Variable Block size Motion Estimator

Revision History

- 1/22/10. Major revision to memories, and how to handle special cases.

Introduction: This project aims at designing a part of Variable block size motion estimator used in Low power H.264 Video Compression Architectures. The aim of this project is to find motion vectors between two successive motion frames. Project Details and Required Definitions are provided below.

Background. The class on complexity goes through how to design a fixed block size motion estimator. It is useful to have reviewed that class before proceeding with this document. Basically, often in video the only difference between much of the content in subsequent frames is motion (imagine panning a camera slowly over a fixed scene). If this is true the difference between subsequent frames can be coded solely as a set of motion vectors, indicating the translation. This provides a very high level of compression.

You are going to implement a Variable block motion estimator, as might be implemented in a black and white 1080p HD video camera. Each full frame of video consists of 1072 x 1920 pixels (there are actually blank pixels in each row, but this approximated for this exercise.). The frame rate is 24 frames per second.

Pixel: The basic image information is called pixel. Each pixel is an 8 bit value. A group of pixels makes an image.

Macroblock: Each frame of image is divided into 16X16 pixels, called Macroblock. Thus there are 67 x 120 Macroblocks in each video frame. You have to sustain a rate of completing one Macroblock search at 192960 macroblocks/s (67x120x24).

Reference Frame: Reference Frame consists of group of pixels taken from a reference image which are compared with the pixels in the search frame. Compression of a video is done by measuring the motion of this reference frame between the current and the previous search frame. This generally has the size of a Macroblock.

Search frame: The motion of each reference frame from its position in the reference image frame to another position in the current or previous image frame is assumed to occur within a 48x48 search frame around the position of the reference frame. The basic reference frame is searched in this search frame. Search frame and Reference frame are explained with an example in the following sections. (Note: There are many reference frames and search frames in one complete 1072x1920 frame.)

There are different types of Motion Estimators. Full search Motion Estimation is the most basic method of motion estimation. Full search Motion estimation uses Block-matching technique to estimate Motion Vectors. Variable Block size Motion Estimation is developed on the concepts of Full Search Motion Estimation.

Full search method of Motion estimation: This is the basic method of estimating the motion vector. This is done the following way.
A window size of reference block size is moved over the search frame. For each window location \((i,j)\) the Sum of Absolute Difference (SAD) is determined as follows:

\[
\text{SAD}(i,j) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \text{abs}(C(k,l) - R(i+k,j+l))
\]

where \(\text{abs}\) is the absolute value[1]

Where \(C(k,l)\) is the search frame and \(R(i+k,j+l)\) is the value of window in the search frame.

The vector with the least SAD is selected as the motion vector.

Example using 4 x 4 reference frame and 8 x 8 search frame Block matching and finding motion vectors in two successive frames is explained using the following example.

The 8 x 8 search frame is shown above. The motion of Block highlighted in Blue is to be predicted in both the Search Frames. The block highlighted in red is the window size whose size is equal to reference frame size.

It is clear from the above figure, the blue block has moved from (4, 3) in Previous Search Frame to (3, 4) in Current Search Frame. This motion is estimated using block matching method. Sum of Absolute difference (SAD) between Reference frame and the block highlighted in red is calculated, it is estimated to be 32. Now this block is moved to next position and SAD is calculated. The reference frame is moved through all the blocks in the search Frame and SAD’s are calculated. SAD for block highlighted in blue is estimated to 0, which is minimum. Now the Reference frame is moved over current Search Frame to find minimum SAD. Minimum SAD is obtained with co-ordinates (4, 3) on Previous Search Frame and with Current Search Frame Minimum SAD is obtained at co-ordinates (3, 4). So we can conclude that the block has moved from position (4, 3) to (3, 4).

The above mentioned method cannot detect internal motion in the 16X16 reference block and is not very accurate, so a new technique called variable block size motion estimator (VBSME) is used. In this project
a part of VBSME is implemented. Motion vectors and Best Partition are calculated using two reference frames as explained below.

**Variable block size Motion Estimation (VBSME)**

The basic idea in VBSME is dividing the 16X16 Window size into 41 smaller blocks[2].

The blocks are divided as the way explained below.

The big 16X16 block is divided into sixteen 4X4 blocks, eight 8X4 blocks, eight 4X8 blocks, four 8X8 blocks, two 8X16 blocks, two 16X8 blocks and one 16X16 block. Total number of blocks is 41. Same convention of numbering the blocks is to be maintained.

While calculating the SAD of the Motion Block (16X16), the SADs of 16 4X4 pixels are calculated and these SADs are reused in calculating SAD’s of other blocks. For example SAD of the 24th subblock of size 4X8 is obtained by adding SAD’s of sub blocks 0 and 4 4X4 blocks. Similarly all the SAD’s are calculated. In the next topic the basic VBSME is explained which is to be implemented for the project.

The High level block diagram of the Variable Block size Motion Estimator is given below. Each block is explained below.[1]
Each Block is explained in detail below:

**Current MacroBlock Memory:** The current MacroBlock Memory will have the pixel information of Reference window size 16X16. Current memory has width x length as 256x8. This memory has 1 read port.

**Search Frame memory:** Search frame memory is 48x48 pixels memory. In the project one search frame memory is provided. Search Frame memory has width x length as 2304x8. The memory has 2 read ports. Memory is arranged the following manner.

An example shows how 3X3 memory is arranged. Please Extend this to a 48X48 memory.
<table>
<thead>
<tr>
<th>Memory size</th>
<th>3x3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
</tr>
</tbody>
</table>

Where A1 represents first pixel at (0,0) location, B1 represents pixel at (1,0) location.

**Processing Units:** The processing unit is responsible in finding the absolute difference between the pixel in current MacroBlock memory and search Frame MacroBlock memory.

**Adder Tree:** Adder tree will group the SAD values obtained from the 16 4X4 blocks to form different SAD’s of other blocks. For example it combines two 4X4 blocks SAD to get SAD of 4X8 or 8X4 block. The output of Adder tree block will be 41 SAD’s.

**Comparator Unit:** This unit compares the 41 SAD from the adder tree with the previously generated 41 SAD’s. The output of the Comparator unit will be best 41 SAD’s for the whole search area frame.

**Decision Unit:** The decision unit decides the best partition and the Motion Vector from the 41 best SAD’s obtained from the Comparator based on the threshold levels decided. The Threshold level SAD for 4X4 block is 8. Threshold level SAD for 4X8 and 8X4 block is 16. Threshold level for SAD for 8X8 block is 32. Similarly Threshold level SAD for 8X16, 16X8 blocks is 64. Threshold level for SAD for 16X16 block is 128.

Out of the 41 SAD values, all the blocks with SAD value less than threshold are selected as explained below.

**Best Partition:** Best partition is the sub block which satisfies the threshold conditions. For example if the threshold value of sub block of size 16X16 is less than the threshold value then we consider 16X16 as the best partition and the corresponding co ordinates as motion vector, though the other sub blocks 4 x 4, 8 x 4, 4 x 8, 8 x 8, 8 x 16 and 16 x 8 have SADs satisfy or dissatisfy threshold conditions.

Consider another example; the 8X8 partition is made up of two 4X8 partitions or 8X4 partitions.

If the SAD value of 8X8 partition is 30 which is less than set threshold, and the SAD value of 4X8 is 17 and other 4X8 has SAD of 10. Though one of the 4X8 is failing the threshold conditions, since 8X8 is less than the threshold, the partition 8X8 is considered. The value of co-ordinates representing 8X8 sub block is considered as motion vector.

Consider another example, this example selects 8x8 partition as best partition, though the sub-blocks don’t satisfy threshold conditions. (an 8x8 partition is made of four 4x4 sub blocks).
SUMMARY of Memory Units:

<table>
<thead>
<tr>
<th>Memory</th>
<th>Ports</th>
<th>Width x Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current MacroBlock</td>
<td>2 Read</td>
<td>256X8bits</td>
</tr>
<tr>
<td>Search Frame</td>
<td>2 Read</td>
<td>2304x8bits</td>
</tr>
<tr>
<td>Working Memory</td>
<td>1 Read, 1 Write</td>
<td>Designer can decide</td>
</tr>
<tr>
<td>Output Memory</td>
<td>1 Read, 1 Write</td>
<td>32 x 24bits</td>
</tr>
</tbody>
</table>

Current Macro Block and Search Memory are the input memory files. Please use tutorial 3 to generate your memory units. For this project, you are required to find out the best partitions and their absolute position vectors in a single current search frame only. In reality, there is an extra subtractor unit which calculates the relative motion of the macroblocks between the current search frame and the previous search frame.

Outputs are represented the following way in output Memory file:

After finding the best partition and motion vectors, outputs are written into output memory.

All entries in the output memory are initialized to 0’s. Each entry is a 24 bit representation: the first 8 bits represents partition number (value between 0 and 40); the next 8 bits the first coordinate and the last 8 bits represent the other coordinate. For example if the best partition is 35 and the motion vector corresponding to that is (4, 6), it is represented in the output memory as 24’h230406. The output memory should contain one entry per reference block in row-major order. If there is no motion vector suitable, put (0,0) in the memory.

**Turn In Requirements**

You will be required to turn in you Verilog code and Synopsys scripts. Your Verilog code must be in two components and must use the following names:

**design.v** // contains the synthesizable design. The top module MUST be called top and MUST appear first in design
The script must be called script.tcl

**Design Objectives**

Design a Motion Estimator capable of processing 1080p video (1920 x 1080 pixels) in real time (24 frames per second). We will provide you with sample frame sequences to work on. Your design goal is to achieve this throughput at the minimum total average energy per frame. Power is to be measured for both your logic and ALL memories.

**References:**