Microprocessors and Microsystems 45 (2016) 105–114

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Efficient VLSI design of adaptive rood pattern search algorithm for motion estimation of high definition videos

Rohan Mukherjee[a,*], Baishik Biswas[a], Indrajit Chakrabarti[a], Pranab Kumar Dutta[b], Ajoy Kumar Ray[a]

[a] Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur-721302, India
[b] Department of Electrical Engineering, Indian Institute of Technology, Kharagpur-721302, India

ARTICLE INFO

Article history:
Received 8 October 2015
Revised 1 April 2016
Accepted 13 April 2016
Available online 14 April 2016

Keywords:
Fast motion estimation
ARPS algorithm
VLSI architecture
Interleaved memory organization
Early SAD technique

ABSTRACT

Block-based motion estimation plays a significant role in video codecs by exploiting and reducing the temporal redundancies that exist between consecutive frames in a video sequence. Adaptive Rood Pattern Search (ARPS) is one of the most popular fast motion estimation algorithms. In this paper, VLSI design for the ARPS algorithm is proposed that involves reasonably limited hardware resource without compromising the real-time speed for transmitting HD videos. To tackle the adaptive nature of the algorithm, the proposed design avoids systolic arrays and introduces novel pattern generation methodology that can tackle the adaptive nature of the algorithm. Further, the design incorporates interleaved memory organization with a well–defined sharing strategy to re-use data and ensures high throughput. Working at a frequency of 112 MHz, the present design can process 30 Full HD 1080p (1920×1080) frames using only 47.15 K gates. Hence, the proposed VLSI architecture can be incorporated in video codecs that can be suitably used in devices like camcorders, tablets and smart phones.

1. Introduction

The on-going demand of real-time video applications has put forward new challenges in the domain of video coding. One of them has been the efficient design of motion estimation (ME) unit that contributes most to the computational complexity of the entire codec. The motion estimation unit plays a significant role by exploiting and reducing the temporal redundancies that exist between consecutive frames in a sequence. The widely accepted Block Matching Algorithms (BMA) provides simple and efficient ways to perform ME and have been adopted in current video coding standards like HEVC, H.264, H.263 etc. The basic approach of BMA is based on dividing the entire current frame into small rectangular blocks of pixels (say 16×16 or 8×8), followed by a process of finding the best possible matching block in the previously stored frame (reference frame) for each block in the current frame through certain well-defined criteria (Sum of Absolute Difference or Mean of Absolute Difference) [1]. The displacement between any block in the current frame and its best possible matching block in the reference frame is estimated by the motion vector (MV). The Full Search Block Matching Algorithm is the most exhaustive search. But its high cost (in terms of time and resources) poses great difficulty for real-time implementation [2]. In this scenario, the last two decades have witnessed the development of many fast BMAs like Three Step Search (TSS) [3], Enhanced Three Step Search (ETSS) [4], New Three Step Search (NTSS) [5], Diamond Search (DS) [6], Hexagonal Search (HEXBS) [7], and Adaptive Rood Pattern Search (ARPS) [8] algorithms. The shape and size of the search pattern used in Three Step Search and its variants (ETSS, NTSS etc.) contribute mostly to the performance of the searches. But in videos with large motion, these algorithms have high chances of getting trapped in local minima and sometimes proceed in wrong direction when large patterns are used in the initial step of search. The widely accepted center-biased search techniques like Diamond Search, Hexagonal Search exploit the nature of MV distribution and outperform TSS and its variants in terms of prediction accuracy and computation. But these techniques suffer when a certain mismatch occurs between the size of the pattern and the magnitude of actual motion. The above mentioned challenges for the search methodologies are well addressed by the adaptive search techniques that exploit the spatial and temporal correlation existing between neighboring blocks to predict the motion vector accurately [1,2]. The Adaptive Rood Pattern Search is the most popular among the adaptive search algorithms and achieves better prediction efficiency than DS or HEXBS with less number of computations [8].

Several methodologies have been proposed in the last two decades for VLSI implementations of the fast ME algorithms

* Corresponding author. Tel.: +91 9051849608.
E-mail address: mukherjee.rohan666@gmail.com (R. Mukherjee).

http://dx.doi.org/10.1016/j.micpro.2016.04.003
0141-9331/

[9–32]. One of the most popular techniques to realize hardware for motion estimation is the use of systolic array-based mapping of fast BMAs. The architectures consist of multiple number of Processing Elements (PE) that is arranged in mesh formation and the arrays are categorized by their dimensions. The PEs arranged in 1-D or 2-D formation ensures a regular data flow. Parallelization is the main motive behind the systolic array based operation. These structures are best suited for fast and regular ME algorithms like TSS, ETSS, NTSS, HEXBS and DS where the set of search points are pre-defined. But the basic challenge in these architectures is to guarantee maximum PE utilization and preserve the data flow mechanism for fast and adaptive algorithm. On the contrary, the initial search points in adaptive algorithms like ARPS do not follow a typical pattern and is highly dependent on the motion content of the scene. Further, mapping these adaptive algorithms to hardware is a complex phenomenon resulting in significant possible under-utilization of hardware resources. The architecture for 3 step hierarchical search (3SHS) is proposed by Jehng et al., [9] which can be considered as motivational work related to 2-D array based architecture. This scheme is designed using a 4×4 2-D array of PEs followed by SAD tree arrangement to accumulate the differences generated at the PEs. To store the search window region, the design by Jong et al., [10] uses an on-chip buffer and further use of multiple processing elements array increases the area of the chip. A 2-D array of 25 PEs has been considered in the design by Ding et al., [13] for the diamond search algorithm where buffer arrangements have been provided to upgrade the search window. The DS algorithm has been modified further for hardware implementation by Ndili et al., [14] and the architecture works efficiently at a frequency of 246.5 MHz with mesh structure and FPGA block RAMs. The recent architecture by Yin et al., [17] makes use of 256 PEs organized in three hierarchical structures. This bears a large hardware cost while dissipating high power. The hardware cost incurred by the design in [17] is 382 K. Reconfigurable systolic array based design has been introduced by Tasdizen et al., [18] for Dynamically Variable Step Search. Another recent work by Tsai et al., [19] proposes an effective algorithm with reduced search point and its 2-D systolic array architecture that works at a frequency of 200 MHz and consumes an area of 191 K gate equivalent.

There exist certain hindrances while adopting these designs for adaptive algorithms like ARPS. The buffer memory and the PE arrays can remain largely unutilized due to the variable number of search locations. Such smooth data reuse schemes of the mentioned designs are not suitable for the unpredictable and irregular data flow in ARPS which often leads to much complex memory address generation and cannot mitigate the control complexity. Previous work on development of VLSI for ARPS algorithm [21] had certain limitation. The design follows sequential execution by using a single processing element and can process frames of CIF format.

The presents work proposes VLSI architecture for the Adaptive Rood Pattern Search algorithm that involves less hardware resource without compromising the real-time speed for transmitting HD videos. The ARPS algorithm is a very popular fast search technique and its performance surpasses the other well-accepted searches such as DS, HEXBS. The proposed design avoids systolic arrays and introduces a novel pattern generation methodology that can tackle the adaptive nature of the algorithm. An early SAD check technique reduces the clock cycles required to find a MV for a macro-block. Further, the design incorporates interleaved memory organization with a well–defined sharing strategy to re-use data and thus enhances the throughput. The present design can process 30 Full HD 1080p (1920×1080) frames using only 47.15 K gates which is smaller than other architectures that can process HD frames. The area and the timing performance of the present work is compared with other fast ME architectures and the proposed architecture surpasses them in performance. Section 2 describes the ARPS search procedure whereas Section 3 presents the proposed architecture. Results are depicted in Section 4 followed by the conclusion in Section 5.

2. The adaptive rood pattern search algorithm

The ARPS algorithm proposed by Nie and Ma [8] consists of a symmetrical search pattern referred to as the 'rood' shape or the cross shape. There are two distinct stages of search called the Adaptive Rood Pattern (ARP) and the Unity Rood Pattern (URP) which differ primarily in the separation of the search points. In addition to the symmetrically placed points, the ARP stage has an additional search point given by the predicted motion vector. The initial stage is called adaptive as the rood arm length is determined by the motion content of the neighboring macro-blocks. The ARPS algorithm is iterative in nature; the ARP stage is used once, whereas the lightweight URP search is applied iteratively until the search converges. The following steps further elucidate the ARPS algorithm:

Step 1: The center of the search window is chosen as the center of the ARP stage. In Fig. 1, the center O of the ARP denoted by the square ABCD is fixed at the center of search window. The SAD is computed for all the points including the predicted point P as shown in Fig. 1.

Step 2: The minimum SAD point obtained from the previous step is chosen as the center of the URP and all its points are checked. If the new minimum point coincides with the current search center, then the search terminates; else step 2 is followed again. As shown in Fig. 1, points B and Q are the minimum SAD points. The search terminates at Q.

The arm length of the adaptive rood pattern i.e., OB or OA in Fig. 1 is determined from the motion vectors of the co-located blocks. The ARPS algorithm defines a set of co-located blocks called the ROS as shown in Fig. 2 where macro-block E is one whose MV is required. Although there are various prediction schemes, to ease the complexity the following has been adopted.

$$MV_{predicted} = MV(i, j-1) = MV_A \tag{1}$$

$$\begin{aligned} &\text{ARP arm length (T)}(= OA = OB = OC = OD) \\ &= \text{Max}\left\{\left|MV_{predicted}(x)\right|, \left|MV_{predicted}(y)\right|\right\} \end{aligned} \tag{2}$$

It has been shown in [8] that such a simplified prediction of the motion vector does not degrade the PSNR performance. However, it is highly suitable for hardware implementations of the ARPS algorithm.

3. Proposed architecture

Motion estimation is the process of finding the best match of the current frame macro-block in the reference frame. The operations involved in such a process is the computation of the Sum of Absolute Difference (SAD), which is expressed as

$$SAD = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} |\text{Re}f(x_p + i, y_p + j) - Curr(x_o + i, y_o + j)| \tag{3}$$

where (x_0,y_0) and (x_p,y_p) are the current frame and reference frame macro-block starting addresses. The search algorithm, APRS in this case, specifies the reference frame search points (x_p,y_p) for a given current frame point (x_0,y_0). The architecture proposed here computes the SAD for each of the search points (x_p,y_p) and decides on the minimum distortion point. The reference and the current frame search windows are stored in small interleaved