Project Spec. v 1.0

# EECS150 Final Project, Fall 2003: Block Motion Estimator

R.S. Fearing
Dept. of Electrical Engineering and Computer Sciences
Univ. of California, Berkeley


## 1 Objectives

The final project for this semester is to design a video block motion estimator. The core system will continuously capture data from a video camera, store it in a frame buffer, and display it on a video monitor using NTSC video standards. In parallel, a block motion estimation engine will calculate velocity vectors for each $8 \times 8$ block of the previous frame, and overlay these motion vectors on to the video image. All video capture and processing will be done on the gray scale image, with motion vectors displayed in color.

### 1.1 General Philosophy

This document describes the input/output specification for the optical flow estimator, and gives the schedule for completing requirements for this project. As in the real world, the user/external interface is specified; it is up to you to specify most of what goes into the system. You can consider the course staff as being the customers for your project, and you have contracted to deliver a working system. We have checkpoints and demonstrations along the way to see that our contract will be satisfied. Our contract also has a clause that you won't get paid (i.e. credit) if you don't deliver the working pieces on the specified dates. Exceptions can only be made for serious medical problems. Now you may want to deliver extra features, which is fine, but the basic system needs to be working first. As customers, we aren't going to pay for simulations, we need to see the real thing.

### 1.2 General Tips

Just because a design works does not make it a good design. Use good design practices such as:

- top-down design, with data path/control unit decomposition
- design, simulate, and debug in modules
- use synchronous design methodology (e.g. use only synchronous reset parts). (One global asynchronous reset for the whole design using the STARTUP block is fine). Use BUFG for a single global clock. Pay attention to timing for portions of the system running at more than 25 MHz, (use Xilinx timing analyzer tool to check for worst case path).
- divide work up with your partner for individual modules (but make sure you have agreed on the interface!)

- document your design as you go, using detailed block diagrams, ASMD diagrams, and desired functional timing diagrams

Budget plenty of time to get your project done. With this level of complexity, it takes at least three times as long to debug a systems as it does to design and enter it. In fact, however long you think it will take to complete a task, it will take 3 times longer. Plan accordingly. Also, Calinx boards are limited, and workstations will need to be shared. Plan accordingly. Simulate on stations without a Calinx board. It will be difficult to simulate the entire project due to the large number of cycles.

## 2 Project Description

A top level block diagram of the project is shown in Figure 1. Other configurations may be possible, but given the routing and memory resources available, this configuration is sensible. Check with your TA about major departures from this suggested design.

There are five main blocks in the system:

1. The video input interface converts the analog data from a video camera into 8 bits of digital data (one byte per pixel) using an ADV7185 chip and a video decoding block. The video decoding subsystem determines horizontal and vertical synchronization, and provides input frames to the video frame buffer. We will use a video frame size of 240 rows by 320 columns of 8 bit grayscale data. To avoid interlace confusion with the motion estimation, We will use only the even field of video, which is updated at 30 Hz. This block will be built and debugged in checkpoint 2.

2. Memory interface. The SDRAM will be used to store current (frame $n$) and previous frame (frame $n-1$). As the camera is asynchronous with respect to display and motion calculations, you may want to have a frame $n+1$ which will be filled while comparing frames $n$ and $n-1$. For convenience, you probably want columns in the SDRAM to correspond to columns in the image, so each frame would use $256 \times 512 \times 8$ bits of memory. To increase bandwidth you may want to use 4 pixels of video for each 32 bit word, so a frame would take $256 \times 128 \times 32$ bits of memory. This module is substantially built in checkpoint 1.

3. The video output interface continuously displays the contents of the most recent frame in the frame buffer on a video monitor using NTSC standard video generated by the ADV7194 chip. The basic video output will be built and debugged in checkpoint 2. A color overlay of motion estimate vectors will be superimposed on the image data to visually show correct operation of the motion estimator.

4. The Block Motion Estimator Engine will continually compare $8 \times 8$ blocks from the previous frame with a $16 \times 16$ block from the current frame and estimate the best match using a Minimum Absolute Error metric. The relative offset of the best match location within the $16 \times 16$ block for the $8 \times 8$ block from the previous frame corresponds to the block motion estimate of $(\Delta row, \Delta column) = (\pm 4, \pm 4)$. If no adequate match is found, a VOID output will be returned. One of the biggest challenges of the project is fitting this computation in to the limited memory and bandwidth of the Xilinx FPGA.

5. The controller module is responsible for generating all clock, timing, user input/output, and control signals. It is probably better to have a number of simple, modular FSMs controlling various functions, than one huge FSM controlling everything.

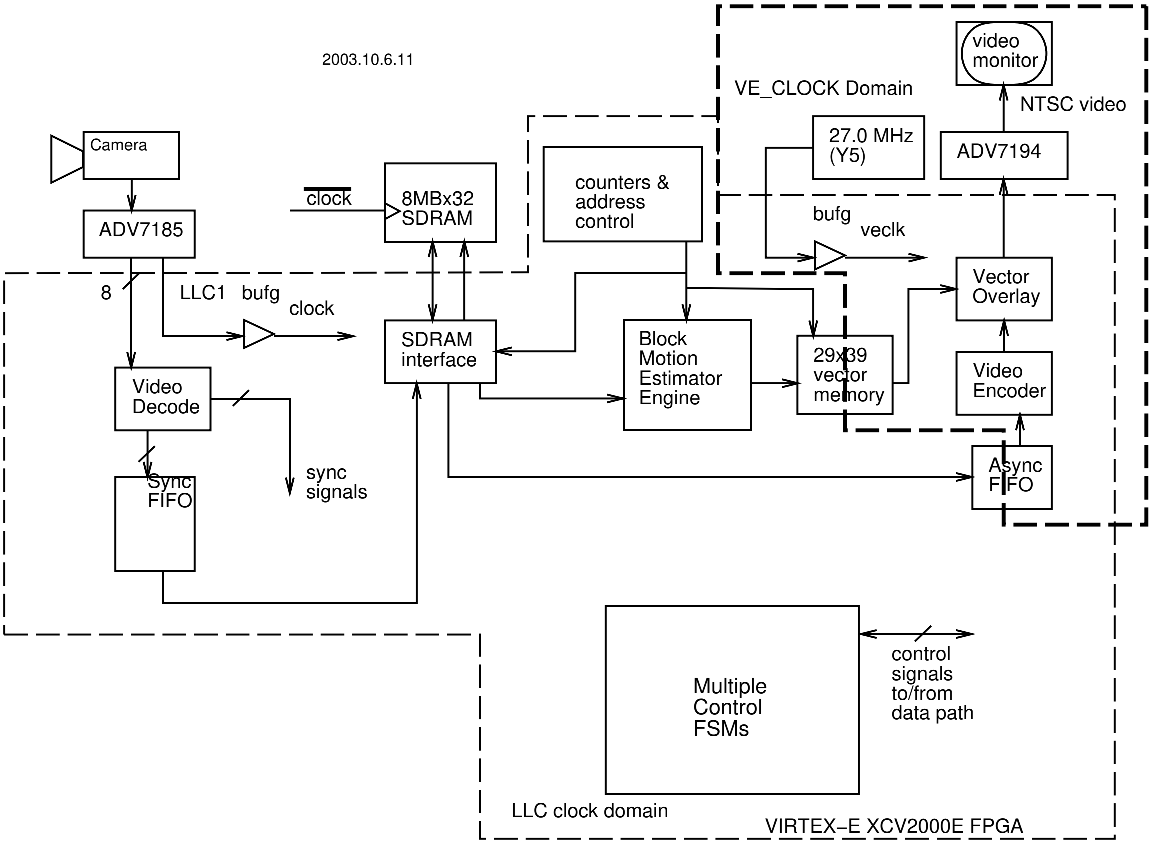


Figure 1: Top Level Block Diagram.

### 2.1 Operation Mode 1: Video Frame Capture

In mode 1, the system operates as a frame grabber. This is a basic operational check mode. First an image is captured in 1/60 of a second (one *field* time) and stored in an SDRAM frame buffer. Complete frames are read from the frame buffer and displayed. Images should be clean with no jumpiness or glitches.

### 2.2 Operation Mode 2: Block Motion Estimation

In mode 2, the system continually captures video data, estimates the block motion, and displays motion vectors overlayed on the video. For debug purposes (and checkpoint 5), you may want to have a mode where only two frames are captured, and you continually display the motion estimation just for those two frames.

## 3 Block Motion Estimation

The block motion estimation algorithm is used in the MPEG video compression scheme to allow sending predicted motions of blocks, rather than resending a complete updated image. See reference [ ] for more detail.

$$MAE = \min w.r.t(l,m) \quad \Sigma_{i,j}|I_{n-1}(i,j) - I_n(i+l,j+m)| \tag{1}$$

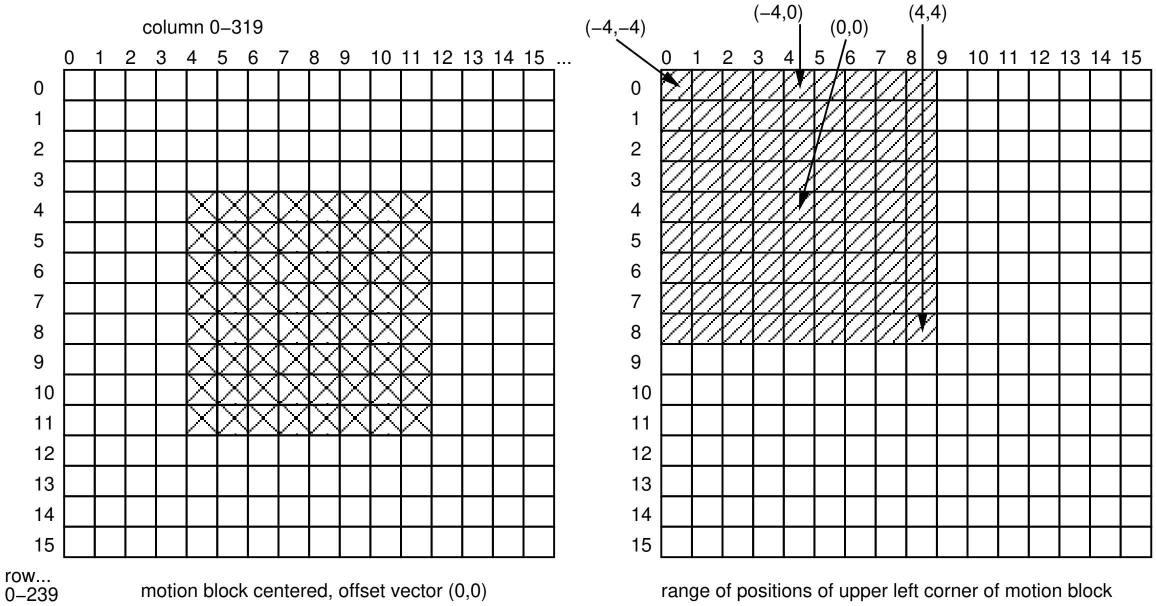


Figure 2: Matching of $8 \times 8$ block within $16 \times 16$ neighborhood.

with $-4 \leq l \leq 4$ and $-4 \leq m \leq 4$, and where $I_n$ is the current image block, and $I_{n-1}$ is an $8 \times 8$ block from the previous frame.

# 4 Project Deadlines

This section summarizes the functionality you need to demonstrate, as well as the steps in the design you need to follow. The number in parentheses is points out of 100 total for the project. Design Review assignments will only be checked off during your lab section during scheduled meeting times. (Both partners need to show up to get individual Design Review credit).

For checkpoints 2-5, there will be a formal design review with your TA during your scheduled lab section. You will need drawings and printouts on paper to hand in as part of a brief presentation on your progress and plans so far. Points are maximum, and will be based on completeness and quality of presentation as well as soundness of reasoning.

See weekly handouts on checkpoints for more details.

## 4.1 Checkpoint 1 Part 1. Due in lab week of 6 Oct.

- (2) Simulation of SDRAM read and write tests.

## 4.2 Checkpoint 1 Part 2. Due in lab week of 13 Oct.

- (4) Demonstrate SDRAM operation on logic analyzer, data retention.

## 4.3 Checkpoint 2. Due in lab week of 20 Oct.

Design Review:

- (2) Detailed block diagram of project data path (to level of Xilinx library components, counters, comparators, registers, etc).

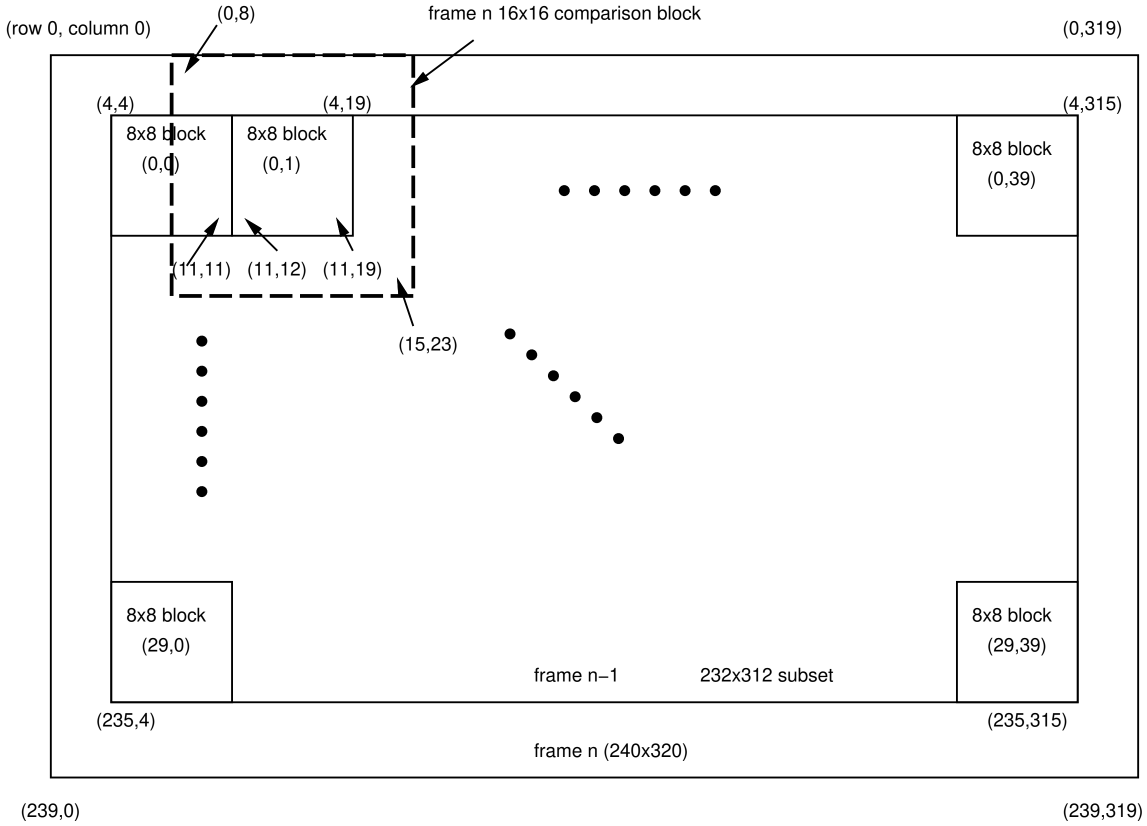


Figure 3: Decomposing 232 by 312 frame into 29 by 39 $8 \times 8$ contiguous blocks.

- (2) ASMD diagram for controller (initial attempt)
- (1) Project plan: number of CLBs, block select+RAM, cycles

Lab:

- (4) Demonstrate single line video decode/video encode using FIFO.

### 4.4 Checkpoint 3. Due in lab week of 27 Oct.

Design Review:

- (3) ASMD diagram for controller for project, complete. High level language block motion estimation code verification.

Lab:

- (5) Demonstrate capture and display 240x320 frame of video.

### 4.5 Checkpoint 4. Due in lab week of 3 Nov.

Design Review:

- (2) revised block diagram, functional timing diagram or RTL, ASMD diagrams, and Verilog code for block motion estimation engine

Lab:

- (4) Demonstrate dummy vector overlay on captured video frame.

### 4.6 Checkpoint 5. Due in lab week of 10 Nov.

Design Review:

- (2) Complete ASMD diagrams for project, Verilog for data path and controller. Simulation output for controller.

Lab:

- (5) Demonstrate static block motion estimation on 2 consecutive frames, showing motion vectors on video display.

### 4.7 Lab week of 17 Nov.

Lab:

- Early Completion Bonus. +5 points if you demo the complete basic project to your TA during scheduled slots on Thurs. Nov. 20.

### 4.8 Project Week 11/24, 11/25, 11/26

- (40) Demo completed project. Sign up for demo slots with your TA during your scheduled lab section or on Wed. 11/26.
- (10) Hand in printout of Verilog code along with project demo.

### 4.9 Project Week 12/1

- (14) Final Report is due Friday 12/5/03 in lab.

### Acknowledgements

Thanks to Alex Krasnov and Greg Gibeling for project design and verification.