ME64 – A PARALLEL HARDWARE ARCHITECTURE FOR MOTION ESTIMATION IMPLEMENTED IN FPGA

Diogo Zandonai [1,2], Sergio Bampi [1] and Marcel Bergerman [2]
[1] *UFRGS – Federal University of Rio Grande do Sul, Porto Alegre, Brazil;*
[2] *GIT – Genius Institute of Technology, Manaus, Brazil l*

Abstract: Digital video compression is a computationally intensive task, in which motion estimation accounts for a significant portion of the arithmetic operations. This paper presents ME64, a dedicated scalable hardware architecture for fast computation of motion vectors. ME64 is a highly parallel architecture, based on a matrix of 64 processing elements at its core, an I/O interface, and comparison and control units. The proposed architecture was implemented in an FPGA to treat reference and search blocks of 8x8 and 15x15 pixels, respectively. ME64 is scalable to be able to cover larger search blocks if needed. It implements the full search algorithm using the SAD criteria. ME64 was fully described in VHDL and prototyped in the Xilinx XC2S150 FPGA device, with a maximum frequency of 33 MHz. Using this FPGA device, ME64 reaches 2.1 GOps (billions of 8-bit operations per second) and 107.32 frames (640x480 pixels) per second. The results herein presented validate the ME64 against a software implementation, using an external I/O data driver.

Key words: Hardware Architecture for Motion Estimation, Motion Estimation, Video Compression.

1. INTRODUCTION

1.1 Motivation

Digital video has a growing number of applications, such as in DVDs, digital television, videophone, and PC multimedia. All these applications require a large communication bandwidth and/or storage space. Compression makes these applications feasible by reducing the amount of data necessary to represent the video information.

Figure 1 shows a block diagram of a generic video compression system. The pre-processing block performs color conversion and sub-sampling, when necessary. Compression occurs inside the blocks: static image compression and motion estimation, which are responsible for removing spatial and temporal redundancy, respectively. Once compression is performed, the resulting data is packed in a bitstream according to some standard, MPEG-2, for example.

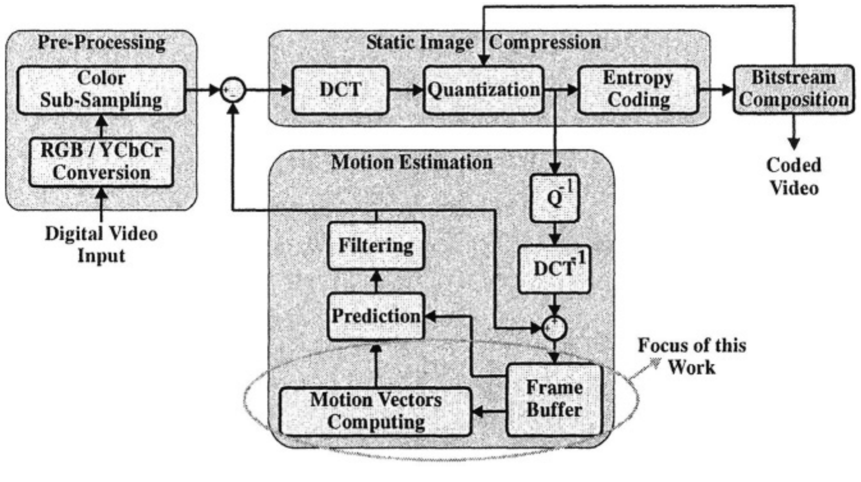

Figure 1. Generic video compression block diagram

Table 1 shows the computational effort to implement the main video compression tasks in MOp (millions of 8-bit operations) per frame. This table shows that the computational effort for motion estimation is more than three times the effort for image compression.

Table 1. **Computational effort to run the main video compression tasks in MOp/frame**

Video Format	Resolution (@ 30 fps)	Motion Estimation	Image Compression
CIF	352x288	6.49	2.13
VGA	640x480	19.66	6.45
SVGA	800x600	30.72	10.08
SVGA	1024x768	50.33	16.52
SXVGA	2048x1536	201.32	66.06

Since motion estimation is the most computationally intensive video compression task, its implementation in a dedicated hardware device saves hundreds of millions of operations and speeds up the task when compared to software solutions.

1.2 Motion vectors

Motion vectors are used to represent the reference frame based on the search frame, as shown in Figure 2. The reference blocks are represented by a portion of the search block that has the same size of the reference block. The motion vector points to the portion of the search block with the lowest distortion when compared to the reference block. Each possible portion is called a motion vector hypothesis.

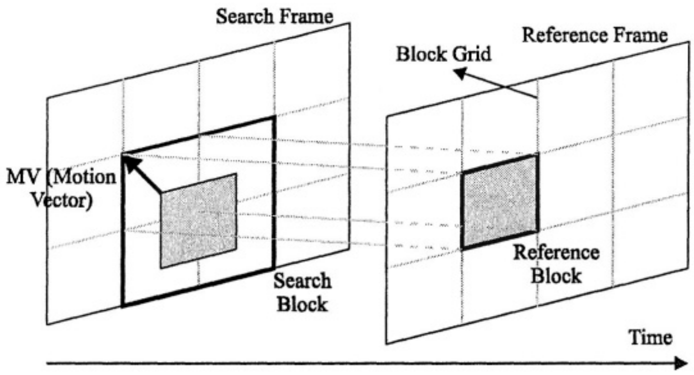

Figure 2. Motion vectors, search block and reference block

To find the best motion vector hypothesis, an algorithm that defines the search procedure and which hypotheses to consider is utilized in association with some criterion for distortion computing. In practice, the most common criterion is the SAD (sum of absolute differences) [5] [8]. Table 2 presents relevant algorithms for motion estimation and their operations requirement in MOp per frame.

Table 2. Algorithms for motion estimation and their operations requirement (MOp/frame)

Algorithm	Ref.	Operations Req.
Full Search	[1]	530
Circular Search	[8]	20
Logarithmic Search	[7]	37
Edge Matching Search	[2]	245
Hierarchical Search	[3]	20
Block Clustering	[4]	72

1.3 Previous works

Some relevant architectures for motion estimation have been developed, as in [5] [4] [6] [9]. Their common features are that they are comprised of linear or two-dimensional arrays of processing elements and all of them utilize the SAD as the distortion calculation criterion. Their main differences are the search block size, the I/O interface to input video data, the level of hardware parallelism and the clock frequency.

The remainder of article is organized as follows: Section 2 presents the proposed architecture, Section 3 presents the prototype used for validation and Section 4 presents the results and conclusions.

2. THE ME64 ARCHITECTURE

2.1 General considerations

ME64 implements the full search algorithm for block matching-based motion estimation. This algorithm was chosen due to its regularity and precision. Full search is the most precise search algorithm, since it returns the optimal motion vector hypothesis for a given search block. The analysis is performed in a very regular way, which allows ME64 to save CPU time by speeding up memory access through an efficient I/O interface and high level of parallelism.

In ME64, the criterion for distortion computation is the SAD. This is a common criterion in motion estimation hardware implementations [4] [6] [9] because it does not involve multiplications or divisions.

ME64 was designed to treat reference and search blocks of 8x8 and 15x15 pixels, respectively; therefore, 64 hypotheses are considered. One motion vector is computed every 64 clock cycles. Motion estimation is performed based on luminance data [1].

2.2 Architectural description

Figure 3 presents the ME64 high-level block diagram. The input reference (Y) and search (`SO` and `S1`) data are organized by the I/O interface in a way suitable for input to the Processing Matrix. The Processing Matrix computes the distortion for all motion vector hypotheses and presents one valid distortion value to the Comparison Unit at each clock cycle. The Comparison Unit analyses these hypotheses and indicates to the Control Unit through a `NEW_MV` signal pulse that a better hypothesis occurred. The Control Unit, upon receiving this pulse, generates the `MV` signal based on its own internal state.

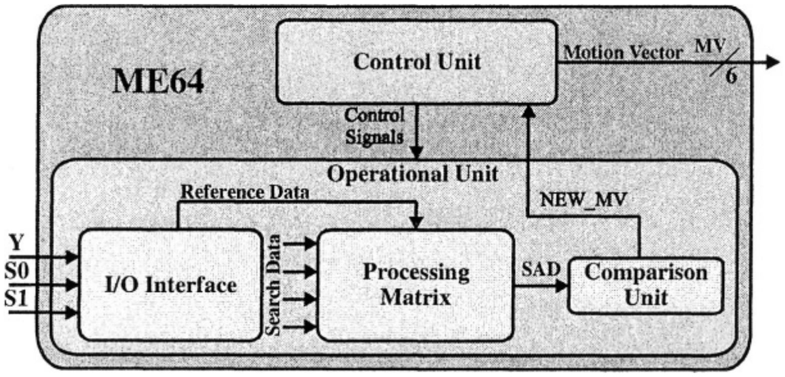

Figure 3. ME64 high-level block diagram

The Processing Matrix is composed of 64 processing elements (PE), each one responsible for the calculation of the distortion for one motion vector hypothesis. The PE architecture is presented in Figure 4. The reference data input `Ri` is stored in a register and presented in the output `Ro`, allowing for a pipeline organization of the PEs. The difference between the `B` and `Ri` signals is computed by `ADR0`. This difference may be inverted, depending on its signal, by a controlled inverter logic gate, implemented through XOR gates. The difference is then accumulated by `ADR1` in the `ACC` register, which is 6 bits larger than the input signal to avoid overflow. After 64 clock cycles `ACC` stores the distortion in such a way that the `SADij` signal is valid.

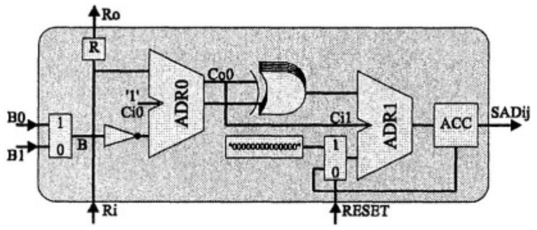

Figure 4. Processing element architecture

The Processing Matrix is presented in Figure 5. Each PE is named `PEij`, in which `i` stands for the line index and `j` stands for the column index of the element's position in the Processing Matrix. The `R` input signal feeds all PEs through a pipeline created by connecting a PE's `Ro` signal to the next PE's `Ri` signal. The four global buses feed local buses. The data from the `GB1` and `GB3` buses pass through a delay line with addressing function. Two local buses feed one line of PEs. The local buses named `LBi0` feed the `B0` input to the PEs while the local buses named `LBi1` feed the `B1` input to the PEs. Each `PEij` is responsible for the calculation of the distortion of the block whose first pixel is the one located at the coordinates (i,j) in the search block. Each PE starts computing one clock cycle that is delayed with respect to the previous PE in the pipeline. For this reason, only one `SADij` value is valid at each clock cycle. The `SADij` outputs from PEs feed the `M64` multiplexer which selects the unique valid `SADij` signal to feed the `SAD` signal.

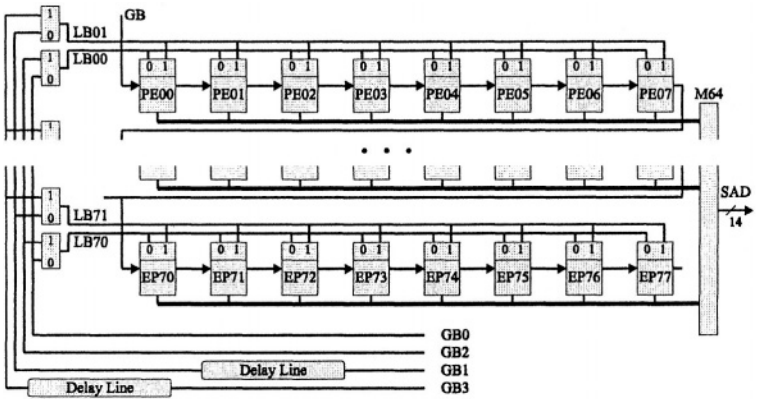

Figure 5. Processing Matrix architecture

The `SAD` signal is the input to the Comparison Unit. This unit, at each clock cycle, analyses one distortion value and, if it represents a minimum, it is stored in the `MIN` register and a `NEW_MV` pulse is generated.

The Control Unit is a finite state machine implemented using an 8-bit counter. The control signals are generated by applying a combinatorial logic to some bits of the 8-bit counter. The Control Unit is also responsible for generating the motion vector by sampling the addressing signal END at the time a pulse is received from the NEW_MV signal.

2.3 Scalability

The proposed architecture may be instantiated to support larger search blocks. For k^2 instances, the search block size is (k*8+7)x(k*8+7). Figure 6 presents an example of four ME64 instances to support a search block of 23x23 pixels.

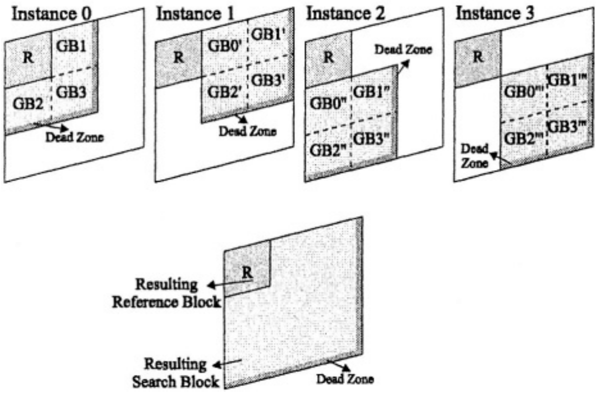

Figure 6. Scalability property of the ME64 architecture

The R bus of each region receives the same data while their global buses receive different partial search blocks. Note that the dead zones from inner instances are covered by adjacent instances.

3. SYNTHESIS RESULTS

The proposed architecture was validated in simulation with the *ModelSim 5.5* tool and by a software tool (running on the PC platform) that feeds data to the actual ME64 hardware implementation and reads back the values of the motion vectors. In addition, to verify the quality of the ME64 hardware computation, the motion vectors were also computed by software in the PC, and compared to the ME64 results for the same frames. The main development tools used were:

- Hardware development tool *WebPack 4.2,* from Xilinx. This tool is integrated with the simulation tool *ModelSim 5.5*, from Mentor Graphics.
- Xilinx Spartan II Evaluation Kit, with the target FPGA prototyping device XC2S150.
- Pentium III 733 MHz microcomputer connected to a video source, running MSWindows.

The ME64 full description was written in VHDL language. In this description, the bit-width of the input signals `Y`, `S0`, and `S1` were defined based on a generic parameter named `n`.

During the development, simulation was exhaustively used for validation. It also showed that the description with `n`=8 (as video is usually distributed) occupies 1,918 logic blocks, which is more than the 1,728 available in the XC2S150 device. Figure 7 presents the number of logic blocks taken up by ME64 for various values of `n`.

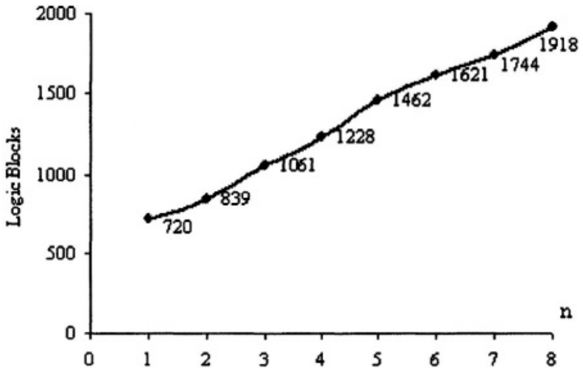

Figure 7. Number of logic blocks versus n

The XC2S150 device offers 12 configurable 4096-bit memory slices. The ME64 architecture requires simultaneous access to its ten 256-bit memory blocks. So unfortunately, each of ME64's memory blocks had to be mapped to different memory slice of the XC2S150, thus using up most of the available memory.

The software developed interfaces to the prototype via the PC parallel port. Due to the restrictions in the number of pins for communication using the parallel port and the low availability of logic blocks in the XC2S150 device, the prototype was initially tested with `n`=4.

The prototype was tested at a slow speed (9.76 KHz) to accommodate the slow communication channel provided by the PC parallel port. The hardware results for the motion vectors of a full frame were compared to the software calculation done in the PC for n=8. This way, the ME64 hardware calculations were validated.

4. CONCLUSION

With the FPGA running at 33 MHz, its maximum operating speed, the proposed architecture can estimate motion for video at a resolution of 640x480 pixels at the rate of 107.32 fps (frames per second) or 41.96 fps for a resolution of 1024x768 pixels.

Comparing the prototype against the PC software implementation, the FPGA hardware prototype is 19.67 times faster and performs 437 times more operations per second than a Pentium III 733 MHz running the compiled version of the motion estimation software.

The ME64 latency is 192 clock cycles. The prototype uses 16 I/O pins (for n=4), and the n=8 version would use 31 I/O pins. The prototype uses 71.1% of the logic blocks and 83.3% of the memory blocks available in the XC2S150 FPGA device.

Considering the high ME64 computing power there is no reason to use an algorithm other than the most precise, the full search. Moreover, in the implementation of another algorithm, the expected decrease in the operations rate requirement would not be directly converted to an increase in the speed, due to the limitations imposed by the I/O rate that the FPGA can sustain.

Table 3 presents a comparison of the proposed architecture against other solutions. The frame rate normalization was done considering frame size, reference block size and search block size.

Table 3. **Comparison between ME64 and others**

Architecture	Ref.	Max. Freq. (MHz)	Operations Rate (GOps)	Absolute Values		Normalized Values	
				fps	ms	fps	ms
STi3220	[5]	18.25	2.65	44.01	22.72	237.65	4.21
Architecture Based on Block Clustering	[3]	15	0.12	18.49	54.08	6.10	163.89
EST256	[6]	20	11	48.22	20.74	260.39	3.84
Linear array of 8 Processing Elements	[7]	8.32	0.06	3.38	295.86	3.38	295.86
ME64	-	**33**	**2.11**	**107.32**	**9.32**	**107.32**	**9.32**

The ME64 is among the fastest in Table 3. It is scalable, i.e. the frame rate may be increased further and the search block size parameterized. The

ME64 has a low I/O pin count and has a good ratio of operations to frame rate. Given the ME64 efficient I/O interface and its pipeline architecture, the hardware usage is 100% after initial pipeline fill latency.

Future developments include the design of different architectures of processing elements. The proposed architecture can be used to implement different algorithms such as hierarchical search or block clustering search. The prototype can be integrated with external memory and an image sensor aiming at a prototype running at the maximum simulated clock frequency. Another tool to be developed is an automatic generator of VHDL descriptions of motion estimation architectures based on the scalability property of ME64. These descriptions would have the same throughput as ME64 and a configurable search block size.

REFERENCES

[1] BHASKARAN, Vasudev; KONSTANTINIDES, Konstantinides. **Image and video compression standards:** algorithms and architectures. 2. ed. Massachusetts: Kluwer Academic Publisher, 1999. 454p.

[2] CHAN, Yui-Lam; SIU, Wan-Chi. Block motion vector estimation using edge matching: an approach with better frame quality as compared to full search algorithm. In: INT. SYMP. ON CIRCUITS AND SYSTEMS, Hong Kong, 1997. **Proceedings**, [S.l.]: IEEE Press, 1997, p. 1145-1148.

[3] CHU, Chung-Tao; ANASTASSIOU, Dimitris; CHANG, Shih-Fu. Hierarchical global motion estimation/compensation in low bitrate video coding. In: INT. SYMP. ON CIRCUITS AND SYSTEMS, Hong Kong, 1997. **Proceedings**, [S.l.]: IEEE Press, 1997, p. 1149-1152.

[4] FUJITA, Gen; ONOYE, Takao; SHIRAKAWA, Isao. A new motion estimation core dedicated to H.263 video coding. In: INT. SYMP. ON CIRCUITS AND SYSTEMS, Hong Kong, 1997. **Proceedings**, [S.l.]: IEEE Press, 1997, p. 1161-1164.

[5] QUEROL, Marc. **STI3220:** motion estimation processor codec. [S.l.]: SGS-Thomson Microelectronics, 2001. Available at: <http://www.st.com/stonline/books/pdf/docs/1648.pdft>. Access in: July, 18, 2002.

[6] SANZ, César; GARRIDO, Matías J.; MENESES, Juan M. VLSI architecture for motion estimation using the block-matching algorithm. In: AUTOMATION AND TEST EUROPE CONF., Paris, *1998*. **Proceedings**, [S.l.: s.n.], 1998, p. 45-49.

[7] SHI, Yun Q.; SUN, H. **Image and video compression for multimedia engineering:** fundamentals, algorithms and standards. United State: CRC Press, 2000. 480p.

[8] TOURAPIS Alexis M.; AU, Oscar C.; LIOU, M. L. Predictive motion vector field adaptive search technique (PMVFAST) enhancing block based motion estimation. In: VISUAL COMMUNICATIONS AND IMAGE PROCESSING, San Jose, 2001. **Proceedings**, [S.l.: s.n.], 2001.

[9] ZANDONAI, Diogo; BAMPI, S.; CARRO, L. An architecture for MPEG motion estimation. In: WORKSHOP IBERCHIP, 7., Montevideo, Uruguay, 2001. **Proceedings**, Montevideo, Uruguay: Universidad de la Republica, 2001, '1' CD.