

# Design and Implementation of an Efficient Mixed Parallel-Pipeline SAD Architecture for HEVC Motion Estimation



Mamidi Nagaraju, Santosh Kumar Gupta, Vijaya Bhadauria, and Devarshi Shukla

**Abstract** Video compression is a computationally intensive task that generally demands high performance for portable device application. This high performance is mainly attained by motion estimation (ME) in the video encoder. With the larger block size and flexible block, partitioning in High Efficiency Video Encoding (HEVC) makes ME block more complicated. The sum of absolute difference (SAD) is extensively used as a distortion metric in the ME process. It is highest computing task with more calculation time and hardware resource. In this paper, an efficient mixed parallel-pipeline SAD architecture is proposed for high frequency. The performance optimization techniques are used at different levels of abstraction: Architecture level (parallel and pipeline), RTL level (grouping and resource sharing) and Implementation level (retiming). The proposed architecture has been simulated, synthesized and prototyped using 28 nm Artix-7 Field Programmable Gate Array (FPGA). The result shows that the proposed design provides a significant increment in the maximum frequency of 498.4 MHz when compared with other existed work in the literature.

**Keywords** Sum of absolute difference · Video coding · HEVC · Motion estimation · FPGA

## 1 Introduction

To meet the continues growing demand of video, technologically, the design constraints such as performance, power and efficient memory at better transfer rate

---

M. Nagaraju (✉) · S. K. Gupta · V. Bhadauria · D. Shukla  
Department of ECE, Motilal Nehru National Institute of Technology Allahabad, Prayagraj,  
Uttar Pradesh 211004, India  
e-mail: [rel1651@mnnit.ac.in](mailto:rel1651@mnnit.ac.in)

S. K. Gupta  
e-mail: [skg@mnnit.ac.in](mailto:skg@mnnit.ac.in)

V. Bhadauria  
e-mail: [vijaya@mnnit.ac.in](mailto:vijaya@mnnit.ac.in)

D. Shukla  
e-mail: [devarshi@mnnit.ac.in](mailto:devarshi@mnnit.ac.in)

are required for a wide range of applications such as high definition (HD) (1080 p resolution) and ultra-high definition (UHD) (4 K and 8 K resolution). To meet out this, robust and enhanced video compression efficient designs are required. HEVC or H.265 is the present popular video compression algorithm designed to increase the video coding efficiency significantly as compared to its previous standard called H.264 or Advance-Video-Coding (AVC) [1]. The HEVC is a video coding standard which supports higher resolutions and can achieve up to 50% bit-rate savings as compared to earlier H.264 for the same video quality [1, 2]. The considerable increase in throughput is due to the many enhanced tools, techniques and methodologies which has been introduced in HEVC. Some of this enhancements are included in the ME prediction process, which is the critical and complex block of video encoding. The experimental results show that the ME block is occupied approximately 70% of the total video encoding load [2].

The main objective of the ME process is to identify the suitable matching block in a search area (SA) of reference frame (RF) corresponding to a current frame (CF) and provides a map of the displacement directions using a motion vector (MV). Figure 1 shows the general motion estimation procedure.

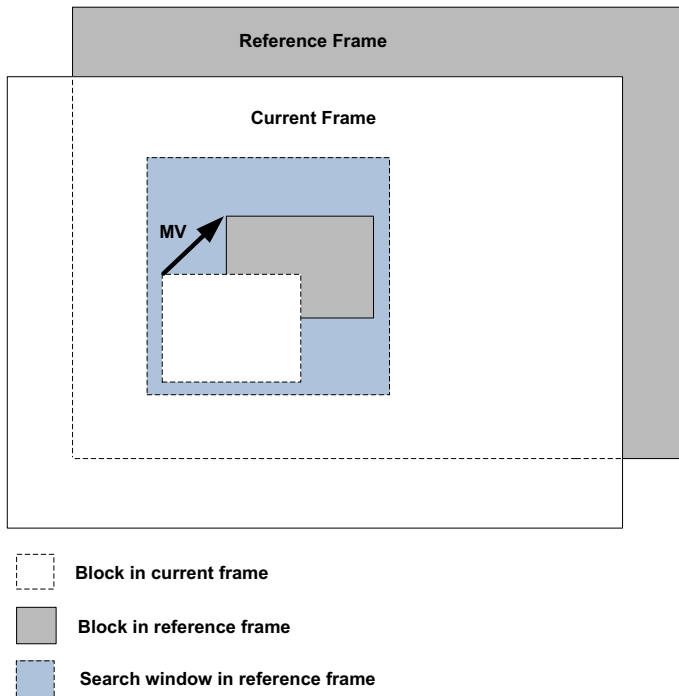


Fig. 1 Basic motion estimation process [3]

In the HEVC, a video frame encoding is performed using basic block called coding units (CU) of varied sizes ( $8 \times 8$  to  $64 \times 64$ ) obtained by dividing the frames. The CU maximum size is  $64 \times 64$  which increases complexity in the design of ME. In every level of CU, HEVC basically supports eight different partitions in the prediction block. As shown in Fig. 2, the block partitions from (a) to (d) are called symmetric mode (SM) partition and from (e) to (h) are called asymmetric mode (AM) partition.

For block-based ME, the SAD is the commonly used metric which perform addition of the absolute differences between corresponding elements in a current and reference block. The main bottleneck of the ME process lies in implementation with an appropriate and efficient SAD design. The SAD calculations are responsible around 65% of the total ME process [3]. The general prediction of SAD value between the current block (CB) and reference block (RB) is performed using the following equations [5].

$$SAD(i, j) = \sum_{k=0}^{M-1} \sum_{t=0}^{N-1} |CB(k, t) - RB(i + k, j + t)| \tag{1}$$

$$SAD_{min} = \min(SAD(i, j)) \tag{2}$$

where  $N \times M$  is block size of the current and reference blocks,  $i$  and  $j$  are coordinates of the block and  $SAD_{min}$  is the minimum SAD value. The HEVC defines different partitioning sizes for current blocks (CB) to reduce ME complexity. Among all, one of the constraint is disable the asymmetric motion partitioning (AMP) option with

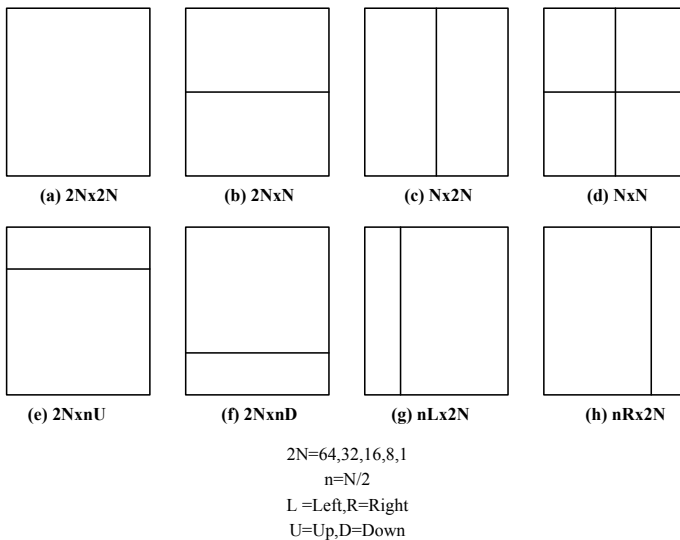


Fig. 2 Different block partitions and sizes for HEVC motion prediction unit [4]

**Table 1** The sum of the total SADs required for every partition sizes in a  $64 \times 64$  unit [6]

S. No.	Block sizes	No.of SAD's	S. No.	Block sizes	No. of SAD's
1	$64 \times 64$	1	14	$8 \times 32$ (left)	8
2	$32 \times 64$	2	15	$16 \times 16$	16
3	$64 \times 32$	2	16	$8 \times 16$	32
4	$64 \times 16$ (up)	2	17	$16 \times 8$	32
5	$64 \times 16$ (down)	2	18	$16 \times 4$ (up)	32
6	$16 \times 64$ (right)	2	19	$16 \times 4$ (down)	32
7	$16 \times 64$ (left)	2	20	$4 \times 16$ (right)	32
8	$32 \times 32$	4	21	$4 \times 16$ (left)	32
9	$16 \times 32$	8	22	$8 \times 8$	64
10	$32 \times 16$	8	23	$4 \times 8$	128
11	$32 \times 8$ (up)	8	24	$8 \times 4$	128
12	$32 \times 8$ (down)	8	25	$4 \times 4$	256
13	$8 \times 32$ (right)	8	–		
Total No. of SADs		316 (Square) + 340 (Symmetric) + 168 (Asymmetric)			

fixed block size  $N = 4$  [4]. Table 1 lists partitions for  $64 \times 64$  CTB (coding tree block) and the number of SAD required for each partition.

By supporting variable block size, the compression coding performance can be improved significantly. Also, the encoding computational complexity is increased dramatically. The coding block size in HEVC is upto  $64 \times 64$  used for HD/UHD resolutions as compared to  $16 \times 16$  in H.264 makes the SAD process critical. Obviously, the number of SAD computations vastly increases resulting into substantial increase in the processing gate delay and hence a decrease in overall performance. For instance, the UHD video ( $4320 \times 2160@30$  frames/s) required the computation of 18,345,885,696,000 absolute differences (ADs) and 71,663,616,000 multi-value summation operations/s [5].

In the literature, several VLSI architectures for SAD computation are reported [4, 6–11]. These report a trade-off between speed, performance and power for the hardware implementation of SAD. The architectures considered are mainly either for low-power or high-performance implementation either on FPGA or on ASIC. The present available FPGAs contain a lot of dedicated hardware resources which are more suitable for high-performance optimization providing less opportunity for power optimization, whereas ASICs can be a good choice for both power and performance optimization. In this paper, FPGA is chosen for performance enhancement of SAD process. The main work contributions are outlined as follows.

1. On-chip memory and its efficient access architecture for SAD have been proposed and designed.
2. Performance optimization has been done at different abstraction levels, viz. **a** at architectural level by deploying mixed parallelism and pipelining, **b** at RTL level

with grouping and resource sharing and **c** at implementation level using retiming techniques.

3. Verified with test bench simulations, debug on FPGA with logic analyzer and constraint-driven synthesis for better performance.
4. Prototyped on FPGA using two consecutive video frames and calculated corresponding residual image and displayed output on monitor using display controller. The interfaces at input and output sides have also been developed.

For HEVC ME process, different SAD architectures were proposed in past targeting FPGA [4, 6–11] with various design and implementation approaches. Dinh Vu et al. [4] introduced a parallel SAD architecture to reduce the hardware cost and calculation time. They decreased the number of registers used but the speed degraded due to higher delay. Joshi et al. [7] reported a high-speed processing architecture with highest frequency of 475 MHz. Nalluri et al. [8] reported another high-speed architecture with trade-off between delay and hardware resources. Medhat et al. [6] reported a highly parallel architecture. It uses 64 PUs operating in parallel with two memory banks for each frame. The optimal data flow from the memory for SAD computation provides better speed and lower delay among above discussed works.

From the above brief literature survey, it is evident that there is a trade-off between speed and processing delay. So, the development of high-speed processing architecture for SAD on FPGA is of great interest. Hence, an efficient mixed parallel-pipelining architecture is proposed, designed and verified on FPGA and results compared with relevant SAD architectures. The implementation results show that the present architecture can process the video with a significant delay reduction in critical path and hence increased frequency which can efficiently meet the requirements of 4 K resolution at 30 frames/s for high-speed portable applications. The paper is organized as follow. Section 1 gives an introduction and overview of HEVC, ME and SAD process and provides literature review and related works reported till date. Section 2 explains the proposed SAD unit architecture and sub modules. Section 3 shows the hardware implementation. Section 4 presents result and comparison with existing works in terms of logic delay, hardware resource usage and operating speed. Finally, Section 5 concludes the findings and features of the proposed work.

## 2 Proposed Architecture and Design Process

The proposed architecture shown in Fig. 3 is mainly composed of: (i) External random access memory (RAM) for current and reference block data, (ii) on-chip memory register files (RF), (iii) processing element (PE), (iv) absolute difference (AD) circuit and (v) adder tree structure to calculate the  $SAD_{\min}$ .

It receives the data related to reference and current pixels from external RAM. This data is pushed into on-chip memory RF. The data from on-chip memory RF is further fed to PE for calculation of ADs and not accessed from external memory

directly. This reduces the access time of data, and hence, processing time is improved. Finally, ADs are fed to adder tree for calculating the SADs.

### 2.1 Memory Organization and Parallel Processing

The memory architecture for calculating all possible SADs value from  $4 \times 4$  to  $64 \times 64$  blocks of HEVC encoder has been considered.

At architecture level,  $64 \times 64$  block unit of current and reference frames are fetched from external memory which are stored in the on-chip RFs. Each  $64 \times 64$  block is divided into sixteen  $32 \times 32$  blocks at level 1 and  $16 \times 16$  blocks at level 2. The  $16 \times 16$  block is divided into sixteen  $4 \times 4$  block to be used for SAD calculations,

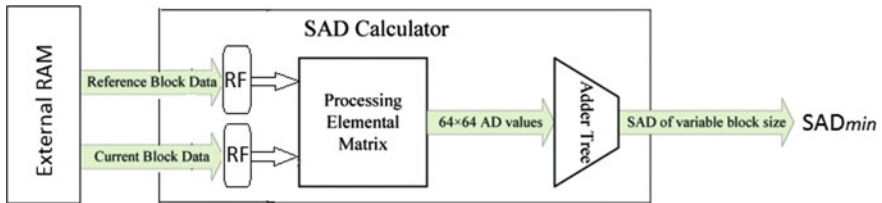


Fig. 3 Proposed SAD architecture

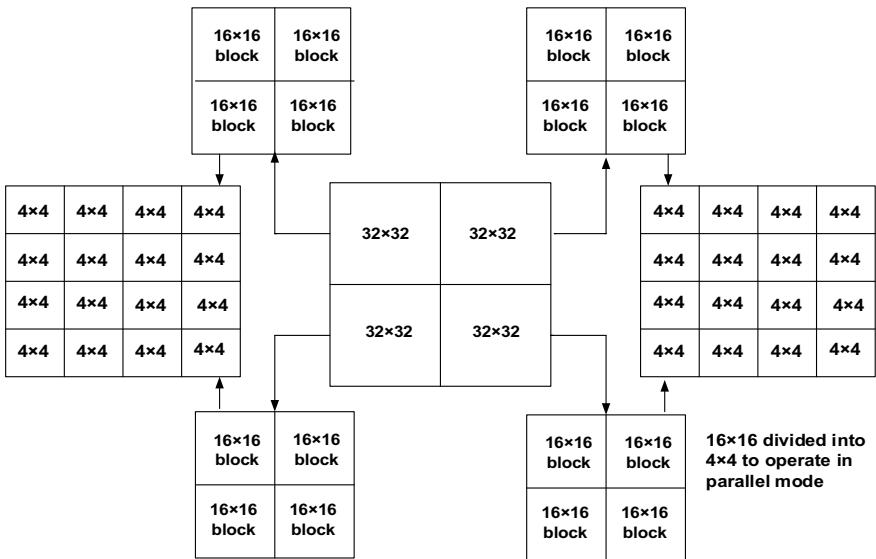


Fig. 4 Memory organization of  $64 \times 64$  block unit

as shown in Fig. 4. Here, this process has been adopted to exploit more degree of parallelism.

### 2.2 Register File as On-Chip Memory

Register File (RF) stores the data of each current and reference block of  $64 \times 64$  size. Each column and row data of  $64 \times 64$  matrix are stored in associated memory address. The size of each RF requires approximately 4 KB of on-chip memory required. In FPGA, block random access memory (BRAM) is sufficient for storing the above mentioned data. However, for high resolution videos having higher number of bits per pixel requiring higher memory than 4 KB, look up tables (LUTs) can be used in the place of RF for on-chip data storage. In case for higher requirement, so, LUT can be used for RF.

### 2.3 Processor Element (PE)

The processor element (PE) block implementation is shown in Fig. 5, CP refers to current pixel and RP refers to reference pixels. It uses 4 ADs to calculate one row of  $4 \times 4$  block in parallel. The AD values of each row are added to calculate 4 SAD values stored in next level RF memory which forms the primitive block for 64 SAD computations.

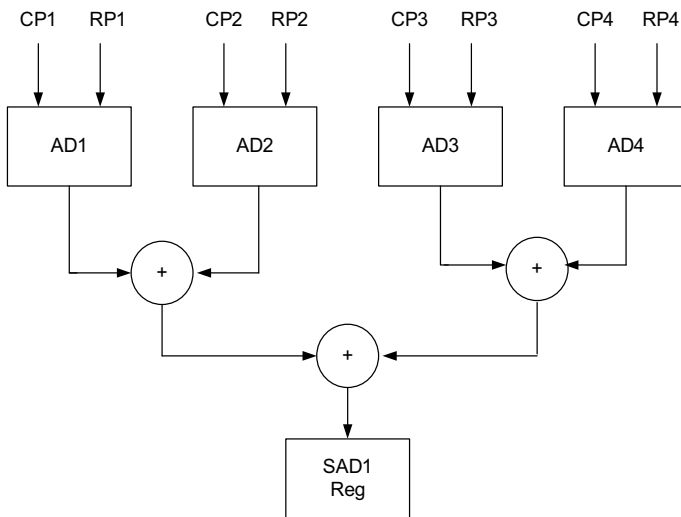
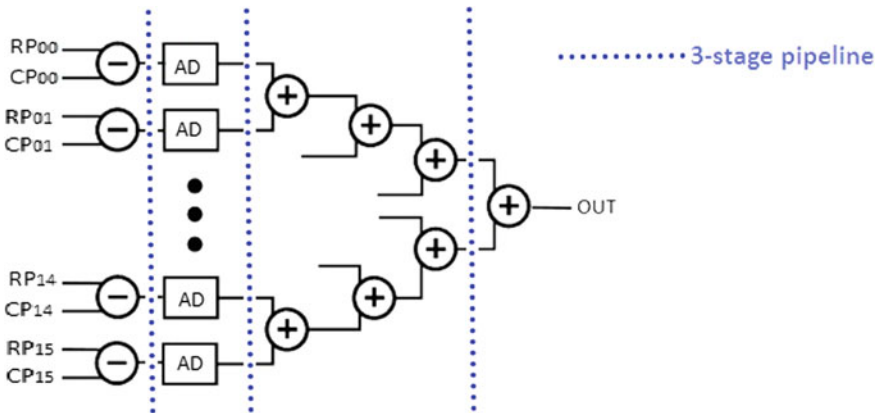


Fig. 5 The architecture of processor element



**Fig. 6** 16 × 16 pipelined architecture

The 16 × 16 array is implementing using a 4 × 4 block. The data block of 16 pixels is splitting into four groups and each group holds 4 pixels data and sent it to hierarchical SAD summation block.

The 16 × 16 size has implemented a three-stage pipeline structure as shown in Fig. 6, where all 4 × 4 blocks are operated in parallel. The critical path has a long path which is a tremendous amount of adder computation. The pipeline helps to decrease critical path delay.

### 2.4 Absolute Difference (AD)

An absolute difference (AD) circuit is 4 × 4 calculation of the absolute differences between current and reference pixels. Figure 7 shows AD block diagram implemented by using 8-bit comparator and subtractor. The comparator determines the higher value from current and reference pixel. The subtractor subtracts the lower value from higher to provide the AD value which is stored in an on-chip register file (RF) by 4-bit subtraction. Equation (3) represents AD operation [3].

$$AD = |C - R| = \begin{cases} C - R = C + \bar{R} + 1 & \text{if } C > R \\ R - C = R + \bar{C} + 1 & \text{if } R > C \end{cases} \quad (3)$$

where  $C$  is current pixel,  $R$  is reference pixel.



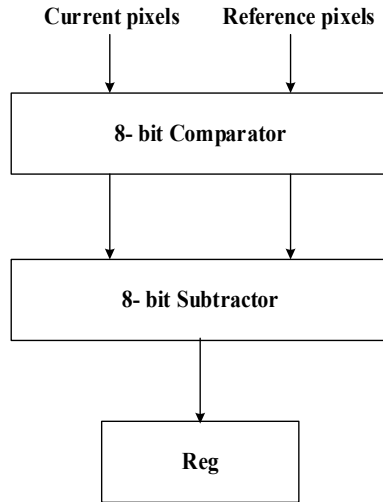


Fig. 7 Absolute difference circuit block diagram

### 2.5 Adder Tree Structure

The adder tree structure is used to calculate the SAD of different block sizes using SAD of  $4 \times 4$  primitive block size. Figure 8 shows the hierarchical SAD adder tree structure. This is similar to the HEVC quad-tree CU structure in which each  $2n \times 2n$  blocks consisting of four  $n \times n$  blocks.

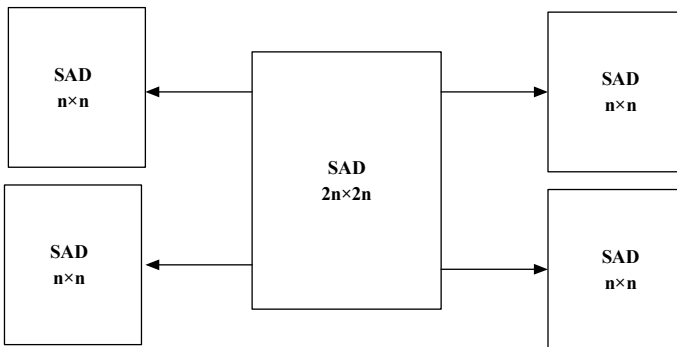


Fig. 8 The hierarchical view of SAD adder tree structure

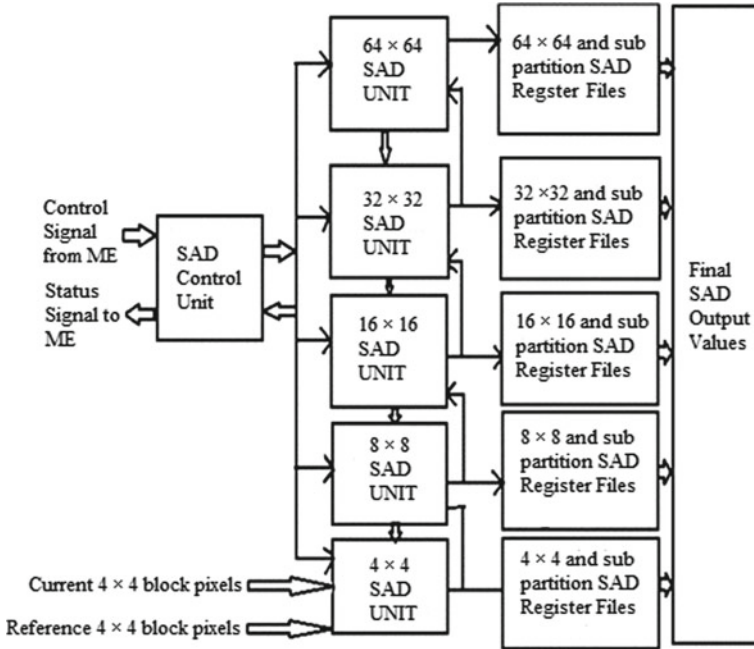


Fig. 9 SAD top-level architecture

### 2.6 SAD Top Level

The basic  $4 \times 4$  block has been reused four times to obtain the SAD for  $8 \times 8$  block size. Likewise, SAD of other block sizes viz.  $16 \times 16$ ,  $32 \times 32$  and  $64 \times 64$  can also be calculated using reuse process. Figure 9 shows the top-level architecture of SAD calculation.

### 2.7 Minimum Frequency Requirement

In this design, each CTB of  $64 \times 64$  size is decomposed into 256 number of  $(4 \times 4)$  block size for SAD calculation. Therefore, it takes 2048 cycles ( $256 \times 8$  bit) to process each CTB. To achieve high throughput, for the application of 4 K video ( $3840 \times 2160$  at 30 frames/s) encoding, the following is minimum necessary operating frequency ( $f_{min}$ ) [12].

$$f_{min} = (2048 \text{ cycle/CTB}) * \frac{(3840 * 2160 * 30 \text{ pixels/s})}{(64 * 64 \text{ pixels/CTB})} = 124\text{MHz} \quad (4)$$

On the other hand, Fan and Yibo, et al. [13] reported the maximum necessary frequency for 4 K at 30 frames/s to be approximately 500 MHz. So, when a  $4 \times 4$  SAD tree is adopted, the minimum and maximum frequency is 124 MHz and 500 MHz, respectively.

### 3 Hardware Implementation and Prototype

The proposed SAD architecture is designed in Verilog HDL and prototyped using Artix-7 FPGA. The test bench has been modeled for logic and functional verification with “foreman” video sequence of quarter common intermediate format (QCIF)  $176 \times 144$  resolution [13]. The reason for choosing QCIF resolution is to keep it simple for simulation observations.

The simulation result of the proposed SAD architecture for two consecutive frames is shown in Fig. 10.

The two consecutive frames data are given to the test bench. It generates the corresponding SAD. The design is also verified with an on-chip debugging tool (chip-scope pro logic analyzer) as shown in Fig. 11.

Figure 12 presents the RTL schematic view of  $4 \times 4$  SAD block in which PE and adders tree are operating in parallel. The synthesized logic can be further improved by (a) resource sharing, (b) grouping logic and (c) retiming pipeline.

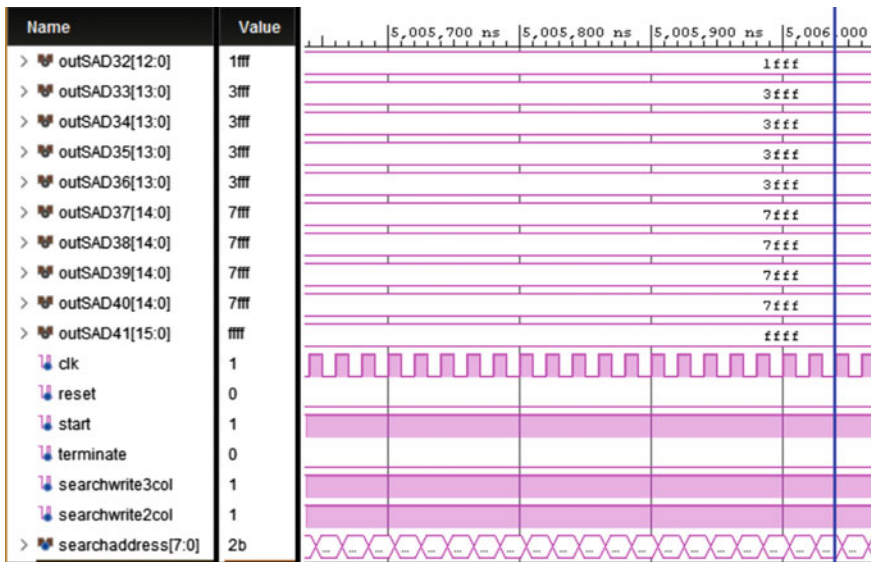
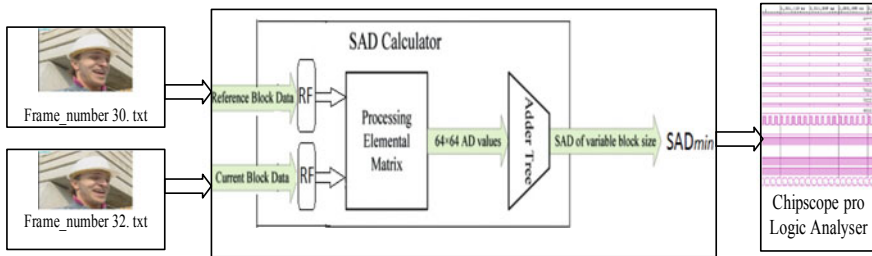
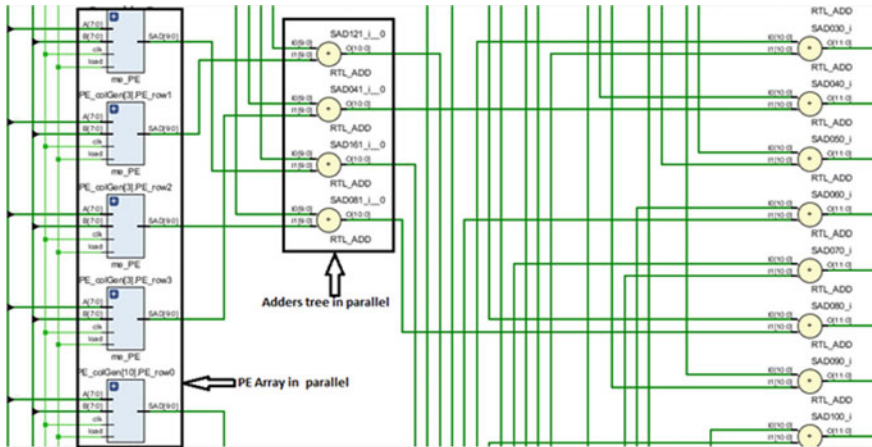


Fig. 10 Test bench simulation of the “foreman” test sequence for SAD calculations



**Fig. 11** Simulation and on-chip debug flow



**Fig. 12** RTL elaborated view of SAD  $4 \times 4$  block

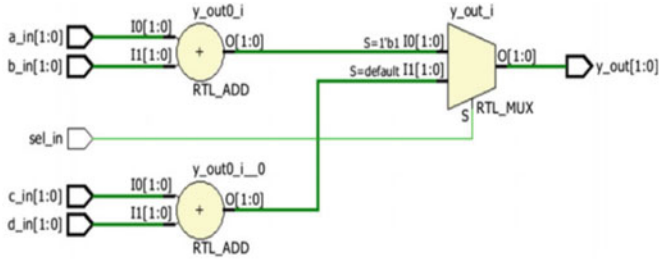
### 3.1 Resource Sharing [14]

At RTL level, resource sharing technique can be utilized for optimizing the performance parameter (such as speed). As shown in Fig. 13a and b, for some logics, if adders are used, it consumes more area. To overcome this problem, multipliers are used instead of adders to reduce the area. Moreover, we can use the optimized DSP48E1 slices for multipliers in FPGA.

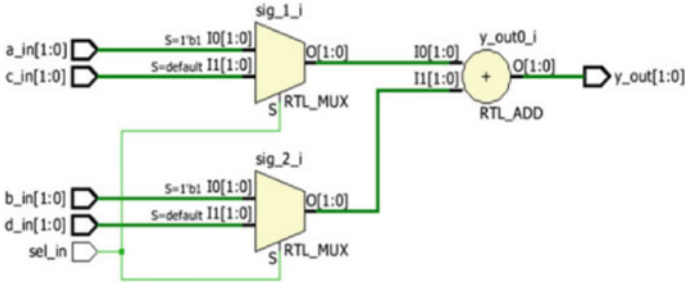
This technique can also optimize the critical path of the logic designs which can decrease delay and increase processing speed.

### 3.2 Grouping Logic [14]

Similarly, the design performance can be improved by grouping the resources at RTL level, as shown in Fig. 14.

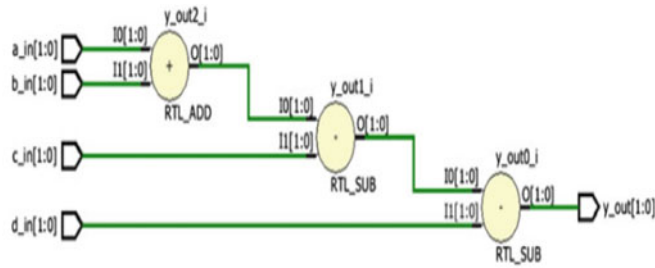


(a). without resource sharing

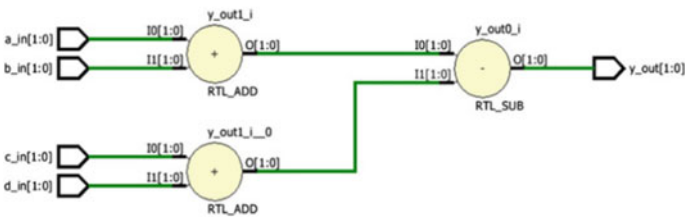


(b). with resource sharing

**Fig. 13** Typical resource sharing techniques after synthesis

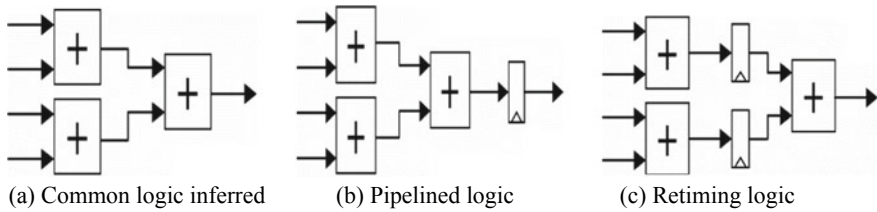


(a). without grouping



(b). with grouping

**Fig. 14** Typical grouping techniques after synthesis



**Fig. 15** Concept of retiming after synthesis

Figure 14a shown without grouping a cascaded logic comprising of one adder and two subtractors. The cascaded logic is a priority logic with estimated delay value of  $n * t_{pd}$ , where  $n$  is number of adders/subtractors with  $t_{pd}$  propagation delay. For the present case, the estimated delay is  $3t_{pd}$ , whereas in Fig. 14b shows the realization of same logic using grouping technique which utilizes two adders followed by one subtractor and estimated delay become  $2t_{pd}$  only.

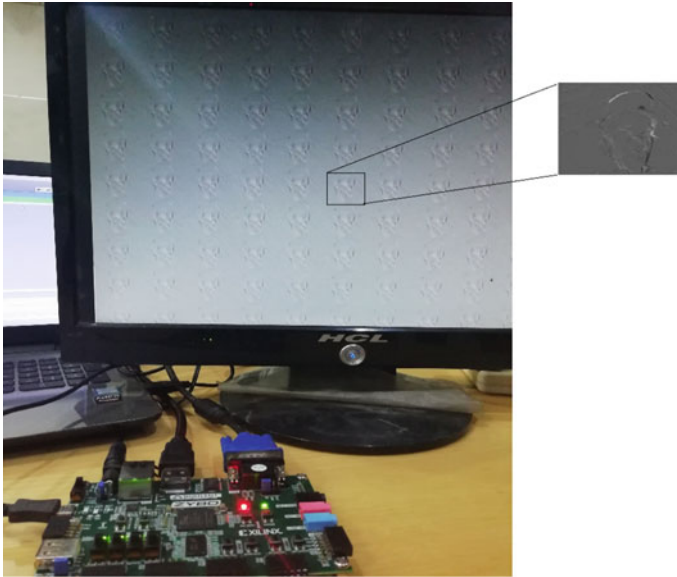
### 3.3 Retiming Pipeline [15]

Register timing concept is used at implementation level to improve timing by reordering the combinational and sequential logic in the data path. This technique can optimize registers and the balance pipeline to achieve higher operating frequency. Figure 15a–c shows the common logic inferred with combinational delay, pipeline register logic and retiming registers logic, respectively. For proposed SAD design at implementation level, retiming pipeline has been used to improve the speed.

### 3.4 Hardware Prototyping Setup

Represented in Fig. 16, the hardware was set up for prototyping on Xilinx Zybo kit (Artix-7 FPGA). The “foreman” video sequence has been prepared as data set for testing. The frame number 32 used as a current frame and frame number 30 used as a reference frame, which are stored in BRAM. These two data sets have been given as input to the SAD logic which computes the difference between the given two frames and stores it in flash memory.

Further, the flash memory data is provided as input to video controller unit to display the reconstructed residual frame on monitor as shown in inset of Fig. 16. Due to default tiles configuration of monitor (output display), output image of video controller unit displays all over the monitor screen.



**Fig. 16** Residual frame generation on FPGA

### 4 Results and Discussion

The design comparison of proposed work with existing design is shown in Table 2. More work has been reported on 65 nm technology node in past ten years [4, 7, 8], whereas very few work has been reported on 28 nm technology node [6]. The proposed design shows higher operating frequency because of the techniques employed as shown in Table 3, which optimizes the speed on the expense of higher number of LUTs and register as compared to 65 nm technology node.

**Table 2** FPGA implementation results and comparison

	[4]	[7]	[8]	[6]	This work
FPGA	Virtex-5	Virtex-5	Virtex-5	Virtex-7	Artix-7
Technology node (nm)	65	65	65	28	28
No. of used registers	8841	4266	9180	39,901	22,880
No. of LUTs	17,992	14,761	15,453	24,957	25,072
Max. Frequency (MHz)	190.785	475.21	171.9	458.7	498.4
Block size (4 × 4 to 64 × 64)	Yes	Yes	Yes	Yes	Yes
Delay for every 64 × 64 block (ns)	167.73	44.19	372.2	45.98	32.06
AMP support	Yes	Yes	Yes	Yes	Yes

**Table 3** List of optimization techniques employed

S. No	Technique	Levels
1	Parallel	Architecture
2	Pipeline	
3	Grouping logic	RTL modeling
4	Resource sharing	
5	Retiming	Implementation

The SAD design without optimization operates at 368 MHz with delay of 58.07 ns for each  $64 \times 64$  block, whereas with optimization, it operates at 498.4 MHz with delay of 32.06 ns for each  $64 \times 64$  block.

## 5 Conclusion

A mixed parallel-pipelined processing SAD architecture for ME has been presented. The architecture was synthesized and prototyped on Xilinx Artix-7 FPGA. The design uses 256 number of  $4 \times 4$  SAD blocks which are operating in a mixed mode of parallel-pipeline to calculate SAD values of  $64 \times 64$  block. The design operates at frequency of 498.4 MHz with the delay of 32.06 ns for each  $64 \times 64$  block. The design performance has been improved at different levels viz. architecture, RTL modeling and implementation. This design can be used for any kind of ME processes including fast and full search algorithms. It can also be used in consumer electronics where the performance is most crucial.

## References

1. Sullivan GJ, Ohm J-R, Han W-J, Wiegand T (2012) Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans Circ Syst Video Technol* 22(12):1649–1668
2. Zheng, J, Lu C, Guo J, Chen D, Guo D (2019) A hardware efficient block matching algorithm and its hardware design for variable block size motion estimation in ultra-high-definition video encoding. *ACM Trans Des Autom Electron Syst (TODAES)* 24(2):15:1–21
3. Jia L, Tsui C-Y, Au OC, Jia K (2018) A low-power motion estimation architecture for HEVC based on a new sum of absolute difference computation. *IEEE Trans Circ Syst Video Technol*
4. Dinh VN, Phuong HA, Duc DV, Ha PTK, Tien PV, Thang NV (2006) High speed SAD architecture for variable block size motion estimation in HEVC encoder. In: *IEEE sixth international conference on communications and electronics (ICCE)*. Vietnam pp 195–198
5. Silveira B, Paim G, Abreu B, Grellert M, Diniz CM, Ceşar da Costa EA, Bampi S (2017) Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design. *IEEE Trans Circ Syst I: Regul Papers* 64(12):3126–3137
6. Medhat A, Shalaby A, Sayed MS, Elsabrouty M, Mehdipour F (2014) A highly parallel SAD architecture for motion estimation in HEVC encoder. In: *IEEE Asia Pacific conference on circuits and systems (APCCAS)*, Japan, pp 280–283



7. Joshi AM, Ansari MS, Sahu C (2018) VLSI architecture of high speed SAD for high efficiency video coding (HEVC) encoder. In: IEEE international symposium on circuits and systems (ISCAS). Italy, pp 1–4
8. Nalluri P, Alves LN, Navarro A (2014) High speed SAD architectures for variable block size motion estimation in HEVC video coding. In IEEE international conference on image processing (ICIP). France, pp 1233–1237
9. Abreu B, Santana G, Grellert M, Paim G, Rocha L, da Costa EAC, Bampi S (2018) Exploiting partial distortion elimination in the sum of absolute differences for energy-efficient HEVC integer motion estimation. In: 31st symposium on integrated circuits and systems design (SBCCI), Brazil, pp 1–6
10. Porto R, Agostini L, Zatt L, Roma N, Porto M (2019) Power-efficient approximate SAD architecture with LOA imprecise adders. In: IEEE 10th latin american symposium on circuits & systems (LASCAS), Colombia, pp 65–68
11. Vanne J, Viitanen M, Hamalainen TD, Hallapuro A (2012) Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. IEEE Trans Circ Syst Video Technol 22(12):1885–1898
12. He G, Zhou D, Li Y, Chen Z, Zhang T, Goto S (2015) High-throughput power-efficient VLSI architecture of fractional motion estimation for ultra-HD HEVC video encoding. IEEE Trans Very Large Scale Integ (VLSI) Syst 23(12):3138–3142
13. Foreman QCIF Video sequence: <http://www2.tkn.tu-berlin.de/research/evalvid/qcif.html>. Last accessed 18 July 2019
14. Vaibbhav T (2019) Advanced HDL synthesis and SOC prototyping, 1st edn. Springer Singapore, Singapore
15. Eguro K, Hauck S (2008) Simultaneous retiming and placement for pipelined netlists. In: 16th international symposium on field-programmable custom computing machines. California, pp 139–148