A Novel SAD Architecture 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—**Motion estimation (ME) is one of the critical and most time consuming tasks in video coding. The increase of block size to 64x64 and introduction of asymmetric motion partitioning (AMP) in HEVC makes variable block size motion estimation more complex and therefore requires specific hardware architecture for real time implementation. The ME process includes the calculation of SAD (Sum of Absolute Difference) of two blocks, the current and the reference blocks. The present paper proposes low complexity SAD (Sum of Absolute Difference) architecture for ME of HEVC video encoder, which is able to exploit and optimize parallelism at various levels. The proposed architecture was implemented in FPGA, and compared with other non-parallel SAD architectures. Synthesis results show that the proposed architecture takes fewer resources in FPGA when compared with results from non-parallel architectures and other contributions.**

Keywords—Motion Estimation; SAD architecture; HEVC.

I. Introduction

HEVC is the latest video coding standard 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]. The coding structure in HEVC is hierarchically generalized into CUs (Coding Units) and TUs (Transform Units). Each CU (Coding Unit) has a maximum size 64x64, and can be partitioned recursively until 4x4 pixel blocks. CUs consists of PUs (Prediction Units) of either of the types intra, inter, skip and merge. The Motion estimation (ME) is one of the time consuming and high complexity tasks in the video processing. The objective of the ME unit is to find the best matched block in the reference (past/future) frame search window (region of interest), for every block of the current frame such that the reconstructed frame contributes to the lowest residual information [3]. The ME is done for all the block sizes (variable block size ME) for a given maximum block size. The coding block size in HEVC is increased to 64x64, compared to 16x16 in H.264/AVC, which adds increased complexity to ME. Furthermore, due to AMP (Asymmetric Motion Partitioning), the number of modes also increases, as shown in Fig.1 (e) to (h). Hence, performing motion estimation for all the block sizes demands fast processing strategies which may resort to real time encoder implementations. Considering FPGA based implementation, it is possible to reconfigure the architecture and exploit the usage of parallelism in order to improve system performance. This is particularly relevant for the optimized design of SAD units. The SAD is one of the matching criteria which are widely used in ME. The SAD between a current and reference blocks of sizes MxN pixels each can be calculated using (1).

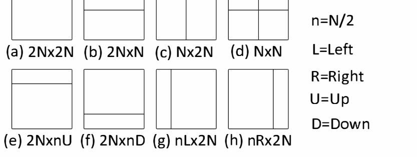


Fig.1 Inter prediction unit partition sizes in HEVC

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

where, CB represents the current block pixels and RB represents the reference block pixels. The bottleneck for the ME system design lies in the implementation of an appropriate SAD architecture. The implementation of SAD units has been addressed in many previous works, targeting both ASIC [4-6] and FPGA [7-8] design approaches. In [4-5], partial propagate SAD and SAD tree architectures were proposed. Partial propagate SAD architecture is best suited for low resolution video sequence and low complexity applications, whereas SAD tree architectures are able to provide best performance even for high resolution and high complexity applications with higher resources. In [7], an FPGA based SAD architecture was proposed which employs a partial product reduction scheme for adding the absolute difference values of pixels. In [8], an SAD architecture able to store partial sums and targeting their reuse under the same search window was presented.

The aforementioned architectures were implemented targeting H.264/AVC video codecs. SAD design increases in complexity under the new HEVC video standard due to block size extension to 64x64. Another problem is that in partial propagate SAD architectures, the buffers (or registers) that are required to accumulate partial SAD values and the delay of the system will increase much more when they are used for asymmetric mode partition (AMP). The present paper proposes a novel SAD architecture for FPGAs complying with HEVC block size (64x64) and able to support AMP mode by reusing the partial SAD values of lower block sizes. To reduce the delay, the proposed architecture employs parallelism at 4x4 block size level unit and hence reduces the overall delay of the system. Section II explains the proposed architecture and its operation, section III discusses the subsystem architecture, section IV analyses the performance results and finally section V concludes the paper.

Table 1. Summary of total number of SADs for each partition size in a 64x64 block

Block	Sub-block sizes	Number of partitions	Total SADs for a 64x64
8x8	8x2, 8x4, 8x6, 2x8, 4x8, 6x8,4x4	4x4 – 4 partitions, rest of the sizes – 2 partitions each.	4x4 – 256, rest of the sizes – 128 each
16x16	16x4,16x8,16x12, 4x16,8x16,12x16,16x16	8x8 – 4 partitions, rest of the sizes – 2 partitions each	8x8 – 64, rest of the sizes – 32 each
32x32	32x8,32x16,32x24, 8x32,16x32,24x32,32x32	16x16 – 4 partitions, rest of the sizes – 2 partitions each	16x16 – 16, rest of the sizes – 8 each
64x64	64x16,64x32,64x24, 16x64,32x64,48x64, 64x64	32x32 – 4 partitions, rest of the sizes – 2 partitions each	32x32 – 4, rest of the sizes – 2 each
		Total	1360

II. Proposed SAD Architecture

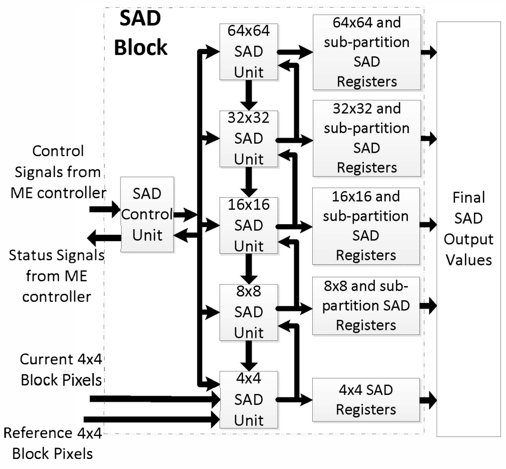
As mentioned before, motion estimation (ME) is one of the most demanding tasks in video estimation. The ME unit generates motion vectors resulting from the comparison between the current frame and a reference frame. One of the design bottlenecks in ME is the cost function computation. Several approaches can be employed, such as the sum of absolute differences (SAD) between pixels of current and reference blocks. In HEVC, the maximum block size is 64x64 pixels, representing a factor of 16 times more pixels than in H.264. Designing a SAD unit able to handle these large pixel blocks, demands special care. For H.264, both serial and parallel units were successfully employed to improve SAD performance. In HEVC serial and parallel computation may also be employed implying necessary design tradeoffs between performance and required resources. This section compares these two design approaches, presenting to SAD architectures compliant with HEVC standard.

A. Variable block size SADs

There are eight SAD sizes for each depth of CU partition – 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N, NxN, as shown in Fig.2. Table 1 lists the total number of partial SADs for a 64x64 PU. The second column in the Table 1 shows sub-block sizes, and the third column shows the number of partitions for each sub-partition size. The last column shows the total number of SADs for each sub-partition size. For instance, in an 8x8 block, there are seven sub-partition sizes and each has partitions (for example two 8x4 partitions, two 4x8 partitions etc.), except 4x4 size which are four in number for one 8x8 size. Hence there are total of 256 4x4 blocks in one 64x64 PU, and 128 blocks for each size - 8x2, 8x4, 8x6, 2x8, 4x8, 6x8. Therefore, the complexity of SAD computation will be high, unless parallelism is employed using an appropriate architecture.

B. Architecture and operation

The complete architecture, without any parallelism is shown in Fig 2. The architecture consists of one 4x4 absolute difference calculation block, and five variable block size SAD calculators corresponding to five depth values (0 to 4) of a PU. In each cycle, one 4x4 current block pixels and one 4x4 reference block pixels are given as inputs to 4x4 block SAD unit. The 4x4 SAD is calculated using absolute difference block, along with 4x2 and 2x4 block SADs. The SADs are sent to 8x8 size unit and stored in SAD registers.



SAD Block
64x64 SAD Unit
32x32 SAD Unit
16x16 SAD Unit
8x8 SAD Unit
4x4 SAD Unit
64x64 and sub-partition SAD Registers
32x32 and sub-partition SAD Registers
16x16 and sub-partition SAD Registers
8x8 and sub-partition SAD Registers
4x4 SAD Registers
Final SAD Output Values
Control Signals from ME controller
SAD Control Unit
Status Signals from ME controller
Current 4x4 Block Pixels
Reference 4x4 Block Pixels

Fig.2. SAD architecture without parallelism

The 8x8 block initially requests 4x4 SAD block four times, and takes four values of 4x4, 4x2, 2x4 SAD units, corresponding to each 4x4 block in a 8x8 block. Hence, it finally calculates SADs of one 8x8 size, two 8x4 size, two 4x8 size, two 8x2 size, two 8x6 size, two 2x8 size and two 6x8 size. These SAD values are again sent to upper depth block which is 16x16 SAD block unit and stored in registers. The 16x16 SAD unit request to calculate the 8x8 SAD unit four times and takes four SAD values of 8x8, 8x4 and 4x8 size. Hence, the 16x16 SAD unit outputs one 16x16 SAD, two 16x8 SADs, two 8x16 SADs, two 16x4 SADs, two 4x16 SADs, two 16x12 SADs and two 12x16 SADs. The operation continues till 64x64 SAD block. The input request control signal to 64x64 comes from SAD control unit.

C. Parallelism

The total number of cycles that will be taken in the architecture shown in Fig.2 can be calculated using (2),

$$delay = 4^{d-p} + k \qquad (2)$$

where, delay is the number of clock cycles required for one SAD computation, d denotes the maximum depth of SAD unit (here $d_{max} = 4$), p is the number of parallel stages, and k represents a constant delay overhead due to the controller. In the non-parallel architecture described before $p=0$, and the total delay will be *256* cycles plus controller delay k. Since this is not practical for high speed encoders, parallel SAD units can be inserted to decrease the delay. The architecture with parallelism at depth *0* (4x4 SAD unit) is shown in Fig.3. In each cycle, four 4x4 SADs are calculated in parallel and sent to 8x8 SAD block and hence the total delay now reduces to *64* clock cycles for each SAD unit.

The number of parallel stages is constrained by the internal data bus width size from memory to the SAD unit. Each pixel is 8 bit width and one 4x4 block of pixels require 128 bits. Hence for the non-parallel version of Fig.2, the bus width size is 256 (128 for current block and 128 for reference block

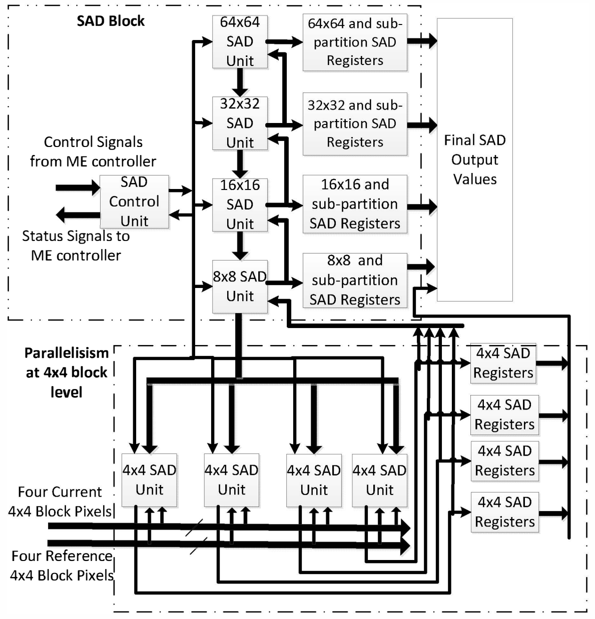

SAD Block
64x64 SAD Unit
64x64 and sub-partition SAD Registers
32x32 SAD Unit
32x32 and sub-partition SAD Registers
16x16 SAD Unit
16x16 and sub-partition SAD Registers
8x8 SAD Unit
8x8 and sub-partition SAD Registers
Final SAD Output Values
Control Signals from ME controller
SAD Control Unit
Status Signals to ME controller
Parallelisism at 4x4 block level
4x4 SAD Unit
4x4 SAD Registers
Four Current 4x4 Block Pixels
Four Reference 4x4 Block Pixels

Fig.3. Proposed SAD architecture with parallelism at depth = 0

pixels). For depth-0 parallel version of fig. 3, the total number of reference block bit lines required will increase to 512 (=128x4) and similarly 512 lines for current block pixels making a total of 1024 lines. The number of data lines for any parallel version of SAD architecture is given by (3).

$$W = 256 \times 4^{p} \tag{3}$$

If the parallelism is further increased to depth-1 stage (8x8 SAD unit), there will be in total two parallel stages, and hence the total number of data lines required will increase to 4096, thus increasing resource usage. For comparison purposes, two SAD units have been considered: i) one using the non-parallel topology of fig. 2; and ii) the other, employing the depth-0 parallel version of fig. 3. The parallel architecture may also be designed to operate with parallelism at depth-1 (8x8 SAD), if area and resources area not a constraint. This approach may improve performance in high speed applications.

III. Absolute Difference and Adder Circuits

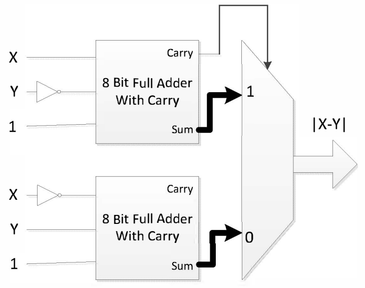
The main sub-systems that are involved in the architecture are absolute difference calculation (AD) circuit, and the adder circuit. The next sections explain each one in detail.

A. *Absolute Difference Circuit*

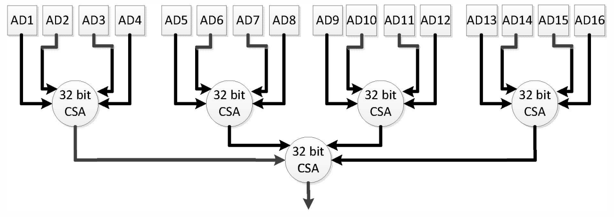
The absolute difference (AD) calculation circuits are only located in the 4x4 blocks. There are many ways to compute the absolute difference. The fastest way is to use two's complement as shown in (4),

$$|X-Y| \begin{cases} = X+Y'+1, \text{ if } MSB = 1 \\ = X'+Y+1, \text{ if } MSB = 0 \end{cases} \tag{4}$$

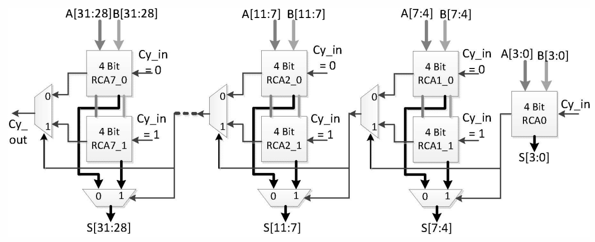
where, *MSB* is the most Significant bit, *X'* and *Y'* are the complements of *X* and *Y*, respectively. In a two's complement addition, the MSB is *1* only if *X* is greater than *Y*. Hence the AD circuit for one pixel (8 bit), can be designed using the circuit shown in Fig. 4(a).



(a) Absolute Difference Circuit


AD1 AD2 AD3 AD4 AD5 AD6 AD7 AD8 AD9 AD10 AD11 AD12 AD13 AD14 AD15 AD16
32 bit CSA

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



(c) Internal Circuit of 32 bit Carry Select Adder

Fig 4. Architecture of absolute difference and adder circuits

B. *Adder Circuit*

The addition operation is performed for sixteen pixels (8 bit absolute difference numbers) in one 4x4 SAD operation. The combinational delay (and hence the critical path) can be reduced using adder trees. The addition is performed using 32 bit carry select adders (CSA). Each CSA adds four pixels with combinational delay T_{CSA}. Hence, for a 4x4 block, there will be four CSAs in the first stage of adder tree, as shown in Fig.4 (b). In the second stage, there is one CSA. Hence the total combinational delay of the circuit is $2T_{CSA}$. The CSA internal circuit using 4-bit RCAs (Ripple Carry Adders) is shown in Fig. 4(c). The delay of each 32 bit CSA is four times the full adder (T_{FA}) plus the delay of seven multiplexers (T_{MUX}) as shown in (5).

$$\begin{gathered} T_{CSA} = 4T_{FA} + 7T_{MUX} \\ T_{Total} = 2T_{CSA} = 2(4T_{FA} + 7T_{MUX}) = 8T_{FA} + 14T_{MUX} \end{gathered} \tag{5}$$

The aforementioned adder circuit is optimized for performance and resource usage. For instance, if the configuration for adder tree architecture is designed using 16 bit CSAs, then the number of stages would be four, and the delay would be $4T_{CSA16}$, where T_{CSA16} is the delay of the 16 bit CSA. The delay of one 16 bit CSA is $4T_{FA}+3T_{MUX}$, and the total delay of adder tree is $4(4T_{FA}+3T_{MUX})$ and is equal to $16T_{FA}+12T_{MUX}$ which has double the adder delay (and almost same multiplexer delay) than that of delay in (5).

Table 2.Synthesis results of proposed architecture

	Non-parallel SAD architecture	Parallel SAD architecture	Difference (for parallel Architecture)
# Slices (out of 17,280)	8577 (49%)	9182 (53%)	4% more
# Slice LUTs (Out of 69120)	11124 (16%)	15453 (22%)	8% more
# Slice Registers (Out of 69120)	20377 (29%)	20736 (30%)	1% more
Total Equivalent Gate Count	250.28 k	291.27 k	41k more
Max. Freq. (in MHz)	174.673	171.947	2.726 MHz less
No. of clock cycles required (for one 64x64 block)	256	64	75% less
Total delay for one 64x64 block	1.4655 µs	372.2 ns	3.9 times less
Total power (mW)	91.3	136.18	49.15% more

Table 3.Comparison of proposed parallel architecture with previous works

	[7]	[4] partial propagate SAD architecture	[4] 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)
Gate count	375 Virtex-II slices	84.1 k	88.5 k	291.27 k
Equivalent gate count for 64x64 block	1792 k	1345.6 k	1416 k	291.27 k
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

IV. Synthesis Results

The proposed architectures were implemented in Verilog-HDL and synthesized using Xilinx Virtex-5 FPGA [9]. Table-2 shows the synthesis results comparison for the proposed parallel version against the non-parallel version. The synthesis results show that the parallel architecture occupies 4% of slices (Virtex 5) more than the non-parallel version, but the delay is 3.9 times reduced compared to nonparallel version. Due to high switching activity and higher number of 4x4 blocks and inputs, the parallel architecture consumes 49% more power than non-parallel architecture. Hence for high speed operation, the parallel architecture is recommended at the expense of more power consumption. The total delay to process one 64x64 block can be calculated using (6) (without considering control unit delay),

$$delay_{frm} = N_{64x64} \times \frac{1}{freq_{Max}} \tag{6}$$

where, N_{64x64} is total number of clock cycles required to process one 64x64 block.

The proposed parallel architecture was compared against previous works [4-7], as shown in Table 3. From Table 3, it is clear that the proposed architecture outperforms the architecture of [7]. Though the proposed architecture operates at 30-60 MHz less maximum clock frequency (compared to [4]), it calculates SADs from 4x4 until 64x64 pixels size including AMP (Asymmetric Motion Partitioning) sizes. The extrapolated results for gate count are an approximate value calculated directly by multiplying with 16 (=64x64/16x16) for architecture in [4]. In case of [7], the gate count is obtained by multiplying 256 (=64x64/4x4) with 7k (equivalent gate count for 375 Virtex II slices).

V. Conclusion

The present paper proposed novel architectures to compute SADs for all block sizes, for real time HEVC motion estimation engines. It compares two types of architectures - one without parallelism and the other with parallelism. The achieved results show that the parallel architecture is 3.9 times faster than non-parallel architecture. Further, the proposed architecture supports asymmetric modes and hence it is best suited for HEVC motion estimation. To increase speed of the SAD computation further, the proposed architecture can be configured with more parallel stages, at the expense of FPGA resources and power consumption. Future work will focus on the optimization of this architecture with more parallel stages and reduction of power consumption using appropriate power reduction techniques.

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] Sullivan, G.J.; Ohm, J.; Woo-Jin Han; Wiegand, T.; Wiegand, T., "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), September 2012.

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

[5] Tung-Chien Chen; Yu-Han Chen; Sung-Fang Tsai; Shao-Yi Chien; Liang-Gee Chen, "Fast Algorithm and Architecture Design of Low-Power Integer Motion Estimation for H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, vol.17, no.5, May 2007.

[6] Ching-Yeh Chen; Shao-Yi Chien; Yu-Wen Huang; Tung-Chien Chen; Tu-Chih Wang; Liang-Gee 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.

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

[8] Niitsuma, H.; Maruyama, T., "Sum of Absolute Difference Implementations for Image Processing on FPGAs," Inter. Conf. on Field Programmable Logic and Applications (FPL), 2010, Sept. 2010.

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