We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,500
Open access books available

118,000
International authors and editors

130M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

High-Efficient Video Transmission for HDTV Broadcasting

Yasser Ali Ismail

Abstract

Before broadcasting a video signal, redundant data should be removed from the transmitted video signal. This redundancy operation can be performed using many video coding standards such as H.264/Advanced Video Coding (AVC) and H.265/High-Efficient Video Coding (HEVC) standards. Although both standards produce a great video resolution, too much data are considered to be still redundant. The most exhaustive process in video encoding process is the Motion Estimation (ME) process. The more the resolution of the transmitted video signal, the more the video data to be fetched from the main memory. This will increase the required memory access time for performing the Motion Estimation process. In This chapter, a smart ME coprocessor architecture, which greatly reduces the memory access time, is presented. Data reuse algorithm is used to minimize the memory access time. The discussed coprocessor effectively reuses the data of the search area to minimize the overall memory access time (I/O memory bandwidth) while fully using all resources and hardware. This would speed up the video broadcasting process. For a search range of 32 × 32 and block size of 16 × 16, the architecture can perform Motion Estimation for 30 fps of HDTV video and easily outperforms many fast full-search architectures.

Keywords: HDTV broadcasting, motion estimation, H.264/AVC, H.265/HEVC, video coding, video transmission

1. Introduction

Broadcasting is the distribution of audio or video content to a dispersed audience via any electronic mass communication medium but typically is the one using the electromagnetic spectrum (radio waves). Recently, broadcasting operations are not used only by television signal, but also it includes many smart devices such as cell phones, video phones, and video conferencing. Two main problems were highlighted because of the high demand of video applications that need broadcasting [1–3]. The first problem is the huge bandwidth needed for transmitting such huge video data. The second problem is the delay in the video transmission process due to the huge computations required for video coding and transmission process [4, 5]. Many video coding standards tried to solve such problems. H.264/Advanced Video Coding (AVC) and H.265/High-Efficient Video Coding (HEVC) are the most recent video coding standards that tried to tackle the aforementioned problems [6, 7]. Although the two video coding standards provide a great video resolution and a low bit rate (BR), the computational complexity required for the video encoding process

is very high [1, 3, 8]. Consequently, the video transmission speed will slow down. As a result, using H.264/AVC and/or H.265/HEVC standards may not be suitable for real-time video broadcasting applications.

H.264/AVC [9] is a video coding standard that was developed by ITU and ISO [10]. The main advantage of H.264/AVC standard is to minimize the bit rate of the transmitted video signal. It achieves up to 50% savings in the transmitted video bit rate if compared to other previous standards (please see **Figure 1**). As a result, H.264/AVC standard is used in many video applications such as HD-DVD, multimedia streaming, remote video surveillance, medical image processing field, video conferencing, HDTV broadcasting, video on demand, and multimedia messaging [5, 6, 11]. Multiple reference frames, half-pel and quarter-pel accurate Motion Estimation, using small block size, exact-match transform, adaptive in-loop deblocking filter, and enhanced entropy coding methods, and variable block size techniques are used to achieve low bit rate and high resolution of the transmitted video signal [12].

Due to the high demand of high-resolution video applications and the traffic constrains of the network infrastructure, the offered transmission bit rate of H.264/AVC standard is not suitable to fulfill such application needs. ITU-T VCEG and ISO/IEC MPEG [13] developed a new video coding standard called H.265/High-Efficient Video Coding (HEVC) standard. H.265/HEVC standard was developed for three main goals. The first goal is to be able to encode high-resolution video sequences such as 4 K and ultrahigh definition (UHD). The second goal is to lower the video transmission bit rate by approximately 50% compared to the H.264/AVC standard. The last goal is to speed up the video coding and transmission process by utilizing parallel processing operations. It is worth mentioning that H.265/HEVC encoder is four times more complex than older standards [1, 14]. However, from the previous aforementioned discussions, it is clear that both H.264/AVC and H.265/HEVC standards require processing too much data in order to obtain higher compression and accordingly low bit rate. This is why the implementation of both standards is not that easy because of the high demand of huge memory size that is being able to process such huge data. This is the main bottleneck in such video coding standards,
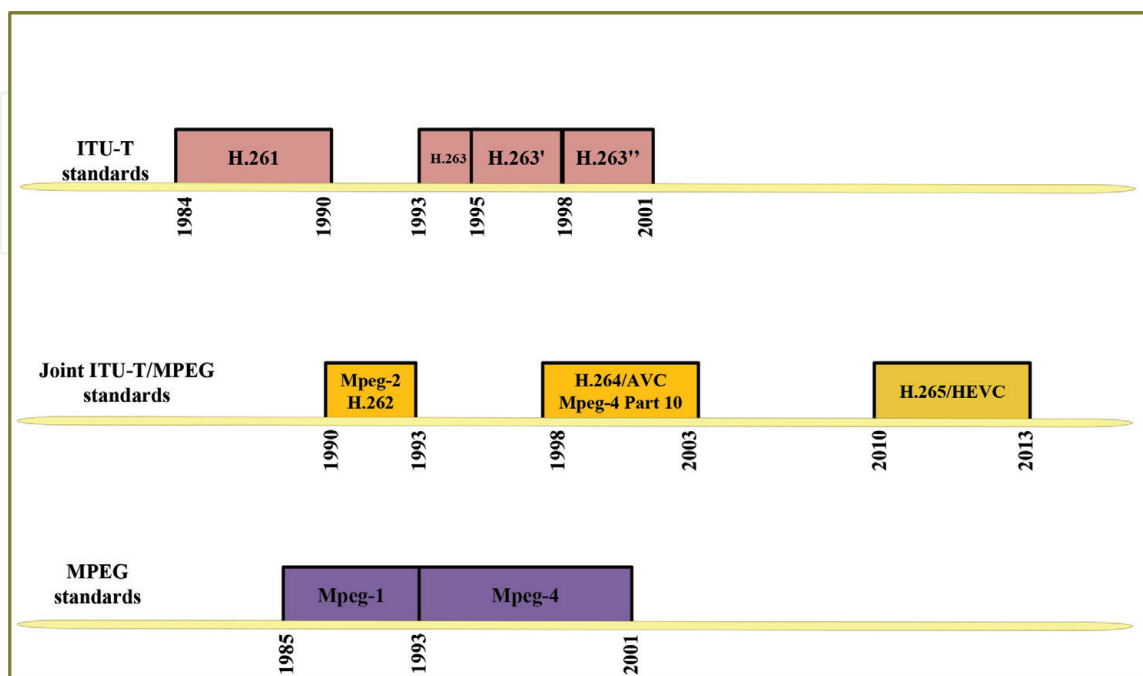

Figure 1.
Video coding standards.

since fetching such huge data from memory will increase the memory access time and slow down the encoding process. As a result, those video coding standards may not be suitable for real-time video applications such as HDTV broadcasting.

The huge data and memory access time that are needed for HDTV broadcasting, if using either H.264/AVC or H.265/HEVC standards, are the recent problems under study for many researchers nowadays. Some techniques tried to reduce the search area size in the Motion Estimation process in order to reduce the memory access time. Adaptive search window size (ASWS) technique [15, 16] adaptively decides the size of the search area according to the motion activity of the Current Block. The authors in [15] used three window sizes 3 × 3, 7 × 7, and 15 × 15. The main drawback of such algorithm is the lake of video visual quality since the selected window sizes are not enough to cover fast motion activities of certain Current Blocks. In [2], this problem is avoided by adaptively adjusting the search window size according to some model equations that used some parameters. Those parameters changed according the motion activity of a Current Block and considering the motion activity of surrounding blocks. The accuracy to decide the search window size in [2] is very high, and, accordingly, the memory access time is less than the case of using the conventional full-search Motion Estimation (FSME) process. Although the previous techniques greatly reduce the memory access time, their very-large-scale integration (VLSI) implementation is not easy [17]. They are not suitable for hardware implementation as they lose the regularity of data flow, but they can greatly reduce the computational complexity. Some straightforward VLSI architectures are used to implement the FSME of the video encoder (either for H.264/AVC or H.265/HEVC standards). Such implementation has many advantages such as they are greatly reducing the memory access time. Additionally, the data flow of their architecture is uniform. This gives such architectures simplicity in their design. The architecture given in [4] is a good example for such architectures. The authors in [4] used a smart algorithm with a simple local memory to reuse data (data reuse level A and level B) of the search area. It means that data could be fetched only once from the main memory. This greatly decreases the memory access time required for the Motion Estimation process. The design proposed in [4] is a flexible one, at which it can be used either for H.264/AVC or H.265/HEVC standards with slightly modifications in the hardware.

The chapter is organized as follows. Section 2 illustrates the problem formulation in details. Section 3 describes the memory I/O bandwidth reduction techniques. Motion Estimation co-processor with data reuse is described in details in Section 4.

2. Problem formulation

The most exhaustive part in video encoding process is the Motion Estimation process [18]. As seen in **Figure 2**, Motion Estimation process consumes up to 53 and 70% of the entire encoding process in case of using one and four reference frames, respectively. This is due to the huge data and computations that are required to perform such operation. As a result, we emphasize on reducing both the computations and the data fetched from the main memory in order to speed up the video encoding process in order to be available for real-time HDTV broadcasting application. In Motion Estimation process, the current frame ψ should be divided into blocks called Current Blocks (please see **Figure 3**). The size of each block is M × M pixels. The main idea of the Motion Estimation process is to reduce the temporal redundancy in the transmitted video sequences. This can be done by searching for a best match candidate block of each Current Block within a search area in the reference frame
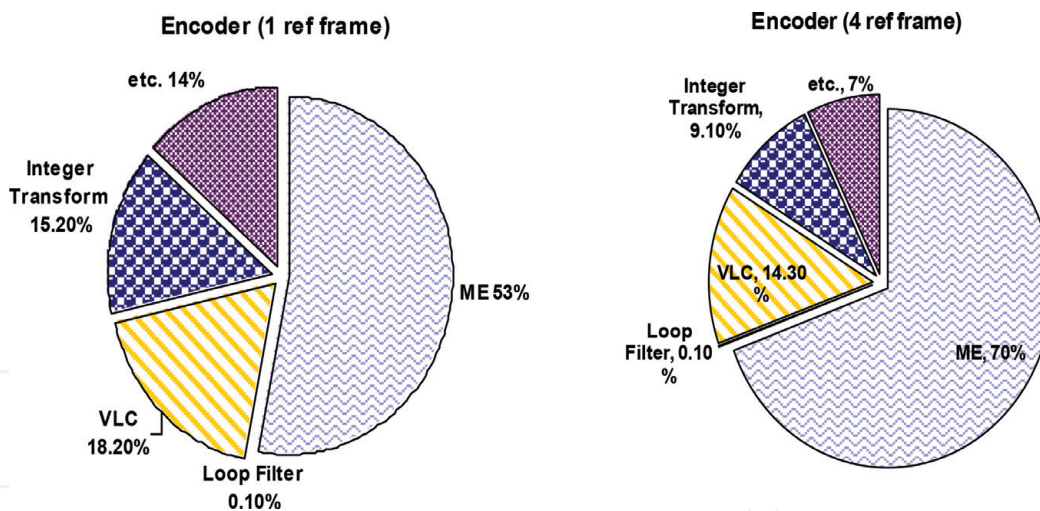
Figure 2.
Motion estimation computational complexity using one and four reference frames, respectively, and using H.264.AVC standard.

$\psi - 1$ as seen in **Figure 3**. Referring to **Figure 3**, the best match candidate block can be calculated by exhaustively searching all candidate blocks allocated in every pixel in the search area. The search area is allocated at frame $\psi - 1$. The search area size is $2X_{max} \times 2X_{max}$, where $2X_{max}$ is the range of the selected search area. The sum of absolute difference (SAD) metric is used to search for the best match candidate block. The pixel in the search area corresponding to the minimum SAD value (i.e., point (k,l) in **Figure 3**) represents the best match candidate block. Two outputs of the Motion Estimation process are seen in **Figure 3**. The first output is the residue between the Current Block and the best match candidate block. The second output is the displacement between the center of the search area and the best match candidate block, represented by the motion vector (MV). The MV can be represented as

$$MV(x, y) \;=\; \arg\min\{SAD(x, y; k, l)\} \tag{1}$$

where $-X_{max} \leq k, l \leq X_{max}$, x, and y are the center coordinates of the search area and k and l are the coordinates of the best match candidate block.
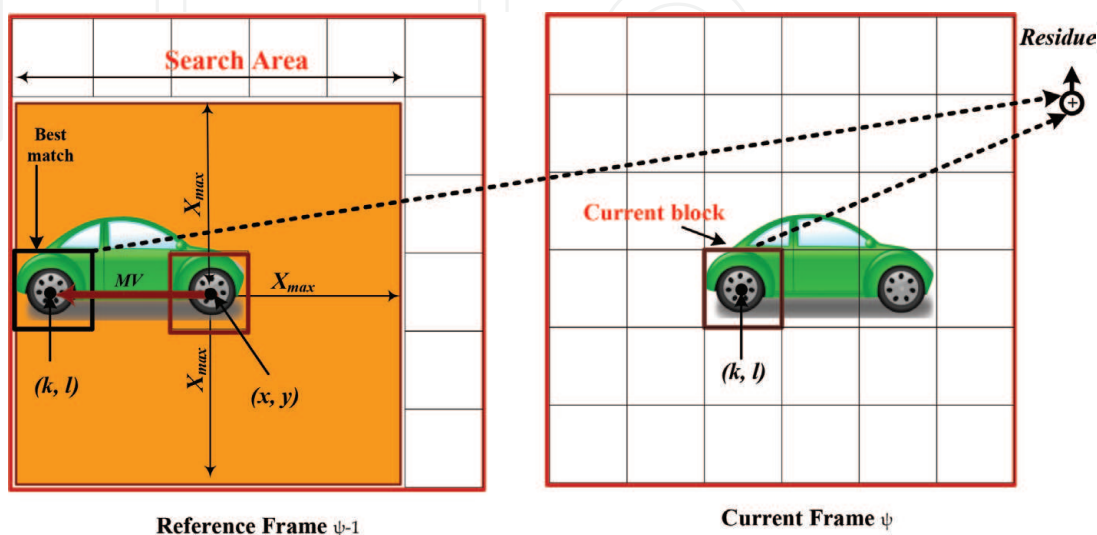

Figure 3.
The current and the reference frames in ME process.

	Frame size (pixels)	Frame rate (frames/sec)
HDTV broadcast	1920 × 1080	30
SDTV broadcast (D1)	720 × 486	30
Digital cinema (DC)	4096 × 2160	24 fps
Standard definition (SD)	720 × 486	30 fps
Video conferencing (SIF)	352 × 240	30
Internet streaming video (QSIF to SIF)	176 × 144 352 × 240	30
Desktop video phone (QSIF)	176 × 144	15–30

Table 1.
Video encoding formats [4].

It is worth mentioning that video applications consume much data compared to some other multimedia sources such as image, speech, and text. The number of bits required to transmit video sequences depends on the video frame resolution. The lager the frame resolution is, the larger the number of bits required to be transmitted. Consequently, the transmission bit rate will also increase. Different video frame formats are shown in **Table 1**. Given that the number of bits/pixels required for specific video resolution is B, the number of pixel per line of one video frame is P, the number of lines per video frame is L, and the transmitted frame per second for such video resolution is F, the video transmission bit rate (BR) can be calculated as in Eq. (2):

$$BR = B \times P \times L \times F \qquad (2)$$

It is noticed from **Table 1** that frame size is increased due to the increase of consumer demand for higher resolution [4]. The more the video frame size is, the more the required search area size will be. This increases the data to be fetched from the memory. Accordingly, more memory I/O bandwidth is required. This is a big problem since the memory I/O bandwidth is limited. As an example, HDTV broadcasting requires much data to be fetched from the memory than the Internet streaming video which uses QSIF or SIF video formats. Many recent researches are directed to create different techniques for a better use of the available memory I/O bandwidth. In the following section, a review of most recent techniques that have been used for reducing the I/O bandwidth problem will be performed.

3. Memory I/O bandwidth reduction

The memory I/O bandwidth is the main problem in video encoding process. The more the data is needed from the memory, the more the required memory I/O bandwidth is. As a result, the delay of the video encoding process will be high. This will prevent the use of such video encoding process in real-time applications such as HDTV and SDTV broadcasting. To avoid such problem, the search area data required for the Motion Estimation process in a video encoder should be reused. In other words, the required data for the Motion Estimation process should be fetched only once from the main memory. This will speed up the video encoding process and allow the video encoder to be used in real-time video applications.

During the full-search Motion Estimation, the current frame is divided into nonoverlapped blocks. Each block (Current Block) has a size of M × M pixels. The selected search area has a size of $SA_x \times SA_y$. The size of the Current Block and the search area is different according to the frame size. As an example, the Current Block and search area sizes for SIF video sequences are 16 × 16 and 32 × 32, while for SDTV and HDTV are 32 × 32 and 64 × 64, respectively. The search area can be divided into $\{SA_x - M + 1\} \times \{SA_y - M + 1\}$ candidate blocks or search pixels. It is worth mentioning that the candidate blocks for the contour pixels use additional padding (please see dashed line area in **Figure 4**). The padding pixels will have maximum values to be sure that they will be excluded from our selections.

Considering the candidate blocks #1 and #2 in **Figure 4**, there will be M − 1 overlapped pixels that should be fetched twice from the main memory. If we continue horizontally, there will be a complete strip that contains several overlapped areas between the candidate blocks of the search area. The horizontal strip of overlapped Data Pixels can be reused and fetched only once from the main memory. In this case, such data is called data reuse level A. If the Motion Estimation process is proceeded by going one step down, huge Data Pixels will be overlapped. This is called data reuse level B, if fetched only once from the memory. It is worth mentioning that a smart architecture of the Motion Estimation processor is required in order to be able to reuse data and reduce the I/O memory access bandwidth. Given that the size of the Current Block is M × M, the number of Data Pixels (DP) required from the memory to perform the Motion Estimation process for only one Current Block is given by Eq. (3). There are much redundant (repeated) data that are fetched from the main memory. This increases the I/O memory bandwidth which is not practical for real-time applications such as HDTV broadcasting:
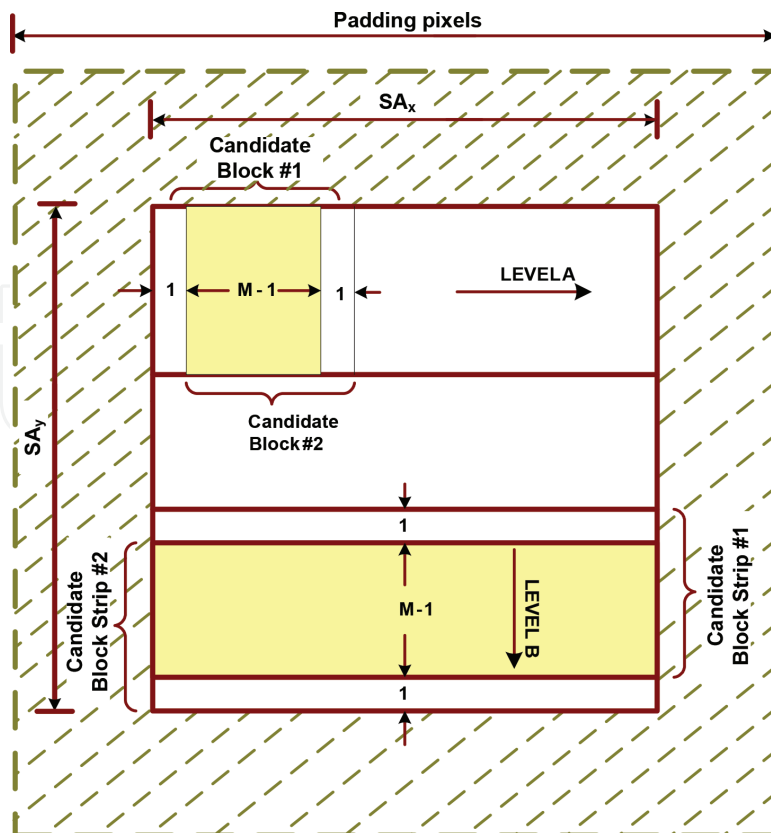

Figure 4.
Data reuse (levels A and B).

$$DP = \{SA_x - M + 1\} \times \{SA_y - M + 1\} \times (M \times M) \qquad (3)$$

Data reuse is the most recent efficient algorithm that is used to reduce the I/O memory bandwidth while performing Motion Estimation process [4, 19, 20]. Many redundant memory access times are skipped during the Motion Estimation process while performing the data reuse algorithm. The algorithm skips the calling of the same pixels multiple times and accesses the required Data Pixels only once while performing the Motion Estimation process. This is how it reduces accessing the main memory many times, and, consequently, it reduces the I/O memory bandwidth. For a maximum reduction of the I/O memory bandwidth, there are four levels of data reuse: levels A, B, C, and D [4, 19]. The four levels are described in the following subsections.

3.1 Data reuse level A

For a horizontal strip as seen in **Figure 4**, there will be an overlapped Data Pixels between any two consecutive blocks. The overlapped area size is $\{SA_y - M + 1\} \times \{M - 1\}$. Data reuse level A principle is achieved by loading only new data from the memory and ignoring old data that are already fetched before from the main memory [4, 19]. This requires a smart local memory in the Motion Estimation processor to keep reusing the old fetched data. As an example, seen in **Figure 4**, when performing the SAD operation between the Current Block and candidate block #2, only M column of Data Pixels will be fetched from the memory. This is a huge saving in the memory I/O bandwidth.

3.2 Data reuse level B

As seen in **Figure 4**, there is another vertical level of overlapped data if the SAD operation is performed one pixel in the down direction [4, 19]. The size of the overlapped Data Pixels is $SA_y \times [M - 1]$. The overlapped Data Pixels will be while performing the SAD operation between the Current Block and both of candidate blocks strip #1 and #2 as seen in **Figure 4**. Data reuse level B is achieved by loading only SA_x Data Pixels when moving from one horizontal strip to the next strip below it.

3.3 Data reuse level C

For two horizontal consecutive search areas $SA1_x \times SA_y$ and $SA2_x \times SA_y$, there will be an overlapped area of a size $\{|SA1_x - SA2_x|\} \times SA_y$ as seen in **Figure 5**. These overlapped Data Pixels could be fetched only once if level C data reuse is applied while performing Motion Estimation process [4, 19].

3.4 Data reuse level D

For two vertical consecutive search areas $SA_x \times SA1_y$ and $SA_x \times SA2y$, there will be an overlapped area of a size $SA_x \times \{|SA1_y - SA2_y|\}$ as seen in **Figure 6**. These overlapped Data Pixels could be fetched only once if level D data reuse is applied while performing Motion Estimation process [4]. Performing data reuse levels C and D require higher complexity in the design of the Motion Estimation process to reuse data already loaded from the former search area strips to the latter search area strips. This is a trade-off between the higher design complexity and the reductions in the memory I/O bandwidth.
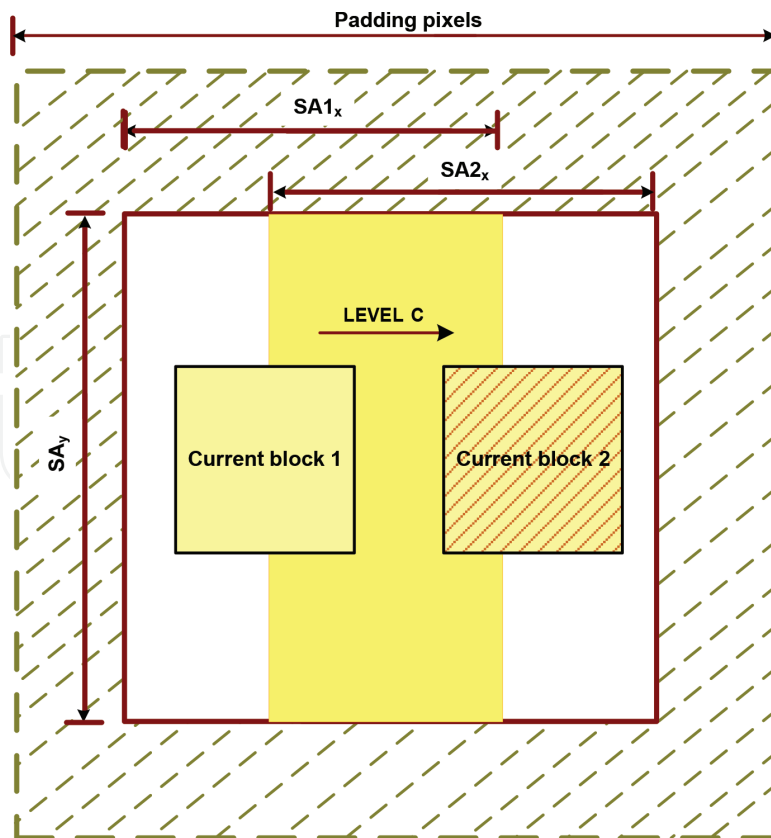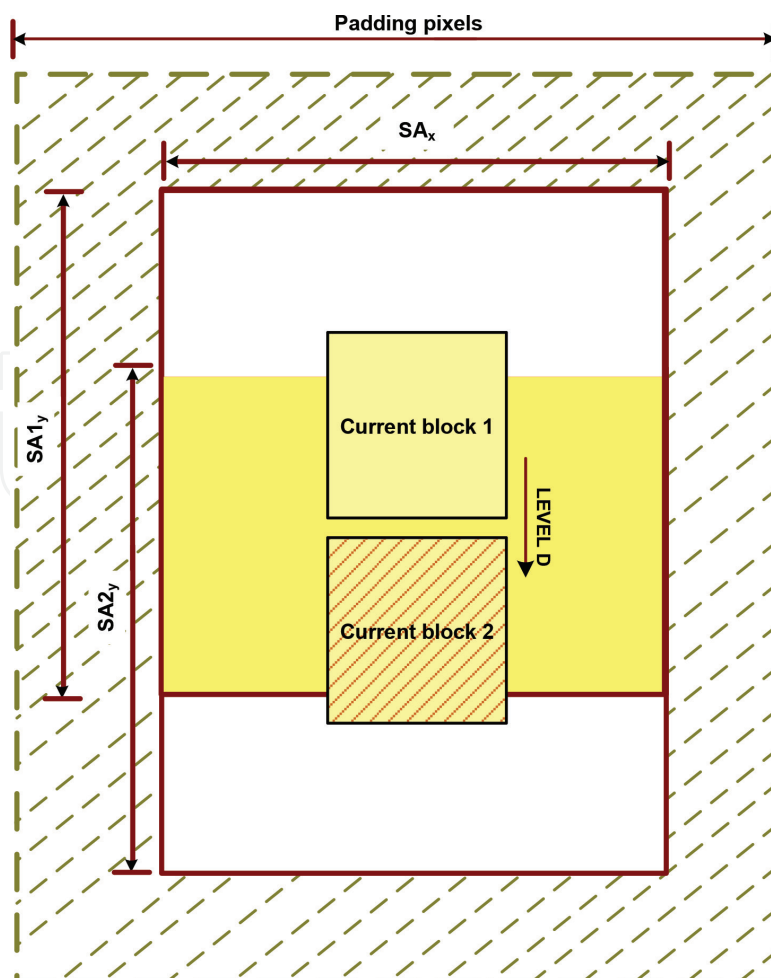
Figure 5.
Level C data reuse.

Figure 6.
Level D data reuse.

4. Motion estimation co-processor with data reuse

The main idea that is discussed in this section is how to design a smart Motion Estimation co-processor that could perform data reuse for a maximum reduction in the memory I/O bandwidth. So, such co-processor can be used for real-time video applications such as HDTV broadcasting. The design of the co-processor is preferred to be simple. This requires implementing only level A and level B data reuse since the other two levels require too much hardware complexity. One smart design that is modified and discussed in this chapter is the one proposed in [4]. The modified Motion Estimation co-processor can be used for both H.264/Advanced Video Coding (AVC) and H.265/High-Efficient Video Coding (HEVC) standards. Although the used window size for such co-processor is 32 × 32 pixels with a Current Block size of 16 × 16 pixels, it can be generalized for a higher number of pixels in both the search area and the Current Block to be suitable for HD video sequences. A modified top-level architecture of the proposed Motion Estimation co-processor in [4] is shown in **Figure 7**.

This co-processor is designed for H.264/AVC; however, it could be used for H.265/HEVC standard with a little modification in the hardware. The structure in [4] is designed to perform the Motion Estimation process for a Current Block of size 16 × 16. The search area can be 32 × 32 or more. To increase the speed of the co-processor proposed in [4], the Current Block is directly loaded from the main memory, and the search area will be loaded into a local memory as will be discussed later. The structure consists of a processing element (PE) array. The PE array consists of a 16 × 16 PE that computes the absolute difference (AD) values between the Current Block and a candidate block in the search area. The Current Block will be directly loaded into the PE array from the main memory, while the candidate block will be loaded into the PE array from the local memory. Data reuse levels A and level B are implemented using a local memory. This local memory is very simple and consists only of bank of
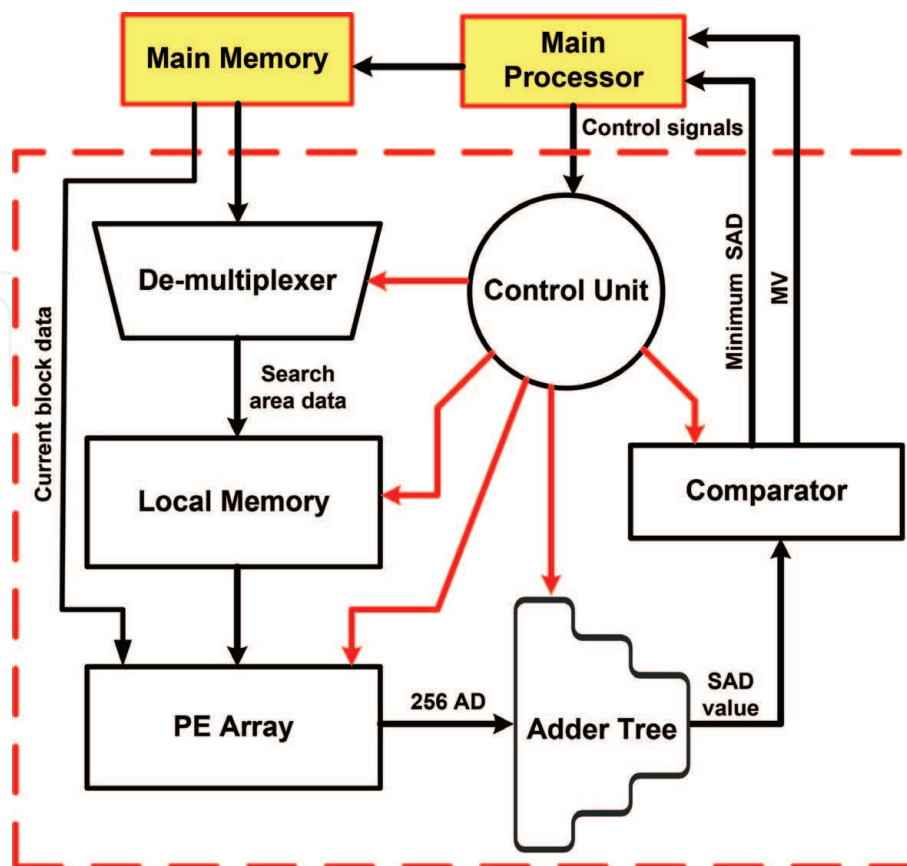

Figure 7.
Motion estimation co-processor.

registers. Using smart control signals, this local memory can succeed to greatly reduce the memory I/O bandwidth. Once the ADs are calculated using the PE array, an adder tree is used to add all of them and calculate a final SAD. The adder tree is adding all values in parallel fashion so that the addition process will be very fast. The SAD will be compared with the calculated SAD so far, and the minimum value will be selected. The candidate block in the search area with the minimum SAD so far will be selected to be the best match candidate block and the corresponding motion vector. The demultiplexer is used to smartly feed the candidate blocks into the PE array.

4.1 Design the local memory

Implementing data reuse levels A, B, C, and D can be performed using a local memory inside the Motion Estimation co-processor. This local memory could fetch the search areas only once from the main memory. This means less memory access time is required. A smart algorithm is needed to control the data flow in/out of the local memory. The local memory in **Figures 7** and **8** is designed to perform level A and level B data reuse. However, it can be used with slight modifications in its hardware and the operating algorithm to perform the other two data reuse levels (i.e., level C and level D data reuse). The modified architecture of the local memory [4] is shown in **Figure 8**. As mentioned before, this co-processor is mainly used for the Motion Estimation process of the H.264/AVC encoder.

The Current Block (CB) is loaded only once per one search area directly from the main memory into the PE array. The 16×16 Current Block is loaded into the PE array row by row as seen in **Figure 9**. The CB loading process starts from the bottom PE row and shifted in the up direction as seen in **Figure 9**. Each pixel of the CB is loaded into a register R_x of each PE via pin X_{in} as seen in **Figure 8**. While loading a new row of the CB into the PE array, the content of the register R_X in each PE will be shifted up via the pin X_{out}. This process will be continued until all the CB is loaded into the PE array. Loading the CB into the PE array consumes 16 clock cycles. The local memory is used to load the PE array with the search area of size $SA_x \times SA_y$ (including the padding area). The local memory is very simple in its design. It consists of two banks of registers. Each bank is 16×16 byte registers. No addressing modes are required for such local memory. Only a simple 5-bit counter is used for addressing. There is an additional (1×2) demultiplexer (Demux) used to control loading the PE array by the search area.

The Motion Estimation process starts by loading the Current Block inside the PE array as mentioned before. The local memory starts loading the search area by loading the top left 16×16 candidate block into Register Bank #1 of the local memory as seen in **Figure 8**. This will be performed via connecting terminal 1 of the Demux to the main memory. Loading operation will be performed by loading the candidate block of the search area line by line into Register Bank #1. This will take additional 16 clock cycles.

In clock cycle 33, two operations will be performed at the same time. First, since the first top-left candidate block is available on Register Bank #1 of the local memory, the Motion Estimation co-processor will start loading this candidate block into the PE array column by column starting by the left column of the Register Bank #1 of the local memory. A simple 5-bit counter will be used to read such columns. The pixels of the candidate block will be loaded into the PE via pin Y_{in} and stored into register R_Y (please see **Figure 8**). It is worth mentioning that every time a column of a candidate block is loaded into the PE array, it will be shifted to the left direction via pin Y_{out} (as seen in **Figures 8** and **9**) until the PE array is filled with the candidate block. Second, the second group of the search area will be loaded into the Register Bank #2 of the local memory. This will be performed by connecting terminal 2 of the Demux to the main memory.
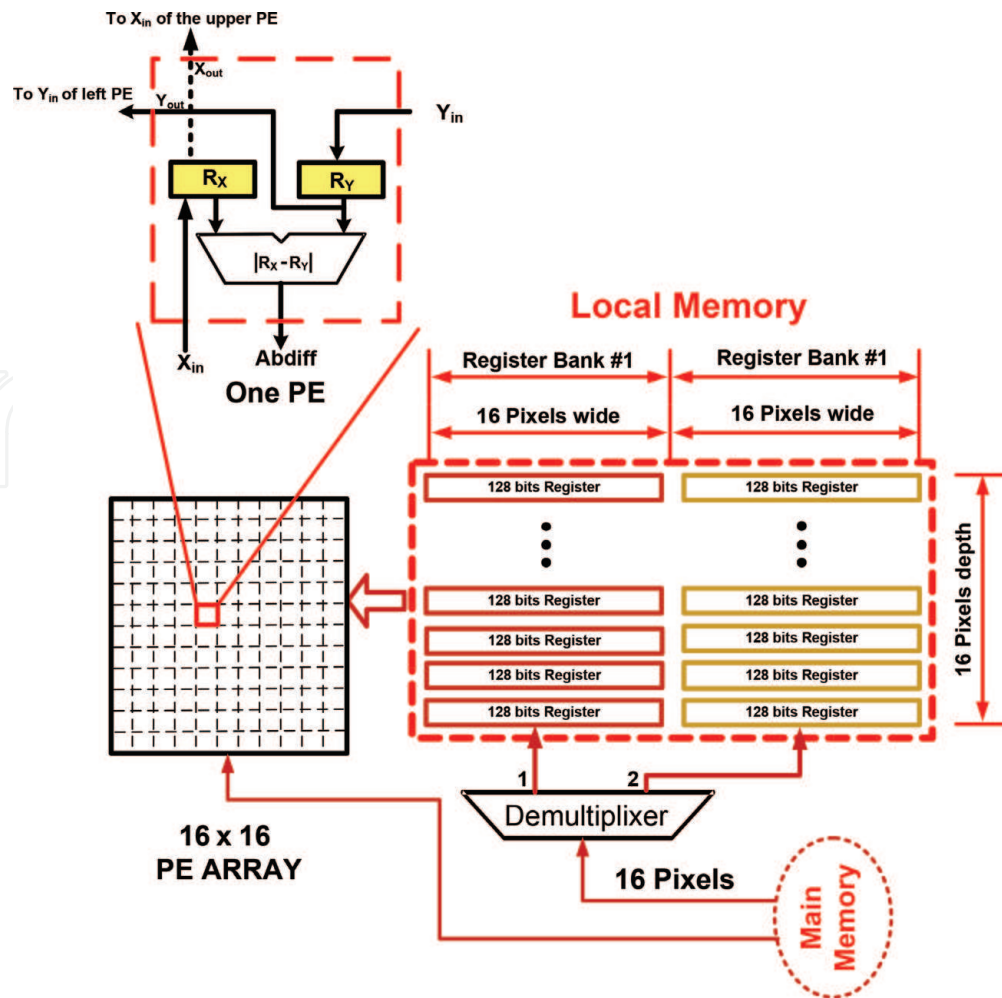
Figure 8.
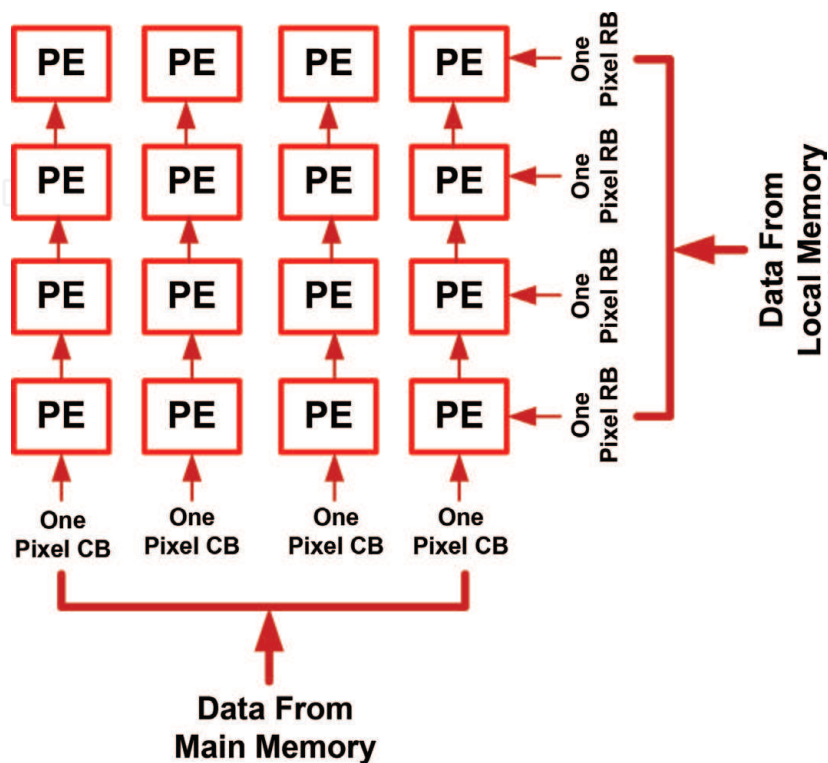*Local memory design for the motion estimation co-processor.*

Figure 9.
Loading the current block (CB) and the candidate block (RB) into a 4 × 4 PE array.

Once the candidate block is loaded into the PE array, the PE array will start calculating the AD between each pixel in the Current Block and the corresponding pixel in the candidate block using the adder of each PE. It is worth mentioning that level A data reuse will be performed by shifting only one column from the Register Bank #2 of the local memory into the PE array. While loading the first strip candidate blocks inside the PE array using the Register Bank #2 of the local memory, level B data reuse could be performed by loading new horizontal line into the left bank registers of local memory (Register Bank #1) via terminal 1 of the demultiplexer. Once the last candidate block of the first strip is loaded into the PE array, the counter will back to load a new candidate block (using level B data reuse) by loading the first left column of the Register Bank #1 of the local memory into the PE array. This operation will be continued until all candidate blocks are loaded into the PE array.

Every time a new candidate block is loaded into the PE array, 256 ADs will be calculated by the PEs. The adder tree will calculate the SAD value of the 256 ADs. The comparator will compare the current SAD value with the minimum SAD so far. The final minimum SAD will be decided after all candidate blocks are checked. All of these operations are performed at no stall at all and 100% utilization of the PE array. This is the main advantage of the architecture discussed in this section; this is why such architecture is suggested for real-time high-speed HDTV broadcasting applications. The final output of the co-processor will be the best candidate block with the minimum SAD value and the corresponding MV. These two values will be sent to the main processor to calculate the residue as seen in **Figures 3** and 7.

5. Conclusion

This book chapter provides a solution for the huge memory access time required by the high-resolution video broadcasting operation (HDTV broadcasting). A high-speed ME architecture that greatly minimizes the memory access time, if adopted into high-resolution H.264/AVC and H.265/HEVC video standards, has been presented. The proposed architecture can perform ME process for 30 fps of HDTV video. It can easily outperforms over many fast full-search architectures due to the great reduction in the memory access time required for the ME process. The proposed architecture effectively uses pipelining, parallelism, and data reuse to achieve a high-throughput rate while maintaining 100% PE utilization. The proposed co-processor is suitable for high-performance and high-accuracy real-time video applications such as HDTV and SDTV broadcastings. As a future work, the author of the book chapter is welling to reduce the processing time required for the ME process. This can be achieved by adopting some dynamic models inside this co-processor to adaptively select the proper size of the search area. The size of the search area is decided using these model equations according to the motion activity of the Current Block. Accordingly, the computations and the memory access time required for the ME process are expected to be greatly reduced.

Acknowledgements

Author details

Yasser Ali Ismail
Electrical Engineering Department, Southern University and A&M College,
Baton Rouge, USA

*Address all correspondence to: yasser_ismail@subr.edu

IntechOpen

References

[1] Jamali M, Coulombe S. Fast HEVC intra mode decision based on RDO cost prediction. IEEE Transactions on Broadcasting. 2018:1-14

[2] Ismail Y, McNeely JB, Shaaban M, Mahmoud H, Bayoumi MA. Fast motion estimation system using dynamic models for H.264/AVC video coding. IEEE Transactions on Circuits and Systems for Video Technology. 2012;**22**:28-42

[3] Sotelo R, Joskowicz J, Anedda M, Murroni M, Giusto DD. Subjective video quality assessments for 4K UHDTV. In: 2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB); 2017. pp. 1-6

[4] Ismail Y, El-Medany W, Al-Junaid H, Abdelgawad A. High performance architecture for real-time HDTV broadcasting. Journal of Real-Time Image Processing. 2014;**11**(4):633-644

[5] Vayalil NC, Kong Y. VLSI architecture of full-search variable-block-size motion estimation for HEVC video encoding. IET Circuits, Devices and Systems. 2017;**11**:543-548

[6] Rao KR. High efficiency video coding. In: 2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA); 2016. pp. 11-11

[7] Hannuksela MM, Yan Y, Huang X, Li H. Overview of the multiview high efficiency video coding (MV-HEVC) standard. In: 2015 IEEE International Conference on Image Processing (ICIP); 2015. pp. 2154-2158

[8] Ismail Y, Elgamel MA, Bayoumi MA. Fast variable padding motion estimation using smart zero motion prejudgment technique for pixel and frequency domains. IEEE Transactions on Circuits and Systems for Video Technology. 2009;**19**:609-626

[9] Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A. Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology. 2003;**13**:560-576

[10] Advanced video coding for generic audiovisual services. International Telecommunications Union, Telecommun. (ITU-T) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC 1, Recommendation H.264 and ISO/IEC 14 496-10 (MPEG-4) AVC; 2003

[11] Ohm J, Sullivan GJ, Schwarz H, Thiow Keng T, Wiegand T. Comparison of the coding efficiency of video coding standards 2014 including high efficiency video coding (HEVC). IEEE Transactions on Circuits and Systems for Video Technology. 2012;**22**:1669-1684

[12] Lee TK, Chan YL, Siu WC. Adaptive search range by depth variant decaying weights for HEVC inter texture coding. In: 2017 IEEE International Conference on Multimedia and Expo (ICME); 2017. pp. 1249-1254

[13] ITU-T and JVT. High Efficiency Video Coding (HEVC) Text Specification Draft 6. Joint Video Team of ISO/IEC MPEG and ITU-T VCEG; 2012

[14] Kuang W, Tsang SH, Chan YL, Siu WC. Fast mode decision algorithm for HEVC screen content intra coding. In: 2017 IEEE International Conference on Image Processing (ICIP); 2017. pp. 2473-2477

[15] Goel S, Ismail Y, Bayoumi MA. Adaptive search window size algorithm for fast motion estimation in H.264/AVC standard. In: 48th Midwest Symposium on Circuits and Systems; 2005; Vol. 2. 2005. pp. 1557-1560

[16] Kibeya H, Belghith F, Ayed MAB, Masmoudi N. Adaptive motion estimation search window size for HEVC standard. In: 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT); 2016. pp. 410-415

[17] Mukherjee R, Banerjee A, Chakrabarti I, Dutta PK, Ray AK. Efficient VLSI design of CAVLC decoder of H.264 for HD videos. In: 2017 7th International Symposium on Embedded Computing and System Design (ISED); 2017. pp. 1-4

[18] Celebi AT, Yavuz S, Celebi A, Urban O. One-dimensional filtering based two-bit transform and its efficient hardware architecture for fast motion estimation. IEEE Transactions on Consumer Electronics. 2017;**63**:377-385

[19] Tuan J-C, Chang T-S, Jen C-W. On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture. IEEE Transactions on Circuits and Systems for Video Technology. 2002;**12**:61-72

[20] Goel S, Ismail Y, Devulapalli P, McNeely J, Bayoumi MA. An efficient data reuse motion estimation engine. In: IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS '06); 2006; 2006. pp. 383-386