Low-power, High-throughput Deblocking Filter for H.264/AVC

Muhammad Nadeem[1], Stephan Wong[1], Georgi Kuzmanov[1], Ahsan Shabbir[2],
Muhammad Faisal Nadeem[1] Fakhar Anjam[1]
[1]Delft University of Technology, Delft, The Netherlands
{M.Nadeem, J.S.S.M.Wong, G.K.Kuzmanov, M.F.Nadeem, F.Anjam}@tudelft.nl
[2]Eindhoven University of Technology, Eindhoven, The Netherlands.
A.Shabbir@tue.nl

ABSTRACT

In this paper, we present a low-power, high-throughput hardware implementation of deblocking filter core in H.264/AVC for battery-powered multimedia electronic devices. The hardware implementation is based an optimized deblocking filter algorithm with 50% less number of addition operations. The evaluation of full or partial filtering skip scenarios is employed at an early stage in the filter processing chain to avoid un-necessary operations. Moreover, independent processing blocks are identified and are implemented with gated clock. Thus an efficient control block to activate/deactivate these independent processing blocks dynamically and pipeline implementation enable us to achieve low-power at one hand and high-throughput design for deblocking filter on the other. Experimental results suggest that the dynamic power consumption is reduced up to 50%, when compared with state-of-the-art designs in the literature. The deblocking filter core consumes 43 mW dynamic power on a Xilinx Virtex II FPGA and consumes 16.36 µW, when synthesized using 0.18µm CMOS standard cell library. The FPGA implementation on Virtex II can work at 76 MHz whereas the maximum operating frequency for 0.18µm process technology is 200 MHz. Our deblocking filter hardware implementation can easily provide real-time filtering operation for full-HD video format (1920×1080) @ 30 fps with an operating frequency as low as 59 MHz.

Keywords

Low-power, High-throughput, H.264/AVC, Deblocking filter.

1. INTRODUCTION

The latest video coding standard H.264/AVC [1], jointly developed by ITU-T and ISO/IEC MPEG, significantly out-performs the previous video coding standards (like H.263 and MPEG-2/4) in terms of bit-rate reduction. H.264/AVC offers perceptually the same video quality with at least 2 times better compression when compared with MPEG-2 [2], [3] and up to 30% better compression when compared with H.263+ and MPEG-4 Advanced Simple Profile (ASP) [4]. The H.264/AVC video coding standard has already been adopted for a wide range of applications, from low bit-rate mobile video to high-definition TV. This improved video quality and reduced bit-rate, however, is achieved at the cost of increased complexity of the video coding algorithms in H.264/AVC.

Like other block-based video coding standards, H.264 also suffers from the blocking artifacts in the reconstructed video. The block-based discrete cosine transform (DCT) and block-based motion compensation (MC) are two major sources of these blocking artifacts [5] [6]. The H.264/AVC uses an adaptive in-loop deblocking filter to remove such artifacts in the reconstructed video frames. According to the analysis of run-time profile of the H.264 decoder sub-functions, the deblocking filter consumes about one-third of the computational resources [7]. These demanding characteristics suggest a high-throughput and low-power hardware implementation for such a deblocking filter for portable battery-powered multimedia electronic devices like mobile phones, digital cameras and PDAs etc.

In recent years, a lot of research have been conducted in designing different architectures for deblocking filter with main focus on very high-throughput to meet the computational requirement of real-time processing of high definition videos for Digital Television (DTV) application domain. However, not much work has been reported in the literature for low-power with significantly high-throughput deblocking filter hardware implementation, for battery-powered electronic devices such as mobile phones.

With this work, we address the dynamic power constraints and propose a design for H.264/AVC deblocking filter core which consumes less power at one hand and provides significantly higher throughput on the other, when compared with other state-of–the-art proposals in literature.. More specifically, in this paper, we provide the following contributions:

- A proposal to split up the filter processing into independent processing units based on macroblock (*MB*) components (luminance vs chrominance pixels), filtering modes and conditional filtering within these filtering modes.
- Early evaluation of all filter conditions and a proposal for provision to skip the filtering process (full or partial) on the basis of these evaluated filter controls.
- Use of optimized deblocking filter algorithm with 50% reduction in arithmetic operations for both the filtering modes (*Strong Filter Mode* and *Weak Filter Mode*) in H.264/AVC.

The rest of the paper is organized as follows: Related work is presented in section 2. Section 3 provides a brief overview of the deblocking filter algorithm. Algorithmic optimizations, with low-power hardware implementation of deblocking filter point of view, are described in section 4. Section 5 explains the proposed design of the deblocking filter. Experimental results are provided in section 6 and section 7 concludes this paper.

2. RELATED WORK

Several different architectures and their hardware implementations have been presented in literature for last few years. Most of these architectures targets the HDTV application domain and therefore very high-throughput is the primary focus to meet the real-time processing requirements for such applications. Very few of them, however, targets to meet the requirements of battery-powered multimedia application domain, like mobile phones, where reasonably high-throughput along with significantly low-power designs are key to success.

For instance, in [12][13], the authors presented two different low power hardware implementations for deblocking filter. Their design consists of 2 stage pipeline data path. The first pipeline stage includes a 12-bit adder and two shifters. The second stage includes a 12-bit comparator, several two's complement units and multiplexers for conditional branching. The dynamic power consumption of the deblocking filter is significantly reduced with such a simplified data path. However, because of lack of sufficient storage registers, the intermediate results are to be stored and fetched from the memory several times during filtering processing. This results in a large number of cycles to process a single MB (5248 cycles/MB). Thus the throughput of the filter is reduced quite significantly. Moreover, no attempts are made, or at least reported to remove the redundancy in the filter algorithm. The hardware implementation works at 72 MHz in a Xilinx Virtex II FPGA and can process up to Common Intermediate Format (CIF, 352 × 288) video frame resolution only.

Similarly Byung-Joo et al. also proposed a low-power deblocking filter in [14]. They attempt to reduce the power by skipping filtering process completely or partially by exploiting the relationship between block level encoding parameters and filtering conditions. The proposed design, however, requires 2272 cycles to process one MB and therefore can process CCIR601 video frame format (704 × 576) in real time. The proposed design though provides better throughput, but still do not meet the real time processing requirement of a wide range of portable multimedia devices where standard-definition (SD, 720×480), high-definition (HD, 1280×720) or even higher resolutions are to be supported.

Very recently, Nam Thang et al.[15] also proposed a low-power, high-throughput 4-stage pipelined architecture, implemented using 0.18 µm CMOS standard cell technology, for deblocking filter in H.264/AVC. Four stages pipeline implementation of deblocking filter and hybrid processing order enable them to process a single macroblock in 192 cycles. While working at 220 MHz, it can provide a throughput up to 1146K MB/sec. The filter core consumes 34.8 µW for QCIF frame resolution at an operating frequency of 220 MHz. The proposed design uses five 4×4 block buffers for intermediate result storage during MB processing to achieve high throughput and attempts to reduce power consumption by adopting clock gating for these blocks. However no such attempts are made for 2nd and 3rd pipeline stages where most of the filter processing take place. Moreover, no partial or full filtering process skipping scenarios are taken care of in the proposed architecture to reduce the dynamic power.

3. DEBLOCKING FILTER ALGORITHM OVERVIEW

This section provides a brief overview of the adaptive deblocking filter algorithm in H.264/AVC. The detailed description of the algorithm can be found in [1].

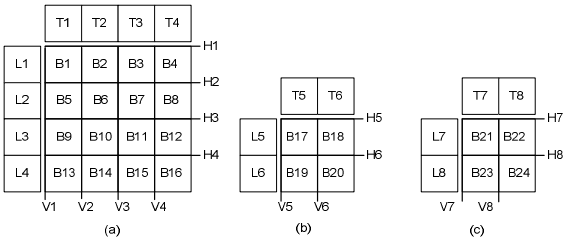

Fig 1. Vertical and Horizontal 4×4 block edges in a macro-block.

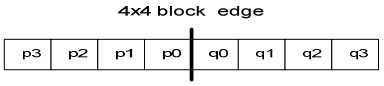

Fig. 2 Convention to describe pixels across Vertical/Horizontal edge.

Pixel Level Filter Control Flags

FSF = (Bs ≠ 0) And (|p0-q0|<α) And(|p1-p0|<β) And (|q1-q0|<β) (1)
SFF_P = (Luma edge) And (|p2-p0|<β) And (|p0-q0|<((α/4)+2)) (2)
SFF_Q = (Luma edge) And (|q2-q0|<β) And (|p0-q0|<((α/4)+2)) (3)
WFF_P = (Luma edge) And (|p2-p0|<β) (4)
WFF_P = (Luma edge) And (|q2-q0|<β) (5)

Strong Filter (BS= 4)

p0' = (p2 + 2*p1 + 2*p0 + 2*q0 + q1 + 4) >> 3 (6)
p1' = (p2 + p1 + p0 + q0 + 2) >> 2 (7)
p2' = (2*p3 + 3*p2 + p1 + p0 + q0 + 4) >> 3 (8)
p0'' = (2*p1 + p0 + q1 + 2) >> 2 (9)

q0' = (p1 + 2*p0 + 2*q0 + 2*q1 + q2 + 4) >> 3 (10)
q1' = (q2 + q1 + q0 + p0 + 2) >> 2 (11)
q2' = (2*q3 + 3*q2 + q1 + q0 + p0 + 4) >> 3 (12)
q0'' = (2q1 + q0 + p1 + 2) >> 2 (13)

Weak Filter (BS= 3)

delta = clip(-c1, c1, ((((q0-p0)<<2) + (p1-q1) +4)>>3)) (14)
p0' = clip(0,255,p0 + delta) (15)
q0' = clip(0,255,q0 - delta) (16)

p1' = p1+clip(-c0,c0,(p2+((p0+q0+1)>>1)-(2*p1))>>1) (17)
q1' = q1+clip(-c0,c0,(q2+((p0+q0+1)>>1)-(2*q1))>>1) (18)

Fig. 3(a) Pixel level Filter Controls equations, (b) Filter equations for Strong Filter Mode , (c) Filter equations for Weak Filter Mode

A macroblock at 4 × 4 block level is depicted in Fig. 1. Blocks B1 through B16 belong to Y (Luma) component of the current MB whereas B17-B20 and B21-B24 are from U, V (Chroma) components of the same MB respectively. The blocks (T1-T8) are the top neighbor 4×4 blocks and (L1-L8) represent the left neighbor 4×4 blocks of a MB in Figure 1. The convention to describe the pixels across the horizontal and vertical edge is depicted in Figure 2. The bold line between pixels p0 and q0 is either a vertical or horizontal edge between two adjacent 4×4 blocks. Pixels q0-q3 represents the pixels in the current 4×4 block whereas pixels p0-p3 are from corresponding left or top neighbor 4×4 block across the vertical or horizontal edge respectively.

The filtering operation is performed on MB basis after reconstruction of the picture, with all MBs in a picture processed in order of increasing MB address. The filtering process of current MB also requires pixels from left and top neighbor macroblocks in order to filter the left most edges (V1, V5 and V7) and the top most edges (H1, H5 and H7), respectively.

The H.264/AVC deblocking filter is a highly adaptive filter. The *boundary strength* (*BS*) parameter controls the filtering strength at 4×4 block level and varies from 4 (strong filtering) to 0 (no filtering). For BS values 1, 2 and 3, a weak filtering process is invoked on the pixels across the block edge. The decision process

to compute the *BS* value is given in [1]. The selected filtering mode is turned ON or OFF depending upon the value *of Filter Sample Flag (FSF).* The *FSF* is derived using Eq. (1) in Fig. 3(a).

In *Strong Filter Mode* (*BS*=4), at most 3 pixels are modified on either side of the edge. The new pixel values for the pixels on left or top side of the edge (p-pixels) are computed using Eq. (6) - Eq.(8) Figure 3(b), provided the *Strong Filter Flag* for p pixels (*SFF_P*) is set. If the *SFF_P* flag is NOT set, only one pixel (p0) is filtered and the new value for this pixel is derived by Eq.(9). Similarly, the pixels in the current block (q-pixels) are computed using Eq.(10) – Eq.(12) Figure 3(b), provided the corresponding *Strong Filter Flag* (*SFF_Q*) is set. In case *SFF_Q* flag is NOT set, only one pixel (q0) is filtered using Eq. (13) and rest of the pixels remain unchanged. The values for *SFF_P* and *SFF_Q* flags are derived from Eq.(2) and Eq.(3), respectively.

In the *Weak Filter Mode* (*BS* = 1,2 or 3), at most 2 pixels on either side of the edge are filtered. The new filtered values for the pixels p0 and q0 across the edge are derived from Eq. (15) and Eq. (16) respectively, whereas Eq. (17) and Eq. (18) are used to provide the filtered values for pixels p1 and q1, if the corresponding *Weak Filter Flag* (*WFF_P, WFF_Q*) is set. In case the *Weak Filter Flag* is NOT set, the filtering operation for these pixels is turned OFF. The values of *WFF_P* and *WFF_Q* are derived from Eq. (4) and Eq. (5), as shown in Figure 3(a). No filtering is applied for *BS* = 0

4. OUR APPROACH TO REDUCE THE DYNAMIC POWER CONSUMPTION

In this section, we shall elaborate on the optimizations carried out at the algorithm level and shall also describe other design level considerations along with their justification and/or motivation for low-power hardware deblocking filter implementation.

Processing Units for Luma/Chroma Components: The H.264/AVC supports YUV 4:2:0, YUV 4:2:2 and YUV 4:4:4 video frame formats. In case of YUV 4:2:0 , 33% of the pixels of a macroblock belong to chrominance component. This number increases to 50 %, and 66% for YUV: 4:2:2 and YUV 4:4:4 video frame formats respectively. Since the chrominance filtering is much cheaper than the luminance filtering in terms of required arithmetic operations (13 Vs 45), it is potentially beneficial to divide the processing units between the chrominance and the luminance sub-blocks**.** These separate luma-, chroma processing units, therefore, can be enabled or disabled independent of each other and depending upon the *ChromaEdgeFlag* status.

Processing Units for Strong / Weak Filter Modes: Within the luminance or the chrominance components of a MB, the filtering mode for the pixels across an edge is determined on the basis of *BS* value for that edge. Since only one mode is active at any time during processing, we can implement the arithmetic operations for both filter modes in separate processing units employing clock gating. Thus, depending on the filtering mode, either of these processing units (Strong or Weak filtering units) can be deactivated dynamically to reduce the power consumption.

Full or Partial Filter Process Skip: If the *BS* value of an edge is zero, the filtering process for that 4×4 block edge can be completely skipped. Similarly, if the *FSF* is not set because of local pixel values across the edge and corresponding 4×4 block level filter control threshold values, the filtering process for current pixel-row or pixel-column (Vertical or Horizontal filtering

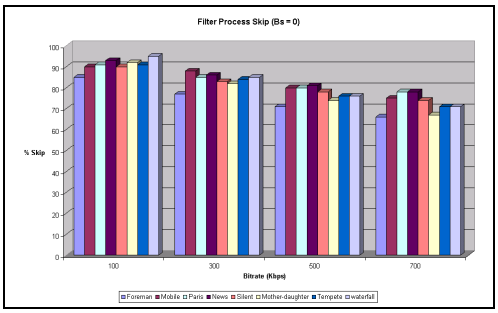

Fig. 4. Full filtering process skip (Bs = 0 case)

cases) within the 4×4 block can also be skipped.

Fig. 4 illustrates the % cases when the complete filter process can be skipped because of the *BS* value being zero for a variety of video test sequence encoded at different bit-rate values (100 – 700 Kbps). There is no need to even compute the *FSF* and other conditional flags for such a 4×4 block case. Similarly, Fig. 5 suggests that even when the *BS* value is non zero, there is quite a significant number of cases when the *FSF* is False and therefore filtering can be skipped partially for current pixel-row or pixel-column within the 4×4 block..

We propose to implement the processing for *FSF* along with other conditional filter flags in the first pipeline stage as independent processing unit with gated clock. This unit is activated only if the *BS* value is non zero. Thus, the filtering process for the whole 4×4 block can be skipped for a scenario when boundary strength value is zero. In case the *BS* value is not zero, the *FSF* along with other conditional flags can be used to generate the appropriate control signals for the rest of the processing units in the 2nd and 3rd pipeline stage of deblocking filter to partially skip the filtering process and thus reduce the dynamic power

Processing Units for Conditional Filtering: In order to provide more insight for the conditional filtering scenario, we computed the percentage of cases when either of the conditional filter flags (*WFF_P, WFF_Q, SFF_P, SFF_Q*) for each of the filtering modes (Strong and Weak Filter Modes) is false. The data is generated for various video test sequences encoded at 100 – 700 Kbps bitrate. As suggested in Fig. 6 and 7, there are quite a significant number of cases during processing of any video frame when the conditional filtering is OFF. Therefore, we propose to split the processing for these cases into separate blocks supported by clock gating. This shall help to deactivate any of these processing blocks to reduce the dynamic power consumption when any of the conditional filtering flags is not set.

Optimized Deblocking Filter Algorithm: In order to achieve the low-power deblocking filter implementation, the removal of arithmetic operations redundancy within the filtering algorithm itself is also very important. This is achieved by intelligent decomposition of the filter kernels so that the common intermediate results can be used as much as possible and thus reduces the total number of arithmetic operations to carry out filtering process. In the original filter equations in H.264/AVC video coding standard [1], 49 addition operations along with 5 clip operations are required in total for full pipeline implementation of the algorithm.

We proposed a novel decomposition of the filter kernels to remove the arithmetic operations redundancy in our previous work [17]. The proposed optimization of the filter equations

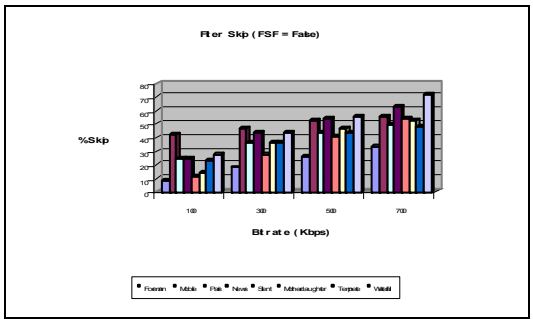

Fig. 5. Partial filtering process skip (Bs = 1, 2, 3 or 4)

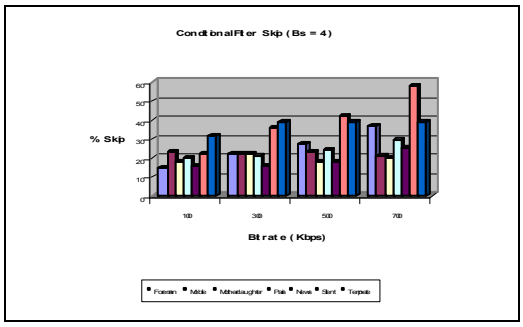

Fig. 6 Conditional filtering process skip (Bs = 4)

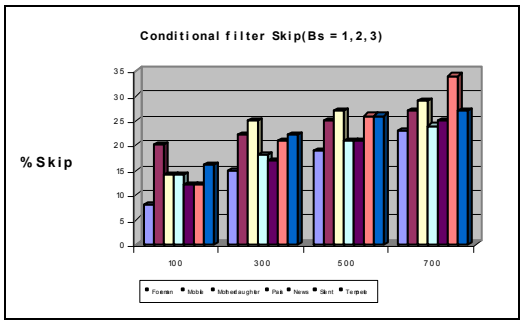

Fig. 7 Conditional filtering process skip (Bs = 1, 2, or 3)

reduces the total number of adder instances from 49 to 24 [17]. This more than double reduction of addition operations does not only pay off in terms of less area requirement for its implementation but also helps to reduce the signal activity in the combinatorial logic between different pipeline stages. The later facilitates the reduction of power consumption for the entire hardware deblocking filter unit.

5. HARDWARE DEBLOCKING FILTER ORGANIZATION.

In this section, we shall first introduce the top level design of deblocking filter implementation. Later, the internal design of the deblocking filter core module shall be elaborated. The independent processing blocks proposed in section 4 for low-power design shall also be indentified and explained within the deblocking filter core module.

Top Level Design :

The block diagram of the hardware deblocking filter is depicted in Fig 8. The design is based on a single filtering unit (*DBF core*), and two RAM units (*Filter Control Data RAM, Neighbor 4×4 Block Data RAM*). Since the filter process is identical in both the

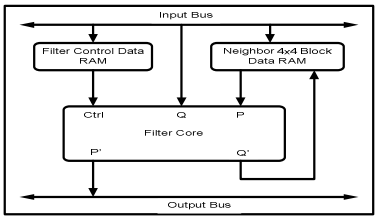

Fig. 8 High level organization of the deblocking filter hardware implementation

directions (Horizontal and Vertical filtering). Therefore, a single filtering unit, *DBF core*, is used to filter the current *MB* data across all vertical edges followed by the filtering across all horizontal edges. During the filtering process of vertical or horizontal edge, the pixel data across the edge is also required from left or top 4×4 neighbor block respectively. This left or top neighbor 4×4 block pixels data is stored in the *Neighbor 4×4 Block Data RAM*. Similarly, the filter control threshold parameters (*Alpha, Beta, Tc0, BS* values etc) at 4×4 block level are stored in the *Filter Control Data RAM* unit. The *Neighbor 4×4 Block Data RAM* unit is composed of 32 x 32-bit dual port SRAM whereas the *Filter Control Data RAM* is composed of 16 x 32-bit SRAM. All the data paths in Fig. 8 are 32-bit wide.

The 4×4 block level filter control data along with the neighbor pixel data is first transferred to the deblocking filter and is stored in the corresponding RAM units. Later, the filtering process starts as soon as the first pixel-row of the current 4×4 block *B1* is received at *Q* input (Fig. 8) of the *DBF core* from the external memory via *input bus*. The corresponding neighbor pixel data across the edge (*P* data in Fig. 8), is provided to the *DBF core* from *Neighbor 4×4 Block Data RAM* unit. The partially filtered macroblock data (*Q'*) of the current 4×4 block is temporarily stored in the *Neighbor 4×4 Block Data RAM* unit to be used as neighbor data for the next 4×4 block. Whereas the filtered data *P'* is transferred to the external frame memory via *Output bus*.

Deblock Filter Core Module.

The *DBF core* is the main processing block where the entire filter related conditional flags are computed and the appropriate filter kernels are used to filter the input pixel data. The design choices made for this unit plays an important role in determining the throughput, power consumption and on-chip area requirement for its implementation.

Pipeline processing is an effective implementation choice to achieve high throughput. In order to meet the real time processing requirement of up to full-HD (1920×1080) resolution video frame format, while keeping the operating frequency as low as possible with low-power design point of view, we implement the *DBF core* in pipeline fashion. For full-HD video @ 30 fps, a throughput of 243 KMB/sec is required to meet the real time filter processing requirement. If we filter the current MB on-the-fly, then for 192, 4×4 block edges per MB, the minimum operating frequency required shall be ~47 MHz. Thus, the *DBF core* with a 3 stage pipeline implementation of the complete filtering algorithm can easily meet these throughput requirements. The block diagram of the *DBF core* is depicted in Fig. 9.

The straightforward pipeline implementation of deblocking filter shall consume high dynamic power because of parallel processing in different pipeline stages. Therefore, we implement the *DBF core* based on the optimized algorithm mentioned in section 4.

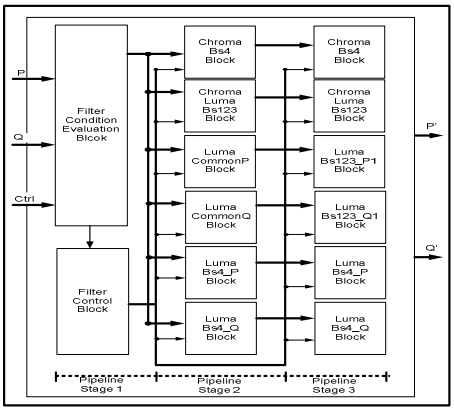

Fig. 9 Block diagram of the *DBF core*.

The 1st pipeline stage of *DBF core* implements the processing for the evaluation of all the filter control flags (*FSF, WFF_P, WFF_Q, SFF_P and SFF_Q*). These flags are used in *Filter Control Block* pipeline stage 1, to generate the appropriate control signals for the rest of the processing blocks in the subsequent pipeline stages of *DBF Core.*

The 2nd and 3rd pipeline stages of DBF Core are based on processing blocks identified in previous section with gated clock. *ChromaBs4Block* in Fig. 9, as the name suggests, implements the filter processing for chrominance component of the macroblock for *Strong Filter Mode i.e.* Eq.(9) and Eq.(13) in Fig 3(b). The filtering process of pixels p0 and q0 is identical for the luma / chroma components of the MB for Weak Filter Mode case. Therefore, for this case, the *ChromaLumaBs123Block* implements all the arithmetic processing for pixels p0 and q0. i.e Eqs. (14) – (16) in Fig 3(c). Similarly, the overlapped data path for conditional filtering in strong and weak filtering modes is implemented in *LumaCommonPBlock and LumaCommonQBlock* whereas the *LumaBs4_PBlock, LumaBs4_QBlock* implements the rest of the processing for strong filter mode case [17].

In case of *Strong* or *Weak Filter Modes* for chroma component of the *MB*, one can see from Fig. 9, either the *ChromaBs4Block* or *ChromaLumaBs123Block* is to be enabled respectively and rest of all the processing blocks in 2nd and 3rd pipeline stages are deactivated for 33% of the processing time in a MB. Similarly, for the other filtering scenarios, the filter Control Block in 1st pipeline stage generates the appropriate control signals to activate only the respective processing block(s) in the 2nd and 3rd pipeline stage. If the *BS* value is zero, all the processing blocks are deactivated and the unfiltered pixels are transferred to the output of the DBF core via by-pass stage (Not shown in Fig. 9). The experimental results show that our hardware implementation of deblocking filter core requires significantly less dynamic power, when compared with state-of-the-art low-power deblocking filter in the literature.

6. DESIGN EVALUATION

The design proposed in this paper was described in VHDL. The RTL simulation results were verified with that of the JM13.2 reference software [16]. For FPGA implementation, VCD file was generated to record signal activity using ModelSim 6.5 tool for number of video test sequences. Later this VCD file was used to estimate dynamic power consumption using Xilinx XPower tool for various video test sequences. Similarly for ASIC

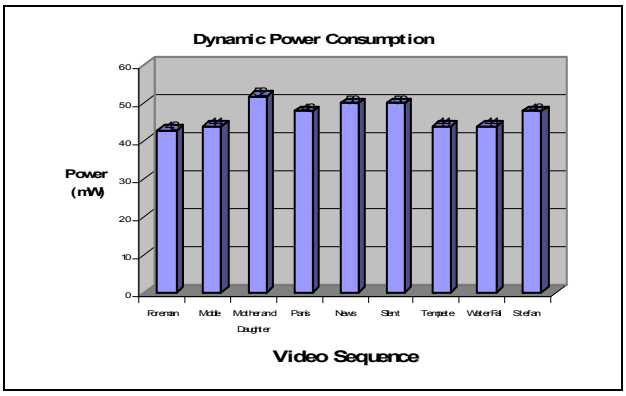

Fig. 10 Dynamic Power Consumption

TABLE I. FPGA RESOURCES USAGE

HW Resources	Usage
Slices	552
DFF	604
4 input LUTs	662

implementation, the same design was synthesized using Synopsys Design Compiler (ver v2002, rev05) under UMC 0.18 µm CMOS standard cell library (v1.5). The dynamic power consumption was estimated for signal activity recorded in the VCD file for gate level simulation with "foreman" video test sequence.

The dynamic power consumption for a number of video test sequences on Xilinx Virtex II FPGA device is illustrated in Fig 10, where as the hardware resources used are provided in Table 1. Since the dynamic power consumption for designs already in the literature are provided either for FPGA implementation or on ASIC for different set of parameters. Therefore, a detailed comparison for both cases is provided in rest of this section.

FPGA Implementation Comparison: Mustafa et.al. [12] provided the power consumption results of their design for "Forman" video test sequence with CIF(352 x 288) resolution, on 2V8000FF1157 Xilinx Virtex II FPGA device with speed grade 5. For a correct comparison, we also implemented our design on the same FPGA device and computed the power consumption using the same video test sequence. The test results indicate that our design consumes only 43 mW (Vs 85.76 mW). Thus our deblocking filter implementation consumes half of the dynamic power consumed by [12], for the same video test sequence on the same FPGA device. As far as the throughput is concerned, with maximum operating frequency of 76 MHz and 192 cycles per MB, we can provide a maximum throughput of 395 KMB/sec. Hence, our hardware deblocking filter implementation can process full-HD(1920×1080) at 30 fps with an operating frequency as low as 59 MHz. In contrast, the deblocking filter implementation in [12] cannot provide real-time processing beyond CIF(352×288) video frame format. Further comparison is provided in Table II.

ASIC Implementation Comparison: NamThang et al. [15] reported power consumption for their implementation for 0.18 µm CMOS standard cell library. Their deblocking filter core consumes 34.8 µW for processing a QCIF video frame @ 30 fps. We synthesized our design using Synopsys Design Compiler (ver v2002, rev05) under UMC 0.18 µm CMOS standard cell library

Table II. Comparison Table

Ref	Clks/Mb	Max Frequency(MHz)		Throughput (KMB/sec)		Dynamic Power Consumption	
		FPGA VirtexII	***0.18 um CMOS***	***FPGA VirtexII***	***0.18 um CMOS***	***FPGA VirtexII***	***0.18 um CMOS***
[12]	5248	72 MHz	200 MHz	14	38	85.76 mW	NA
[15]	192	NA	220 MHz	-	1146	NA	34.8 µW
Prop.	192	76 MHz	200 MHz	395	1041	43 mW	16.36 µW

(v1.5). The power consumption was estimated for gate level simulation with "Forman" QCIF video test sequence @ 30fps. The estimated power consumption for our implementation is 16.36 µW (Vs 34.8 µW). Thus our design consumes 53% less power for the same process technology (0.18 µm CMOS standard cell) and for the same video test sequence (Forman QCIF @ 30 fps). As far as the throughput of our implementation for 0.18 µm process technology is concerned, the maximum operating frequency of our implementation is 200 MHz and thus can easily meet the real-time processing requirements of full-HD video frame format with an operating frequency of 59 MHz. Further comparison details are provided in Table II.

The higher throughput for our implementation of deblocking filter is mainly achieved by pipeline implementation design choice, small number of cycle to process a MB, and higher maximum operating frequency. The power consumption is mainly reduced because of optimization of filter algorithm (more than 50% reduction of addition operations), implementation at finer granularity of processing sub-blocks that can be independently deactivated during filtering process and early detection of filter process skipping conditions to deactivate the complete processing chain.

7. CONCLUSION

This paper presented a low-power, high-throughput hardware implementation of the deblocking filter core in H.264/AVC. The implementation is based on an optimized deblocking filter algorithm, where the number of addition operations to carry out the filtering process is reduced by a factor of two. This optimization not only helped to reduce the area requirement for its implementation but also reduced the power consumption during the filtering process. Furthermore, we identified the independent processing units within the optimized algorithm to be implemented separately with clock gating. Thus the dynamic power consumption is reduced by a factor of 2, when compared with state-of-the-art designs in the literature. Our deblocking filter can easily process full-HD (1920× 1080) resolution at 30 fps in real-time with an operating frequency as low as 59 MHz.

8. REFERENCES

[1] Joint Video Team of IT-T VEG and ISO/IEC MPEG, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification", ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[2] ISO/IEC JTC1/SC29/WG11, "Report of the Formal Verification Tests on AVC/H.264", doc. N6231, Waikoloa, USA, 2003.

[3] G. Sullivan, P. Topiwala and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", SPIE Conference On Applications Of digital Image Processing, vol.5558, pp. 454-474, 2004.

[4] J. Ostermann, J. Bormans, P. List, D. Maroe, M. Narroschke, F. Pereira, T. Stockhammer and T. Wedi, "Video Coding with H.264/AVC: Tools, Performance and Complexity", IEEE Circuit and Systems Magazine, vol 4, no. 1, pp. 7-28, 2004.

[5] Y. L. Lee and H. W. Park, "Loop Filtering and Post-filtering for Low bitrates Moving Picture Coding", Signal Processing: Image Communication, vol 16, pp. 871-890, 2001.

[6] P. List, A. Joch, J. Lainema, G. Bjontegaard and M. Karczewicz, "Adaptive Deblocking Filter", IEEE Transactions on Circuits and Systems for Video Technology, vol 13, no. 7, pp. 614-619, 2003.

[7] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysis", IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 704-716,2003.

[8] M. Shirasaki *et al.*, "A 45 nm Single-chip Application-and-Baseband Processor Using an Intermittent Operation Technique," in *IEEE ISSCC Dig. Tech. Papers,* pp. 156–157, Feb. 2009.

[9] M. Ito *et al.*, "A 65 nm Single-chip Application and Dual-Mode Baseband Processor With Partial Clock Activation and IP-MMU," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 83–89, Jan. 2009.

[10] S. Nomura *et al.*, "A 9.7 mW AAC-decoding, 620 mW H.264 720p 60 fps decoding, 8-core media processor with embedded forward-bodybiasing and power-gating circuit in 65 nm CMOS technology," in *IEEE ISSCC Dig. Tech. Papers,* pp. 262–263, , Feb. 2008.

[11] H. Mair *et al.*, "A 65-nm Mobile Multimedia Applications Processor With an Adaptive Power Management Scheme to Compensate for Variations," in *Dig. Symp. VLSI Circuits*, pp. 224–225, Jun 2007.

[12] M. Parlak, I. Hamzaoglu, "A Low Power Implementation of H.264 Adaptive Deblocking Filter Algorithm", Second NASA/ESA Conference on Adaptive Hardware and Systems, Aug. 2007.

[13] M. Parlak, I. Hamzaoglu, "Low Power H.264 Deblocking Filter Hardware Implementations", IEEE Transactions on Consumer Electronics, vol. 54, no. 2, pp. 808-816, May 2008.

[14] B. Kim, J. Koo, M. Hong, S. Lee "Low-Power H.264 Deblocking Filter Algorithm and Its SoC Implementation", IEEE Pacific–Rim Symposium on Image and Video Technology, LNCS 4319, pp. 771-779, Dec. 2006

[15] N. Ta, J. Youn, H. Kim, J. Choi, S. Han "Low-power high-throughput deblocking filter architecture for H.264/AVC", IEEE International Conference on Electronic Computer Technology, pp. 627-631, Feb. 2009.

[16] JVT H.264/AVC Reference Software JM 13.2.

[17] M. Nadeem, S. Wong, G. Kuzmanov, A. Shabbir, " A High-throughput, Area-efficient Hardware accelerator for Adaptive Deblocking Filter in H.264/AVC", IEEE Workshop on Embedded Systems for Real-time Multimedia , pp. 18-27, Oct. 2