RECONFIGURATION FOR POWER SAVING IN REAL-TIME MOTION ESTIMATION

S.R. Park and W. Burleson

Department of Electrical and Computer Engineering
University of Massachusetts at Amherst, MA 01003
{srpark, burleson}@ecs.umass.edu

ABSTRACT

Motion estimation presents a class of algorithms well-suited to reconfigurable hardware due to their variable computational load, highly structured array architectures, robust reduced complexity algorithms, and a motivation for low power implementations in portable video products. Motion estimation is the most computationally demanding part of video compression algorithms and hence usually requires hardware support for real-time implementation. However dedicated hardware usually requires that the algorithm and most of its parameters be hardwired. Reconfigurable hardware based on FPGAs allows the parallelism of hardware implementations with the flexibility of software.

The statistics of motion vectors can be monitored on a frame by frame basis to choose appropriate algorithm and hardware configurations. Unlike some proposed applications of dynamic reconfiguration, this rate can easily be supported by existing FPGA technology. Another novel aspect of this work is that we use power savings as a motivation for the reconfiguration. Although FPGAs are not a very power efficient technology, careful design of array architectures can allow power to be saved by avoiding unnecessary computation by adjusting the search area according to the changing characteristics of an input video signal. Another more general result is that further power saving can be achieved by utilizing free FPGA resources as local memory to avoid power-hungry off-chip communication. Practical implementation issues using Xilinx 6200 series FPGAs are also discussed.

1. INTRODUCTION

Visual communication is a rapidly growing area for telecommunications, computers and multimedia. There are now several video compression standards such as H.261, MPEG-1 and MPEG-2, which are all based on block based motion compensation and Discrete Cosine Transform(DCT) to reduce the temporal and spatial redundancy, respectively[1]. To reduce the enormous date rate of a video signal, DCT is used in intraframe coding and motion estimation /compensation is used in interframe coding.

Motion estimation is the most computationally demanding part of video coding. It exploits temporal correlation between two consecutive frames and provides a large (typically a factor of five) coding efficiency [2]. Most motion estimation algorithms implemented in VLSI up to now do not have flexibility at all or have only a small amount of flexibility. Most significantly, the search area size can not be varied at run-time. If the search area can be adjusted dynamically according to the changing characteristics of an input video signal, the search area size can be tailored to avoid unnecessary computation, and hence power saving without severe loss of picture quality.

In this paper, we propose a reconfigurable approach to motion estimation. Rather than proposing new architectures or new matching criteria, we focus on how reconfigurability can be exploited in motion estimation with well-known architecture and matching criteria. The power saving due to reconfiguration is analyzed by using a simple model.

2. MOTION ESTIMATION

For estimating motion by means of a block matching algorithm, the image is divided into blocks of $n \times n$ pixels. Usually n is 16. The blocks resulting from the segmentation of the current and previous frames are called the current and previous block, respectively. For each current block, the best matching previous block is found within a search area surrounding the previous block. The previous blocks in a search area are called candidate blocks. Suppose the search area extends on both sides over p pixels in the horizontal and vertical directions, then the search area is $(2p+n)^2$, and the total number of the candidate blocks in search area is $(2p+1)^2$. If the picture size is M by N pixels, then the number of blocks in one frame is MN/n^2. We assume that M and N are integer multiples of n throughout this paper.

To compute the motion vector, the Mean Absolute Difference (MAD) criterion is widely used.

$$D(k,l) = \sum_i \sum_j |x(i,j) - y(i+k, j+l)| \qquad (1)$$

$$V_{min} = (k,l)_{D\,min} \qquad (2)$$

where $x(i,j)$ is the luminance value of a pixel in the current block and $y(i+k, j+l)$ is the luminance value of a pixel in the candidate block. V_{min} is called the motion vector. The displacement calculated is limited to a search area range such that $-p \leq k, l \leq p$. Fig.1 shows a block and a block matching algorithm. Basically block matching algorithms can be divided into two categories: full search block matching algorithm(FS BMA) and 'intelligent' or 'directed' search

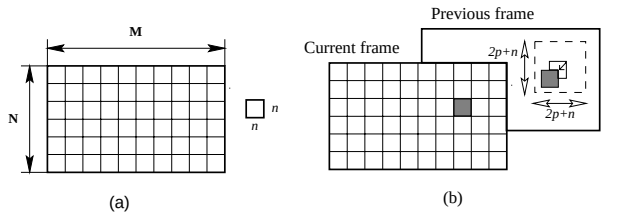

Figure 1: (a) Block in a frame (b) Motion Estimation by block matching

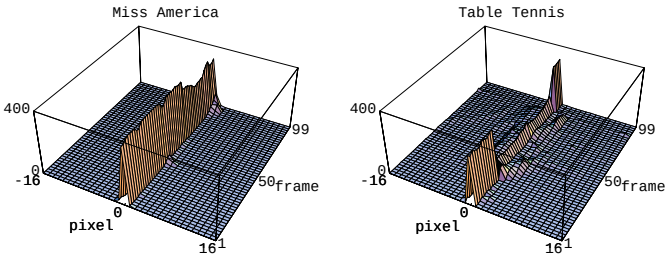

Figure 2: Motion Vector Distribution over time in 'Miss America' and 'table tennis' video sequences.

algorithms. In FS BMA, all candidate blocks in a search area are compared to the current block and a best match is determined. To reduce the computational overhead 'intelligent' search algorithms compare only a small number of candidate blocks in a search area while introducing mild performance degradation.

As for the matching criteria, Mean Absolute Difference(MAD) is widely used instead of Mean Squared Difference(MSD) due to the simpler operation. Reduced Bits Mean Absolute Difference(RBMAD)[4], and Pel Difference Classification(PDC)[5] can be used to reduce the hardware complexity at the expense of mild performance degradation.

3. RECONFIGURABLE MOTION ESTIMATION ALGORITHM

In general, FS BMA with MAD matching criteria is used in VLSI implementation due to its regularity and simple basic calculations. However reconfigurable VLSI opens up new possibilities which better support irregularity to exploit the temporal variation in video sequences. Reconfigurable algorithms and architectures have the potential to reduce power consumption without severe degradation in picture quality by adjusting its search area. In addition, more power can be saved by utilizing unused hardware resources as local memory.

3.1. Motion Vectors and Search space

The statistics of motion vector vary considerably between and within video sequences. Fig.2 shows the distributions of the horizontal component of the motion vector of 'Miss America' and 'table tennis' video sequences. The first 100 frames were used for collecting the motion vectors. As can be seen, the range and the shape of the distribution of motion vectors are different mainly due to the changing characteristics of the video signal. In the 'table tennis' sequence, even in the same sequence, the shape of the motion vector distribution is different along with the frame number because of the changing characteristics of picture content. These observations are one of the motivations for changing the search window size and hence the motion vector range according to the input video signals. Fig.3 shows how compression varies with search space size for 'flower garden' sequence.

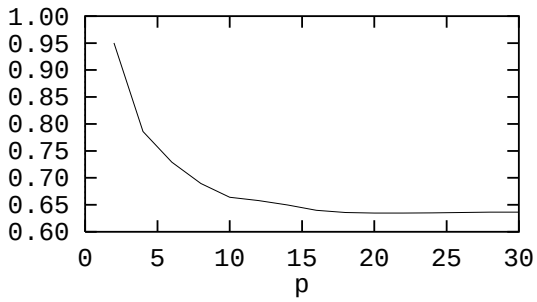

Figure 3: Bit per pixel(bpp) vs. p in 'flower garden' video sequence

3.2. Guidelines for Reconfigurable Motion Estimation Algorithm

In this subsection we discuss some guidelines for reconfiguration and suggest a simple algorithm. Two basic questions are how often a search area should be adjusted and how large a search area should be. The time interval between reconfigurations can be as short as a frame-by-frame basis or as long as several minutes. The reconfiguration rate should be determined by considering the followings: 1)the changing characteristics of the input signal, 2)the reconfiguration time of a specific FPGA being used, 3)penalties for gathering the motion vector statistics. Hence it should be determined by considering the system and FPGAs being used.

We suggest a simple reconfiguration algorithm that collects motion vector statistics over 10 frames and adjusts the search area to accommodate 95% of motion vectors. It is based on the assumption that the content of a picture does not change too frequently and the fact that unnecessarily larger search window is not so beneficial in picture quality. This is obviously an area for further research.

4. RECONFIGURABLE ARCHITECTURES FOR MOTION ESTIMATION

Full search block matching algorithm (1) can be implemented in hardware in several ways. We consider one of well-known architectures[3] as an example of reconfigurable motion estimation. If the index (i, j) in (1) is mapped into

hardware, then block matching is performed by sequential exploration of the search area(Fig.5(a)), while the computation of each $D(k,l)$ is performed in parallel. Fig.4 shows the practical implementation. 'AD block' computes absolute value and summation and 'R block' is composed of a mux and registers. '+ block' accumulates the results and 'M block' determines the V_{min}. Fig.5(b) is a simplified representation of Fig.4 (b). Fig.5(c) is a reconfigured architecture to support a larger search area than Fig.5(b).

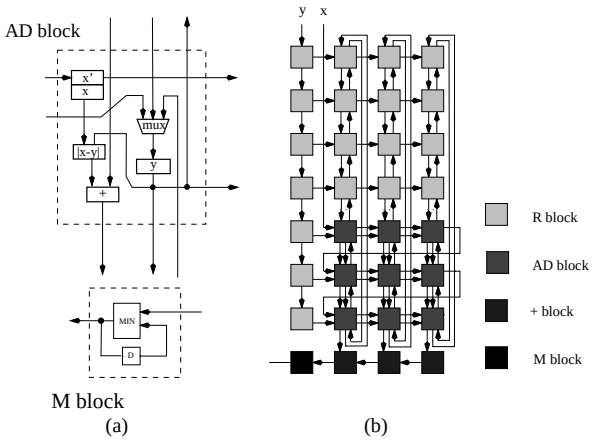

Figure 4: practical implementation[3] (a) details of AD block and M block (b) architecture ($n = 3, p = 2$)

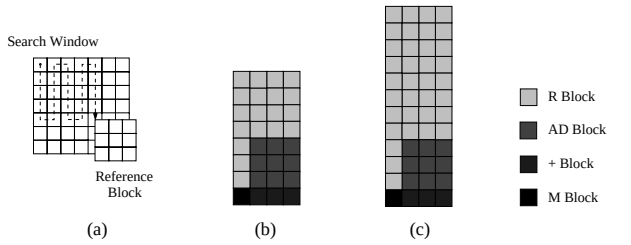

Figure 5: (a) principle of the algorithm[3] (b) simplified block diagram ($n = 3, p = 2$) (c) reconfigured architecture to support a larger search area ($n = 3, p = 4$)

4.1. Power Estimation

In this subsection we use a simple model to figure out how much power can be saved by reducing the search area. The power consumed in the motion estimation processor can be approximated as follows.

$$P_{total} = P_{comp} + P_{I/O} \tag{3}$$

where P_{comp} and $P_{I/O}$ are the power consumed by computation and I/O, respectively.

The first term, the power consumed by computation, in (3) can be assumed to be proportional to the number of operations to find a motion vector. To compute $D(k,l)$, various operations are needed: e.g., difference, absolute value, summation. We denote the power to compute a distance as $P_{distance}$. Since $D(k,l)$ should be calculated for every pixel in the current block and comparison is only needed for each candidate block in the search area, P_{comp} can be expressed as follows.

$$P_{comp} = (P_{distance} \times n^2 + P_{compare}) \times (2p+1)^2 \tag{4}$$

As we can see from (4), the power consumed by computation is reduced quadratically as p is decreased.

The second term in (3) is the number of I/O operations. Since the number of motion vectors in a frame is much less than the number of pixels in a frame, we consider only the number of pixel accesses to figure out the power consumed by I/O. If the Motion Estimation(ME) processor has a large enough on-chip memory to store the entire current and previous frame, then the memory traffic is

$$2 \times M \times N \times f_{frame} \tag{5}$$

If the ME processor has a limited amount of on-chip memory to hold one current block and one search area(or a small part of a search area) data, then it should read the current and search block data each time it computes the motion vector. In this worst case, memory traffic is given by

$$(M \times N + (2p+n)^2 \times M \times N/n^2) \times f_{frame} \tag{6}$$

where $M \times N \times f_{frame}$ and $((2p+n)^2 \times M \times N/n^2) \times f_{frame}$ correspond to memory traffic for the current and previous frame, respectively. The ratio of (6) and (5) is $0.5 \times (1 + ((2p+n)/n)^2)$. If $p = 8$ and $n = 16$, then the ratio is 2.5, i.e., it needs 2.5 times more off-chip memory accesses.

Fig.6 shows the power consumed by computation and I/O with $n = 16$ and parameter p. Each curve is normalized to 1 when $p = 8$. (The actual ratio of I/O to computation power is technology-dependent.)

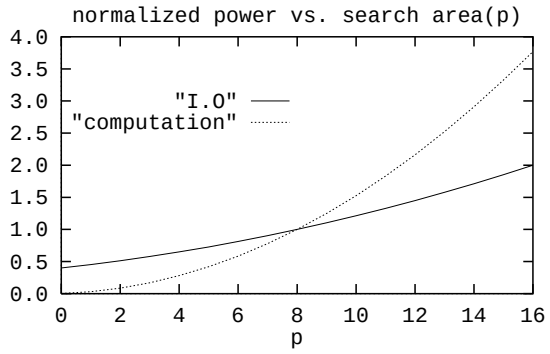

Figure 6: Power vs. Search area(p)

4.2. Resource Reuse

In principle a search area is larger than a current block. Therefore overlapped areas exist among the adjacent search areas. These overlapped areas increase the memory traffic for a previous frame as a search area increases[7]. In reconfigurable motion estimation, unused hardware resources or returning resources by reducing the search area can be

used as local memory, resulting in further power saving. Fig.7 shows the overlapped areas in adjacent search areas. Unused resources or returning resources from reducing a

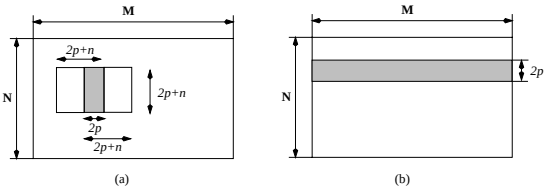

Figure 7: Overlapped area in Search window(a) horizontal overlapped area (b) vertical overlapped area

search area can be used as local memory to avoid reading the overlapped area date again.

The following table shows how much power can be saved by reducing the search area and using the returning resources as local memory.

search area(p)	P_{comp}	$P_{I/O}$	$P_{I/O}$ with local memory
8	1.00	1.00	1.00
7	0.78	0.90	0.85
6	0.59	0.81	0.71
5	0.42	0.73	0.58
4	0.28	0.65	0.50

Eq. (4) and (6) were used to estimate P_{comp} and $P_{I/O}$, respectively. Each term was normalized to 1 when $p = 8$. The picture size is 720×576(CCIR 601 format) and $n = 16$. When estimating the $P_{I/O}$ with local memory, we assumed that we build local memory only by returning resources from reducing the search area. Returning resources are first used for local memory for horizontal overlapped area(Fig.7(a)). If there are more resources than horizontal overlapped area, then the remaining resources are utilized as local memory for vertical overlapped area(Fig.7(b)). How much local memory can be built from returning resources depends on the FPGAs being used. When we estimate the $P_{I/O}$ with local memory, we used Xilinx XC6200 series and the estimation results on 2 bits RBMAD in the next subsection.

By reducing p from 8 to 4, P_{comp} reduced by 72 % and $P_{I/O}$ with local memory by 50 %. The effect of utilizing unused or returning resources as local memory is larger than one may expect. When reducing p from 8 to 5, $P_{I/O}$ reduced by 27 % but with local memory total 42 % of reduction in $P_{I/O}$ can be obtained.

4.3. Estimation of the Number of CLBs and Reconfiguration time

We estimate the number of Configurable Logic Blocks(CLBs) and reconfiguration time in Xilinx XC6200 series. To reduce the hardware complexity, We used 2 bits RBMAD matching criterion[4]. We estimate 23 CLBs are needed for 'AD block' with 2 bits in accumulation and 4 CLBs for 'R block'. If 1 bit RBMAD is used, approximately a half of above resources are needed.

As for the reconfiguration time, the XC6200 series requires 40 nsec/cell for reconfiguration[6]. When we change the search area, p, by one, about 5 usec is needed for 2 bits RBMAD for reconfiguration. This is a reasonable amount of time considering that the time interval between two successive frames is 33 msec.

5. CONCLUSION

In this paper we proposed a reconfigurable approach to real time motion estimation. Rather than proposing new architectures or new matching criteria, we focused on how motion estimation can get benefits from reconfiguration with well-known architectures and criteria. We believe that motion estimation is one of the most appropriate new applications that can get benefits from reconfiguration. By adjusting the search area according to the changing characteristic of an input signal, unnecessary computation can be avoided, hence saving power. More power can be saved by utilizing the unused resources for local memory. Furthermore the rate of computation and of reconfiguration required by the algorithm are well matched by existing FPGA technology.

Future challenges include: 1) reconfiguration to support more sophisticated motion estimation algorithms, 2) more detailed performance studies over a wider range of video sequences, 3) generalization of this concept to other algorithms and architectures, 4) modification to FPGA architectures to support the use of logic cells as local memory.

6. REFERENCES

[1] D. L. Gall, "MPEG: A video compression standard for multimedia Applications," Communications of the ACM, vol.34, pp.46-58, Apr. 1991.

[2] J. Villasenor, C. Jones, and B. Schoner, "Video communications using rapidly reconfigurable hardware," IEEE Trans. Circuits Syst. Video Technol., vol.5, pp.565-567, Dec. 1995.

[3] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI Architectures for Video Compression -A Survey," Proc. of IEEE, vol.83, pp.220-246, Feb, 1995.

[4] Y. Baek, H.-S. Oh, and H.-K. Lee, "An efficient block-matching criterion for motion estimation and its VLSI implementation", IEEE Trans. Consum. Elec. Vol.42, pp.885-892, Nov. 1996

[5] H. Gharavi and M. Mills, "Block-matching motion estimation algorithm - New results", IEEE Trans. Circuits Syst. vol.37, pp.649-651, May, 1990

[6] D. Conner, "Reconfigurable Logic: Hardware speed with Software flexibility," EDN, pp.53-64, Mar. 28, 1996

[7] F. Cattor, F. Franssen, S. Wuytack, L. Nachtergaele, and H. De Man, "Global communication and memory optimizing transformations for low power signal processing systems," Proc. IEEE workshop on VLSI signal processing, La Jolla, CA, Oct. 1994

[8] Xilinx web site, http://www.xilinx.com