ABSTRACT

Block matching motion estimation takes a great part of the processing time for video encoding. To accelerate this process is must to reach real time video coding. The best motion vector is obtained by full-search block matching algorithm which has to be usually implemented by hardware. In recent years, several FPGA based designs have been proposed since these devices support high number of process elements in parallel mode. In this paper a survey of recent architectures to perform the full-search block matching algorithm in FPGAs is presented. A further comparison on terms of frames per second reached, hardware cost in CLB slices and system frequency is presented.

1. INTRODUCTION

The full-search block matching algorithm (FSBMA) is usually used in the hardware implementation of motion estimation (ME), because of its simplicity, regularity, and optimum result.

The most commonly used metric to determine the best match for FSBMA in hardware is the Sum of Absolute differences (SAD). Computing the minimum SAD from among all the candidate blocks. To do this, a search iteration is performed for each candidate block.

The SAD adds up the absolute differences between corresponding elements in the candidate and reference block,

\[ SAD = \sum_{i=1}^{N} \sum_{j=1}^{N} |c_{i,j} - r_{i,j}|, \]

(1)

where \( r_{i,j} \) are the elements of the reference block, and \( c_{i,j} \) the elements of the candidate block.

Field programmable logic devices supports a high number of processor elements (PE) in parallel mode. This property can be used to process, at the same time, all SAD operations from a MPEG macroblock in a search area. With this real time (RT) video encoder for ME can be reached.

The high performance of current FPGA technology permits to implement new designs to solve the ME problem, this work presents a survey about recent ME designs.

2. RECENT ARCHITECTURES IN FPGA

In [1] two ways to parallel the ME process are exposed: To process the same number of pels as the macroblock has, called type-1; and to process the same number of ubications available for the motion vector (MV), called type-2. Most authors implement differences alternatives of the type-1.

To implement a variant of the type-1 architecture two main philosophies are used: To shift the pels through an systolic array of process elements (PE), or to parallel the PEs processing without shift the pels.

In this work the efficiency and the structure of several recent works to parallel the ME will be analysed.

1. Type-1 architectures shifting the pels:

A. Ryszko et al. [8] Implements the four architectures presented in [3]: AB1, AS1, AB2 and AS2.

AB1: in this, the data already fetched from memory can be saved in delay lines for later reuse, this limits the local high memory bandwidth requirements of this architecture. \( N \) data is fetched on every clock cycle from both: area search and reference block. Where \( N \) is the block side.

AB2: based on the preloading of the intermediate sum registers of absolute differences (AD) elements. The input data flow allows sequential computation of consecutive search area lines. AB2 architecture is based on AB1, replicating PEs structure to reuse data, and a high bit rate, \( N \) data from search area, improves related hardware cost than AS1 but increasing the frame rate. High complex and bandwidth are the disadvantages of this architecture.

AS1: this systolic array requires only a sequential data input. This architecture requires the same number of
clock cycles as AB1 to obtain a MV. Replicating AB1 structure, to reduce the high bit rate to 2 is possible, data is shifter on internal registers to reuse it. Multiply by 10 the hardware is necessary.

AS2: is the 2D extension of the AB1 architecture, where \( N \times N \) PEs are used. This architecture requires a high increasing of hardware.

M. Mohammadzadeh et al. [5] Also implements the AS1 architecture. An \( 8 \times 8 \) solution is proposed and how to extend to \( 16 \times 16 \) block is not explained.

2. Type-I architectures processing pels in parallel mode:

S. Wong et al. [9] Implements a \( 16 \times 1 \) SAD unit, called SAD16, which is equivalent to a macroblock row for MPEG. This design is inspired on the adder-tree model presented in [2]. The authors state briefly how to extend the design to compute a \( 16 \times 16 \) SAD reusing the original SAD16 unit to compute the remaining rows.

N. Roma et al. [7] Presents an innovative processing scheme based on a cylindrical structure and on the zig-zag processing sequence proposed in [1]. This cylinder is based on active and passive PEs. Active PEs processing while new data are being read in the passive pels. So, regular data flow and a simple reuse of overlapped pels is reached, also the hardware cost increases highly. \( N \) active PEs process a new block line on every clock cycle, an adder tree computes the SAD value and a final comparator evaluates if it’s appropriate to be the new MV.

J. Olivares et al. [6] This work presents a novel design based on online arithmetic (OLA). OLA works in bit-serie mode, this is, each bit is processed in successive clock cycles, operating with the most significant digit (MSD) at first, this model facilitates the absolute difference and the comparison operations, also, to process in bit-serial mode simplify to process \( N^2 \) pels at once. In this architecture the absolute difference improves also the arithmetical representation conversion with no computational cost, an online adder tree allows to process one macroblock at once, and a final online comparator allowing the early stop if the current SAD must be reached. Since this architecture improves a pipeline to process to bit level, a singular value of clock cycles to obtain a MV is calculated by the expression showed in Table 2.

3. Type-2 architectures:

H. Loukil et al. [4] Propose to process the SAD of \( 17 \times 1 \) ubications at time. And \( N^2 \) clock cycles are required to obtain each one.

2.1. The processor elements structure

Since PE is one of the main indicators to describe each architecture, to explain the structure of everyone is must to understand the element complexity:

Wong PE: is based on one carry generator, one inversor, and two exor. It function is to determine the smallest of the two pels, to invert the smallest operand and pass both operands to an adder tree.

Roma, Ryszko and Mohammadzadeh PE (called AD): This unit calculates the absolute difference of two pels and adds the result to a previously calculated partial sum of absolute differences. The partial sum of absolute differences is given from one processor element to the next processor element and finally the complete SAD is calculated.

Loukil PE (called SAD unit): This calculates the sum of all absolute differences until an external signal resets the accumulator of the SAD summation.

Olivares PE: This computes the absolute difference operating to bit level, and is performed multiplexing each bit from both pels.

To estimate the number of clock cycles required for each architecture two parameters, \( N \) and \( p \), are used, where \( N \) is the block side and \( p \) is the maximum displacement allowed for the block into the search area, typical MPEG values are \( N = 16, p = 8 \) and \( N = 8, p = 4 \).

3. COMPARISON

In section 2 several motion estimation architectures are presented. In Table 1 a comparison on terms of frames per second reached, hardware cost in CLB slices and system frequency are showed.

Also the family of the device used for every implementation is detailed. Because this, a qualitative comparison is necessary, and some appreciations must be exposed. In Table 2 structural parameters to describe and to compare the different architectures are showed. These are: The number of clock cycles required to obtain the best MV. The number of 8bit inputs is showed as I ports. The field AD PEs indicates the number of PEs for AD operations. The number of Adders. The number of comparators, Comp. is showed in the last column.

Notice that all authors uses 8bit adders excepting [6], that uses 1-bit adders. Also, other difference between all authors and [6] is in the PE architecture. Most author uses a PE based on carry-save adders, but in [6] only three 2-bit registers are used to implement the AD. To explicitly this difference is necessary because, the number of components can be used to compare several architectures, but in [6] all OLA components have less hardware cost, than conventional 8bit based arithmetics. The hardware cost (for a implementation in ISE 6.2i platform and Spartan3 technology), of an 8bit
adders appears: due to digit serial nature it reduces the number of signal lines connecting modules, the MSD first computation allows subsequent calculations to occur at much earlier stage, and it eliminates carry propagation chains since it uses a redundant number representation system. Non conventional arithmetics, SD OLA in this case, allows work with non conventional architectures, in the design proposed in [6], this feature is used to operate at bit level. Operate at bit level maybe has no sense for common processors with registers of 8, 16, 32 or more bits, but can be powerful in field programmable logic devices where nets and registers are configurable.

Notice that the low frame rate reached in [8] is using obsolete FPGA technology. To reach a high frame rate is expected for a recent FPGA technology. In fact, AS1 architecture is implemented by [5] reaching RT for 4CIF.

Architectures, [7] [6] [5], offers RT compression for ME in 4CIF. [7] obtains about 60% more frame rate than [6] but about 1280% hardware increasing is required.

The architecture presented in [7] reaches a high frame rate reducing drastically the number of clock cycles. So, the clock cycle to obtain a MV is about the 13% of [6], the 8% of [9] or the 25% of [4]. But this reduction also involves a high increase of hw components, where about the 40% of the PEs are passive. This increment of PEs involves a more complex connection and a frequency decreasing.

2D architectures (those that have \(N^2\) PEs) are faster and more complex than 1D (with \(N\) PEs), a non linear progression appears between the number of PEs and the clock cycle rate. This is because 2D models requires a high number of input ports, and a complex memory interface is required, this must to be computed in a global system to compare strictly with 1D models. Notice the memory interface can contribute over 50–70% of the global system for 2D models. In this way, 2D models cannot be compared appropriately with 1D models without the memory interface.

### 4. CONCLUSION

Recent FPGA implementations of FSBMA for ME video coding are analysed. Thanks to the high degree of parallelism that FPGA devices permits, FSBMA is accelerated to reach RT processing. Parallel computation of the pels corresponding to one candidate block, is preferred for most of the authors. The performance of the presented architectures and hw cost are compared based on the number of frames processed per second and the CLB slices respectively. The results show that FPGA is suitable for RT for 4CIF video sequences. Also the frame rate obtained for HDTV720p towards to be a reference point in future works.
Table 1. FPGA architectures performance comparison.

<table>
<thead>
<tr>
<th>Design</th>
<th>Block Size</th>
<th>4CIF (fps)</th>
<th>HDTV 720p (fps)</th>
<th>CLB slices</th>
<th>Freq. (MHz)</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>16 × 16</td>
<td>10.25</td>
<td>4.51</td>
<td>≈ 955</td>
<td>197.0</td>
<td>Altera Flex20KE</td>
</tr>
<tr>
<td>[7]</td>
<td>16 × 16</td>
<td>47.61</td>
<td>20.98</td>
<td>≈ 29430</td>
<td>76.1</td>
<td>Xilinx XC40250</td>
</tr>
<tr>
<td>[8] AB1</td>
<td>8 × 8</td>
<td>1.25</td>
<td>0.55</td>
<td>184</td>
<td>25.0</td>
<td>Xilinx XCV3200E</td>
</tr>
<tr>
<td>AS1</td>
<td>8 × 8</td>
<td>1.20</td>
<td>0.53</td>
<td>1214</td>
<td>24.0</td>
<td>Xilinx XCV3200E</td>
</tr>
<tr>
<td>AB2</td>
<td>8 × 8</td>
<td>12.50</td>
<td>5.5</td>
<td>948</td>
<td>30.0</td>
<td>Xilinx XCV3200E</td>
</tr>
<tr>
<td>AS2</td>
<td>8 × 8</td>
<td>18.25</td>
<td>8.03</td>
<td>3732</td>
<td>22.0</td>
<td>Xilinx XCV3200E</td>
</tr>
<tr>
<td>[4]</td>
<td>16 × 16</td>
<td>11.70</td>
<td>5.15</td>
<td>1654</td>
<td>103.8</td>
<td>Altera Stratix</td>
</tr>
<tr>
<td>[5]</td>
<td>8 × 8</td>
<td>31.25</td>
<td>13.75</td>
<td>300</td>
<td>191.0</td>
<td>Xilinx VirtexII</td>
</tr>
<tr>
<td></td>
<td>8 × 8</td>
<td>32.41</td>
<td>14.26</td>
<td>657</td>
<td>401.9</td>
<td>Xilinx Spartan3</td>
</tr>
</tbody>
</table>

Table 2. FPGA architectures functional comparison.

<table>
<thead>
<tr>
<th>Design</th>
<th>I ports</th>
<th>AD PEs</th>
<th>Adders</th>
<th>Comp.</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>2N</td>
<td>N</td>
<td>1</td>
<td>(27 + N − 1)x(2p + 1)^2</td>
<td></td>
</tr>
<tr>
<td>[7]</td>
<td>3</td>
<td>N^2</td>
<td>1</td>
<td>2x(2p)^2 + N x(2p + N − 1)</td>
<td></td>
</tr>
<tr>
<td>[8] AB1</td>
<td>2N</td>
<td>N</td>
<td>1</td>
<td>N x(2p + 1)x(2p + N)</td>
<td></td>
</tr>
<tr>
<td>AS1</td>
<td>2</td>
<td>2p + 1</td>
<td>2p + 1</td>
<td>2p + 2</td>
<td></td>
</tr>
<tr>
<td>AB2</td>
<td>2N</td>
<td>N^2</td>
<td>1</td>
<td>(2p + 1)x(2p + N)</td>
<td></td>
</tr>
<tr>
<td>AS2</td>
<td>2N + 2p</td>
<td>(2p + 1)xN</td>
<td>2p + 1</td>
<td>2p + 2</td>
<td></td>
</tr>
<tr>
<td>[4]</td>
<td>3N</td>
<td>2p + 1</td>
<td>2p + 1</td>
<td>N + 1</td>
<td></td>
</tr>
<tr>
<td>[5]</td>
<td>2</td>
<td>2p + 1</td>
<td>2p + 1</td>
<td>2p + 2</td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td>2</td>
<td>N^2</td>
<td>2x(N^2 − 1)</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

5. REFERENCES


