Homeland security video surveillance system utilising the internet of video things for smart cities

Yasser Ismail1 | Mohamed Hammad2 | Mahmoud Darwich3 | Wael Elmedany2

1Electrical Engineering Department, College of Sciences and Engineering, Southern University and A&M College, Baton Rouge, LA, USA
2College of Information Technology, University of Bahrain, Sakhir, Bahrain
3Department of Mathematical and Digital Sciences, Bloomsburg University of Pennsylvania, PA, USA

Abstract
In the contemporary video surveillance system, there have been many efforts made to maintain high security and extend the coverage areas in most of the countries around the world. The deployment of many surveillance cameras and sensors capable of detecting abnormal and meaningful events on the territories' streets and airports is an aspect of internal security. There are two main problems that affect the homeland security system, and the security cameras and sensors are not enough to cover all areas in the country. This is because of the high cost of the video surveillance cameras and the sensor installations, and the non-standardisation of security cameras and sensors manufacturing, which is due to the differentiated infrastructure of companies or organizations that provide home security. The authors introduce a design and hardware implementation of a motion estimation (ME) co-processor that can be used for video surveillance cameras in homeland security. The proposed ME co-processor, if adopted in video surveillance cameras, can be connected utilising an internet of video things infrastructure (IoVT). The proposed co-processor is suited for high-efficiency encoding video surveillance systems (H.265/HEVC). Furthermore, to reduce the memory I/O, data reuse Level A and Level B have been used in the proposed architecture while taking full advantage of the hardware resources. Moreover, an effective local memory has been used to reuse the data during the process of loading both the search area and the current block into the processing element array (PE array). The performance of the proposed architecture has been calculated using subjective and quantitative measures techniques and compared to the full search block-based motion estimation (FSBB-ME) algorithm. Moreover, the proposed architecture achieves a very high video resolution accuracy that is similar to the accuracy of the FSBB-ME algorithm. Modelism-version10.4a has been used for simulation and time verification testing proposes. The proposed ME co-processor can be embedded in the compressing decompressing and high definition broadcast for video surveillance systems.

1 INTRODUCTION AND LITERATURE REVIEW

The internet of things (IoT) brings significant benefits to users and companies of smart cities. It has been introduced to connect billions of physical devices to the internet and allows these devices to communicate with each other and transfer data over the internet [1, 2]. Video surveillance cameras, vehicles and sensors are examples of these physical objects that can exchange data between them. The development of smart cities depends on the future technology of the IoT. There is a wide range of applications of the IoT in smart cities such as smart shopping systems, smart healthcare, smart transportation and homeland security [3–7]. The main challenges of the IoT implementation and connecting many devices through the IoT are the size of the connected devices, the power consumption and the transmission data rate of the connected devices.

Video surveillance cameras are becoming a common feature of smart cities where they are useful for terrorist detection, forensic evidence, and crime prevention. The IoT-based video surveillance system is the future technology, which is under the research scope of scientists nowadays,
where security cameras are connected through the internet to monitor many different places [8–13]. The deployment of several video cameras and sensors to discover abnormal and purposeful events in public places is one aspect of internal security. Additionally, the increase in using video cameras and connected sensors in the video surveillance system increases the required transmission data rate as well as the overall cost of the surveillance system. This increase in the required transmission data rate increases the complexity of the hardware design, which may decrease the overall speed of the security system.

With the fast growth and advancement of the IoT, a huge amount of data is generated from IoT devices. Video surveillance cameras are one of the most important applications of IoT devices that generate a huge amount of data used in multimedia data storage and transmission in smart cities. Some many techniques and algorithms are used to achieve a high quality of video compression and coding in real-time video surveillance cameras [14, 15]. Most of the proposed H.265/HEVC encoders in the literature are focusing on achieving high performance, better quality, low complexity and configurable architecture. A joint explorations model that is based on distributed video coding (DVC) has been introduced in [16] to achieve higher performance and effective quality consistency control. Furthermore, an high efficient video coding (HEVC) video with a new data hiding method has been introduced in [17], and the proposed technique can be applied for a coding block size of 4*4, 8*8 and 16*16 pixels. In this method, the Bose, Chaudhuri and Hocquenghem (BCH) error correcting code has been used to improve the robustness, and the HEVC encoding process is shown in Figure 1 [17]. A hardware implementation of integer ME for HEVC/H.265 encoder with high quality has been introduced in [18]. The proposed integer ME architecture is shown in Figure 2. The proposed architecture uses different adaptive multiresolution strategies coupled with accurate full processing element modes integer ME at the finest level. The reference search window and original pixels are first stored in the RAM arrays and then passed to the computing unit with the cooperation of the control module.

Note that a configurable H.265/HEVC video encoding architecture has been introduced in [19] using FPGA. The proposed H.265/HEVC encoder is structured using four stages coding tree pipelined unit, and targeting a flexible FPGA platform. A high-efficiency video coding (HEVC) encoder for vehicular ad-hoc networks (VANETs) with low complexity is proposed in [15]. The authors claimed that the proposed architecture reduces the complexity by 50% for VANETs. Another research for reducing the complexity of H.265/HEVC video coding by using the maximum entropy principle has been introduced by authors in [14].

The IoT includes many sensors that are connected to the internet and provides real-time reading through the connected sensors. On the other hand, the IoVT includes video cameras connected to the internet and provides real team video streaming [20]. The next generation of video surveillance system (VSS) integrates the IoT devices and smart objects to come up with IoVT [21]. It will extend the variety of

![Diagram](image-url)
applications from national security to home security applications, such as intelligent urban surveillance systems and smart cities. The IoVT architecture in [22] includes three different layers, edge, fog, and cloud. The edge fog cloud framework architecture is arranged together to provide efficient video streaming capabilities that can be used in many different applications. This architecture allows efficient communication between devices on the edge of fog or fog to cloud. Devices in the edge layer are designed to capture images and videos and send this visual surveillance data to the fog layer for performing computational tasks that require low latency. The cloud layer provides storage resources and performs the largest amount of computation and analysis. Internet of multimedia things (IoMT) is the another subset of IoT, which integrates many devices and smart multimedia things including video cameras at the internet edge [23, 24].

1.1 Main contribution

Connecting video cameras using the IoVT infrastructures in most of the previously discussed systems is impossible if we are targeting to cover the whole area of the United States. This is because of the huge number of surveillance cameras needed to cover the United States and the very high cost of such surveillance cameras.

The main contribution of this article is to establish a low-cost video surveillance camera that can be adopted into a video surveillance system, where the security cameras are connected through IoVT infrastructure. The main target of this research is to cover all states in the US, by increasing the number of security cameras. Also, to reduce the overall cost, the size of the security cameras should be minimised. Additionally, the hardware cost of the security cameras that are used in video surveillance systems should be minimised. The hardware cost can be reduced by reducing the overall computations that are needed for the video encoding process. Reducing the computations of the encoding process not only reduces the overall hardware cost of security surveillance cameras but also increase the video transmission speed of such cameras. As a result, video surveillance cameras can process and transmit online videos in a high-speed fashion and can be connected using IoVT infrastructure.

Designing an optimised video surveillance camera having a minimum size, low cost of the hardware design and good quality of video transmission, is our target to get an acceptable transmitted bit-rate and low-cost video surveillance system. The proposed architecture here when adopted in a video surveillance camera, is expected to be used in homeland security-based applications such as smart traffic, schools, airports and hospitals.

It can be concluded from the previous discussion that our target is to increase the number of video surveillance cameras that can be connected utilising the IoVT infrastructure without a major increase in the overall cost of the surveillance system. This can be achieved by reducing the hardware cost and the area of each video surveillance camera in the surveillance system. The work presented by the authors discusses the implementation of a high-fidelity ME (ME) co-processor that can be adopted into the H.265/HEVC encoder of a video surveillance camera. As a result, more cameras can be adopted into the surveillance system with no extra cost. These large number of cameras can be connected through a traditional IoT infrastructure since the expected transmission data rate will not increase due to the high video transmission coding efficiency of the proposed ME co-processor.

Our ME co-processor has a low hardware-cost that leads to less area and power consumption. Consequently, the transmission speed will increase due to the reduction in both the area and the hardware cost of our co-processor. The resultant ME co-processor, if adopted into the H.265/HEVC coder decoder (CODEC) system, will be suited for the IoVT infrastructure which requires less area and power consumption as well as high-speed transmission of the used video devices. Data reuse is used to reduce the huge video data required for the ME process of the H.265/HEVC encoder. Utilising the data reuse principle in the proposed architecture not only increases the video encoding speed but also decreases the hardware needed for fetching the video data from the external memory. As a result, fewer hardware costs and areas are obtained in our architecture. The proposed architecture is validated using the very high-speed integrated circuit (VHDL) language and Matlab tools. It is worth mentioning that the proposed work in this journal is an extension of the work presented in [25].

The remaining part of the article is organised as follows. Section 2 discusses in detail the problem formulation; in Section 3, ME process is discussed; the memory I/O Bandwidth Reduction is discussed in Section 4; in Section 5, the architecture of the introduced ME is discussed; Section 6 discusses the design of local memory; in Section 7, the simulation results and discussion are elaborated; and finally the conclusion is given in Section 8.
2 | PROBLEM FORMULATION

Now, to reduce the total cost of the video camera, the calculations needed for the coding and signal transmission has to be reduced. As a result, the overall required resources of the hardware design, and the security camera area, will be reduced. This will allow more cameras to be used in the surveillance system at a similar cost or maybe less. As a result, additional areas can be monitored by more surveillance cameras at no additional cost. The proposed architecture can be used in airports, public streets, shopping malls, hospitals, agriculture, traffic monitoring and other governmental security-based applications.

Figure 3 shows the consumed time distribution of HEVC encoder with H.265, which is used for high quality (HQ) signal transmission in video surveillance systems [26]. Since ME is the most consumed process in HEVC-H.265 codec, it consumes almost 84% of the total process as seen in Figure 3, and so to reduce the time delay of the video coding process, ME computations must be reduced.

3 | H.265/HEVC MOTION ESTIMATION PROCESS

ME is a video coding and transmission technique, where the transformation of a current block (CB) (located at the current frame) from a reference block (RB) (located at the reference frame) is determined by the motion vectors of the two frames. The most common algorithm of ME is the full Search block matching estimation (FBMA) [16–18]. In the FBMA process, all frames are divided into N × N blocks. The blocks should be non-overlapped inside the frame, and each block in the current frame is matched to another block within a given search range in the previous frame. Figure 4 shows how to implement the FBMA algorithm. The best match RB can be obtained through an exhaustive search at all points of the search area that has a dimension of \(2P_{\text{max}} \times 2P_{\text{max}}\) search points. The value \(2P_{\text{max}}\) represents the width and length of the selected search area. The best match RB is selecting the search point with the minimum cost that is represented by the sum of absolute difference (SAD) value. The actual motion vector (AMV) seen in Figure 4 will be used by both the encoder and the decoder to reconstruct the RB. The AMV represents the distance from the centre of the search area to the best match RB. The AMV can be calculated as follows:

\[
AMV(k, l) = \arg \min [SAD(k, l; u, v)]
\]

Where \(-P_{\text{max}} \leq u, v \leq P_{\text{max}}, K,\) and \(l\) are the coordinates of the CB of the search centre, \(u\) and \(v\) are the coordinates of the best match candidate block, and \(\text{argmin}\) is a Matlab function used to find the minimum value in an input vector.

Video applications consume much data compared to other applications such as speech applications. Table 1 shows various video data formats and as an example, the SIF video sequences, \(32 \times 32\) search area, and \(16 \times 16\) CB size are needed. While for SDTV and HDTV video sequences, \(64 \times 64\) search area, and \(32 \times 32\) CB size are required [8]. We concluded from Table 1 two main important notations:

1. There is a need for higher number of pixels of the search area from the memory as the resolution of a video has been increased [27]. As an example, ultra high definition television (UHDTV) transmission requires more frame rate and sample bit depth than video Conferencing, which uses source input format (SIF). The presented architecture by the authors allows the reuse of existing data for better use of the available memory input/output bandwidth. Consequently, there is no need for excessive use of the external memory for fetching a large amount of data.
2. Also, to achieve high video resolution, a fast FBMA algorithm is needed, which requires more computations. Here, a fast FBMA algorithm will be implemented for great savings in the ME computations.

There are two main problems in H.265/HEVC standard: the huge number of data required for performing the ME process and the high computational complexity compared to the previous standards [28]. Both problems prevent the use of H.265/HEVC CODEC into an IoT infrastructure. The huge data results from two main reasons. First, by increasing the density of each pixel; for example, 4K and UHD pictures have 10-bits/pixel. Second, by increasing both the CB and the search area sizes, the size of the CB and the needed search area is \(32 \times 32\) pixels and \(64 \times 64\) pixels, respectively. Consequently, large computations are required to process such huge data.

The problem of the huge number of pixels to be processed in the ME process can be solved using a smart internal memory in our co-processor which is important for data reuse. The data reuse principle is very important to reduce the amount of data needed from the main memory. The high computational complexity problem can be solved by utilising smart data flow and optimised hardware for the designed co-processor. Combining such two previous techniques can greatly reduce the required computations for the ME process. As a result, the encoding speed, power consumption and
will the take of time required of one algorithm. High speed the data from process bandwidth (HD).

The used time the algorithm will discussed or algorithm 8 bandwidth only algorithm. Th for huge process the 4 ME which two videos memor reuse infrastr.

As sa the are reduced. the I/O data required are to needed is time huge achieve accessing access for um definition allow only the problem redundant T treatly y minimize only algorithm process one bandwidth ws C utilising bandwidth y result, by IoT been the o o memor, I/O been the o memor. Here, I/O structure e video hing an main muc I/O hing ive reduced data staining an main, I/O requirement due, a for required y real algorithm reuse in reuse required efficient encoding, y video up video y for els, SIF has been split into 4 levels; A, B, C and D [8]. Here, only two levels are discussed in the following sub-sections.

4 | THE MEMORY INPUT/OUTPUT BW REDUCTION

The main encoding problem of high definition (HD) videos is the requirement of huge memory I/O bandwidth perform the ME process. To reduce the required I/O bandwidth for the ME process, a data reuse algorithm has been used for fetching data from external memory. The huge savings in the I/O bandwidth is due to fetching the video data from the memory only one time. The data reuse algorithm will speed-up the overall process of video encoding, and allow real-time video applications to take place, which are connected utilising the IoT infrastructure.

The data reuse algorithm is an efficient video encoding technique used to decrease the required bandwidth for I/O memory in the ME process [8]. Utilising data reuse allows skipping much redundant memory access time required for the ME process. This is achieved by accessing the required data pixels, needed for the ME process, only one time. As a result, the I/O memory bandwidth will be greatly reduced. To achieve the minimum bandwidth of the I/O memory, the data reuse

| Resolution         | Pixels/line | Lines/frame | Frames per second |
|--------------------|-------------|-------------|-------------------|
| UHD 8k             | 7680        | 4320        | 30                |
| UHD 4k             | 3840        | 2160        | 30                |
| HDTV broadcast (D1)| 1920        | 1080        | 30                |
| SDTV broadcast (D1)| 720         | 486         | 30                |
| Video conferencing (SIF) | 352  | 240         | 30                |

4.1 | Data reuse (Level A)

For a CB of size M x M pixels shown in Figure 5, the size of the search area is SAx x SAy; where SAx and SAy are equal to 2Pmax. For the contour pixels of the search area, additional padding pixels are used and see the dashed lines area in Figure 5.

Figure 5 shows levels of A&B data reuse, and there are two blocks 1&2, where the two blocks are overlapped in many search area. To reduce the overall access times to the external memory, the overlapped areas will be fetched only one instead of two access times as for the M-1 overlapped pixels shown in Figure 5.

4.2 | Data reuse (Level B)

During the search process for the best match RB (or candidate block) in the search area, the search process proceeds by searching the second line of the search area. As seen in the bottom part of Figure 5, huge data pixels will be overlapped if

FIGURE 4 Adjacent frames in ME encoding process (H.265/HEVC)

TABLE 1 Video transmission formats
searching one line and the next line of a search area. This is called data reuse level B if fetched only one time from memory. The overlapped data pixels has a size of \(SA_r \times (M - 1)\). It will be performed while calculating the sum of the absolute difference (AD) between the CB and both the candidate blocks strip #1 and the candidate blocks strip #2 as seen in Figure 5. The Data reuse for level B is achieved during the process of loading \(SA_s\) data pixels and when moving from one horizontal strip down to the next strip.

5 | THE PROPOSED ME ARCHITECTURE

Now, an efficient ME co-processor architecture suited to be adopted in the H.265/HEVC video CODECs that are connected by utilising proposed the IoT infrastructure. The proposed architecture uses a smart data reuse level A and level B to decrease the I/O memory band width (BW). As a result, the required data to be fetched from the memory will be less. ME process requires fetching huge data from the memory, and by using data reuse, fewer data will be fetched from the memory. As a result, both the memory access and the waiting time to fetch the search area will decrease and consequently increase the speed of the ME process. We use a CB of size \(32 \times 32\) pixels which is recommended by the H.265 Standard [29] in our implementation. Additionally, the size of the search area (fetched from the main memory) is \(64 \times 64\) pixels which are compatible with UHD 4K video sequences [30]. Figure 6 shows the top level of the implemented ME co-processor architecture.

The AD values between the CB and the RB (in the search area) will be calculated by the PE array. The PE array consists of \(32 \times 32\) PEs. A small local memory is used to stores the pixels of the search area and to perform data reuse. This will reduce the memory access and the data needed from the memory to perform the ME process. The adder tree the sum of the 1024 AD (that result from the \(32 \times 32\) PEs) in parallel. The sum of the AD calculated by the adder tree will be sent to the compare unit. The comparison unit is used to calculate the minimum distortion \((\text{SAD}_{\text{min}})\) between the CB and all candidates in the search area. Once the \(\text{SAD}_{\text{min}}\) is calculated, the AMV will be decided using a lookup table. All control signals are generated by the control unit.

Moreover, an external memory in the main processor is used to store both the current frame and the reference frame. During the ME operation, the CB and the search area pixels will be loaded from the external memory. Both the CB and the
search area pixels are loaded into the local registers of the PE array and the local memory, respectively. It is worth mentioning that a smart local memory is designed to maximise the coding performance while loading both the CB and the search area. The implemented structure in Figure 6 achieves both data reuse and regular data flow. This will guarantee a high-speed of the proposed architecture as well as simplicity in its design structure.

5.1 The processing element array

The PE array is responsible for calculating the AD values between the current and the reference blocks. The PE array consists of 32 × 32 PEs, each PE is responsible for computing the AD between a pixel from the CB and a pixel from the RB located at the search area. In the implemented ME co-processor, the CB remains the same whereas the RB data moves horizontally from right to left. The pre-load time is defined as the required time to load all CBRs and RBRs in the PE array with the CB and the candidate block, respectively, to begin the ME operations. Referring to Figure 7, during the first 32 cycles, the de-multiplexer (DEMUX) output will feed the PE array with the CB pixels. During the pre-load time, the CB data will be pushed horizontally from right to the left directly into the PE array in forms of 32-pixels column by column. In the next 32 clock cycles, the RB (candidate block) is pushed into the PE array exactly like the CB, but in this time the DEMUX output will be connected to the terminal output #1. Figure 8 shows the data flow of both the CB and the RB into the PE array. Connecting each PE only to its immediate neighbour, therefore, shorter interconnects are obtained. This adds many advantages such as smaller delays, less area, low cross-talk noise and less power consumption. In the implemented ME co-processor, each register data in the search area is accessed only once. A small internal memory is used to perform the data reuse principle. Data reuse is achieved within a row of RB, by pushing new 32 pixels column data every clock cycle using the RBRin terminal as seen in Figure 8. Before data is written into the RBR registers, the old data is shifted to the left sideways. Using this method, the data in the PE array is reused.

Using a [−32, 31] search window, each new 32-pixels row of reference blocks takes 64 clock cycles to finish.

5.2 Local memory

The data reuse is achieved by using the local memory unit. It is used to store the search area data considering the reduction of the I/O memory BW. Additionally, only one external memory access is required for each search area. Figure 9 shows the used architecture of the local memory connected to the 32 × 32 PE array. The local memory consists of (1) DEMUX and the sub-memory units. Each sub memory consists of 32 × 32 pixels. Each pixel is represented by 10 bits as this is defined in the H.265 standard. Consequently, each sub memory consists of 32 × 32 10 bits registers as seen in Figure 10.
The output terminal 0 of the DEMUX is used to pass the CB pixels values from the external memory to the PE array. Output terminals 1, 2 and 3 of the DEMUX are used to pass the search area to sub memories 1, 2 and 3, respectively. A 320-bits (32 pixels wide) data bus is used to move the data from the external memory. A 32 pixels-row of data is pushed from the bottom row registers using the DEMUX while the existing data is shifted to the top coming from external memory. Data is loaded to the PE array as 32-pixel columns.

The pre-load time is 96 clock cycles. The first 32 clock cycles are needed to fill the PE array with the CB pixels values via terminal 0 of the DEMUX as seen in Figure 9. During the second 32 clock cycles, sub-memory 1 is filled via terminal 1 of the DEMUX. During the last 32 clock cycles of the pre-load time, all values inside sub-memory 1 will be installed inside the PE array, and at the same time, sub-memory 2 will be filled as seen in Figure 9. After that, sub-memory 3 is filled while loading the PE array with sub-memory 2 data column by column. Using the previous operations schedule, level A data reuse is achieved.

5.3 | Adder tree

The 1024 AD values coming from the PE array are added together to get the SAD value using an adder tree. Adder tree can perform additions in a parallel fashion to speed up the addition process.
The whole $32 \times 32$ AD values resulting from the PE array are divided into $32 \times 4 \times 4$ arrays shown in Figure 11. Each group uses the adder tree to add all of its AD values. Multiple parallel operations are performed at the same time to get the final SAD value. The $4 \times 4$ sub-array is selected to reduce the complexity of the summation operation and achieve high-speed operation. The more the size of the sub-array is, the slower the adder tree will be.

### 5.4 | The ME controller

The ME controller is responsible for producing all the control signals that are needed for the designed ME co-processor. The control unit is considered to be the main and the most complex unit in the whole architecture. It controls all components in the implemented architecture. The ME controller consists mainly of an up-counter sub-component and control signals subcomponent. The controller has only three inputs, which are enabled, reset and the clock as seen in Figure 12. As the name complies, the up-counter subcomponent is used for counting the number of clocks that are needed for a complete ME process. In the implemented architecture the search area is $64 \times 64$ which requires a 13-bit counter to finish the whole ME process with 6116 clock cycles. The counter can be started using enable, reset and system clock. The counter value should be reset every new ME process. This counter is used also to locate the candidate block inside the search area. The control signals are generated based on the value of the up-counter. These control signals are used to control and initialise all resources of the whole ME design. The ME controller produces many control signals during the ME as shown in Table 2.

| Control signal name | Purpose |
|---------------------|---------|
| Selector            | Demux selector to choose the required sub-memory |
| AVM_mem_en          | To enable/disable AMV memory |
| mem1_en             | To enable/disable Sub memory 1 |
| mem2_en             | To enable/disable Sub memory 2 |
| mem3_en             | To enable/disable Sub memory 3 |
| comp_en             | To enable/disable comparator unit |
| Pos                 | To count for the AMV value |

**FIGURE 13** The high-level structure of the designed local memory

**TABLE 2** The control signals
6  DATA FLOW AND THE LOCAL MEMORY

The local memory is used for storing the required data for the search area, considering the reduction of the I/O memory BW. Additionally, the external memory access is reduced to only one access for each search area. Figure 13 shows the used architecture for the local memory connected to the $32 \times 32$ PE array. The local memory consists of DEMUX and the sub-memory units. Each sub memory consists of $32 \times 32$ pixels. Each pixel is represented by 10 bits as this is defined in the H.265 standard. Consequently, each sub memory consists of $32 \times 32$ 10-bits registers.

The CB pixels values are passed directly to the processing element array, where it is transferred from the external memory as seen in Figure 13. Output terminals 1, 2, and 3 of the DEMUX are used to pass the search area to sub memories 1, 2 and 3, respectively. Also, a 320-bits (32 pixels wide) data bus is used to move the data from the external memory. Furthermore, a 32-pixels row of data is pushed from the bottom row registers using the DEMUX while existing data is shifted to the top coming from external memory. Data is loaded to the PE array as 32-pixels columns.

The pre-load time is 96 clock cycles. The first 32 clock cycles are needed to fill the PE array with the CB pixels values via terminal 0 of the DEMUX as seen in Figure 13. During the second 32 clock cycles, sub-memory 1 is filled via terminal 1 of the DEMUX. During the last 32 clock cycles of the pre-load time, all values inside sub-memory 1 will be installed inside the PE array. And at the same time, sub-memory 2 will be filled as seen in Figure 13. After that, sub-memory 3 is filled while loading the PE array with sub-memory 2 data column by column. Using the previous operations schedule, level A data reuse is achieved. Level B data reuse will be achieved by loading Register Bank #1 by one row through terminal #1 of the DEMUX. This operation will start when Register Bank #2 starts loading the PE array. The operations will be repeated until all search area pixels are loaded into the PE array to be processed. The whole ME process in the architecture is taking $(64 \text{-row search area}) \times (95 \text{ clock cycles for reading one slice using sub-memory 1, 2 and 3}) + (36 \text{ setup clock cycles})$ which are 6116 clock cycles in total.

7  SIMULATION RESULTS AND DISCUSSIONS

7.1  Simulation setup

Frames are divided into blocks of size $32 \times 32$ pixels per block. The size of the search area is $64 \times 64$ pixels. The VHDL (Modelism-version10.4a) was used as a time verification tool of the proposed ME co-processor. A 100 ns clock cycle was used in the implementation. Video frames of size $720 \times 486$ and $3840 \times 2160$ pixels are used. In the following sub-sections, all individual components of the proposed co-processor are tested and evaluated using the VHDL tool.

7.2  Process element (PE)

The AD unit calculates the difference between the two input pixels from the CB and the RB. The output of this unit goes as an input to the ADR. Figure 14 shows the output of the PE for different inputs. The result is the AD between these two input values. The input values are $(|3-1| = 2), (|1-5| = 4), (|7-1| = 6), (|0-4| = 4)$, and $(|10-13| = 3)$. It is worth mentioning that each PE takes one clock cycle to get an output.

7.3  PE row

This subsection focuses on the verification of the operation of one row of the PE array. 32 clock cycles are needed to enter 32 pixels data in both the current and the reference registers of each PE. The output will be calculated in clock cycle number 33 as shown in Figure 15. The following matrices show the value of CBR and RBR in one row of the PE array.

$$
\text{CBR} = [01\ 02\ 07\ 08\ 0F\ 01\ 0A\ 09\ 0B\ 08\ 01\ 09\ 0E\ 01\ 0A\ 04\ 01\ 02\ 07\ 08\ 0F\ 01\ 0A\ 09\ 0B\ 08\ 01\ 09\ 0E\ 01\ 0A\ 04]
$$

FIGURE 14  Simulation of the PE
**Figure 15** Simulation of the PE row

**Figure 16** Simulation of PE Array for the first 32 clock cycle

**Figure 17** Simulation of the SAD unit
FIGURE 18  Simulation of compare unit

\[
\begin{array}{c|c|c|c|c}
\text{now} & \text{0 ns} & \text{100 ns} & \text{200 ns} & \text{300 ns} \\
\hline
\text{cursor} & 0 \text{H} & 0 \text{H} & 0 \text{H} & 0 \text{H} \\
\text{rbr} & 20 \text{H}001 & 20 \text{H}002 & 20 \text{H}003 & 20 \text{H}004 \\
\text{sysclk} & 1 \text{2H}001 & 1 \text{2H}002 & 1 \text{2H}003 & 1 \text{2H}004 \\
\text{sysad} & 1 \text{2H}001 & 1 \text{2H}002 & 1 \text{2H}003 & 1 \text{2H}004 \\
\end{array}
\]

\begin{itemize}
\item RBR = [03 05 03 05 06 0C 03 04 02 03 04 05 04 03 01 02 03 05 03 05 06 0C 03 04 02 03 04 05 04 03 01 02]
\item PE Row Output = [02 03 04 03 09 0B 07 05 09 05 03 04 02 09 02 02 03 04 03 09 0B 07 05 09 05 03 04 02 09 02]
\end{itemize}

The below calculation is performed to get the output shown in Figure 15. \([|1\text{-}3| = 2, \ (|2\text{-}5| = 3, \ (|7\text{-}3| = 4, \ (|8\text{-}5| = 3, \ (|15\text{-}6| = 9, \ (|1\text{-}12| = B, \ (|10\text{-}3| = 7, \ (|9\text{-}4| = 5, \ (|10\text{-}2| = 9, \ (|8\text{-}3| = 5, \ (|1\text{-}4| = 3, \ (|9\text{-}5| = 4, \ (|14\text{-}4| = A), \ (|1\text{-}3| = 2, \ (|10\text{-}1| = 9, \ and \ (|4\text{-}2| = 2).\]

7.4  PE array

Figure 16 shows the output of the PE array using the following matrices, and the input data is only for one PE for each row to simplify the simulation results. The following matrices are 32 × 32 CB and 32 × 32 RB, the output matrix is calculated to show the AD between the CB and RB matrices. Thirty-two clock cycles are needed to enter 32 rows of data from right to left, and the output will be calculated in clock cycle number 33 as shown in Figure 16.

\[
\text{CB} = \begin{bmatrix} 00 & 00 & 00 & \ldots & 00 & 00 & 02 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 05 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 04 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 06 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 0F \\ 00 & 00 & 00 & \ldots & 00 & 00 & 02 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 0F \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \end{bmatrix}
\]

\[
\text{RB} = \begin{bmatrix} 00 & 00 & 00 & \ldots & 00 & 00 & 01 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 02 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 03 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 05 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 03 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 04 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 01 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \\ 00 & 00 & 00 & \ldots & 00 & 00 & 00 \end{bmatrix}
\]

FIGURE 19  First reference frame with a white background and 32×32 black block in position 20H
FIGURE 20  Reference frame (ducks_take_off. yuv frame number 0)

FIGURE 21  Current frame (ducks_take_off. yuv frame number 1)

FIGURE 22  Simulation results of the ME process

PE Array Output =

\[
\begin{array}{c}
\begin{bmatrix}
00 & 00 & 00 & \ldots & 00 & 01 \\
00 & 00 & 00 & \ldots & 00 & 03 \\
00 & 00 & 00 & \ldots & 00 & 01 \\
00 & 00 & 00 & \ldots & 00 & 01 \\
00 & 00 & 00 & \ldots & 00 & 0C \\
00 & 00 & 00 & \ldots & 00 & 08 \\
00 & 00 & 00 & \ldots & 00 & 0E \\
00 & 00 & 00 & \ldots & 00 & 00 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
00 & 00 & 00 & \ldots & 00 & 00 \\
00 & 00 & 00 & \ldots & 00 & 04 \\
\end{bmatrix}
\end{array}
\]
7.5 | SAD unit

Figure 17 shows the simulation results of the SAD unit. The output (AD values) of the PE Array will be sent to the SAD unit to calculate the SAD value. In this case, we select the values of each pixel in the CB matrix to be (1023)_{10}. The pixels of the RB matrix are selected to be 0s. The SAD unit will calculate the SAD between CB and RB matrices. In the first 32 clock cycles, the PE array will be filled after that 2 more clock cycles are needed to get the final SAD value which is ‘FFFFC00’ as shown in Figure 17, so the total clock cycle number for SAD is 34.

7.6 | Compare unit (comparator)

The compare unit is used to compare two SAD values and decide the minimum SAD value. The output of the compare unit should be the minimum SAD and its corresponding position in the search area. As an example, assume two groups of two SAD values group 1 [‘FFFF’, ‘0FFFE’], and group 2 [‘07FFE’, ‘1FFF6’] with the corresponding positions 1, 2, 3 and 4 respectively. As shown in Figure 18, the minimum value according to the input values of SAD in the first group is ‘0FFFE’ at position number 2. And the minimum value according to the input values of SAD in the second group is ‘1FFF6’ at position number 4. The minimum SAD and its position are stored in a register to be compared with the next SADs.

7.7 | Full processor design simulation

In this sub-section, we are going to elaborate on the whole architecture of the ME co-processor given in Figure 6. Four different cases are considered to verify the functionality of the

---

**FIGURE 23** Hardware implementation comparison results: (a) current block (b) best match obtained from the proposed ME co-processor and (c) FSBB-ME (Best match in Matlab)

**FIGURE 24** Reference frame Ducks Take Off. yuv Frame No. 0
proposed architecture. The following are discussions about each case individually.

Full design simulation – Case 2

A real HD video (ducks_take_off.yuv with 720 × 486 pixels) is used here to show that the implemented architecture can work with real video sequences to perform the ME process. Figures 20 and 21 show the reference frame and the current frame, respectively. The ModelSim simulation results are shown in Figure 22, where the best match position is \( (BFE)_{\text{Hex}} \) and the minimum SAD \( (\text{SAD}_{\text{min}}) \) is \( (10ACB)_{\text{Hex}} \). The hardware implementation comparison results are shown in Figure 23. The best match block obtained by the proposed ME co-processor is shown in Figure 23(b), and the best match using FSBB-ME implemented in Matlab is shown in Figure 23(c). The best match candidate blocks of Figure 23(b) and (c) are almost the same, which verifies the high accuracy of the proposed ME co-processor in performing the ME process.

Full design simulation – Case 3

The HD ducksTakeOff.yuv video sequence of a size 720 × 486 pixels is used to evaluate the performance of the proposed architecture. The reference frame and the current frame (that are taken from the ducksTakeOff.yuv video sequence) are illustrated in Figures 24 and 25, respectively. Figures 24 and 25 show also the search area and the CB that will be processed by our ME co-processor to get the best match candidate block. Referring to the search area pixels positions, the best match candidate block is allocated at the position \( (7FF)_{\text{Hex}} \) with a minimum SAD equal to \( (01CA2)_{\text{Hex}} \), and see Figure 26. The results obtained by the proposed ME co-processor are verified also by using Matlab software as seen in Figure 27. Figure 27 shows the current block, the best match candidate block obtained by the proposed ME co-processor and the best match candidate block obtained by implementing the FSBB-ME in Matlab, respectively. It is clear from Figure 27 that the
The proposed co-processor accuracy is very high and the determined best match candidate block obtained by the proposed architecture is almost the same as the best match candidate block obtained by FSBB-ME using Matlab. The high similarity between the CB and the best match candidate block obtained by the proposed architecture means that less residue (the difference between the CB and the best match candidate block) will be obtained and the transmitted data-rate will be very small. As a result, the IoT infrastructure can be used to connect too many surveillance cameras that use our co-processor since the transmitted bit-rate will be acceptable.

Full design simulation – Case 4

4k video sequences (NewsProRes.MOV) of size 3840 × 2160 pixels are used in this case study. The reference frame and the current frame (that are taken from the ducksTakeOff.yuv video sequence) are illustrated in Figures 28 and 29, respectively.
Figures 28 and 29 show also the search area and the CB that will be processed by our ME co-processor to get the best match candidate block. The best match candidate block is allocated at the position (822)_{Hex} with respect to the search area coordinates, and with a SAD_{min} equal to (1107E)_{Hex} as seen in Figure 30. Figure 31 shows the CB, the best match candidate block obtained by the proposed ME co-processor, and the best match candidate block obtained by implementing the FSBB-ME in Matlab, respectively. It is clear from Figure 31 that the calculated best match candidate block by our co-processor is almost the same and similar to the original CB. This reflects the high coding efficiency of our co-processor to process 4K video sequences.

8 | CONCLUSIONS AND FUTURE WORK

A high-fidelity ME co-processor that can be adopted into H.265/HEVC encoders of a security monitoring system was introduced and evaluated here. The implemented ME co-processor achieved high video transmission speed by using smart local memory and data reuse strategies. The designed on-chip memory and the data reuse strategy greatly reduce the memory I/O BW. The proposed architecture also achieved high video encoding quality that allows low transmission data-rate. This allows connecting many surveillance cameras using our architecture with a traditional IoT infrastructure. High video resolutions (HD and 4K resolutions) are used to evaluate the proposed ME co-processor. The pipeline uniformity, parallelism and data reuse allowed the introduced ME processor to achieve high throughput with 100% PE utilization. The achieved efficiency and high throughput of the video signal transmission of the surveillance camera that uses our ME co-processor (high video quality and low transmission data-rate) allow our co-processor to be used in real-time video streaming in smart city security applications. The optimised design and high-speed of the introduced co-processor enable the adoption into the homeland security system utilising a traditional IoT infrastructure. It is worth mentioning that the proposed ME co-processor is limited to use a maximum current block size of 32x32 pixels. However, it can use any size of the search area of multiple sizes of 32x32 block pixels. As a future objective, more accuracy may be targeted by considering the motion vector (MV) cost in addition to the SAD. Area, power consumption, transmission data-rate and hardware cost should be considered to be able to connect the surveillance cameras utilising the IoT infrastructure.

REFERENCES
1. Chopra, K., Gupta, K., Lambora, A.: Future Internet: the internet of things—a literature review. In: 2019 International conference on machine learning, Big Data, cloud and parallel computing (COMITCon), Faridabad, pp. 135–139. (2019). https://doi.org/10.1109/COMITCon.2019.8862269
2. Nekrasov, H.A., Polivoda, D.E.: The development of a reference architectural prototype of the internet of things network mode. In: 2019 International conference quality management, transport and information security, information technologies (IT&QM&IS), pp. 554–557 (2019)
3. Shahroz, M., et al.: IoT-based smart shopping cart using radio frequency identification. IEEE Access. 8, 68426–68438 (2020)
4. Qiu, J., et al.: Towards secure and smart healthcare in smart cities using Blockchain. In: 2018 IEEE international smart cities conference (ISC2), pp. 1–4 (2018)
5. Sharma, A., Battula, R.B.: FOOTREST: safety on roads through intelligent transportation system. In: 2020 International conference on information networking (ICOIN), pp. 818–820 (2020)
6. Ojajuni, O., Ismail, Y., Lawson, A.: Distributed Denial-of-Service attack detection and mitigation for the Internet of Things. Intl. J. Technol. Diffusion. 11(15). 02/28 2020
7. Hakak, S., et al.: Securing smart cities through Blockchain technology: architecture, requirements, and challenges. IEEE Netw. 34(1), 8–14 (2020)
8. Ismail, Y., Elmedamy, W., Al-Junaid, H.: High performance architecture for real-time HDTV broadcasting. J. Real-Time Image Process. 11, 05–27 (2014)
9. Goel, S., et al.: An efficient data reuse motion estimation engine. In: 2006 IEEE workshop on signal processing systems design and implementation, pp. 383–386 (2006)
10. Tao, H., Lu, X.: Automatic smoky vehicle detection from traffic surveillance video based on vehicle rear detection and multi-feature fusion. IET Intell. Transp. Syst. 13(2), 252–259 (2019)

11. Chen, D., et al.: Platform choices and design demands for IoT platforms: cost, power, and performance tradeoffs. IET Cyber-Phys. Syst. Theory Appl. 1(1), 70–77 (2016)

12. Kerfa, D., Saidane, A.: An efficient algorithm for fast block matching motion estimation using an adaptive threshold scheme. Multimed. Tools Appl. 79, 24173–24184 (2020). https://doi.org/10.1007/s11042-020-09040-z

13. Kerfa, D., Saidane, A.: An efficient algorithm for fast block matching motion estimation using an adaptive threshold scheme. vol. 10996, pp. 109960C. International Society for Optics and Photonics (2019) https://doi.org/10.1117/12.2505732

14. Jiang, X., Song, T., Katayama, T.: Maximum-entropy-model-enabled complexity reduction algorithm in modern video coding standards. Symmetry. 12, 113 (01/07 2020)

15. Jiang, X., et al.: Low-complexity and hardware-friendly H.265/HEVC encoder for vehicular ad-hoc networks. (in eng), Sensors (Basel, Switzerland). 19(8). 1927 (2019)

16. Duong, D., Cong, H., Hoangyan, X.: A novel consistent quality driven for JEM based distributed video coding. Algorithms. 12(130) 06/28. (2019)

17. Biswas, K.J.S.: A robust and high capacity data hiding method for H.265/HEVC compressed videos with block roughness. Measure Error Correct. Technique. 11, 1360 (2019)

18. Zhu, C., et al.: High-quality hardware integer motion estimation for HEVC/H.265 encoder. IEICE Trans. Electron. (08/13 2019)

19. Ding, D., et al.: Real-time H.265/HEVC intra encoding with a configurable architecture on FPGA platform. Chinese J. Electron. 28(5), 1008–1017 (2019)

20. Mohan, A., et al.: Internet of video things in 2030: a world with many cameras. In: 2017 IEEE international symposium on circuits and systems (ISCAS), pp. 1–4 (2017)

21. Chen, C.W: Internet of video things: next generation IoT with visual sensors. IEEE Internet Thing. J., 1–1 (2020)

22. Sultana, T., Wahid, K.A.: Choice of application layer protocols for next generation video surveillance using internet of video things. IEEE Access. 7, 41607–41624 (2019)

23. Rani, S., et al.: IoMT: a reliable cross layer protocol for internet of multimedia things. IEEE Internet Thing. J. 4(3), 832–839 (2017)

24. Aslam, A., Curry, E.: Towards a generalized approach for deep neural network based event processing for the internet of multimedia things. IEEE Access. 6, 25573–25587 (2018)

25. Ismail, Y., Hammad, M., El-Medany, W.: Homeland security video surveillance system for smart cities. In: 2018 International conference on innovation and intelligence for informatics, computing, and technologies (SICT), pp. 1–4 (2018)

26. Grellert, M., et al.: An adaptive workload management scheme for HEVC encoding. In: 2013 IEEE international conference on image processing, pp. 1850–1854 (2013)

27. Paramkusam, A.V., Reddy, V.S.K.: A novel block-matching motion estimation algorithm based on multilayer concept. In: 2014 IEEE international conference on multimedia and expo (ICME), pp. 1–6 (2014)

28. Mallikarachchi, T., et al.: Decoding-complexity-aware HEVC encoding using a complexity–rate–distortion model. IEEE Trans. Consumer Electron. 64(1), 35–43 (2018)

29. ITU-T, J.: High efficiency video coding (HEVC) Text Specification Draft 6. Joint Video Team of ISO/IEC MPEG and ITU-T VCEG (2012)

30. Ismail, Y., et al.: Fast motion estimation system using dynamic models for H.264/AVC video coding. IEEE Trans. Circuit. Syst. Video Technol. 22(1), 28–42 (2012)