HIGH SPEED SAD ARCHITECTURES FOR VARIABLE BLOCK SIZE MOTION ESTIMATION IN HEVC VIDEO CODING

Purnachand Nalluri[1,2], Luis Nero Alves[1,2], Antonio Navarro[1,2]
(nalluri@av.it.pt, nero@av.it.pt, navarro@av.it.pt)

[1]Institudo de Telecomunicações,
Pólo-Aveiro, Campus Universitário de Santiago,
3810-193 Aveiro, Portugal.

[2]Departamento de Electrónica, Telecomunicações e Informática,
Universidade de Aveiro, Campus Universitário de Santiago,
3810-193 Aveiro, Portugal.

ABSTRACT

HEVC is the latest video coding standard aimed to compress double to that of its predecessor standard H.264/AVC at the cost of increased coding complexity. Motion Estimation (ME) is one of its critical tools in the encoder whose complexity drastically increases due to the increase in coding block size to 64x64 and due to the introduction of Asymmetric Motion Partitioning (AMP). Hence it requires specific hardware architectures for real time implementation. The bottleneck of ME tool is the SAD (Sum of Absolute Difference) circuit architecture which calculates SAD between current block and reference block pixels. The present paper proposes and implements three SAD architectures in FPGA. Synthesis results show that one of the proposed architectures outperforms when compared to results of other contributions, despite supporting all block modes of HEVC.

Index Terms— Motion Estimation, SAD architecture, HEVC.

1. INTRODUCTION

HEVC (High Efficiency Video Coding) developed under joint collaboration of ITU-T VCEG and ISO/IEC MPEG, together under the name JCT-VC (Joint Collaborative Team on Video Coding) [1-2] is the latest video coding standard. It is targeted to double the video compression ratio to that of its predecessor H.264/AVC. The frame-level coding structure of HEVC consists of CUs (Coding Units) of maximum block size 64x64 pixels, and each CU can be partitioned recursively until 8x8 pixel block size. CUs consists of PUs (Prediction Units) for prediction (intra or inter) and TUs (Transform Units) for transformation of residual blocks. Each PU can be of types either intra or inter or skip or merge. Each inter PU sizes varies from 64x64 pixels to 4x4, supporting 8 partition modes at each depth level of CU as shown in Fig.1, where 2N=8 represents depth 0 of a CU, 2N=16 represents depth 1 and so on. Partitions (a) to (d) are called symmetric modes and (e) to (h) are called asymmetric partition modes (AMP).

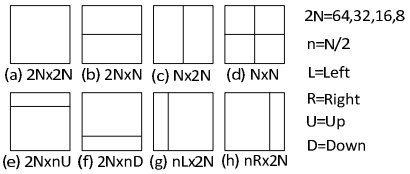


Fig.1 Inter prediction unit partition sizes in HEVC

Amongst all the tasks in video encoding, motion estimation is the most time consuming and complex task. The ME unit finds the best matched block (using a cost function) in the reference (past/future) frame search window for each and every PU block of the current frame and hence the reconstructed frame has lowest residual information for doing transformation [3]. The rate-distortion cost J is shown in (1), where λ represents Lagrange multiplier, R represents bits required to encode the motion vector difference and D represents the distortion function. One of the most widely used distortion function for ME is SAD (Sum of Absolute Difference) that can be defined as shown in (2), where CB represents current block pixels, RB represents reference block pixels and MxN is the size of the current PU block.

$$J_{MV} = D + \lambda_{MV}.R \quad (1)$$

$$SAD = \sum_{i=1}^{M}\sum_{j=1}^{N}|CB(i,j) - RB(i,j)| \quad (2)$$

The aforementioned cost should be calculated for every PU block size of each CU, ranging from 4x4 to 64x64 and for all the reference frames ranging one to four. The SAD is the core of the ME that takes most of the complexity. Each CU size has eight SAD sizes to be computed. The total SADs with their sizes in each 64x64 size CU will be 1361 including 64x64 block size, as shown in our previous work [4]. Hence the complexity of SAD is very high in HEVC and hence this operation demands real time implementation to exploit parallelism and improve system performance.

Many research works have been published in the area of SAD architectures for H.264/AVC with a maximum block size of 16x16 [5-9]. In [5], Rehman et al proposed an FPGA based SAD architecture which uses a partial product

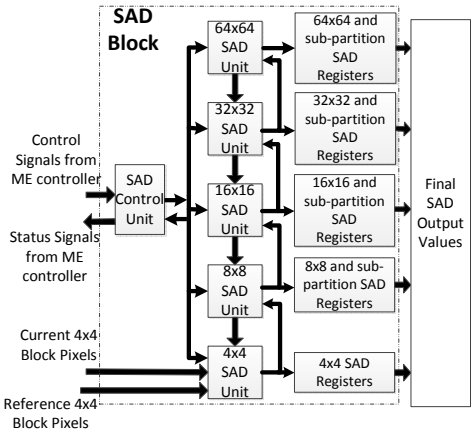


Fig. 2 Sequential SAD architecture

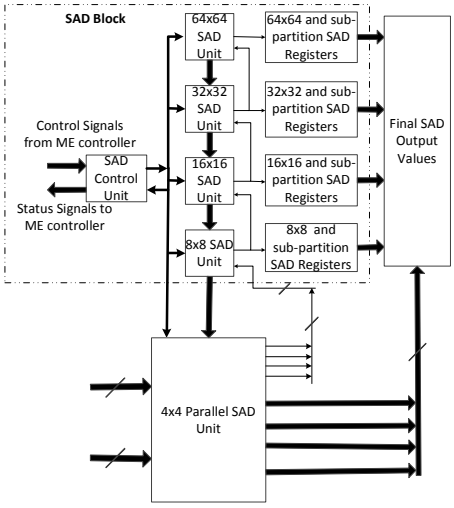


Fig. 3(a) 1-Stage parallel Architecture

reduction scheme for adding the absolute difference values of 4x4 blocks of pixels. The authors propose that multiple 4x4 blocks can be used in parallel, but when it comes for HEVC the number of 4x4 partial SAD blocks need to be very high. Furthermore, there is no explanation to efficient variable block size SAD architecture implementation in [5]. In [6],[7], the authors proposed and optimized partial propagate SAD and SAD tree architectures. In partial propagation SAD architecture, each row of all 4x4 pixel blocks SAD is calculated every clock cycle and at the end of 4th clock cycle they are summed up. For H.264/AVC (maximum block size 16x16) it takes 16 clock cycles to get first sample of all 4x4 partial SADs and then it takes four clock cycles for subsequent 4x4 partial SADs set. While this implementation takes 256 PEs (Processing Elements) in H.26/AVC (with 16x16 block size), it takes 4k PEs in HEVC (with 64x64 block size). Another problem with partial propagate SAD architecture is that it is only efficient

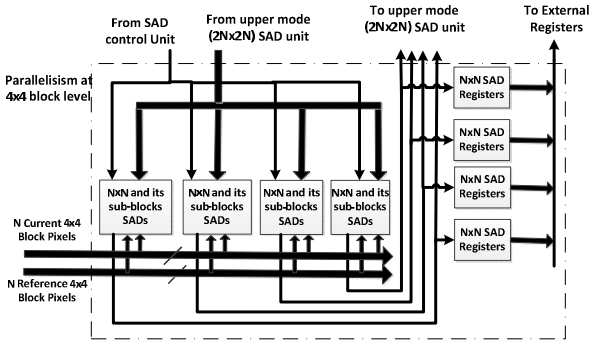


Fig 3(b) Internal architecture of NxN parallel SAD unit (N=4,8)

for full search ME algorithm since neighboring reference blocks can be pipelined.

In [8],[9], the authors propose SAD tree architecture, where all the 4x4 block partial SADs are calculated in parallel in each clock cycle and summed up in the next subsequent cycles. This architecture reduces the pipeline delay but it also takes 4k PEs in HEVC. Hence a controlled mechanism for inserting the number of parallel levels should exist to reduce the number of PEs while being able to efficiently compute in parallel. The present paper proposes a 64x64 SAD architecture, where the number of parallel levels can be supervised in the initial stage of design and traded off with the amount of hardware resources utilized. In our previous work [4], we designed a SAD architecture by using a single parallel stage with four 4x4 SAD computation units. In this paper, we increase the number of parallel levels up to 8x8 computation unit. Section II explains the proposed architecture and its operation. Section III explains the synthesis results and their analysis and finally Section IV draws concluding remarks.

2. PROPOSED SAD ARCHITECTURE

The present paper explains three possible configurations with different levels of parallel stages in each level of architecture. Fig. 2 shows the sequential architecture with no parallel stages. Fig. 3(a) and Fig. 3(b) show the proposed architecture with 1-stage parallelism and Fig. 4 shows the architecture with two parallel levels. The details of each architecture are explained in the following sub-sections.

2.1. Sequential Architecture

The sequential SAD architecture is shown in Fig. 2. It contains only one 4x4 SAD processing block and no parallel stages and hence the processing speed is very low. Initially a set of 4x4 current block pixels and reference block pixels are sent to the 4x4 SAD unit via system data bus. In each clock cycle one 4x4 SAD, two 4x2 SADs and two 2x4 SADs are generated from 4x4 SAD unit. The 4x2 and 2x4 SAD values are used to generate upper mode SADs - 8x2,

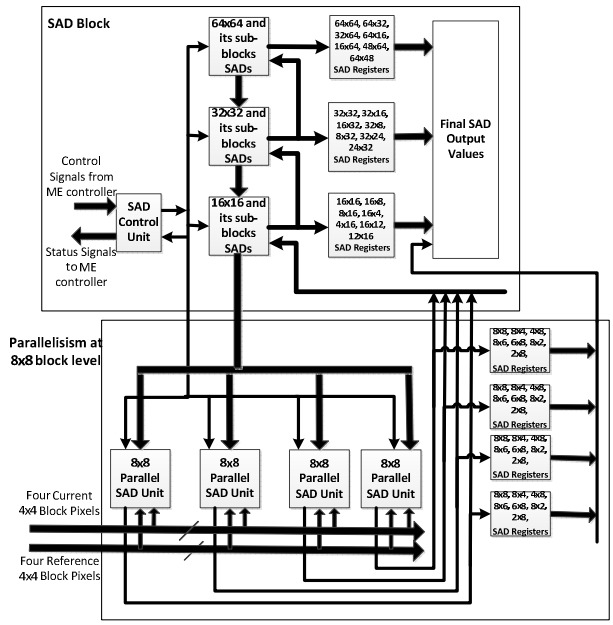


Fig. 4 2-Stage parallel Architecture

2x8 SAD values by adding two 4x2 and two 2x4 SAD values respectively. The 4x4 SAD values are stored in registers and are also sent to 8x8 SAD processing units along with 4x2 and 2x4 SAD values. After four cycles the 8x8 SAD unit generates one 8x8 (2Nx2N) SAD, two 8x4 (2NxN) SADs, two 4x8 (Nx2N) SADs, one 8x2 (2NxnU) SAD unit, one 8x6 (2NxnD), one 2x8 (nUx2N) SAD unit, one 6x8 (nDx2N) SAD unit. These SAD values are stored in registers and are also sent to upper mode SAD unit, which is 16x16 SAD units. The process repeats until 64x64 SAD unit. Since there are no parallel stages, the total clock cycles to process one 64x64 block can be calculated using (3), where d is the maximum depth of SAD unit (here d=4), p is the number of parallel stages (here p=0), and k is the controller delay. Neglecting the controller delay, the total delay for processing one 64x64 block is 256 clock cycles.

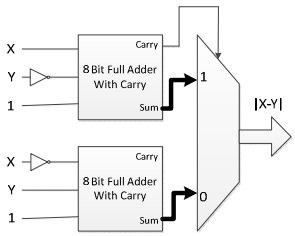
$$delay = 4^{d-p} + k \quad (3)$$

Each pixel requires 8 bits of data bus and hence one 4x4 SAD unit requires 256 bits (128 for current block and 128 for reference block pixels). The total bus width required for any parallel architecture can be derived using (4), where p denotes number of parallel stages (here p=0).

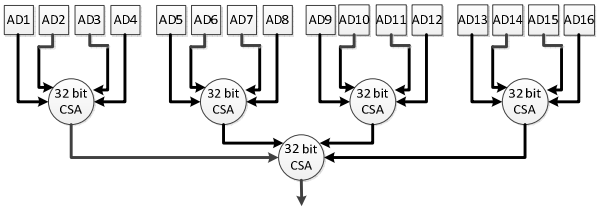
$$W = 256 \times 4^{p} \quad (4)$$

2.2. 1-Stage Parallel Architecture

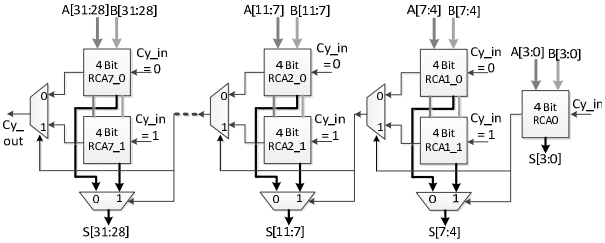
The 1-stage parallel architecture with parallelism at 4x4 block level is shown in Fig. 3(a). The internal architecture of 4x4 parallel unit is shown in Fig. 3(b), with N=4. The operation is similar to that of sequential architecture except



(a) Absolute Difference Circuit



(b) Adder Tree Architecture using 32-bit Carry Select Adder (CSA)



(c) Internal Circuit of 32 bit Carry Select Adder

Fig 5. Architecture of absolute difference and adder circuits

that all the 4x4 SAD units are processed parallel in the same clock cycle and hence improving the system performance. The total delay for processing one 64x64 block can be calculated using (3) and is reduced to 64 clock cycles (d=4, p=1), but the bus width required is increased to 1024 lines (from (4)).

2.3. 2-Stage Parallel Architecture

In 2-stage parallel architecture, there will be two parallel levels, one at 4x4 processing level and the other at 8x8 processing level, as shown in Fig. 4. The internal architecture of 8x8 parallel unit is shown in Fig. 3(b) with N=8. The operation for this architecture is also similar to the above architectures except that all the 8x8 block SAD units work in parallel. The total delay to process one 64x64 PU will be 16 clock cycles, which can be calculated from (3) with d=4 and p=2. Though the total delay reduces, the bus-width is increased to 4k lines, which can be calculated from (4) with p=2.

2.4. PU scanning order

One of the main advantage of the proposed architecture is the customizability of scanning order of PUs. In each 8x8 CU, the 4x4 PUs are scanned in z-order. Similarly the 8x8

CUs are scanned in z-order in a 16x16 CU and this scanning order is followed until 64x64 CUs. Hence the SAD operation can be terminated early and some of the block modes can be skipped if the rate-distortion cost reaches a pre-defined threshold (based on mode decision algorithm). The threshold value for early termination comes from ME controller to SAD control unit.

2.5. Sub-system Architecture

Each 4x4 sad unit contains the 16 absolute difference circuits, that can be realized using (5), where MSB represents Most Significant Bit, X' and Y' represents complements of X and Y. Fig 5(a) shows the AD (Absolute Difference) circuit used. The 16 AD values are added using Carry Select Adders (CSA) as shown in Fig 5(b). The internal circuit of 32-bit carry select is shown in Fig. 5(c). The carry select adder shown in the figure pre-computes 8-bit addition with carry bit (carry=1) and without carry bit (carry=0), using 8-bit ripple carry adders. Since CSAs do not wait for carry bits, they perform addition very fast.

$$|X-Y| = \begin{matrix} X+Y'+1, if\ MSB=1 \\ X'+Y+1, if\ MSB=0 \end{matrix} \quad (5)$$

3. SYNTHESIS RESULTS

The proposed architectures were implemented in Verilog-HDL and synthesized using Xilinx Virtex-5 FPGA [10]. Table 1 shows the synthesis results of the three architectures. The results show that the number of slices occupied for 1-stage parallel architecture is 4% greater than that of sequential architecture, which is insignificant. For the 2-stage architecture, the occupied slices are 16% more than that of sequential architecture, which is not insignificant. The maximum frequency (due to critical path delay) of the system is found to be almost same. The minimum number of clock cycles is reduced by a factor of four when each parallel level is included to sequential architecture. The minimum delay to process a 64x64 block can be calculated using (6), where n_{64x64} is the minimum number of clock cycles required which can be calculated using (3).

$$delay_{64x64} = n_{64x64} \times \frac{1}{freq_{\max}} \quad (6)$$

From Table 1, it can be observed that the total delay for processing one 64x64 block in 2-stage parallel architecture is very low (96.63ns) and for 1-stage parallel architecture is moderate (372.2ns) compared to that of sequential architecture. The total data bus lines required for 2-stage parallel architecture is very high (4k) and hence demands more routing area from on-chip search window memory to SAD processing unit. Due to more switching activity in 4x4 SAD block unit, the power consumption increases with number of 4x4 parallel SAD units. For 1-stage parallel architecture, the power consumption and data lines required are also moderate. In terms of area and on-chip bus width 1-stage parallel architecture is the optimized one.

Table 1.Synthesis results of proposed architecture

	Sequential architecture	**1-stage Parallel architecture**	**2-stage Parallel architecture**
# Slices (out of 17280)	8577 (49%)	9182 (53%)	11738 (68%)
# Slice LUTs (Out of 69120)	11124 (16%)	15453 (22%)	29484 (43%)
# Slice Registers (Out of 69120)	20377 (29%)	20736 (30%)	22236 (32%)
Max. Freq. (in MHz)	174.673	171.947	165.57
No. of clock cycles required (for one 64x64 block)	256	64	16
Total delay for one 64x64 block	1.4655 µs	372.2 ns	96.63 ns
On-chip data bus width	256	1024 (=1k)	4096 (=4k)
Total power (mW)	91.3	136.18	320.86

Table 2.Comparison of proposed parallel architecture with previous works

	[5]	**[6] partial propagate SAD architecture**	**[6] SAD Tree architecture**	**proposed**
Process	0.12 µm FPGA (Xilinx Virtex 2)	TSMC 0.18 µm	TSMC 0.18 µm	65 nm FPGA (Xilinx Virtex 5)
Area Results	375 Virtex-II slices	84.1 k gates	88.5 k gates	9.18 k Virtex-5 slices
Max freq. (MHz)	133.2	231.6	204.8	171.9
Block Sizes	4x4	4x4 to 16x16	4x4 to 16x16	4x4 to 64x64
AMP support	No	No	No	Yes

Table 2 shows comparison of the proposed 1-stage parallel architecture with previous works. The area results in Table 2 are not comparable as they are synthesized with different technologies and designed for different block sizes. The proposed architecture outperforms the architecture in [5], in terms of maximum frequency. Compared to partial propagate and SAD tree architectures of [6] the maximum frequency of the proposed design is 30 and 60 MHz less respectively. Nevertheless the proposed design computes all the block sizes from 4x4 to 64x64 including Asymmetric Mode Partitions (AMPs).

4. CONCLUSION

The present paper proposed three SAD architectures suitable for HEVC based motion estimation engine. Each architectures trades off between total delay and hardware resources used. The architectures are implemented in FPGA and the synthesis results show that the parallel architecture outperforms previous contributions. The proposed architecture also supports AMP modes. The architecture can be more optimized at sub-system level to reduce the critical path delay and increase the system performance. Future work will focus on reducing the critical path delay and reducing the power consumption.

5. REFERENCES

[1] B. Bross, W.-J. Han J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, T. Wiegand, “High Efficiency Video Coding (HEVC) text specification draft 10”, Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-L1003_v34, Jan. 2013.

[2] G.J. Sullivan, J.R. Ohm;Woo-Jin Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circ. and Sys. for Video Tech., vol.22, no.12, Dec. 2012.

[3] N. Purnachand, L. N. Alves, and A. Navarro. "Fast Motion Estimation Algorithm for HEVC." IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin), Sept. 2012.

[4] P. Nalluri, L.N. Alves, A. Navarro, "A novel SAD architecture for variable block size motion estimation in HEVC video coding" International Symposium on System on Chip (SoC), pp. 1-4, Oct. 2013.

[5] S. Rehman; R. Young;C. Chatwin;P. Birch, "An FPGA Based Generic Framework for High Speed Sum of Absolute Difference Implementation," Europ. Jour. Scient. Res., vol.33, no.1, 2009.

[6] Z. Liu, S. Goto, T. Ikenaga, "Optimization of Propagate Partial SAD and SAD tree motion estimation hardwired engine for H.264," IEEE Inter. Conf. on Comp. Design, pp. 328-333, Oct. 2008.

[7] C.Y. Chen, S.Y. Chien, Y.W. Huang, T.C. Chen, T.C. Wang, L.G. Chen, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," IEEE Trans. on Circuits and Systems I: Regular Papers, vol.53, no.3, Mar. 2006.

[8] Y. Huang, Z. Liu, S. Goto, T. Ikenaga, "Cost efficient propagate partial SAD architecture for integer motion estimation in H.264/AVC", 7th International Conference on ASIC, pp.782-785, Oct. 2007.

[9] C.Y. Kao, Y.L. Lin, "A memory-efficient and highly parallel architecture for variable block size integer motion estimation in H.264/AVC", IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no.6, pp. 866–874, Jun. 2010.

[10] Xilinx Virtex-5 FPGA User Guide, Version 5.4, Mar. 2012.