Indian Journal of Science and Technology, Vol 9(39), DOI: 10.17485/ijst/2016/v9i39/100769, October 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Energy Efficient Reconfigurable Architecture for Motion Estimation in Video Coding

S. Savitha* and K. R. Sarath Chandran

Computer Science and Engineering, SSN College of Engineering, Chennai - 603110, Tamil Nadu, India;
saviviji17@gmail.com, sarathchandran@ssn.edu.in

Abstract

Background/Objectives: Reconfigurable architecture has ability to dynamically allocate the hardware resources during runtime. It can be effectively used in computationally intensive application like media processing. As the motion estimation in video coding consumes large amount of computational time and resources, it can be mapped into reconfigurable architecture to effectively manage the power utilization by dynamic reconfiguration. **Methods/Statistical Analysis:** A systolic array based reconfigurable architecture for motion estimation which can be configured based on the properties of input video is proposed. A dynamically reconfigurable hardware is designed which can be worked on different search regions based on the level of motion in frames of input video. For the input video, the level of motion among the adjacent frames is determined by motion analyzer. Based on the level of motion between the frames of video, the search window size for block search is selected and this selection will enable the optimum number of processing elements for processing. This dynamic selection of hardware resources based on the search window reduces the power dissipation and computational complexity. Findings: Two search windows have been fixed for analysis 8×8 and 7×7. For power dissipation analysis, the total logic elements, total registers and fan-out for each design is taken. The performance is analysed by enabling selective number of processing elements for different size of search window. It is observed that power dissipation is high for the search window 8×8, because the resource utilization is higher than 7×7 search window. Instead of using the same fixed search window for performing block batching, different sized search windows can be used based on the level of motion of the video. After analysis, it is positively found that the proper selection of search window will lead to the optimum utilization in terms of power and resources. **Application/Improvements:** The context-aware reconfigurable hardware design for highly computationally intensive applications like video processing would be helpful in optimizing the power and resource utilization in hand held devices like smart phones, cam-coders etc.

Keywords: Motion Estimation, Motion Vector, Power Optimization, Reconfigurable Architecture, Video Coding

1. Introduction

The field of reconfigurable architecture has been familiar to more audience because of the advantages of providing better performance and having less power dissipation than sequential CPU based computing. Media applications such as video processing in cell phone base stations are becoming more complex in computing power. In order to achieve the computational demands of these applications, new reconfigurable architectures are emerging. The Motion estimation is the critical component in video coding as it consumes large amount of computational resources. Based on complexity analysis, different video compression standards show that the Motion Estimation (ME) module is the most computational intensive component. The full search algorithm is the efficient algorithm for motion estimation, because it has higher accuracy. This algorithm compares the macro blocks at each possible location in the search window with the macro block in current frame. This leads to the best possible match of the macro block in the reference frame. But it requires greater number of computations and takes higher power consumption. So the full search algorithm is mapped into reconfigurable architecture to efficiently manage the power and resource utilization. The reconfigurable architecture for motion estimation is supplied

*Author for correspondence

with current-block and search-region data from external memory. This reconfigurable fabric has been designed to be configurable on selecting the best processing elements and distributing these tasks to selective number of workers for achieving the parallelism. Typically, not all the functionality needs to be implemented by the reconfigurable module. The computation that is critical in time consumption lead to the reconfigurable fabric. The motion estimation in reconfigurable architecture can be adapted to comply with different system constraints such as power dissipation and time delay. When the best match is calculated exhaustively then there are possibilities that increase in power dissipation and process of motion estimation may be slower. This motivates to develop the energy efficient architecture for motion estimation.

2. Background and Related Works

In[1], proposes a concept of dynamically reconfigurable approximation, which helps in maintaining better control over application-level quality metrics while simultaneously reduce the power consumption and benefits the hardware approximation. MPEG has been the most preferred video compression scheme. The dynamically reconfigurable approximate hardware architecture varies the degree of approximation during run-time across multiple computational cycles, depending on inputs. In[2], uses the fast motion estimation technique, Adaptive Rood Pattern Search (ARPS) technique. A single processing element and simplified memory addressing is used to reduce the hardware complexity. An optimum area is used while satisfying speed requirements for real-time video processing. The ARPS algorithm considers the proper prediction of the current motion vector and various size of the search pattern based on available motion vectors. The ARP is followed by a Unit Rood Pattern (URP) to refine the search. The ARP pattern has centre point surrounded by four points located in four vertices. In[3], designed architecture to cover most of the video coding standards, including MPEG-2, MPEG-4, H.264, WMV-9 and AVS. The architecture easily handle flexible search ranges without any increase in silicon area and configured prior to the motion estimation process for a specific standard. The important focus is on Block-based ME architecture, Reconfigurable ME systems, Block-based architecture for H.264. In[4], proposes the spiral search for variable block size motion estimation in H.264/AVC. The spiral search provides hardware friendly data flow with efficient resource utilization. The performance is better when compared with three step search. It has lesser computational complexity. In[5], discussed about the evolution of reconfigurable computing in coarse grained architectures. The coarse grained reconfigurable is explored on the basis of hardware aspects of granularity, reconfigurability and interconnection networks. The better performance is achieved by exposing the parallelism into the computation models. The coarse-grained architectures consist of functional units and they are most suited for multimedia and streaming applications.

3. Proposed System

The proposed system architecture is shown in Figure 1. It contains different blocks for generating motion vectors in energy efficient manner. First, the frames cache block is used for storing the sequence of frames from input video. The SAD calculator is designed inside the processing element, for generating the motion vectors of all macro blocks in current frame. Configuration memory is used to store the different configurations according to the output from the selection of processing element block.

3.1 Frame Buffer

Video is defined as the sequence of frames. The sequence of frames in the input video is stored in frame buffer. These frames can be used in the motion analyser block and processing elements block for analysing the motion between the frames and for calculating the SAD values. The video format used in the proposed architecture is YUV video format.

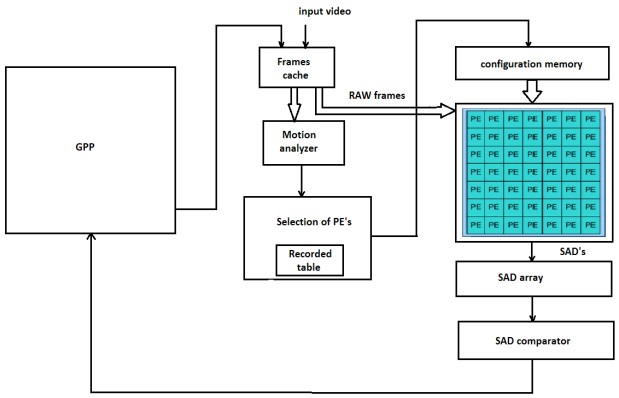

Figure 1. System architecture.

3.1.1 YUV Video

Motion estimation needs an uncompressed video format. So, YUV 4:2:0 video format is used by our system. The YUV formats use 8 bits per pixel location to encode the Y channel, which use 8 bits per sample to encode each U or V chroma sample[6]. A notation called the "A: B: C" notation describes how U and V are sampled relative to Y. The term U is equivalent to Cb, and the term V is equivalent to Cr. The YUV videos include different types of format. Figure 2 shows the original image 8 pixels wide and 4 pixels high, and indicates the boundaries of the chrominance pixels with heavy lines. The dots (white and black) in the Figure.2 represent the chrominance samples. The black dots show the chrominance values. In YUV 4:2:0 video formats, for every Y value, Cr and Cb values are sampled in both the horizontal and vertical dimensions by a factor of 2.

3.2 Motion Analyzer

For the input video, the level of motion among the adjacent frames is determined by motion analyzer[7,8]. The reference frame and current frame can be obtained from frames cache and he absolute difference between these two frames give the level of motion in video. A threshold is maintained to determine level of motion. If the absolute difference is lesser than or equal to threshold, then minimum number of processing element is enabled. High level of motion is detected, when the absolute difference between the consecutive frames are higher than threshold. This is given to selection of processing elements block. The configuration with large size search window is selected when the level of motion is high. Because the best match of the current frame block is obtained inside the large size search window. When the level of motion is low, the smaller size search window is selected. It is because the blocks in the frames may be moved to adjacent positions. So the best match is determined in power efficient manner.

3.3 Processing Elements

The processing elements perform the Sum of Absolute Difference between current frame and reference frame and the formula for Sum of Absolute Difference is given as:

$$Sum\ of\ Absolute\ Difference(i,j) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|c(i,j) - r(i,j)| \quad (1)$$

Where c(i, j) is the current macro-block and r(i, j) is the candidate macro-block in the reference frame in equation(1). The macro-block in the current frame is searched for best match in the reference frame(s). The Figure 3 shows how the reference frame macroblocks are taken to calculate Sum of Absolute Difference.

The detailed working of single processing element is shown in Figure 4. The processing element contains a data cache for storing the macroblocks of current frame

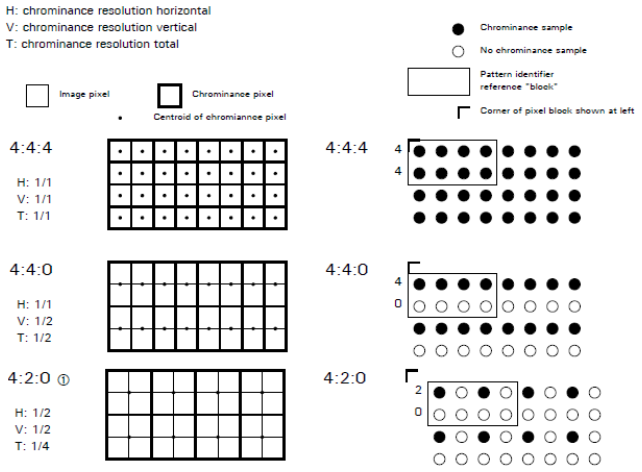

Figure 2. Types of YUV video format.

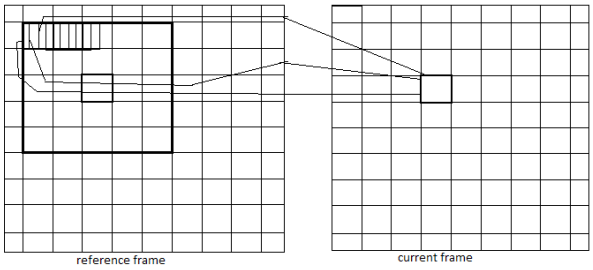

Figure 3. Sum Absolute Difference calculation inside search window.

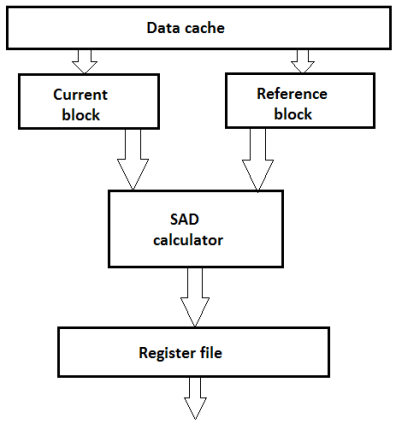

Figure 4. Structure of processing element.

and reference frame. The Sum of Absolute Difference calculator includes subractor and adder and the output is given to comparator.

3.4 Reconfiguration Module

The Reconfiguration module contains group of processing elements. Each processing element requires a current block, a reference block and a subractor to calculate Sum of Absolute Difference, a register to store the output and an enable port which is connected to configuration memory to activate or inactivate the processing element. with respect to the size of the search window in current frame. The selection of number of processing elements selection can be determined by:

$$Selection\ of\ PE's = (N - n + 1) \times (N - n + 1) \quad (2)$$

where N = search window size, n = size of macro block. So, the different configurations are obtained for different search windows. This shows that the reconfiguration of the system reduces power dissipation and efficient in resource utilization. A single PE calculates the Sum of Absolute Difference between two blocks. The SAD's from each PE is given to the SAD comparator. The reconfiguration module contains 25 PE's and 24 comparators.

3.5 Configuration Memory

The configuration memory is used to enable or disable the selective number of processing elements[9], according to size of the search window. Among the total number of PE's, smaller numbers of PE's are active in default. If the system requires maximum number of PE's then the configuration memory is used to enable PE's in addition. This is referred as the configuration patterns. For different size of search of search window, the configuration patterns changed. This can reduce the power dissipation among the system design.

3.6 SAD comparator

The output from the reconfiguration module is given to SAD comparator. The minimum SAD value obtain from the SAD comparator is the motion vector of the current block. So the resulting minimum SAD value is the best match of the current frame macro block.

4. Results Analysis

This section contains the implementation results of reconfigurable architecture. The implementation consists of analysis of input video frames, Motion analyzer, hardware modules, comparator and Configuration array. Altera Quartus II 9.0 was used to design the hardware modules. The Y, U and V components are stored separately for each frame. The pattern of the YUV video is verified using Hex editor tool. The Y components are highly occupied the frame when compared with the other components. The Y values are taken to determine the level of motion between the frames. The difference between the current frame and reference frame is determined by formula:

$$Y'_{i,j} = Y^{1}_{i,j} - Y^{2}_{i,j} \quad (3)$$

where Y'i j is the frame difference(3), Y1i j and Y2i j are the two frames which is used to calculate the difference[7]. Then the level of motion between the frames can be calculated using the formula (4).

$$Level\ of\ motion = \sum \frac{[(Y]'_{ij})}{n} \quad (4)$$

where i, j are the rows and columns of the matrices and n is the number of elements in $Y'_{i,j}$ (4). Table 1 shows the frame differences for different YUV videos. The mean of observed frame difference values[8] are akiyo = 0.3813, foremann = 4.9432, bus = 20.6877, coastguard = 7.7809. From the mean values, the threshold is fixed as 10. The 'bus' video contains the high level of motion because it contains the values at the range of 20 and above. The 'akiyo' contains the low level of motion, since it contains the values less than 1.

The exact match of the current frame macroblock in reference frame is shown in Figure 5. The sample input matrices are given to processing elements to verify the working of hardware modules.

Table 1. Analysis of level of motion

Frame Difference	Video			
	Akiyo	Foremann	Bus	Coastguard
1,2	0.3719	5.5826	20.2791	7.5364
2,3	0.3474	5.5880	20.1928	7.5999
3,4	0.3406	5.0315	20.4931	7.5567
4,5	0.3930	5.1051	20.7525	8.2267
5,6	0.4280	4.5335	20.8020	7.9229
6,7	0.3600	3.8905	21.0227	7,9136
7,8	0.4282	4.8714	21.2721	7.7132

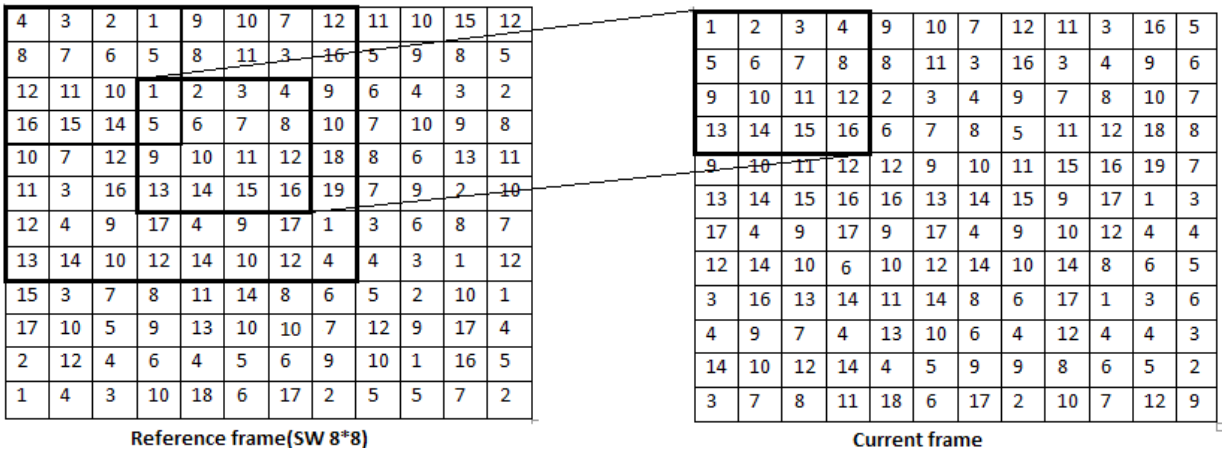

Figure 5. Inputs for ME.

The following are the resulting SAD's. SAD1 = 42, SAD2 = 57, SAD3 = 75, SAD4 = 92, SAD5 = 99, SAD6 = 80, SAD7 = 82, SAD8 = 76, SAD9 = 67, SAD10 = 61, SAD11 = 85, SAD12 = 68, SAD13 = 46, SAD14 = 0, SAD15 = 32, SAD16 = 87, SAD17 = 56; SAD18 = 86, SAD19 = 49; SAD20 = 96, SAD21 = 76; SAD22 = 82, SAD23 = 75, SAD24 = 84, SAD25 = 95. For the search window size 8×8, all the 25 processing elements are enabled. The resulting configuration pattern for the search window 8×8 is '1111111111111111111111111'. The minimum SAD value is obtained from processing element 14.To find out the next minimum SAD and to verify the this hardware modules, the processing element 14 is disabled. So the configuration pattern becomes '1111111111111011111111111'. Table 2 shows the position of blocks inside the search windows. in reference frame for the corresponding current frame blocks. For power dissipation analysis, the total Logic elements, Total registers and Fan-out is taken. The performance is analysed by enabling selective number of processing elements for different size of search window. From Table 3, it is clear that power dissipation is high for the search window 8 × 8, because the resource utilization is higher than 7 × 7 search window. The comparison graph in Figure 6 and Figure 7 show that the total logic elements, total fan-out and total registers used by system for search window 8×8 and search window size 7 × 7. The power dissipation for search window size 8 × 8 is higher than search window size 7 × 7 is shown in Figure 8.

The performance analysis shows that the power dissipation is high for the search window size 8×8, because of the maximum number of processing elements are enabled.

Table 2. Search window indexing table

Current frame block No.	Search Window indexing	Processing element No.	SAD
Block1	Block14	PE14	0
Block2	Block2	PE2	5
Block3	Block7	PE7	0
Block4	Block14	PE14	6
Block5	Block11	PE11	0
Block6	Block17	PE17	0
Block7	Block7	PE7	0
Block8	Block23	PE23	0
Block9	Block13	PE13	0

Table 3. Resource usage and power dissipation

Resource usage	Search window 8 × 8	Search window 7 × 7
Total logic elements	4817	3053
Total registers	111	72
Total Fan-out	14363	9186
Total power dissipation	219.22mW	92.25mW

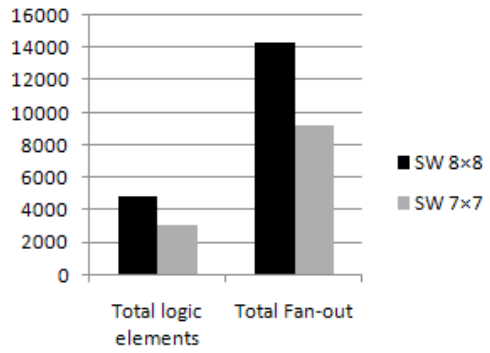

Figure 6. Resource usage.

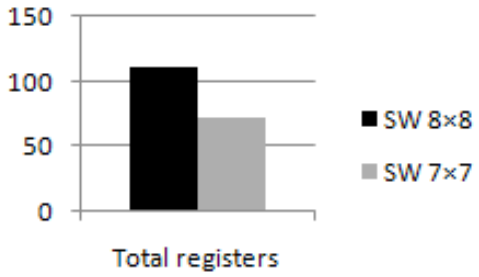

Figure 7. Resource usage.

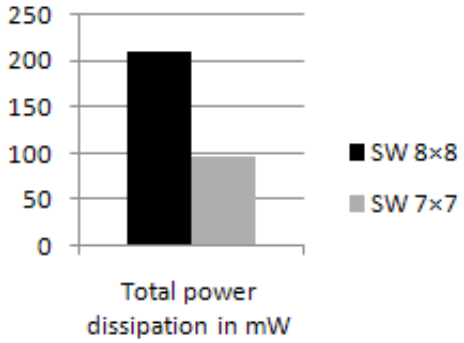

Figure 8. Power dissipation.

5. Conclusion and Future Work

The results conclude that the proper configuration for motion estimation selected, during runtime gives the optimization in terms of power and resource utilization. For future work, the size of the search windows can be refined more accurately by proper analysis. More configurations also can be introduced based on analyzing the level of motions in the input videos. The hardware design can be modified for variable size block matching for identifying fine motions in the frames.

6. References

1. Raha, Jayakumar A, Raghunathan H. Input-Based Dynamic Reconfiguration of Approximate Arithmetic Units for Video Encoding. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2008; 99:1–15.
2. Biswas B, Mukherjee R, Chakrabarti I. Efficient Architecture of adaptive rood pattern search technique for fast motion estimation. IEEE transactions on very large scale integration (VLSI) systems for microprocessors and microcontrollers. 2014; 39:200–9.
3. Lu L, McCanny JV, Sezer S. Reconfigurable system-on-a-chip motion estimation architecture for multi-standard video coding. In Proceedings of IET Computer Digitise Technology. 2010; 349–64.
4. Muralidhar P,.Rama Rao CB, Dwith CYN. Efficient Architecture for Variable Block Size Motion Estimation in H.264/AVC. ACEEE International Journals on Signal and Image Processing. 2014; 5:215–329.
5. Zain-ul-Abdin, Svensson B. Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing. Microprocessors and Microsystems. 2009; 4(33):161–9.
6. Kerr DA. Chrominance Subsampling in Digital Images. In Proceedings of Electronic communications 3(AE). 2014; 107–18.
7. Al-Ani MS, Hammouri TA. Video Compression Algorithm Based on Frame Difference Approaches. International Journal on Soft Computing (IJSC). 2011; 2(4):67–79.
8. Singla N. Motion Detection Based on frame difference method. International Journal of Information and Computation Technology. 2014; 4(15):1559–65.
9. Nithya R, Sarath Chandran KR, Premanand Chandramani V. Run-Time Reconfiguration of Processing Elements through Soft-Core Processor. Third IEEE International Conference on Communication and Signal Processing. IEEE Xplore. 2014 Apr 3-5. p. 813–7. Doi: 10.1109/ICCSP.2014.6949956.