Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 473725, 10 pages
doi:10.1155/2012/473725

Research Article

A Memory Hierarchy Model Based on Data Reuse for Full-Search Motion Estimation on High-Definition Digital Videos

Alba Sandyra Bezerra Lopes,[1] Ivan Saraiva Silva,[2] and Luciano Volcan Agostini[3]

[1] *Federal Institute of Education, Science and Technology of Rio Grande do Norte, Campus João Câmara, 59550-000 João Câmara, RN, Brazil*
[2] *Department of Computer Science and Statistics, Campus Ministro Petronio Portela, Federal University of Piauí, 64049-550 Teresina, PI, Brazil*
[3] *Group of Architectures and Integrated Circuits-GACI, Federal University of Pelotas Pelotas, RS, Brazil*

Correspondence should be addressed to Alba Sandyra Bezerra Lopes, alba.lopes@ifrn.edu.br

Received 20 January 2012; Accepted 19 April 2012

Academic Editor: Alisson Brito

The motion estimation is the most complex module in a video encoder requiring a high processing throughput and high memory bandwidth, mainly when the focus is high-definition videos. The throughput problem can be solved increasing the parallelism in the internal operations. The external memory bandwidth may be reduced using a memory hierarchy. This work presents a memory hierarchy model for a full-search motion estimation core. The proposed memory hierarchy model is based on a data reuse scheme considering the full search algorithm features. The proposed memory hierarchy expressively reduces the external memory bandwidth required for the motion estimation process, and it provides a very high data throughput for the ME core. This throughput is necessary to achieve real time when processing high-definition videos. When considering the worst bandwidth scenario, this memory hierarchy is able to reduce the external memory bandwidth in 578 times. A case study for the proposed hierarchy, using 32×32 search window and 8×8 block size, was implemented and prototyped on a Virtex 4 FPGA. The results show that it is possible to reach 38 frames per second when processing full HD frames (1920×1080 pixels) using nearly 299 Mbytes per second of external memory bandwidth.

1. Introduction

Nowadays, several electronic devices support high-definition digital videos. Applications like internet and digital television broadcasting are also massively supporting this kind of media. In this scenario, the video coding becomes an essential area to make possible the storage and principally the transmission of these videos, mainly when the focus is in high definition.

The most recent and advanced video coding standard is the H.264/AVC (advanced video coding) [1]. This standard includes high complexity on its modules, aiming to achieve high compression rates. This high complexity makes difficult to achieve real time (e.g. 30 frames per second) though software implementations, especially when high definition videos, like 1920×1080 pixels, are considered.

A digital video is a sequence of still images, called frames, typically sampled at a rate of 30 frames per second. In a video sequence, there is a considerable amount of redundant elements, like background scenes or objects that do not have any motion from a frame to another, that are not really essential for the construction of new images. These elements are usually called redundant information [2]. There are three types of redundancy: spatial redundancy (similarity in homogeneous texture areas), temporal redundancy (similarity between sequential frames) and entropic redundancy (redundancy in the bit stream representation). Those redundancies can be removed in order to achieve high compression rates.

The motion estimation (ME) is the most computationally intensive module of a video encoder. This module explores the temporal redundancy to reduce the amount of data needed to represent the video sequence.

A feature of the H.264/AVC and other video coding standards is the use of asymmetric compression algorithms. In this case, the encoder and the decoder have different

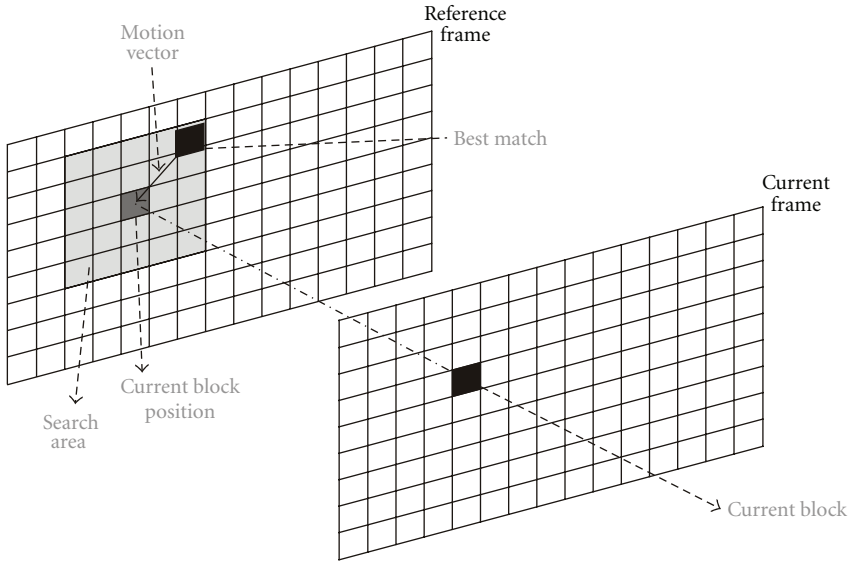

Figure 1: Motion estimation process.

definitions and the decoder, which is used in the higher number of devices, is less complex and cheaper than the encoder. The H.264/AVC standard introduces several new features when compared with others video compression standards. But the implementation of many of these new features is not mandatory [1]. Although the motion estimation is the most complex module in a digital video codec, it is presented only in the encoder side.

The motion estimation requires, besides the high processing throughput, also a very high bandwidth of external memory to realize its operations in real time when considering full HD videos (frames with 1920×1080 pixels) [3]. The throughput can be solved increasing the parallelism in the internal operations. The external memory bandwidth may be reduced using a memory hierarchy. Besides, the increase of the parallelism implies increasing the bandwidth. If more calculations must be performed at the same time, more data must be available to perform these operations.

The motion estimation presents the highest demand for external memory bandwidth, and it shares the external memory interconnection subsystem with others encoder modules like the motion compensation, the intraframe prediction and the deblocking filter. In this context, it is important to explore methods and architectural solutions which minimize the number of external memory accesses. An efficient memory hierarchy is the key point to respect the demands of a video encoder, mainly when high-definition videos are being processed.

This paper presents a memory hierarchy model for a full-search motion estimation core. The proposed memory hierarchy model is based on a data reuse scheme, and it aims to minimize the memory bandwidth and to maximize the data throughput delivered to the motion estimation core. The paper is structured as follows. Section 2 introduces the motion estimation process. Section 3 presents the data reuse exploration scheme. Section 4 presents the proposed memory hierarchy model. Section 5 presents some results and discussions. Section 6 presents the implemented core using the proposed memory hierarchy which was used as a case study. Section 7 presents comparisons with some related works. Finally, Section 8 presents conclusions and future works.

2. Motion Estimation

The ME process (illustrated on Figure 1) uses at least two frames: one current frame and one or more reference frames. The current frame is the frame that is being encoded. The reference frame is a previously encoded frame in the video sequence.

The current frame is divided into nonoverlapped blocks. For each block, a search window is defined in the reference frame. A search process is performed to find the better match for each block in its respective search window. To determinate the better match, a similarity criteria like Sum of absolute differences (SAD) [4] are used. A motion vector is generated for each block to represent where the better match was found. In the H.264/AVC, multiple reference frames can be used. In this case, the same search is done for all available reference frames and the candidate block that presents the better value of similarity among all reference frames is selected.

There are several search algorithms that define how the search process is done inside a search window. The full search (FS) is an algorithm that generates optimal results in accuracy, with a high cost in complexity [5]. Fast algorithms, like diamond search (DS), provide a great complexity reduction with acceptable lossless in accuracy [5].

Besides to provide the best possible motion vector, another important feature in FS algorithm is that it has a regular pattern with no data dependencies.

The FS algorithm looks for the best match in all possible candidate blocks in the search window, from the superior left

border until the inferior right border, shifting pixel by pixel in the search window. For each candidate block, a similarity calculation must be done to measure how similar is the candidate block to the current block. Once the similarity value is computed for all candidate blocks, the best match is the most similar candidate block (for instance, the lowest sum of absolute differences if SAD is used as similarity value). The FS process increases its complexity proportionally to the increase of search window range [6]. For a 32×32 search window and a 16×16 block size, there are 289 candidate blocks. A similarity calculation should be performed for each one of these blocks candidates. If a 64×64 is used, the number of candidate blocks increases to 1089.

Despite the complexity, in the FS motion estimation (FSME), there is no data dependency during the process of similarity calculation. So it is possible to design hardware architectures that provide the necessary resources in order to achieve high performance rates. It is possible to use systolic arrays [7] or tree structures to solve computational problems providing enough processing elements to compute the similarity between samples from the current block and the candidate block in parallel. However, the design of high performance motion estimation hardware is not an easy task. The parallelism exploitation increases the chip area proportionally to the parallelism level. The memory bandwidth also increases with the number of parallel similarity computation. Some criteria must be observed in order to achieve a good tradeoff between some parameters like. the degree of parallelism, the chip area, the memory bandwidth, and the power consumption.

On other hand, software solutions based on current general-purpose processors are not able to encode1080 HD digital video in real time (at a rate of 30 frames per second) when all H.264/AVC coding features are used. Then, the use of dedicated hardware architectures is mandatory to provide the required performance.

3. Data Reuse in Full-Search Motion Estimation

On FSME, data reuse refers to the use of each pixel of the search window (one or more n-bits words) to the similarity calculation of all candidate blocks that share the pixel without additional fetches in memory. A sample can be shared by neighboring candidate blocks of a same search window or by neighboring search windows in a reference frame. The number of candidate blocks that share a single sample depends on the size of the block, the size of the search window, and the position of the sample in the search window. For instance, a sample on the superior left corner of a search window is not shared by other candidates blocks, a single candidate block of a single search window have has sample, while a sample on the center of a search window is shared by many blocks of many search windows. Typically a sample can be shared by as many candidate blocks as the number of samples in a candidate blocks.

The regular pattern of FS algorithm allows the exploration of data reuse along with the exploration of parallelism in high degrees. The data reuse approach provides a way to reduce the bandwidth at the same time that the complexity of FS similarity calculation is reduced to the calculation of the similarity of a pair of block. In this context, data reuse is essential to reduce the bandwidth and the high parallelism is essential to achieve high performance.

As higher is the required video quality and definition, as higher is the required external memory bandwidth. The bandwidth increases with the frame size increase, with the search window increase, and with the frame rate increase. The bandwidth problem can be reduced using an efficient memory hierarchy.

As mentioned before, motion estimation uses at least two frames: one current frame and one reference frame.

The current frame is divided into several nonoverlapped $N \times N$ blocks. In the FSME, the search for the best match of one current block requires that its samples are repeatedly compared with samples from the reference frame. The current block samples lifetime is the time period of motion-estimating one current block [3]. Keeping the $N \times N$ samples from the current block in a local memory during the search reduces the data access from the current frame to the maximum possible, minimizing currant frame bandwidth. Doing this, each sample of the current frame will be fetched from the main memory just once in the whole motion estimation process.

Although adjacent current blocks do not overlap, adjacent search windows are no longer independent and overlap. Each search window in the reference frame is a rectangle ($SW_W * SW_H$) centered on the current block. This generally means that the search window of the current block shares data with the search window of the neighboring block.

In [3], four data reuse levels in the reference frame were defined according to the degree of data reuse: from level A to level D, where level A is the weakest reuse degree and the level D is the strongest reuse degree. Level A and Level B cover data reuse within a single search window. Level C and Level D cover data reuse among different search windows.

The data reuse level is an important factor in dealing with memory bandwidth requirements of ME. As stronger is the reuse level, less memory bandwidth is required.

In the search window, a row of candidate blocks is called a candidate block strip. Adjacent candidate blocks in a candidate block strip only differ by a column of samples. The Level A of data reuse take advantage of this feature to reuse all data loaded from external memory to the calculation of the previously candidate block and load just the column of samples which is necessary for the next candidate block calculation.

Figure 2 shows the level A data reuse scheme, where N represents the block dimension, SW_H and SW_W represents the height and the width of the search window, respectively. At this figure, two adjacent candidate blocks are showed (candidate block 1 and candidate block 2). The gray area is all samples within the search window that can be reused from a candidate block to another.

The level B of data reuse considers that adjacent vertical strips of candidate blocks within the search window also overlap significantly, only differing by a row of samples. All

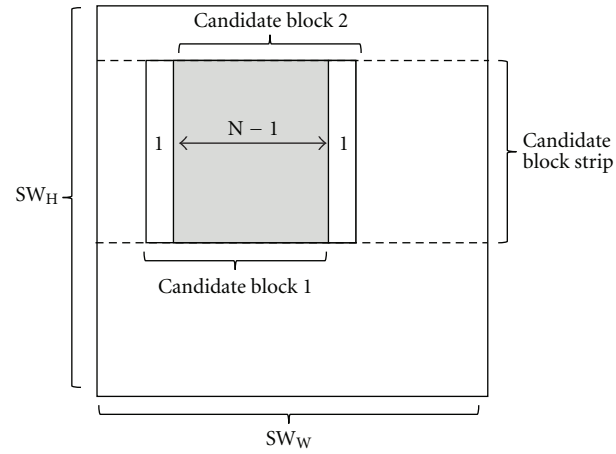

Figure 2: Level A data reuse scheme.

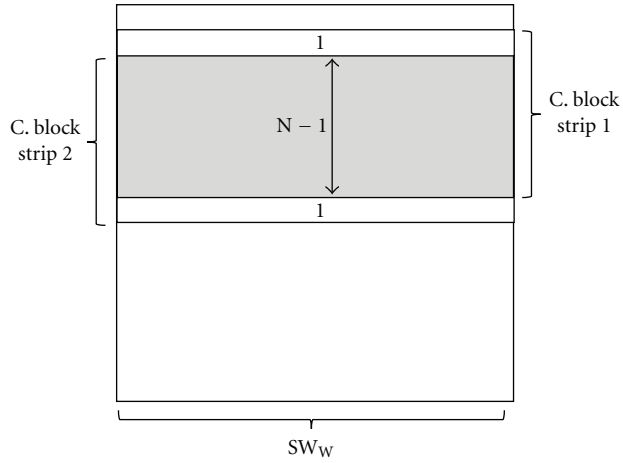

Figure 3: Level B data reuse scheme.

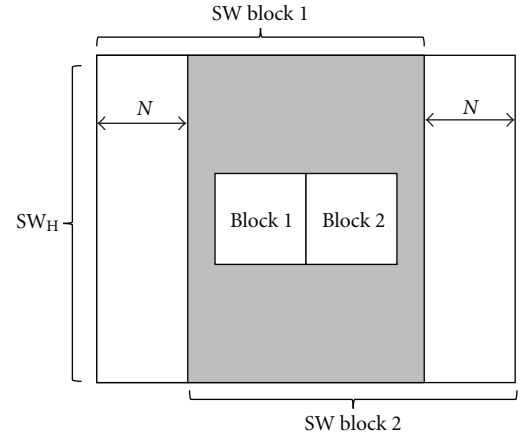

Figure 4: Level C on data reuse scheme.

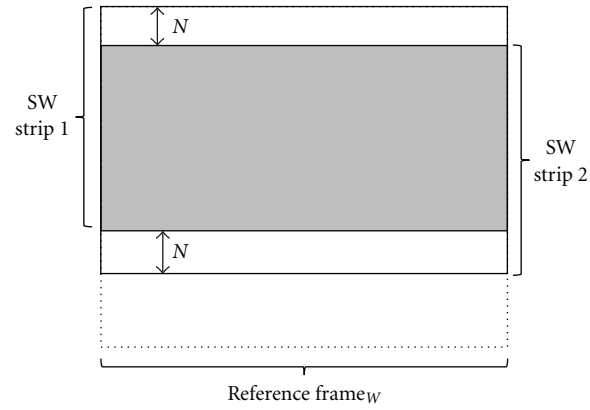

Figure 5: Level D data reuse scheme.

sample loaded for the previously strip of candidate blocks, with one row exception, can be reused.

Figure 3 represent two vertically adjacent candidate block strips (C. block Strip 1 and C. block strip 2). All the gray area represents data that can be reused while processing the next candidate block strip.

The level C refers to data reuse among different search windows. Search windows from neighboring blocks have several samples in common. in Figure 4, two blocks (block 1 and block 2) and their respective search windows (SW block 1 and SW block 2) are represented. The data of the two search windows differ by N columns of samples (where N represents the block dimension). All gray area is data that can be reused in the next block processing.

Finally, level D showed in Figure 5 resembles to level B of reuse data, except that it applies to reuses of samples in the entire search window strip instead of candidate blocks strip. With this level application, each sample of the reference frame is loaded just once during the entire motion estimation process.

The level A scheme uses the smallest size of on-chip memory, but it consumes more off-chip memory bandwidth. On the other hand, the level D uses more on-chip memory but it archives minimal off-chip memory bandwidth.

This work adopts Level C on data reuse scheme to balance on-chip memory size and off-chip memory bandwidth.

4. Proposed Memory Hierarchy

Generally, a memory hierarchy is composed of registers, local memories, and external memories. External memories are cheaper and have high storage capacity. But they are also slower. Local memories are faster than external memories, and their cost is also higher. Registers are the fastest option but with the highest cost among all solutions.

Considering these features, the proposition of a memory hierarchy for the motion estimation process must provide: (i) memory capacity to store at least the data that can be reused, considering the adopted reuse level (in this paper the Level C was chosen); (ii) a combination of high-level and low level memories in the hierarchy aiming to balance memory hierarchy cost, bandwidth, and throughput.

Data that can be reused in the ME in general was previously loaded from external memory. So it must be appropriately distributed between memory modules in the memory hierarchy to meet the goals described above.

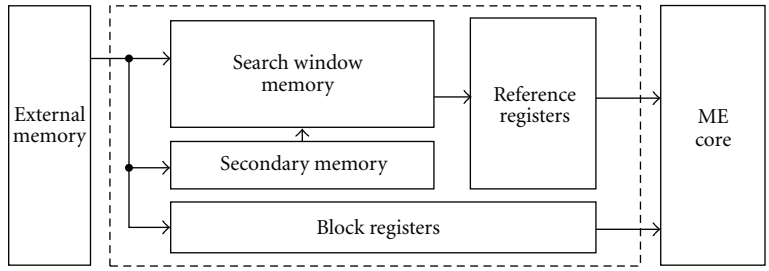

Figure 6: Proposed memory hierarchy model.

According to level C, data reuse scheme presented on [8] and illustrated in Figure 6 was proposed. This model aims to reduce the required memory bandwidth, while it balances cost and throughput. This way, the memory hierarchy will be able to provide the necessary data to the ME core that achieves the required high processing rates.

The external memory stores frames (current and reference frames) that will be used in the motion estimation process. The memory hierarchy levels store data that are recurrently used, avoiding redundant accesses to the external memory. These levels are composed by a search window memory, a secondary memory, and registers (block and reference registers). Each one will be described below.

In the proposed hierarchy, the block registers bank (BRB) contains $N \times N$ registers, each one storing a sample of the current block. The data from current block is used in the similarity calculation of all candidate blocks in the search window. It is important to use registers to store this block because it will be accessed continuously during all motion estimation process so the access to these data must be fast. The data at this bank is loaded just once per each block in the current frame and all values are used at each cycle during all ME process of the current block. Using this block register bank reduces to the minimum possible the external memory access of the current frame.

The reference register bank (RRB) has $N * (\mathrm{SW_W} - N)$ registers, where $\mathrm{SW_W}$ is the search window width. It is loaded with data from search window memory. These data refers to candidate blocks in the search window which are being used in the similarity calculation in relation to the current block. This bank in the way it was designed is sufficient to provide a high throughput to the ME core process. A fast access to these data is required to allow a high throughput in the ME core; thus, registers are also needed to store them.

At the reference register bank, three types of data shifts are allowed: shift right, shift left, and shift up. It was planned to minimize local memory access when the procedure of full search scan is performed as proposed in Figure 7.

In the proposed scheme, the data in RRB are shifted N times left until reaching the search window right border. One shift down is made when borders are reached and N shifts right are done to reach the left border of the search window. In each shift, only one column or row of new data must be stored in the RRB. Each shift is performed in one clock cycle. Every time a search window border (left or right) is reached, a complete data of candidate blocks strip has been provided to the ME core.

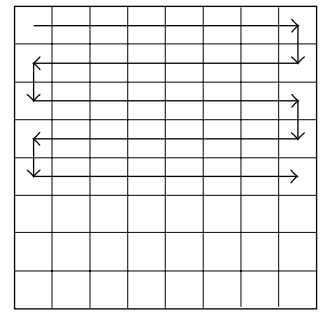

Figure 7: Proposed full-search scanning order.

The search window memory stores the complete search window of the current block. Sets of data from this memory are accessed at each moment from the motion estimation process. Since these data do not need to be simultaneously accessed, it can be stored in a local memory (on-chip memory). So the search window memory has a storage limit of $\mathrm{SR_W} * \mathrm{SR_H}$ samples. This is the memory that provides data to the RRB. After the initial RRB fill, at each clock cycle, the search window memory provides to the RRB on row or column of data.

Together, the RRB and the search window memory cover data reuse within a single search window, which represents level A and level B on the data reuse scheme.

The secondary memory stores the difference of search window data from two adjacent current blocks. So the size of this memory is $N * \mathrm{SR_H}$. These data are loaded from external memory, while the full search scan is being done. Once immediate accesses to them are not required, they can be stored in the local memory. When it is time to process a new current block, no extra clock cycle is expended to request data from external memory. The search window memory keeps the data that can be reused from the search window of the previously current block, and the data at the secondary memory overwrites the data that are no longer needed in the search window memory. So the complete search window from the next current block is available in the memory hierarchy modules.

The search window memory and secondary memory cover the level C on the data reuse scheme that refers to data that can be reused among different search windows.

A complete search window of a current block just need to be loaded from external memory when the current block

is the first one in the strip of current blocks in the current frame.

5. Experimental Results

Table 1 presents the experimental results using the proposed memory hierarchy for three sizes of search window (32×32, 64×64, and 96×96) and two block sizes (8×8 and 16×16).

The first and second columns on the table present the search windows and block sizes, respectively, used in this evaluation. The third column presents the number of candidate blocks existent in the search window. The number of candidate blocks in a $k \times k$ search window can be calculated by the formula $(k - n + 1)^2$, where n represents the block dimension.

The fourth column presents the bandwidth required (in megabytes) to bring data from external memory to process 30 frames in 1080 HD using our memory hierarchy model. The fifth column shows the bandwidth per second (in gigabytes) provided, using the proposed model, to the ME Core. The presented numbers consider the way of our memory hierarchy reuse data, keeping data previously loaded into local memories and registers.

The sixth column presents the number of cycles needed to fill a complete local memory with data from a whole search window. The next one shows the size (in Kbytes) of on-chip memory needed for each pair "search window/block size." This number is the sum of search window memory size and the secondary memory size. The eighth column presents the size (in Kbytes) of used registers (block registers and reference registers).

Finally, the last column shows the PSNR (Peak Signal Noise Rate) reached using the specific parameters of search window and block size. The PSRN results were obtained by software evaluations, using the H.264/AVC reference software [9], testing 10 video sequences (blue sky, man in Car, pedestrian area, riverbed, rolling tomatoes, rush hour, station, sunflower, traffic, and tractor). The PSNR values presented are the average of the values of all 10 video sequences because the values of PSNR may vary according to the video characteristics, like the amount of movement present in each video sequence.

An important relation that must be observed is the increase in the number of candidate blocks. A smaller block size, for a same search window, implies in a higher number of candidate blocks. Considering the search window variation, the number of candidate blocks increases proportionally with the search window increase. These aspects affect directly the computation necessary to the ME, and, consequently, the properly amount of data must be provided to ME core.

Using the proposed memory hierarchy, the external memory bandwidth to the ME is reduced, as shown in Figure 8. While the number of candidate blocks increases exponentially, the bandwidth maintains a linear growth. Comparing the best and the worst case presented on Table 1, while the number of candidate blocks increases approximately 27 times, the bandwidth required is only 4.4 times bigger. Without the proposed hierarchy, both curves presented on Figure 6 would have the same exponential behavior.

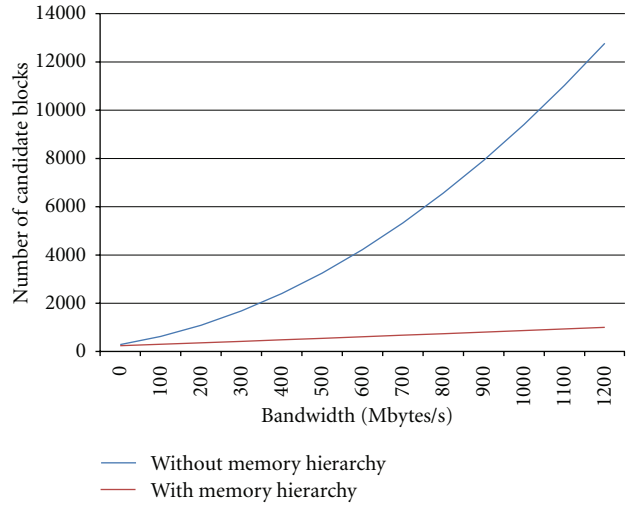

Figure 8: Candidate blocks versus bandwidth required.

According to a DDR SDRAM behavior, the number of clock cycles needed to fill the complete memory hierarchy with data from external memory was estimated.

A complete search window must be loaded from external memory only when the current block is the first one in a strip of current blocks. Considering a 1080 HD frame and a block size 8×8, there are 135 strips of current blocks. Using a 32×32 search window, 74 clock cycles are needed to fill the local memory. So only 9.990 clock cycles are expended with external memory accesses per frame. The fill of secondary memory can be done without introducing any extra clock cycles in the critical path, since this filling is done in parallel with the ME core process.

The on-chip memory presented in Table 1 is the sum of search window memory size and secondary memory size. The search window memory has a fixed size according to the size of search window. Once motion estimation uses only luminance samples (the motion vector to chrominance blocks are calculated based on the luminance ones), with a 32×32 search window and using 1 byte per luminance sample, the size of local memory is exactly 1 Kbyte. But the size of secondary memory depends on the block width and on the search window height. So using an 8×8 block size and the same 32×32 search window, the secondary memory uses 0.25 Kbytes of on-chip memory, totalizing 1.25 Kbytes. This way, the increase of search window memory is proportional to the square size of search window, when the secondary memory increase is proportional to both, the search window and the block size. When comparing the size of secondary memory with search window memory, the search window memory size increases 9 times between the best and worst case, and the secondary memory increases only 1.5 times.

The number of registers used is proportional to the block size and the search window. As bigger is the block size, as bigger is the block register size. Analogously, bigger search windows imply in bigger reference registers banks.

Table 1: Experimental results.

Search window	Block size	Candidate blocks	Bandwidth Ext. mem. (Mbytes/sec)	Bandwidth ME core (Gbytes/sec)	Clock cycles	On-chip memory (Kbytes)	Registers (Kbytes)	PSNR (dB)
32 × 32	16 × 16	289	179.0	15.8	74	1.50	0.5	34.00
	8 × 8	625	299.6	34.8		1.25	0.25	35.90
64 × 64	16 × 16	2,401	302.6	136.3	266	5.00	1	35.41
	8 × 8	3,249	547.8	184.9		4.50	0.5	37.64
96 × 96	16 × 16	6,561	430.1	375.4	586	10.50	1.5	35.90
	8 × 8	7,921	803.9	453.8		9.75	0.75	38.26

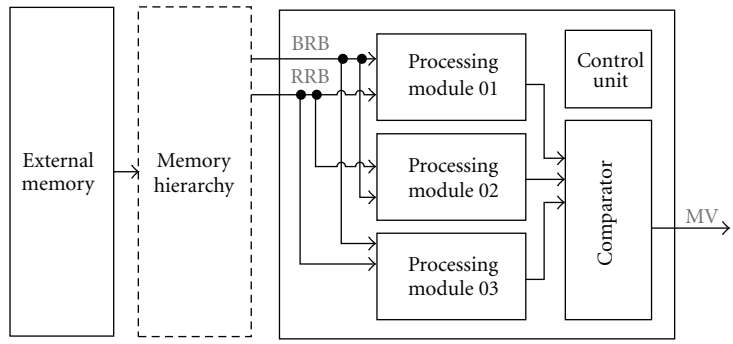

Figure 9: ME core block diagram.

So, according to Table 1, the configuration that spends more registers is that considering a 96 × 96 search window and a 16 × 16 block size.

One important factor that must also be considered when the parameters presented Table 1 in are evaluated is the quality of generated matches. The PSNR (peak signal noise rate) is one of the most used parameters to compare the video quality [6]. It is a quantitative measure that evaluates how much noise exists in the encoded video when compared to the original one. As biggest the PSNR value is, smaller is the noise and better is the video quality.

The PSNR results presented on Table 1 are the average of PSNR results from ten 1080 HD video sequences. As larger is the size of the search window, as higher is the ME chance to find a best match and a best motion vector. On the other hand, as smaller is block size, as higher is the ME chance to find a best match. Analyzing the PSNR results, the variation between worst and best case is more than 4 dB.

Furthermore, as bigger is the search window size and as smaller is the block size, as bigger is the number of computation needed to process the ME and also higher is the memory bandwidth required, as shown on Table 1.

Motion estimation architecture decisions always imply in a tradeoff between the numbers of needed calculations, the quality of ME results, the chip area, and the required external memory bandwidth. So it is important to balance all these criteria when designing a new architecture for motion estimation. But, in all cases, the use of the proposed memory hierarchy will decrease a lot the external memory bandwidth, allowing the ME to reach high throughputs, even for high-definition videos.

6. Case Study

The proposed memory hierarchy model and a functional motion estimation core were implemented and coupled as shown, in Figure 9 to demonstrate the qualities of the proposed memory hierarchy. The complete architecture, including the memory hierarchy and the ME core, was described in VHDL and synthesized to a Xilinx Virtex 4 XC4VLX25 FPGA [10].

The local memories (search window memory and secondary memory) were mapped to FPGA BRAMs.

A module to control the accesses to memory hierarchy was designed. It was designed to make read and write requests at the right moment: write data from external memory into the search window memory, secondary memory or block register; read data from search window memory, and writes in the reference register bank; also read data from secondary memory and writes into the search window memory.

These ME core was implemented using a fixed 8 × 8 block size and the 32 × 32 search window. This parameters were defined after analyzing some of the criteria presented in Table 1, and it is realized that these parameters presents good tradeoff between numbers of needed calculations, the quality of ME results, the chip area, and the required external memory bandwidth.

The ME core uses SAD [4] as similarity criterion. This is the most used criteria for hardware implementations because it executes simple calculations and it offers a good quality answer. It performs the absolute difference between samples from current block and samples from candidate block. The ME core was implemented intending to achieve

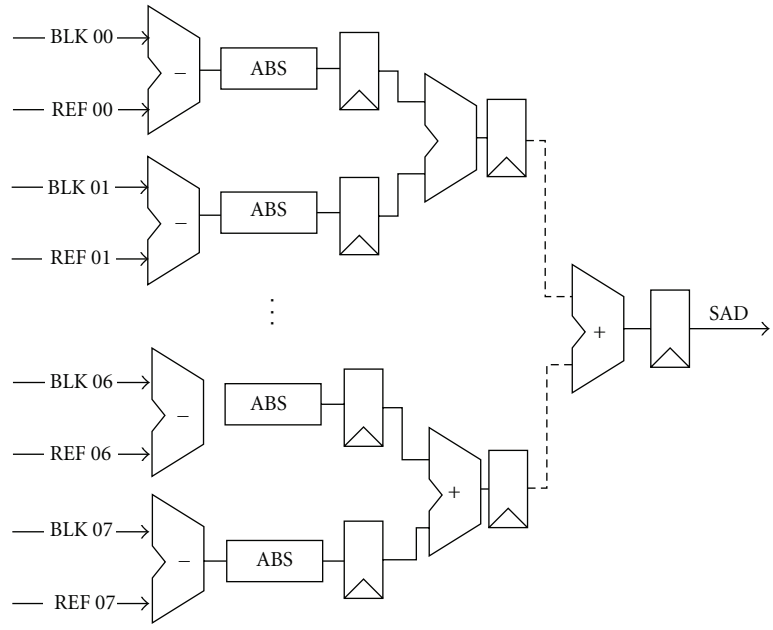

Figure 10: Processing unit RTL scheme.

high processing rates. So it takes all bandwidth provided by the memory hierarchy.

The ME architecture is composed by three processing modules (PMs), each one including units responsible for arithmetic operations; a comparator to decide which block represents the best match; a control unit that is responsible to manage the data flow with control signals.

Each PM is composed by eight processing units (PUs). The processing units (PUs) are responsible to give the results of similarity calculation between the current block and candidate block. Figure 10 presents the PU RTL scheme.

Once there are eight processing units per PM, each PU calculates the absolute difference between one row of the current block and one row of the candidate block. The data of the current block are obtained from the BRB, and the data from the candidate block are obtained from the RRB. In the PM there are pipelined adders that sum the results of the eight PUs and compute the complete SAD for a candidate block.

The comparator (Figure 11) receives the three SAD values and their respective motion vectors (MVs), each one generated for one PM. The three values are compared and the small one is stored in best motion vector register (best vector in Figure 11).

Once the PMs generate SAD values in pipeline, three SAD values are received for the comparator module at each clock cycle. After the comparator's pipeline is full, the comparison of these three values is delivered at each clock cycle. The best value of the three SADs is compared with the SAD stored at the best SAD register. At the end of the ME process, the best motion vector referent to the best SAD will be available at the best vector register.

The architecture control was developed in a decentralized way. Each module has its own manager, and the control unit sends signals for all the modules managers. The control unit has signals to initialize the complete architecture and output signals that signalize when the process of ME has finished.

Table 2: Synthesis results.

	Used	Available	Percent of usage
Slices	6,895	10,752	64%
Slice flip flops	6,469	21,504	30%
4 input LUTs	11,893	21,504	55%
BRAMs	1	72	1%

The synthesized results of the complete architecture to the Virtex 4 XC4VLX25 FPGA are showed on Table 2.

As shown in Table 2 the complete architecture spends 64% of available slices, 30% of flip-flop slices, and 55% of LUTs. Once this device is not a huge one, these values are acceptable for ME cores.

Focusing on BRAM results showed in Table 2, it is possible to realize that, from the total of BRAMs available on the target FPGA device, only 1% was used for the complete architecture. This means that the proposed memory hierarchy is very efficient to reduce the external bandwidth, to feed a high throughput ME core, and presents all these features using a very low amount of hardware resources. This result is also a stimulus to design a new hierarchy model considering the level D of data reuse, as proposed in [3].

Table 3 presents information about the minimum operation frequency required to achieve real time for different video resolutions, considering this implementations. The achieved frequency in this case study using a Virtex 4 was 292 MHz. The complete architecture is capable to process, in real time, since small resolution videos until high-definition videos. With this frequency, the architecture is able to process more than 86 720 HD frames per second and 38 1080 HD frames per second. Considering 1080 HD resolution at a

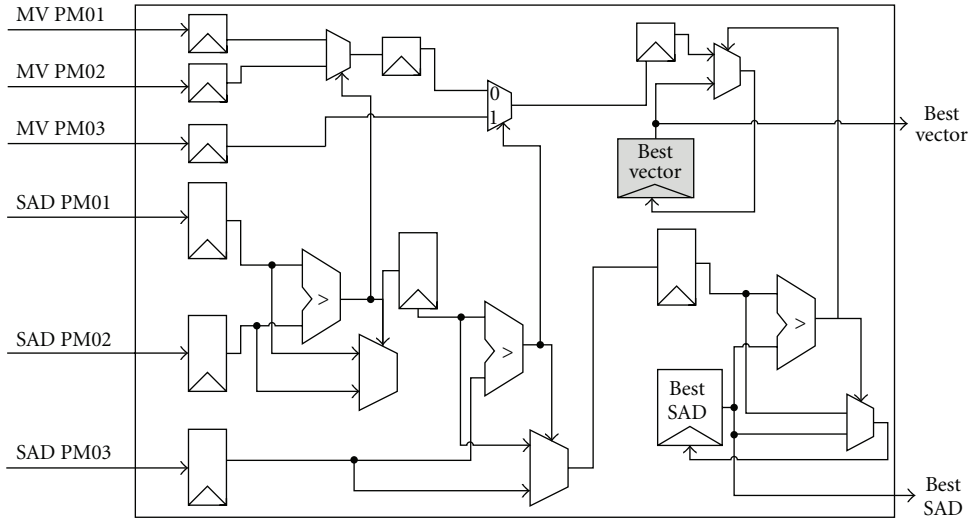

Figure 11: RTL scheme for the comparator module.

Table 3: Frequency and bandwidth required.

Resolution	Min. freq. (MHz)	Max. frames/sec.	Bandwidth (Mbytes/s)
CIF (352 × 288)	11.20	784.10	15.29
VGA (640 × 480)	33.83	259.59	45.26
SD (720 × 480)	38.04	230.84	50.75
720 HD (1280 × 720)	101.29	86.70	133.81
1080 HD (1920 × 1080)	227.75	38.56	299.59

frame rate of 30 fps, the required bandwidth is 299 Mbytes per second.

Using the proposed memory hierarchy, it is possible to develop a functional architecture for motion estimation with good quality results, high processing rates, and mainly with a reduced bandwidth.

7. Related Works

There are many works that focus on motion estimation architectures. But few of them focus on the memory bandwidth reduction. All works presented in this section uses full search as block matching algorithm. Table 4 presents a comparison with this related works.

The work presented in [11] uses variable block size, doing the SAD calculation for 4 × 4 blocks, and reusing these data to generate the SAD for the other block sizes. The architecture stores just the current block and the 4 lines of the search window. No local memory are used, only registers. This architecture attends to the level B data reuse scheme. Besides the use of small amount of registers and no local memory, it requires more external memory bandwidth than our solution.

In [12] is presented an architecture that allows variable block size and uses a 16 × 16 search window. It has a computation mechanism that processes the search window line per line, processing the first line of all candidate blocks in that line, and then, starting the next line, and so on, until the last line of the search window is reached. This architecture attends to the level B data reuse scheme. Although using a small size of search window, this architecture requires a high bandwidth because it does not reuse search window data from neighboring blocks.

Table 4: Comparison with related works.

	[11]	[12]	[13]	This work
Block size	4 × 4 to 16 × 16	4 × 4 to 16 × 16	4 × 4 to 16 × 16	8 × 8
Search window	19 × 19	16 × 16	65 × 65	32 × 32
Level of data reuse	B	B	C	C
Memory (Kbytes)	—	0.25	7.75	1.25
Register (Kbytes)	0.08	0.06	0.5	0.25
Bandwidth (MB/s)	1,397.8	1,008.5	1,054.0	299.5

The work presented in [13] attends to the level C data reuse scheme. It uses a 65 × 65 search window and variable block size. The size of the used search window is higher but the architecture attends to level C data reuse scheme, and then, this work also requires a very high bandwidth with external memory.

The architecture presented in this work using the proposed memory hierarchy model, presents a great tradeoff between search window size, block size and external memory bandwidth requirement.

Considering the external memory bandwidth required to process 30 frames per second, this work presents best result. Even using small search windows, the solutions presented in [11, 12] need a higher bandwidth to process 1080 HD videos in real time. This is because these architectures do not use a high level of data reuse. The work presented in [13] in spite of using level C of reuse of data uses a large search window, requiring a higher bandwidth than our work.

Develop hardware architecture for motion estimation implies always in a tradeoff between several parameters. So it is necessary to verify if the application priority is a better quality, a low cost in on-chip area, a reduction on external memory bandwidth or, an increase in the processing rate, for example. But, balancing all these criteria it is possible to develop efficient motion estimation architectures.

8. Conclusions and Future Works

This paper presented a memory hierarchy model for full-search motion estimation aiming to reduce the external memory bandwidth requirement. The proposed model, besides reducing memory bandwidth, it is also able to provide a high throughput to the ME core allowing the ME to process high-definition videos.

Considering the highest search window and the lowest block size evaluated, this architecture can reduce the external memory bandwidth in 578 times when compared to a solution without hierarchy.

The ME core designed and coupled to the memory hierarchy model proposed in this paper was described in detail. The complete architecture formed by the memory hierarchy and ME core can reach more than 38 frames per second when processing a 1080 HD video using only 299 Mb of external memory bandwidth.

As future works, we plan to evolve the memory hierarchy model to support motion estimation with multiple reference frames. It is also a future work to adopt the level D of data reuse, In this case, more on-chip memory will be necessary, but also the memory bandwidth requirement will be lower.

References

[1] J. V. Team, Draft ITU-T Rec. and Final Draft Int. Standard of Joint Video Spec. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] A. Bovik, *Handbook of Image and Video Processing*, Academic Press, 2000.

[3] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61–72, 2002.

[4] Y. Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering*, CRC Press, 2nd edition, 2008.

[5] P. A. Kuhn, *Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwe Academic Plubisher, Boston, Mass, USA, 1999.

[6] I. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley & Sons, Chichester, UK, 2003.

[7] Y. H. Hu and S.-Y. Kung, *Handbook of Signal Processing Systems*, Springer, New York, NY, USA, 2010.

[8] A. S. B. Lopes, I. S. Silva, and L. V. Agostini, "An efficient memory hierarchy for full search motion estimation on high definition digital videos," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*, pp. 131–136, Joao Pessoa, Brazil, September 2011.

[9] JM15.1, "H.264/AVC JM Reference Software," 2011, http://iphome.hhi.de/suehring/tml/.

[10] Xilinx, "FPGA and CPLD Solutions from Xilinx, Inc," http://www.xilinx.com/.

[11] R. S. S. Dornelles, F. M. Sampaio, and L. V. Agostini, "Variable block size motion estimation architecture with a fast bottom-up Decision Mode and an integrated motion compensation targeting the H.264/AVC video coding standard," in *Proceedings of the 23rd Symposium on Integrated Circuits and Systems Design (SBCCI '10)*, pp. 186–191, September 2010.

[12] R. Porto, L. Agostini, and S. Bampi, "Hardware design of the H.264/AVC variable block size motion estimation for real-time 1080HD video encoding," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '09)*, pp. 115–120, May 2009.

[13] L. Deng, W. Gao, M. Z. Hu, and Z. Z. Ji, "An efficient hardware implementation for motion estimation of AVC standard," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1360–1366, 2005.