EFFICIENT DEBLOCKING FILTER IMPLEMENTATION ON RECONFIGURABLE PROCESSOR

Kausik Maiti[1], Sirish K. Pasupuleti[1], Raj N. Gadde[1], SangJo Lee[2]

[1]Advanced Research Team, DMC.
Samsung R&D Institute India – Bangalore Pvt. Ltd. Bangalore, India
kausik.maiti@samsung.com, sirish.p@samsung.com, raj.gadde@samsung.com.

[2]DMC R&D center, Multimedia Computing Lab.
Samsung Electronics Co., Ltd, Suwon-si, Republic of Korea
sjlee92@samsung.com.

ABSTRACT

As world is moving towards Ultra High Definition (UHD) content and display technology, high quality visual content is becoming a necessity. Deblocking filter is one of the tools used in today's video coding standards to enhance the quality of compressed video. Deblocking consumes significant percentage (≈20%-33%) of total decoding cycles. Therefore, many decoder implementations tend to offload deblocking operation to additional hardware IP block or GPU to achieve real-time performance. However, hardware IP increases die area and lacks flexibility; on other hand GPUs are power hungry. In this paper, we present a reconfigurable processor based software solution along with a deblocking specific intrinsic catering to wide range of video coding standards for handling this performance bottleneck. Our experimental results show, proposed approach improves deblocking performance by a factor greater than 10 and results in processing time in the order of 140 ms for 4K UHD HEVC (60 fps, 30 mbps) stream.

Index Terms **—** Deblocking, HEVC, Reconfigurable processor

1. INTRODUCTION

All video coding standards (like H.264, VC1, VP8, HEVC, VP9) that have come up after MPEG-4 Part 2 have adopted deblocking filter in the feedback path [1-6]. The purpose is two-fold. First, it helps to improve the quality of video by eliminating the blocky artefacts that are introduced in the reconstructed frames due to disjoint block wise processing and quantization. Second, by reducing the artefacts in the reconstructed frames in Decoded Picture Buffer, it improves the accuracy of inter-frame prediction and thereby improves the compression efficiency.

Most of the deblocking algorithms use highly data-dependent computation involving some decision tree model. Consequently, it becomes a challenge for the underlying processor to handle deblocking operation in resource efficient manner. Thus, many codec implementations usually find deblocking as a major bottleneck to achieve real-time performance.

In such a scenario, researchers have come up with different hardware, software solutions. For example, on hardware side, Ozcan et al. proposed a hardware architecture with two data-paths in parallel, where each data-path can be configured to implement all decision and edge filter operations of HEVC deblocking [7]. Shen et al. proposed memory architecture for efficient transmission of data to the deblocking filter and the filtered outputs to an external memory, trying to reduce bandwidth [8]. On the software front, Yan et al. proposed a HEVC decoder implementation by exploiting Single Instruction Multiple Data (SIMD) parallelism in order to boost the performance [9]. Kotra et al. proposed implementations for the HEVC deblocking filter on multiple CPU cores [10]. Diego De Souza et al. proposed efficient implementations of HEVC deblocking filter for heterogeneous (multi-core CPU+GPU) platform [11].

Both lines of proposal have unique pros and cons. The problems with additional hardware IP are increased die area (i.e. increased gate-count) and lack of adaptability to the future algorithms. Whereas software-only solutions usually need high clock frequency and hence are not power efficient. Therefore, achieving power efficiency, real time performance, design flexibility and smaller gate count is a real challenge.

We believe, in order to achieve a balanced trade-off among all the afore-mentioned design parameters, we need a solution that comes mid-way between hardware only and software only design. We need a processor architecture, whose capability can be adjusted with the generic computational pattern of the algorithm. Also, we need a software implementation design that can maximally utilize

the architectural features and resources. Keeping both these points in mind, in this paper, we propose a highly efficient deblocking filter solution on reconfigurable architecture and present the results obtained with our implementation. We take HEVC decoder for the sake of illustration.

The rest of the paper is organized as follows. In section 2, we outline HEVC deblocking algorithm. Section 3 includes brief description on Reconfigurable Processor. Our proposal of dual approach for handling deblocking filter is given in section 4. Section 5 contains the experimental results in detail. And finally, conclusion is drawn in the last section of the paper.

2. DEBLOCKING ALGORITHM

In HEVC, deblocking is applied to all samples adjacent to a Prediction Unit or Transform Unit boundary that does not coincide with video frame boundary. Slice and tile boundaries are also skipped, if syntax elements in Sequence Parameter Set and Slice header indicate so. Deblocking is applied on 8x8 grids (Figure 1) of luma and chroma samples. All vertical edges are filtered first. Then all horizontal edges are processed. In this paper, we describe the key steps involved in vertical luma edge filtering. Exactly similar steps are involved in horizontal luma edge filtering too. Detailed description about luma and chroma deblocking can be found in [12-14].

Few important control parameters of HEVC deblocking are Base Strength (BS), Quantization Parameter (QP), Local Adaptivity Measure (LAM) and QP dependent thresholds t_c and β. BS value depends on intra-prediction flag, non-zero transform coefficient, reference indices, similarity / differences between motion vectors of two blocks P, Q that join at the edge (Figure 1). Possible BS values are 0, 1, and 2 for luma edges, where 0 denotes no filtering. Horizontal LAM is obtained as in equation (1).

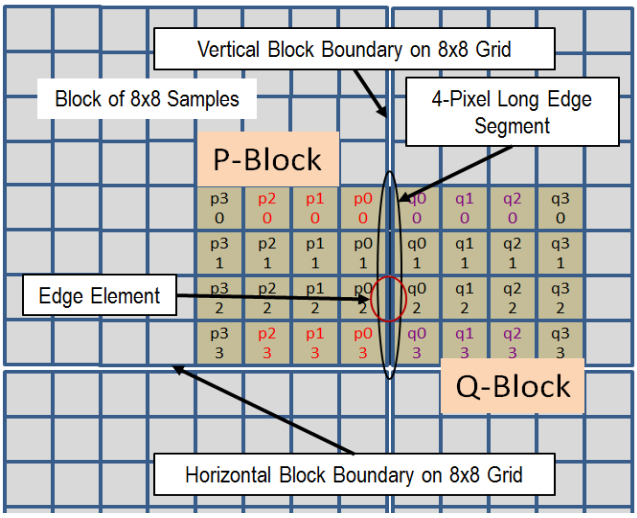

Figure 1. Illustration of 8x8 block boundary and samples involved in vertical Luma edge filtering in HEVC

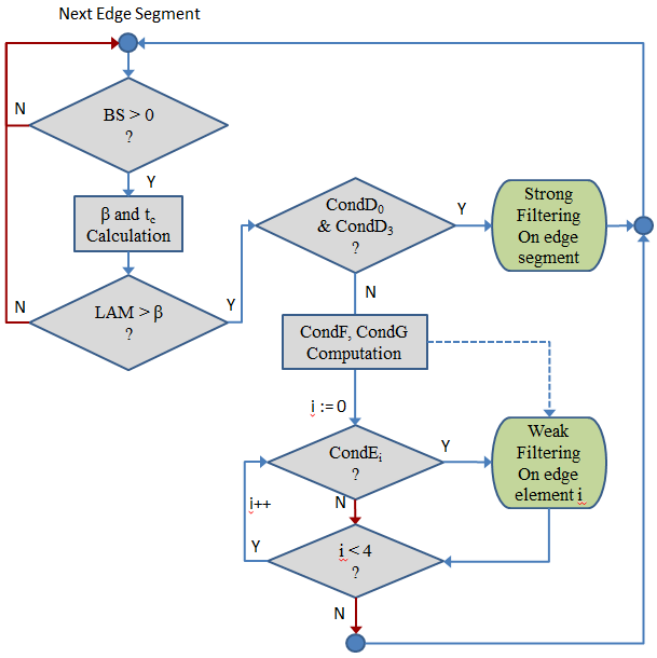

Figure 2. Flow-chart for HEVC deblocking algorithm

$$LAM_H := (|p2_0 - 2p1_0 + p0_0| + |p2_3 - 2p1_3 + p0_3| + |q2_0 - 2q1_0 + q0_0| + |q2_3 - 2q1_3 + q0_3|) \quad (1)$$

For vertical edge filtering, if $LAM_H > \beta$, type of filtering is decided with the following set of equations.

$$CondA_i := |p2_i - 2p1_i + p0_i| + |q2_i - 2q1_i + q0_i| < \beta/8;\ i = 0, 3 \quad (2)$$
$$CondB_i := |p3_i - p0_i| + |q0_i - q3_i| < \beta/8;\ i = 0, 3 \quad (3)$$
$$CondC_i := |p0_i - q0_i| < 2.5t_C;\ i = 0, 3 \quad (4)$$
$$CondD_i := CondA_i \,\&\, CondB_i \,\&\, CondC_i;\ i = 0, 3 \quad (5)$$
$$CondE_i := |d0_i| < 10t_C;\ i = 0, 1, 2, 3$$
$$\text{Where } d0_i = (9(q0_i - p0_i) - 3(q1_i - p1_i) + 8) >> 4 \quad (6)$$
$$CondF := |p2_0 - 2p1_0 + p0_0| + |p2_3 - 2p1_3 + p0_3| < 3/16\ \beta \quad (7)$$
$$CondG := |q2_0 - 2q1_0 + q0_0| + |q2_3 - 2q1_3 + q0_3| < 3/16\ \beta \quad (8)$$

The overall HEVC deblocking control flow is explained using a flow-chart in Figure 2. A strong filtering influences three pixels on either side of the edge. In case of weak filtering, two pixels (adjacent to the edge element) in block P and Q are modified, if CondF and CondG are TRUE respectively. Otherwise, only the nearest pixel adjacent to the edge element is modified.

3. RECONFIGURABLE PROCESSOR

In recent past, reconfigurable processor (RP) has drawn much attention due to its high performance and yet high degree of adaptability with a wide range of computational demand of embedded applications [15-20]. Coarse Grained Reconfigurable Array (CGRA), the core component of RP, coupled with powerful software scheduler presents huge parallelism at very low power consumption [21-26].

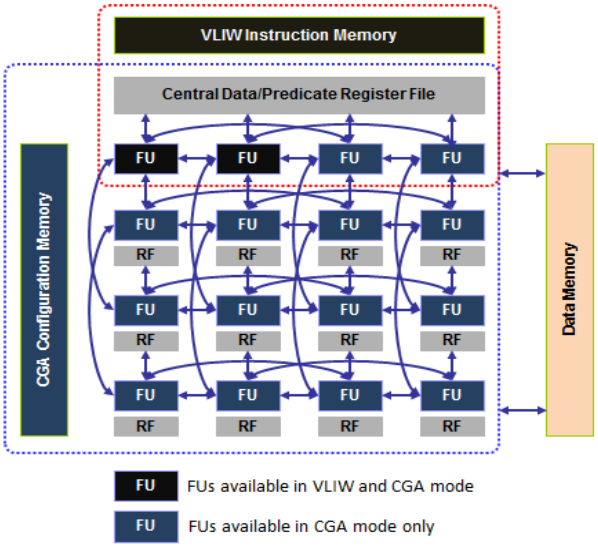

Figure 3. Basic Block Diagram of RP

Figure 3 shows the high level block diagram of RP that we have used in our experiment. It is similar to the Architecture for Dynamically Reconfigurable Embedded Systems (ADRES) designed by IMEC, Belgium [18]. It consists of a coarse-grained array of functional units (FU), global / local register files, internal data memory, instruction cache, configuration memory, and high speed buses for external data transfers. In a RP core, the FUs are arranged in form of regular NxN grid. In our case, value of N is 4. Capabilities of FUs vary widely; some are capable of multiplication, some can perform memory access operations, some might have SIMD capabilities, some may be equipped with application specific instructions or intrinsics etc. A RP core can be made to operate in either of the two modes: Very Long Instruction Word (VLIW) mode and Coarse-Grained Array (CGA) mode. In VLIW mode, only limited number (typically 2 to 4) of FUs are available. And all N^2 FUs are available in CGA mode. VLIW mode is used for handling sequential control flow. CGA mode is meant for executing data massive computationally intensive loops comprising of unconditional statements. CGA compiler analyzes the dependencies among instructions inside a loop and also across successive loop iterations to exploit loop level parallelism (LLP) and arrives at a scheduling that maximizes the throughput. Switching between these two modes can happen seamlessly with the help of special instruction. RP core supports application specific instructions (ASI) in the form of intrinsic. Usually the ASIs are designed from the frequently used patterns or sequence of the basic instructions in the target applications. Adding new intrinsics will increase the gate-count. Hence, among the candidates ASIs, the processor architects choose the ones yielding the maximum performance boost without much increase in hardware cost such as gate counts, critical path length [25].

4. PROPOSED DUAL APPROACH

Our proposed solution resorts to a dual approach. At one hand, it introduces additional architectural capability to suit the need of the deblocking filtering algorithm. On the other hand, it proposes a novel algorithm partitioning approach to ensure maximal utilization of the architectural features and resources.

4.1. Resource Efficient Algorithm Partitioning

RP is best fit to handle iterative execution of block of independent statements without branch / jump operations. Because, in such cases, loops can be mapped in CGA mode and all 16 FUs can be utilized. But, HEVC deblocking algorithm uses highly data-dependent computation along with multiple levels of decision tree. And iterative execution of such an algorithm cannot be mapped to CGA.

An evident approach to enable CGA mapping of deblocking algorithm is branch predication. Branch predication is widely used technique in computer science to avoid conditional branch with the help of conditional instructions [27]. This means, each operation associated with TRUE and FALSE condition paths has to be converted to conditional instruction. The resultant CGA mapping helps to achieve considerable performance gain over VLIW because of higher FUs in CGA mode compared to VLIW mode and the loop level parallelism explored by the CGA compiler. However, this implementation may not produce an optimal performance, since both the TRUE and FALSE paths are executed all the time and it results in significant increase in number of instructions. And, many of the operations can't be executed in parallel, since the highly dependent control and filtering paths of a deblocking algorithm are executed in the same loop.

To produce an optimal solution by exploiting the full capability of CGA, we propose to partition the deblocking algorithm into two stages, namely analysis stage and filtering stage, as shown in Figure 4. In the analysis stage, we make the filtering decisions for all edge segments and store the offset information of only those edge elements that require filtering in an intermediate buffer. And in the filtering stage, we iterate over only the edge-elements entered into the offset buffer. The analysis loop can be operated in CGA mode with the help of the branch predication logic as discussed earlier. But, the filtering stage can now be efficiently mapped to CGA due to absence of conditional operations and also data inter-dependencies. The resultant CGA scheduling can now optimally exploit LLP to maximize the throughput. Moreover, as a good percentage of edge-elements do not require actual filtering, the proposed partitioning also avoids filtering for such edge-elements. Table 1 clearly shows the improvement in both 'Cycles per Iteration' (CPI) and 'Instructions per Cycle' (IPC) with the proposed partitioning as compared to simple branch predication based approach.

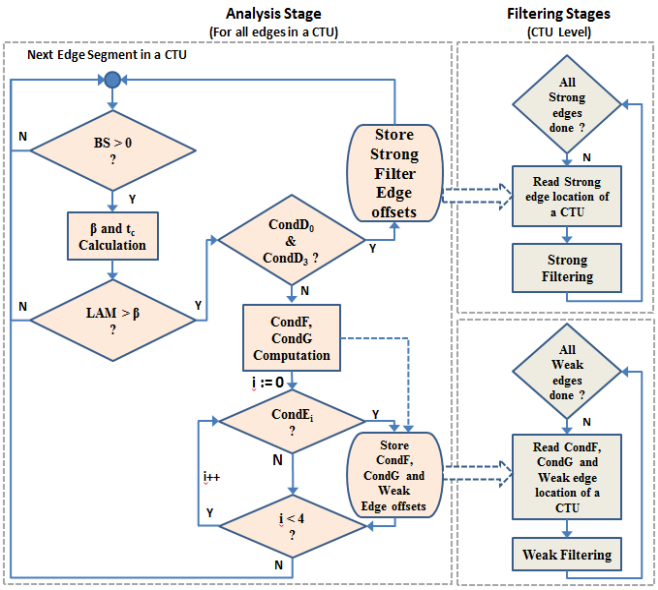

Figure 4. Proposed Algorithm Partitioning

Table 1. Percentage improvement in CPI and IPC for proposed approaches compared to branch predication

Resource utilization	With Algorithm Partitioning	With Algorithm Partitioning + Intrinsic
Improvement in CPI	65 %	82 %
Improvement in IPC	57 %	86 %

4.2. Application Specific Intrinsics

To further accelerate the performance of HEVC deblocking on RP, we propose a generic intrinsic for core deblocking operation. A generic computational pattern of deblocking filter across various video codec standards (H.264, VP8, VC1, HEVC and VP9) is used to form this intrinsic. The schematic diagram of proposed intrinsic is shown in Figure 5. Due to hardware implementation, the deblocking intrinsic takes lesser number of cycles to produce output as compared to its software implementation. Also, this ASI helps the CGA compiler to achieve denser instruction scheduling. Benefit of intrinsic usage is shown in Table 1.

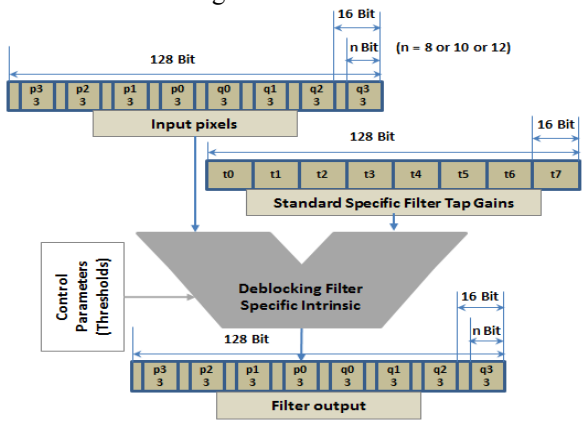

Figure 5. Input output structure of deblocking intrinsic

5. EXPERIMENTAL RESULTS

We have implemented the proposed approaches on HM-15.0 compliant HEVC decoder for RP. We created three different versions of the decoder; one with branch predication, second with algorithm partitioning and the third with both algorithm partitioning and intrinsic approaches. All three versions are tested using 4K UHD input streams with different bit-rates. Time (millisecond) spent in deblocking module alone is shown in Table 2. As it is evident from Table 2, significant improvement (around 90%) of processing time is achieved through our proposed approach. Figure 6 illustrates the percentage improvement in deblocking performance as compared to a simple branch predication based solution for 3 different bit-rates. The intermediate buffer to hold offset information for all vertical edge elements of a 64x64 CTU needs only 1408 Byte [1152 Byte for Luma, 256 Byte for Chroma] and is easily manageable in any embedded system.

Table 2. Milliseconds spent in deblocking module for different 4K UHD 4:2:0 inputs with 8-bit/pixel bit-depth, 60 frames/second frame-rate and 30 Mbps bit-rate

Test inputs	With Branch Predication	With Algorithm Partitioning	With Algorithm Partitioning + Intrinsic
Kimono	1517.7	433.6	139.6
NHK	1651.5	485.6	131.2
Discovery	1754.4	548.3	161.2
Gallery	1588	441.1	152.1

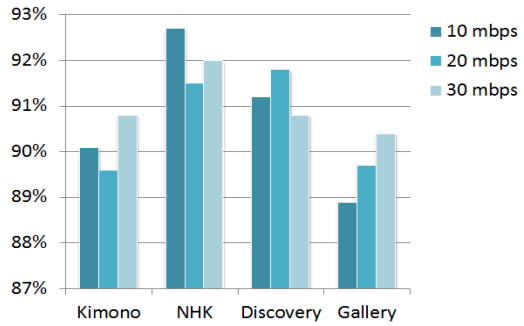

Figure 6. Percentage improvement for different bit-rates with proposed dual approach compared to branch predication

6. CONCLUSION

In this paper, we present a reconfigurable processor based solution for handling computational load of deblocking algorithm used in recent video coding standards like HEVC. The proposed solution employs a dual approach. At one hand, it involves adjustment of processor capability with the generic computational pattern of the algorithm. On the other hand, the algorithm design and software implementation are modified for maximal utilization of architectural features and resources. Our approach helps to achieve more than 10x improvement in deblocking performance and results in processing time in the order of 140 ms for 4K UHD HEVC (60 fps, 30 mbps) stream.

7. REFERENCES

[1] "Information Technology - Generic Coding of Audio-Visual Objects - Part 2: Visual," MPEG-4 standard, ISO/IEC/ JTC 1/SC29/WG 11 N 2688, Seoul, March (1999).

[2] ITU-T VCEG and ISO/IEC MPEG, "Advanced video coding for generic audiovisual services", ITU-T Recommendation H.264 and ISO/IEC 14496-10(MPEG-4 AVC), Version 5, Feb. (2009).

[3] ITU-T Recommendation H.265 and ISO/IEC 23008-2, ITU-T and ISO/IEC JTC 1, High Efficient Video Coding (HEVC), Apr. 2013.

[4] "SMPTE Standard: VC-1 Compressed Video Bitstream Format and Decoding Process", SMPTE 421M-(2006).

[5] De Simone, Francesca, et al. "Performance analysis of VP8 image and video compression based on subjective evaluations." SPIE Optical Engineering + Applications. International Society for Optics and Photonics, 2011.

[6] Mukherjee, Dipankar, et al. "The latest open-source video codec VP9-an overview and preliminary results." Picture Coding Symposium (PCS), 2013. IEEE, 2013.

[7] E. Ozcan, Y. Adibelli, and I. Hamzaoglu, "A high performance deblocking filter hardware for high efficiency video coding," Consumer Electronics, IEEE Transactions on, vol. 59, no. 3, pp. 714–720, 2013.

[8] W. Shen, Q. Shang, S. Shen, Y. Fan, and X. Zeng, "A high-throughput VLSI architecture for deblocking filter in HEVC," in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, 2013, pp. 673–676.

[9] L. Yan, Y. Duan, J. Sun, and Z. Guo, "Implementation of HEVC decoder on x86 processors with SIMD optimization," in Visual Communications and Image Processing (VCIP), 2012 IEEE, 2012, pp. 1–6.

[10] A. M. Kotra, M. Raulet, and O. Deforges, "Comparison of different parallel implementations for deblocking filter of HEVC," in Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, 2013, pp. 2721–2725.

[11] De Souza, Diego F., Nuno Roma, and Leonel Sousa. "Cooperative CPU+GPU deblocking filter parallelization for high performance HEVC video codecs." Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014.

[12] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 22, no. 12, pp. 1649–1668, 2012.

[13] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 22, no. 12, pp. 1685–1696, 2012.

[14] A. Norkin, G. Bjøntegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. V. der Auwera, "HEVC deblocking filter," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 22, no. 12, pp. 1746–1754, 2012.

[15] Page, Ian. "Reconfigurable processor architectures." Microprocessors and Microsystems 20.3 (1996): 185-196.

[16] Rabaey, Jan M. "Reconfigurable processing: the solution to low-power programmable DSP." Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on. Vol. 1. IEEE, 1997.

[17] Bondalapati, Kiran, and Viktor K. Prasanna. "Reconfigurable meshes: Theory and practice." Reconfigurable Architectures Workshop, International Parallel Processing Symposium. (1997).

[18] Mei, Bingfeng, et al. "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix." Field Programmable Logic and Application. Springer Berlin Heidelberg, 2003. 61-70.

[19] El-Ghazawi, Tarek, et al. "The promise of high-performance reconfigurable computing." Computer 2 (2008): 69-76.

[20] Li, Yanbing, et al. "Hardware-software co-design of embedded reconfigurable architectures." Proceedings of the 37th Annual Design Automation Conference. ACM, (2000).

[21] Barat, Francisco, Rudy Lauwereins, and Geert Deconinck. "Reconfigurable instruction set processors from a hardware/software perspective." Software Engineering, IEEE Transactions on 28.9 (2002): 847-862.

[22] Dimitroulakos, Grigoris, Michalis D. Galanis, and Costas E. Goutis. "Performance improvements using coarse-grain reconfigurable logic in embedded SOCs." Field Programmable Logic and Applications, 2005. International Conference on. IEEE, (2005).

[23] Bouwens, Frank, et al. "Architectural exploration of the ADRES coarse-grained reconfigurable array." Reconfigurable Computing: Architectures, Tools and Applications. Springer Berlin Heidelberg, 2007. 1-13.

[24] Joon Ho Song, Won Chang Lee, Doo Hyun Kim, Do-Hyung Kim and Shihwa Lee, "Low-Power Video Decoding System Using a Re-configurable Processor", IEEE Conference on Consumer Elec-tronincs (ICCE), pp. 532-533, (2012).

[25] Minwook Ahn, Soojung Ryu and Jeongwook Kim, "The Acceleration of Various Multimedia Applications on Reconfigurable Processor", IEEE Conference on Consumer Electronics (ICCE), pp. 238-239, (2013).

[26] Kim, Changmoo, et al. "ULP-SRP: Ultra Low-Power Samsung Reconfigurable Processor for Biomedical Applications." ACM Transactions on Reconfigurable Technology and Systems (TRETS) 7.3 (2014): 22.

[27] Taylor, Ryan, and Xiaoming Li. "Software-based branch predication for AMD GPUs." ACM SIGARCH Computer Architecture News 38.4 (2011): 66-72.