAE. Nevertheless, [5] did not use real sequences as stimuli and thus its results tend to be extremely optimistic, while the results of LPMB-BAE tend to be realistic.

VII. Conclusion

This work presented a statistical analysis to verify possible alternatives for increasing throughput and low-power opportunities, considering a hardware BAE architecture. The analysis showed that both regular and bypass bins occur within a burst of the same type bin.

Considering the statistical analysis and the fact that bypass bins have fewer data dependencies among them, a novel multiple bypass bin approach, name Multiple Bypass Bins Scheme (MBBS) is entirely proposed, being able to process up to two bypass bins at the same cycle per BAE core, in order to increase throughput for a given BAE design.

Low-power techniques (i.e., Clock Gating and Operand Isolation) were inserted into a selected baseline BAE architecture, along with the MBBS. The statistical analysis done at the present work was used as the reference for the places within a baseline BAE architecture where the low-power approach has the potential savings.

Three versions of the architecture were described and synthesized for ST 65 nm CMOS technology: a baseline BAE, named BA-BAE; a baseline BAE plus MBBS named MB-BAE; and a baseline BAE plus MBBS and the low-power approach, named LPMB-BAE. Comparisons between the BA-BAE and MB-BAE/LPMB-BAE shown negligible frequency degradation, proving that the insertion of the proposed techniques (i.e., MBBS and low-power approach) has minimal effect related to the maximum clock frequency achievable.

Simulations using recommended test video sequences showed an average throughput of 4.56 bins/cycle for the MBBS versions of the architecture, leading to an increase of 14.36% related to the baseline architecture. The same sequences were used for power analysis on the gate-level netlist of MB-BAE and LPMB-BAE, in order to verify the power savings achieved by the insertion of the low-power techniques. The LPMB-BAE reached power savings around 14.2%.

Comparisons with related BAE architectures of the literature were made. The proposed LPMB-BAE has the highest bins/cycle throughput and thus needs the smallest clock frequency to achieve the constraints of real-time 8K UHD. Power consumption comparison was made with related work, but since the input stimuli for the analyses were not the same, it is not fair to compare the associated values. Nevertheless, since our approach was done using real video sequences stimuli on the gate-level netlist, we assume that the LPMB-BAE has the smallest power consumption for a realistic power analysis scenario.

Finally, both MBBS and the low-power approach are suitable to be integrated with the related BAE works, to increase throughput and save power. The current work chose to implement the proposed techniques into a more straightforward baseline BAE, to verify the efficiency of both within a simpler context and point the potential of the methods for any other BAE design.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001, and also by CNPq and FAPERGS Brazilian research support agencies. The authors would like to thank the support received for the developed work.

References

[1] J. Summer, T. Bracht, D. Eager, and A. Gutarin, "Characterizing the workload of a Netflix streaming video server," in Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), 2016, 43-54.

[2] High-Efficiency Video Coding document ITU-T H.265 (V3)/ISO/IEC HEVC 23008-2, 2015.

[3] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High-Efficiency Video Coding (HEVC) standard," *IEEE Transac-*

tions on Circuits and Systems for Video Technology (TCSVT), 22, 12, December 2012, 1649-1668.

[4] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based binary arithmetic coding in H.264/AVC video compression standard," *IEEE* Transactions on Circuits and Systems for Video Technology (TCSVT), 13, 7, July 2003, 620-636.

[5] Z. Liu, and D. Wang, "One-round renormalization based 2-bin/cycle H.264/AVC CABAC encoder," in Proceedings of the IEEE International Conference on Image Processing (ICIP), 2011, 369-372.

[6] W. Fei, D. Zhou, and S. Goto, "1 Gbin/s CABAC encoder for H.264/AVC," in Proceedings of the European Signal Processing Conference (EUSIPCO), 2011, 1524-1528.

[7] J. Zhou, D. Zhou, W. Fei, and S. Goto, "A high-performance CABAC encoder architecture for HEVC and H.264/AVC," in Proceedings of the IEEE International Conference on Image Processing (ICIP), 2013, 1568-1572.

[8] D. Zhou, J. Zhou, W. Fei, and S. Goto, "Ultra-high-throughput VLSI architecture of H.265/HEVC CABAC encoder for UHDTV applications," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 25, 3, March 2015.

[9] V. Sze, and M. Budagavi, "A comparison of CABAC throughput for HEVC/H.265 vs. AVC/H.264," in Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS), 2011, 1524-1528.

[10] A. Moffat, R.M. Neal, and I. H. Witten, "Arithmetic coding revisited," in Proceedings of the IEEE Data Compression Conference, 1996, 202-211.

[11] F. Bossen, Common Test Conditions and Software Reference (JCTVC-L1100), 2013.

[12] HEVC Test Model, HM 16 [Online] Available https://hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-16.

[13] Q. Wu, M. Pedram, and X Wu, "Clock-gating and its applications to low-power design of sequential circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47, 3, March 2000.

[14] A. Correale, "Overview of the power minimization techniques in the IBM powerPC 4xx embedded controllers," in Proceedings of the International Symposium on Low Power Electronics Design (ISLPED), 1995, 75-80.

[15] Y. H. Chen, and V.Sze, "A deeply pipelined CABAC decoder for HEVC supporting level 6.2 high tier applications," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 25, 5, May 2015.