Coding- and Energy-Efficient FME Hardware Design

Ismael Seidel*† Vanio Rodrigues Filho* Luciano Agostini† José Luís Güntzel*
*Embedded Computing Lab. (ECL), Dept. of Computer Science and Statistics (INE/PPGCC),
Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
†Video Technology Group (ViTech), Group of Architectures and Integrated Circuits (GACI)
Federal University of Pelotas (UFPel), Pelotas, Brazil
Email: ismaelseidel@inf.ufsc.br, vanio.rodrigues@grad.ufsc.br, j.guntzel@ufsc.br, agostini@inf.ufpel.edu.br

***Abstract*—Hybrid video standards rely on encoding prediction residues. To improve coding efficiency of inter-frame prediction, interpolated samples may be generated in fractional positions i.e., between neighbor pixels in the reference frame. However, performing Fractional Motion Estimation (FME) increases the overall encoder complexity. Since portable mobile devices are increasingly used to capture and reproduce videos, energy-efficient FME hardware accelerators are of utmost importance. In this work, we propose and evaluate a coding- and energy-efficient hardware design strategy for FME. Such strategy addresses the main weaknesses of the architectures found in the literature. The architecture designed as case study can achieve 2160p@120fps for the HEVC 8×8 FME. We also provide an insightful area and power breakdown of the synthesized design, to drive the design of FME hardware towards further energy improvements.**

I. Introduction

In 2016, 2 trillion minutes of video crossed the Internet every month, corresponding to 73% of all traffic [1]. By 2021, such percentage could reach 82% [2], which puts in evidence the need for ever higher compression rates. However, to achieve the required compression rates without compromising quality, it is necessary to go beyond a Rate-Distortion (RD) trade-off. That is why every new video coding standard is designed to improve as much as possible the coding efficiency. Those improvements allied with advances in embedded electronics have allowed a myriad of video applications to run in Portable Mobile Devices (PMDs), making them an important gateway to record and reproduce videos. However, to achieve high coding efficiency and still comply with real-time and energy efficiency requirements [3], PMD processors must incorporate hardware accelerators specially tailored for the highest complexity steps of encoding [4].

The improvements of ∼50% in coding efficiency of High Efficiency Video Coding (HEVC) standard [5] over its predecessor [6] came at a huge cost in computational complexity [7]. Motion Estimation (ME) is one of the main culprits for such complexity: it responds for about 40% of total encoding time in HEVC [7], even using fast algorithms. ME is accomplished in two steps. The first one, referred to as Integer Motion Estimation (IME), performs an initial search using candidates taken directly from previously encoded frames. In the second step, called Fractional Motion Estimation (FME), a refinement is carried out in fractional positions (Fig. 1), using pixels obtained through the interpolation of those selected by IME.

This work was partially supported by the Brazilian Federal Agency for the Support and Evaluation of Graduate Education (CAPES) and by the Brazilian Council for Scientific and Technological Development (CNPq) through Project Universal (457174/2014-5) and PQ grants 310341/2015-9 and 309707/2015-3.

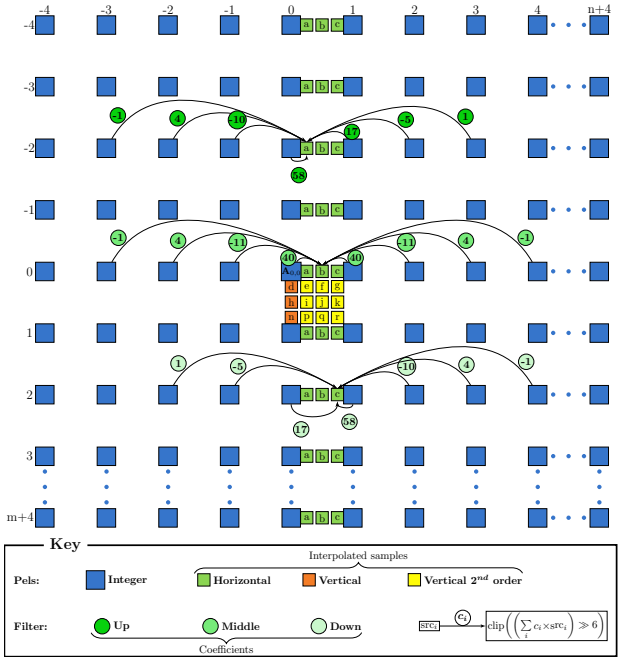


Figure 1. Interpolation of luminance samples in HEVC. Fractional pixel positions are represented with respect to integer ones. Adapted from [8]–[10].

In the HEVC Model (HM), FME is responsible for up to 33.88% of inter prediction encoding time [11]. Moreover, 46.9% of FME time is spent interpolating half and quarter pixels, while the remaining 53.1% is spent calculating the Sum of Absolute Transformed Differences (SATD) over the generated pixels [11]. Disabling FME entirely when encoding Ultra HD sequences accounts for a Bjøntegaard Delta Bitrate (BD-Rate) [12] average increase of 7.01% [11].

Considering its high complexity and its importance for current and forthcoming standards, FME is a natural candidate to be implemented in hardware. In [13] a configurable interpolation filter is proposed, which is able to realize the three HEVC filters. However, only one filter type (up, middle or down) can be used at a time. To reduce the energy consumption of the interpolation filters, [14] uses a Multiplierless Constant Multiplication (MCM) algorithm and explores common sub-expressions among source pixels. Applying both techniques, the architecture from [14] was able to reduce in 48% the energy with respect to a baseline HEVC FME architecture. Both previously works present only interpolation filters, which require them to keep internally the interpolated samples or to write them in an external memory until they are used. The authors of [15] present several hardware design considerations for the

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE SANTA CATARINA. Downloaded on April 13,2023 at 19:46:16 UTC from IEEE Xplore. Restrictions apply.

HEVC ME. Although their designs include FME, no data is presented for such specific part. A hardware design for the HEVC FME is presented in [10]. Their design directly feeds the interpolated samples to search, saving area and power. However, such design uses a filter structure with a dedicated datapath for each interpolated pixel, resulting in redundant operations. Moreover, the authors assume an external source for the pixels in integer positions and do not evaluate the impacts of such decision. Finally, most works that consider the whole FME adopt only a simple distortion metric that disregards rate constraints.

As main contribution, this paper proposes and evaluates a design strategy for FME hardware accelerators which may be used in future video codecs. The resulting architecture has the following features. **1) Coding efficiency**: it searches in all fractional positions while considering the weighted rate, in a fast Rate-Distortion Optimization (RDO); **2) Energy efficiency**: it has a regular dataflow and thus a high degree of parallelism, performs no redundant filter operations, avoids extra Motion Compensation (MC), is memory-aware, thus avoiding unnecessary memory accesses, keeps only useful data in internal buffers, which are designed to minimize wire-length. As secondary contribution, we present area and power breakdowns of the architecture, identifying the largest modules and the sources of power consumption.

The remaining of this paper is organized as follows. Section II brings a few definitions. The proposed design strategy is presented in Section III. Section IV outlines a case study adopting our strategy. Results and comparison with related work are in Section V. Finally, Section VI draws the conclusions.

II. Definitions

Each frame is a pixel matrix, which is divided into partitions called blocks (**B**) [16]. During encoding, each $m \times n$ **B** is referred to as Original block ($\mathbf{B}^{\text{ori}}$). To reduce the entropy of each $\mathbf{B}^{\text{ori}}$, the encoder selects a Reference block ($\mathbf{B}^{\text{ref}}$) and encodes the difference ($\mathbf{B}^{\text{ori}}$- $\mathbf{B}^{\text{ref}}$), called residue ($\mathbf{B}^{\text{res}}$). Thus, the Block Matching Algorithm (BMA) [3] is adopted to find such $\mathbf{B}^{\text{ref}}$. Its goal is to find a Candidate block ($\mathbf{B}^{\text{can}}$), among a set of candidates (S), that minimizes a cost function ($cost$) to be used as $\mathbf{B}^{\text{ref}}$, that is:

$$\mathbf{B}^{\text{ref}} = \underset{\mathbf{B}^{\text{can}} \in S}{\arg\min}\, cost_{m \times n}(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}) \quad (1)$$

To improve coding efficiency, the rate resulting from the selection of a given $\mathbf{B}^{\text{can}}$ must be considered by $cost$, along with its distortion. Thus, encoders often rely on RDO, which uses the Lagrangian rate-distortion cost (j_{cost}) [16], [17] shown in Equation 2, where λ is a pre-calculated Lagrange Multiplier.

$$j_{\text{cost}}(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}) = distortion(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}) + \lambda \times rate(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}) \quad (2)$$

Ideally, $distortion$ should be obtained over reconstructed blocks, i.e., after the upcoming coding steps: transform (T), quantization (Q) and their inverses [18]. Similarly, $rate$ should be obtained after entropy coding, which is performed over the quantized residue. Regardless its good compromise between rate and distortion, RDO also increases the complexity: brute-force RDO is one of the main causes of HM poor speed [7].

A common workaround to such complexity limitation is the adoption of a fast RDO, where the RD is estimated directly over each $\mathbf{B}^{\text{can}}$. Even so, the large cardinality of S brings the need for a simple $distortion$ metric such as the Sum of Absolute Differences (SAD) [19] defined in Equation 3, where $d_{i,j}$ is an element of $\mathbf{D}_{m \times n}$=$\mathbf{B}^{\text{ori}}_{m \times n}$−$\mathbf{B}^{\text{can}}_{m \times n}$.

$$\text{sad}_{m \times n}(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |d_{i,j}| \quad (3)$$

Also, in this fast RDO, the second term of Equation 2 may be a rough estimate [20]. For instance, during ME it can be the $rate$ of the Motion Vector (MV), which encodes the $\mathbf{B}^{\text{can}}$ location. ME can be divided into two main steps: IME and FME. The former executes BMA with candidates obtained in integer pixel positions. The latter uses the $\mathbf{B}^{\text{ref}}$ selected by IME ($\mathbf{B}^{\text{ref-ime}}$) along with its surroundings ($\mathbf{B}^{\text{ref-ime'}}$), to populate a new set of candidates S^{fme} and perform a new BMA.

III. Proposed FME Hardware Design Strategy

A possible approach for FME is to interpolate and perform BMA with candidates from all fractional positions surrounding $\mathbf{B}^{\text{ref-ime}}$. Such approach complies with the two main goals of our strategy: **coding** and **energy** efficiency. If S^{fme} contains all possible fractional candidates, BMA will find the optimal MV around $\mathbf{B}^{\text{ref-ime}}$. Considering the Common Test Conditions (CTC) [21], this approach reduces BD-Rate up to 0.37 % and 0.77 % for Random Access (RA) and Low Delay (LD) profiles, respectivelly. However, it is imperative that at least a fast RDO is used to provide the best trade-off between rate and distortion (j_{cost}), otherwise BD-Rate may increase up to 1.43 % (RA) and 0.86 % (LD). Also, such approach for FME is hardware-friendly, since it leads to a regular dataflow [22], [23]. Such regularity increases the room for data-level parallelism which, by its turn, is able to reduce the required number of cycles. A high degree of parallelism reduces delay at the expense of area and power overheads. If well exploited, the time reduction is worth the overheads, resulting in better energy efficiency [24].

Using the interpolated samples as soon as they are generated [10] is a clever way to reduce power by getting rid of intermediary buffers. Thus, it is necessary that interpolation and BMA are both integrated in the same design (hardware). This restriction does not render MC unable to share the same interpolation, but it just requires a path from the interpolation output to the top level output. Another advantage of an integrated design is the possibility to pipeline both steps.

As proposed in [14], no redundant operations should be performed. Also, the architecture must use MCM, once interpolation filter coefficients are constant. Apart these arithmetic optimizations, to achieve energy efficiency the design must be memory-aware, otherwise memory accesses may exceed 90% of total ME energy [25]. Even using memory hierarchies, data buses and on-chip buffers can be responsible for almost 70% of total power [26]. Thus, it is necessary to keep the required data internally and use it as soon as possible. Moreover, some data require matrix transposition, which can be very demanding of external sources and thus must be done internally using Transpose Buffers (TBs) [27], [28]. Another power reduction technique consists in reducing interconnect distances. It can be induced by designing buffers where data moves in layers, thus avoiding long wires at the expense of more switching activity.

Finally, alongside the common outputs (best cost and related MV), our strategy includes issuing also the $\mathbf{B}^{\text{res-fme}}$. Thus, if the FME candidate is chosen by mode decision, the transform can be applied directly on the outputs, instead of interpolating again the $\mathbf{B}^{\text{res-fme}}$ (avoids unnecessary MC).

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE SANTA CATARINA. Downloaded on April 13,2023 at 19:46:16 UTC from IEEE Xplore. Restrictions apply.

IV. Case Study: The HEVC 8×8 FME

While video coding standards define only the decoder side [15], ME is part of the encoder. However, the interpolation is standardized, since it is also part of the decoder. Thus, we will use the HEVC FME to evaluate our strategy. To compare with state-of-the-art designs, our case study assumes 8×8 blocks.

The hardware architecture resulting from our strategy (Fig. 2) interpolates and searches all fractional samples around $\mathbf{B}^{\text{ref-ime}}$. Thus, this architecture is highly parallel and finishes interpolation in 51 cycles. The interpolation datapath (Fig. 2a) includes the Filter (Fig. 2b), clipping and two TBs, for integer and horizontal samples. Therefore, this design is **memory-aware** because each sample is read from external memory only once, while transpositions are internally performed.

To avoid redundant filter operations, we explored the shared filters coefficients. Since there are 8 distinct filter coefficients (see Fig. 1), their multiplications can be described by a vector function $\vec{ss}(x) : \mathbb{Z} \rightarrow \mathbb{Z}^8$, defined as:

$$\vec{ss}(x) = [-x; 4\times x; -10\times x; -5\times x; -11\times x; 40\times x; 58\times x; 17\times x] \quad (4)$$

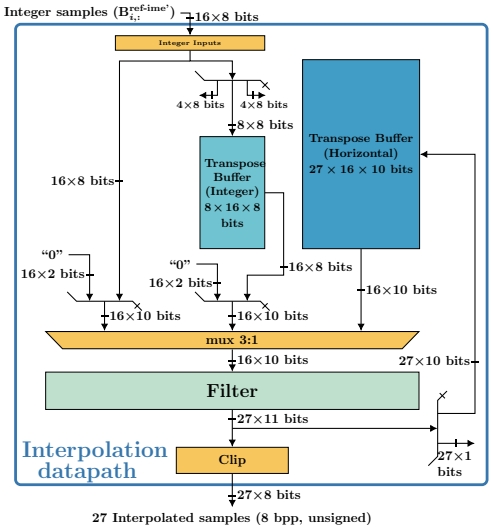
Given $\vec{x} = \mathbf{B}^{\text{ref-ime'}}_{i,:}$, i.e $\vec{x}$ is the i^{th} line of $\mathbf{B}^{\text{ref-ime'}}$, the application of Up, Middle and Down filters (Fig. 1) over $\vec{x}$ results in three vectors, namely $\vec{u}$, $\vec{m}$ and $\vec{d}$. These vectors have 9 interpolated pixels each, and their values can be obtained using $\vec{ss}(x)$, as shown in Equations 5 to 7, where $n \in [-1, 8]$.

$$\vec{u}_n = \{\vec{ss}(\vec{x}_{n-3})_0 + \vec{ss}(\vec{x}_{n-2})_1 + \vec{ss}(\vec{x}_{n-1})_2 + \vec{ss}(\vec{x}_n)_6 + \vec{ss}(\vec{x}_{n+1})_7 + \vec{ss}(\vec{x}_{n+2})_3 + \vec{x}_{n+3}\} \gg 6 \quad (5)$$

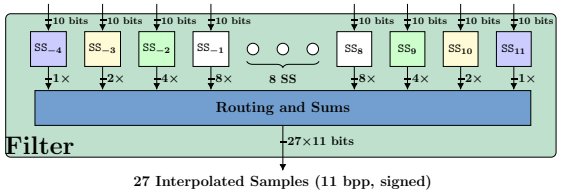
$$\vec{m}_n = \{\vec{ss}(\vec{x}_{n-3})_0 + \vec{ss}(\vec{x}_{n-2})_1 + \vec{ss}(\vec{x}_{n-1})_4 + \vec{ss}(\vec{x}_n)_5 + \vec{ss}(\vec{x}_{n+1})_5 + \vec{ss}(\vec{x}_{n+2})_4 + \vec{ss}(\vec{x}_{n+3})_1 + \vec{ss}(\vec{x}_{n+4})_0\} \gg 6 \quad (6)$$

$$\vec{d}_n = \{\vec{x}_{n-2} + \vec{ss}(\vec{x}_{n-1})_3 + \vec{ss}(\vec{x}_n)_7 + \vec{ss}(\vec{x}_{n+1})_6 + \vec{ss}(\vec{x}_{n+2})_2 + \vec{ss}(\vec{x}_{n+3})_1 + \vec{ss}(\vec{x}_{n+4})_0\} \gg 6 \quad (7)$$

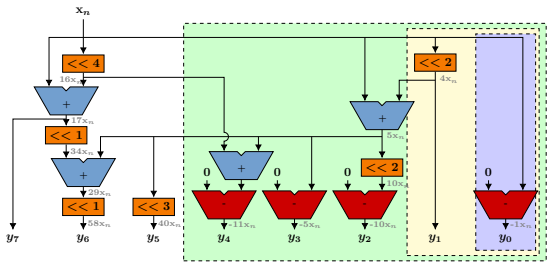
Similarly to [14], we used Spiral tool [29] to generate the filter datapaths corresponding to Equation 4. The resulting datapath (Fig. 2c) was named SS (Sums and Shifts). To increase parallelism, each of the 16 pixels in $\vec{x}$ has its own dedicated SS datapath (Fig. 2b). As the number of dependencies of samples near the borders of $\mathbf{B}^{\text{ref-ime'}}$ reduces, we adopted simplified versions of SS accordingly.



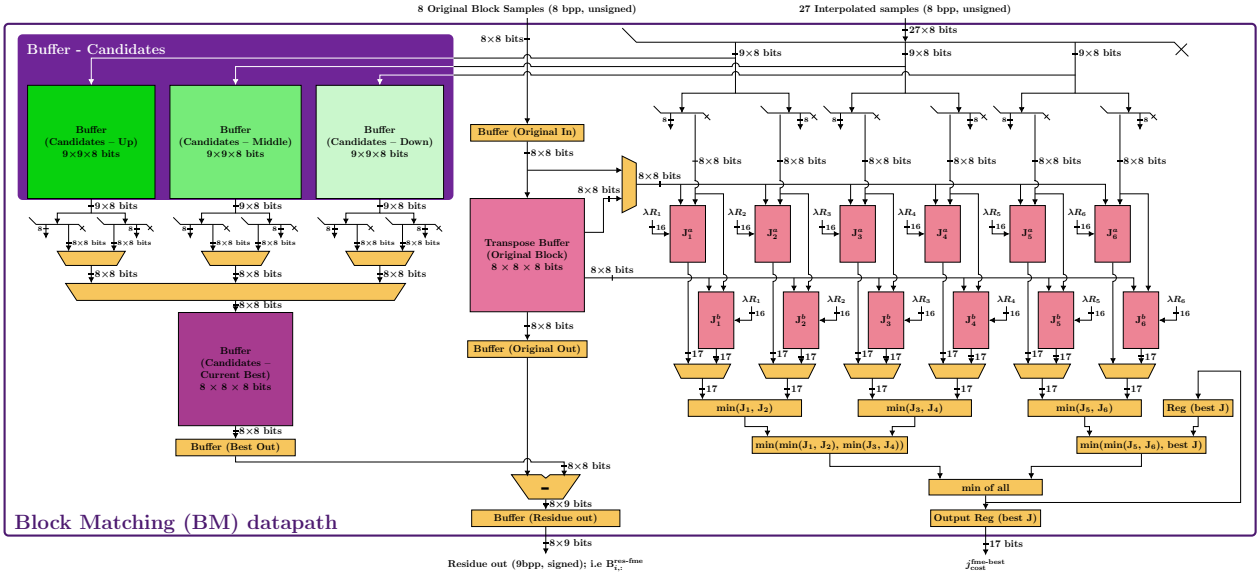
(a) Interpolation.



(b) Filter, which uses 16 SS modules.



(c) Sums and Shifts (SS).



(d) Block Matching (BM).

Figure 2. Register Transfer Level (RTL) datapath schematics for the HEVC 8×8 FME following the proposed hardware design strategy.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE SANTA CATARINA. Downloaded on April 13,2023 at 19:46:16 UTC from IEEE Xplore. Restrictions apply.

Both interpolation and BMA are integrated in a single design and then pipelined such that every 51 cycles one FME is finished. One cycle after the interpolation of candidate pixels, their values are available to the Block Matching (Fig. 2d). Each J module performs the SAD (Equation 3) between $\mathbf{B}^{\text{ori}}$ and each interpolated $\mathbf{B}^{\text{can}}$, which is added to the appropriate $\lambda\times$rate to obtain j_{cost}. The weighted rates must come from an external source, which are loaded to each J accumulator without overheads after 12 extra inputs. To be able to provide the $\mathbf{B}^{\text{res}}$, intermediary $\mathbf{B}^{\text{can}}$ must be kept in Candidate Buffers until their evaluation using j_{cost} finishes. After the decision, only the current best candidate is loaded line-by-line into another buffer, while the Candidate Buffers are overwritten with new candidates. Also, the current $j_{\text{cost}}^{\text{best}}$ is kept in a register to be compared to the remaining $\mathbf{B}^{\text{can}}$.

V. Area and Power Evaluation

The hardware design case study was synthesized with Synopsys® Design Compiler (DC®) [30] in Topographical mode to estimate routing parasitics and thus, obtain realistic timing, area and power estimates. All syntheses used a 45 nm standard cell library from TSMC [31]. Also, input and output delays were conservatively limited to 60% of the clock period. The maximum primary input capacitance was set to 10 times a 2-input AND gate whereas the maximum primary output capacitance was set to 30 times a 2-input AND gate. Finally, to get detailed area and power reports, we restricted DC® to keep the RTL hierarchy during syntheses[1].

We performed four distinct syntheses, each one aiming a pair of resolution and fps, as shown in Table I. Given the FME of a video with resolution of $w\times h$ and f frames per second, the required throughput in $\mathbf{B}^{\text{ori}}_{m\times n}$/second is $(w\times h\times f)/(m\times n)$. The presented architecture finishes an FME each 51 cycles which, multiplied by the throughput, gives the number of cycles per second (its inverse being the target period). Thus, the clock period was set accordingly for each synthesis.

Table I. Synthesis results for TSMC 45 nm @ 0.9 V

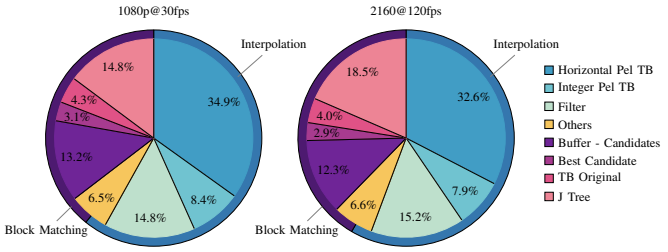
Target	1080@30fps	2160@30fps	2160@60fps	2160@120fps*
Period (ns)	20	5	2.5	1.25
Area (μm^2)	93738.3	93826.5	94439.1	100733.4
Dynamic (mW)	3.05	10.78	21.21	44.04
Static (mW)	0.91	0.91	0.92	1.04
Total (mW)	3.96	11.69	22.13	45.07

* 2160@120fps is equivalent in terms of throughput to 4320@30fps.

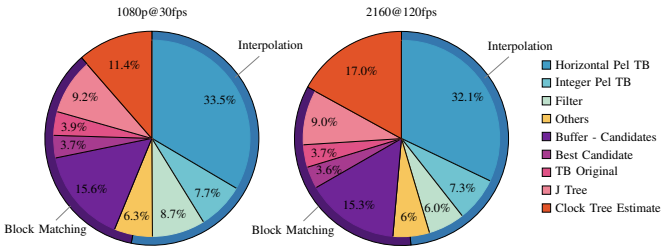
The largest increase in area was only 6.6%, observed when comparing 2160@60fps and 2160@120fps cases. On the other hand, total power increases about 2× when doubling the frequency, which is expected since dynamic power increases with frequency. The share of dynamic power due to nets also increases with frequency: from 18% to up to 23%. This shows the importance of reducing wire-length. To provide some insightful evaluation, Fig. 3 shows area and power breakdowns for the lowest and highest target throughputs.

The increase in area share of Filter and J Tree comes from the critical path optimization performed by the synthesis tool to achieve the higher throughput. In both area and power TB Horizontal is the dominant part of the design. Considering Interpolation and BM shares, the former is larger and demands more power, even considering the large amount of power consumed by the clock tree (up to 17% @ 2160@120fps).

[1]Forcing the synthesis tool to keep RTL hierarchy resulted in 2.5% more power and 10.7% larger area, on average. Also, when hierarchy was not preserved, the tool was unable to meet the required constraints for 2160p@120fps.



(a) Area Breakdown.



(b) Power Breakdown.

Figure 3. Area and Power Breakdown. The outer ring represents the share of Interpolation and Block Matching (BM).

Among related works, the architecture and syntheses of [10] are the most similar to ours. Since it requires the same number of cycles than ours, a direct comparison of energy consumption can be done by evaluating total power. For 1080@30fps our design consumes 20,1% less power than the one in [10], while performing more operations. Moreover, their results do not include the energy spent by external pixel source. On the other hand, for 2160@60fps their design requires 28% less power than ours. Particularly, when raising throughput from 1080@30fps to 2160@60fps (8× the frequency), [10] reported an increase of only 3.19× without justifying such unexpected behavior. For a fair power comparison with [10], we disregarded the power share of the blocks that are exclusive to our design. By doing so, our design consumes 6.4% and 46.5% less power than theirs for 2160@60fps and 1080@30fps, respectively. Finally, it is worth noting that the work in [10] reports lower power than those in [9], [32]–[34].

VI. Conclusions

The proposed design strategy for FME was evaluated by means of a case study consisting of the HEVC 8×8 FME. The hardware architecture of such case study is more energy-efficient than state-of-the-art similar ones. This is because the proposed strategy leads to a BMA that considers all pixel positions to maximize parallelism and adopts a fast RDO during search to guarantee that the optimal fractional motion vector is found. Also, the synthesized design has no redundant filter operations, avoids extra MC, is memory-aware and the internal buffers were designed to minimize wire-length. As result, both coding and energy efficiency are improved.

The obtained power and area breakdowns show that buffers are responsible for up to 64% of both area and power. However, the largest one (more than 30% of area and power) is intrinsic of all memory-aware parallel FME architectures, which is the case. In addition, when enforcing shorter clock periods during synthesis, the largest amounts of area increase were observed in the sequential parts, revealing that the critical paths are in those parts, and not in the large buffers. Thus, future work may address power optimization of buffers through low-power techniques using slower cells [35].

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE SANTA CATARINA. Downloaded on April 13,2023 at 19:46:16 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] Cisco, "VNI complete forecast highlights: Global - 2016 year in review", Cisco, Tech. Rep., Jun. 2017.

[2] ——, "Cisco visual networking index: Forecast and methodology, 2016–2021", Cisco, White Paper, Jun. 2017.

[3] I. Chakrabarti *et al.*, *Motion Estimation for Video Coding: Efficient Algorithms and Architectures*, ser. Studies in Computational Intelligence. Springer International Publishing, 2015.

[4] J. Vanne *et al.*, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs", *IEEE J. of Sel. Topics in Signal Process.*, vol. 22, no. 12, pp. 1885–1898, Dec. 2012.

[5] ITU-T, "Recommendation ITU-T H.265: High efficiency video coding", International Telecommunication Union, Geneva, Recommendation H.265, Apr. 2013.

[6] ——, "H.264 : Advanced video coding for generic audiovisual services", International Telecommunication Union, Geneva, Recommendation H.264, May 2003.

[7] F. Bossen *et al.*, "HEVC complexity and implementation analysis", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[8] C. M. Diniz *et al.*, "A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 2, pp. 238–251, 2015.

[9] V. Afonso *et al.*, "Memory-aware and high-throughput hardware design for the HEVC fractional motion estimation", in *SBCCI'15*, Aug. 2015, pp. 1–6.

[10] V. Afonso *et al.*, "Hardware implementation for the HEVC fractional motion estimation targeting real-time and low-energy", *Journal of Integrated Circuits and Systems*, vol. 11, no. 2, pp. 106–120, 2016.

[11] S. Blasi *et al.*, "Adaptive precision motion estimation for HEVC coding", in *PCS'15*, May 2015, pp. 144–148.

[12] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves", Video Coding Experts Group (VCEG), Austin, Texas, USA, Document VCEG-M33, Apr. 2001.

[13] Z. Guo *et al.*, "An optimized MC interpolation architecture for HEVC", in *ICASSP'12*, Mar. 2012, pp. 1117–1120.

[14] E. Kalali and I. Hamzaoglu, "A low energy HEVC sub-pixel interpolation hardware", in *ICIP'14*, Oct. 2014, pp. 1218–1222.

[15] M. Sinangil *et al.*, "Cost and coding efficient motion estimation design considerations for high efficiency video coding (HEVC) standard", *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1017–1028, Dec. 2013.

[16] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression", *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[17] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression", *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[18] I. E. Richardson, *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons Inc, 2003.

[19] ——, *The H.264 Advanced Video Compression Standard, Second Edition*. John Wiley & Sons Ltd, 2010.

[20] V. Sze *et al.*, *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, ser. Integrated Circuits and Systems. Springer International Publishing, 2014.

[21] F. Bossen, "Common test conditions and software reference configurations", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Shanghai, Document JCTVC-K1100, Oct. 2012.

[22] L. Hu *et al.*, "A hardware-friendly hierarchical HEVC motion estimation algorithm for UHD applications", in *ISCAS'17*, May 2017, pp. 1–4.

[23] W. Xu *et al.*, "High-performance motion estimation for image sensors with video compression", *Sensors*, vol. 15, no. 8, pp. 20 752–20 778, 2015.

[24] J. Rabaey, *Low Power Design Essentials*, 1st ed., ser. Integrated Circuits and Systems. Springer US, 2009.

[25] B. Zatt *et al.*, "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding", in *DAC '11*, New York, NY, USA: ACM, 2011, pp. 1026–1031.

[26] S. Yang *et al.*, "Power and performance analysis of motion estimation based on hardware and software realizations", *IEEE Trans. Comput.*, vol. 54, no. 6, pp. 714–726, Jun. 2005.

[27] V. dos S. Livramento *et al.*, "Evaluating the impact of architectural decisions on the energy efficiency of FDCT/IDCT configurable IP cores", *Journal of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 23–36, Mar. 2012.

[28] I. Seidel *et al.*, "Energy-efficient SATD for beyond HEVC", in *ISCAS'16*, IEEE, May 2016.

[29] C. M. U. Yevgen Voronenko, *Spiral software/hardware generator for DSP algorithm*, 2017. [Online]. Available: http://spiral.ece.cmu.edu/mcm/gen.html.

[30] Synopsys, *Synopsys design compiler, version M-2016.SP4.* 2015.

[31] *TSMC standard cell library tcbn45gsbwptc*, TSMC, 2011.

[32] C. M. Diniz *et al.*, "High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC", in *ICIP'13*, Sep. 2013, pp. 2091–2095.

[33] G. Pastuszak and M. Trochimiuk, "Architecture design of the high-throughput compensator and interpolator for the H.265/HEVC encoder", *Journal of Real-Time Image Processing*, vol. 11, no. 4, pp. 663–673, Apr. 2016.

[34] G. He *et al.*, "High-throughput power-efficient VLSI architecture of fractional motion estimation for ultra-HD HEVC video encoding", *IEEE Trans. VLSI Syst.*, vol. 23, no. 12, pp. 3138–3142, Dec. 2015.

[35] M. Keating *et al.*, *Low Power Methodology Manual - for System-on-Chip Design.* Springer, 2007.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE SANTA CATARINA. Downloaded on April 13,2023 at 19:46:16 UTC from IEEE Xplore. Restrictions apply.