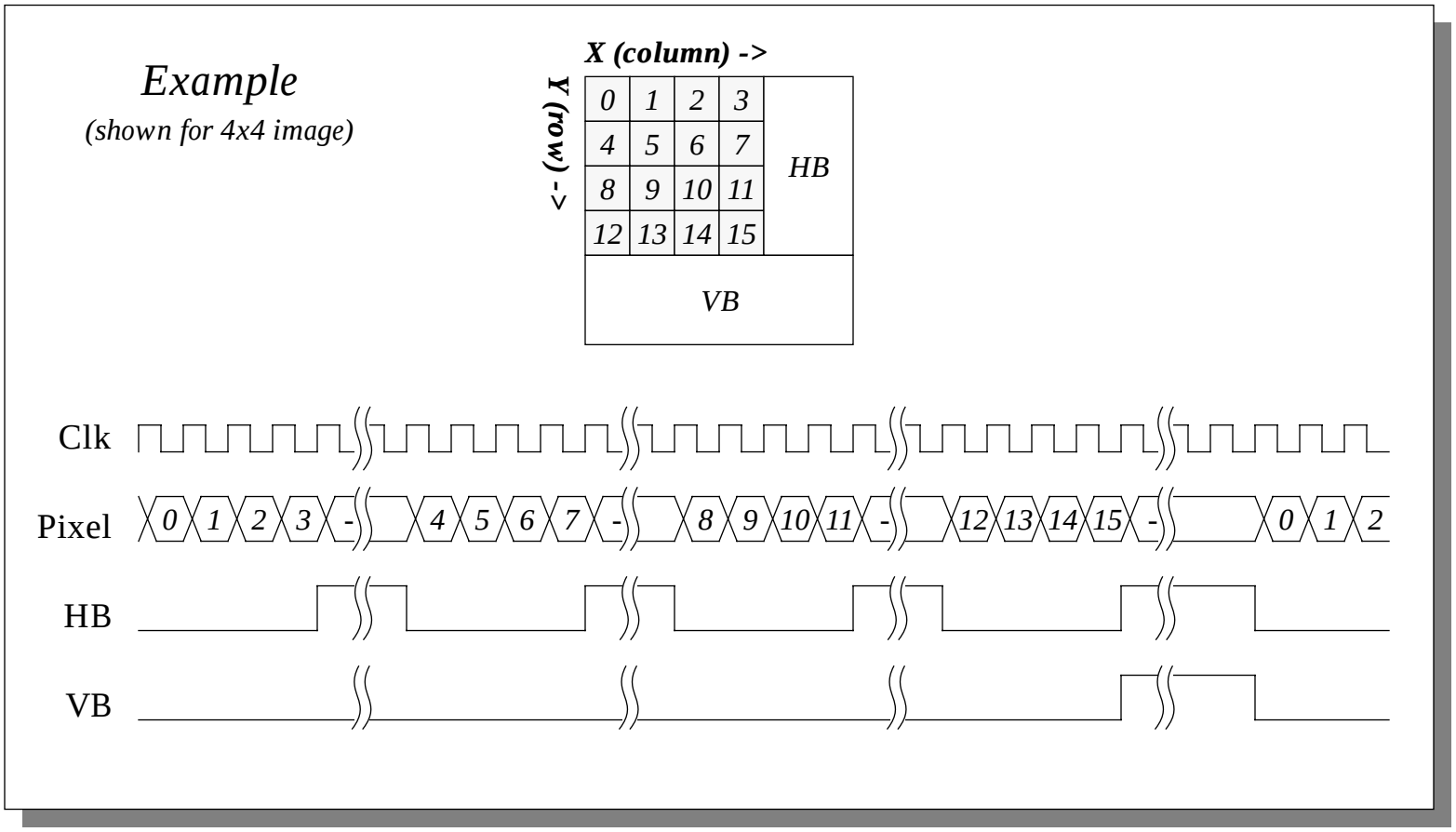
FINAL PROJECT

Due Date May 5, 1998

FINAL PROJECT

THE PROBLEM

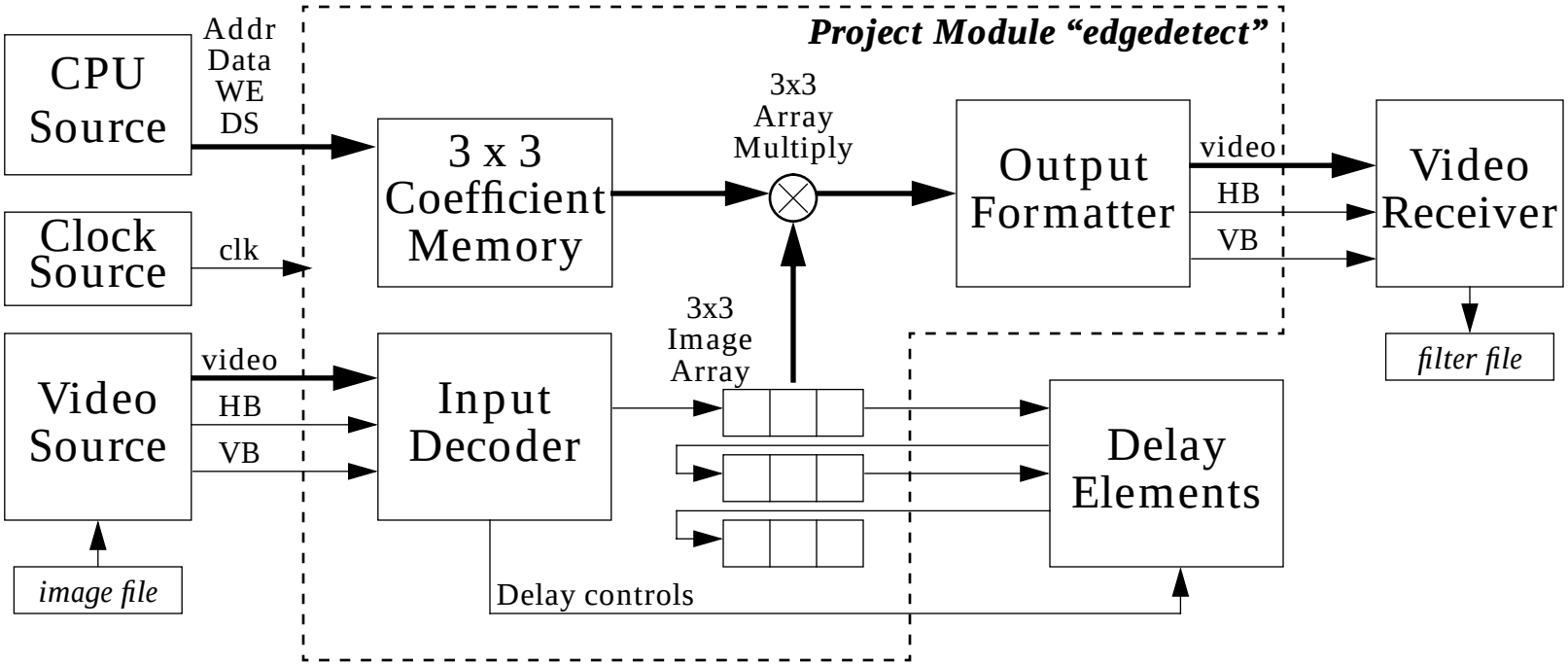
A two-dimensional video image is transferred between processing modules by passing one pixel at a time qualified by horizontal and vertical blank pulses.



FINAL PROJECT

THE PROBLEM (CONTINUED)

- Write a synthesizable VHDL module that accepts the incoming video stream and applies a 3x3 filter (convolution) to the input data. (*inside the dotted box*)
- Write Stimulus and Response module(s) that uses supplied library functions (*see following*) to read an image file and generate a properly formatted video stream and accepts a video stream and writes the resultant image to a file. (*everything else*)

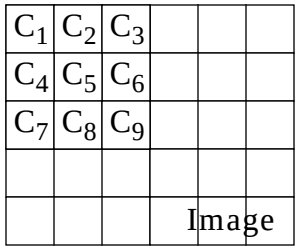


- Write a Testbench Module that instantiates the modules above and allows simulation.

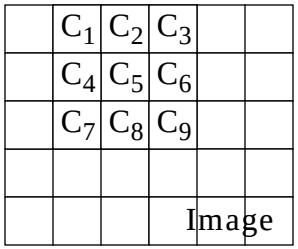
FINAL PROJECT

THE PROBLEM (CONTINUED)

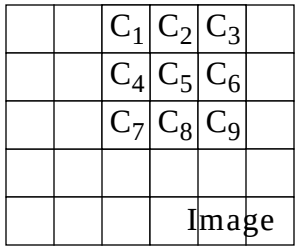
The convolution filter algorithm is shown below.



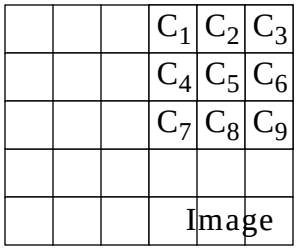
Clock Cycle T



Clock Cycle T+1



Clock Cycle T+2



Clock Cycle T+3

Image Neighborhood

P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9

Coefficient Array

C_1	C_2	C_3
C_4	C_5	C_6
C_7	C_8	C_9

Resultant Filtered Pixel

R

$$R = \sum_{n=1}^{9} P_n * C_n$$

The Image Neighborhood is the 3x3 set of pixels overlapped by the Coefficient Array.

The Row, Column coordinate of the resultant output pixel is defined as the Row, Column coordinate of the Center Pixel of the Pixel Neighborhood.

Every clock cycle, data will shift through the delay pipeline, effectively moving the coefficient window over each point in the image

FINAL PROJECT
WHAT WE NEED TO DO....

Implement the following functions in the **Synthesizable Edge Detector module**

- Input Decoder / Delay Element Controller
  - Accepts an 8-bit video stream, Horizontal and Vertical Blanks, and generates Push and Pop commands for FIFO Delay Elements (line delays) for the 3x3 Image Array.
- Coefficient Memory
  - CPU Programmable (Read and Write) 3x3 Coefficient Array, beginning at Address 0000*h*.
- 3x3 Multiplier Array
  - Performs 9 simultaneous multiplications; each position in the 3x3 image array is multiplied by the corresponding position in the 3x3 coefficient array, the 9 values are summed, and the sum is scaled to 8 bits.
- Output Formatter
  - Sequences out the filtered 8-bit video stream with accompanying Horizontal and Vertical Blank.

Implement the following functions in the **Testbench Stimulus module**

- CPU Source
  - Provides CPU Address, Data, Write Enable, and Data Strobe controls to the Coefficient Memory Array.
- Clock Source
  - Generates a 1 MHz clock.
- Video Source
  - Reads an image file via the supplied procedure *ReadImage* and generates an 8-bit video stream with accompanying horizontal and vertical blanks.
- Video Formatter
  - Accepts an 8-bit video stream with accompanying horizontal and vertical blanks and writes the output image file via the

FINAL PROJECT
WHAT WE NEED TO DO.... (CONTINUED)

- Supporting Files available for your use can be found in the directory */home/clark/project* and should be copied to your working source code directory and compiled along with your own code. Refer to the package files for the parameter lists associated with the various procedures.
  - The package file **std_logic_conv.vhd** contains a 32-bit integer conversion function *conv_integer* needed by **imagepack.vhd**
  - The package file **imagepack.vhd** contains the procedures *ReadImage* and *WriteImage* along with associated data types.
  - the binary sample image file **tiger.pnm** should be used as the input file to your stimulus module.
  - The ASCII sample image file **tiger.pgm** is the same image as *tiger.pnm* and can be useful when debugging the data stream. Alternatively, it can be read in your Stimulus module via the procedure *ReadAsciiImage* instead.

What is due at the deadline, May 5th, 1998

- Turn in the **edgedetect module code** (hardcopy and e-mail)
- Turn in the **driver/receiver module(s) code** (hardcopy and e-mail)
- Turn in the **testbench structural architecture module code** (hardcopy and e-mail)
- Turn in **Leapfrog simulation waveforms** sufficient to demonstrate proper operation of all aspects of the edge-detection module.

  That is, you need not submit traces for the entire simulation run (though you may if you chose), but there must be sufficient information in what you do submit to show that each of the functions is operating properly (e.g., CPU interface, filter algorithm, delay line, output formatting, etc.)
- Turn in the **synthesis report file(s)** generated by Warp, showing the final fit equations.
- Turn in a **written project report** explaining the implementation, any problems encountered, and any comments or explanations you would like to have considered when grading your work.

FINAL PROJECT

ILLUSTRATION

Original Unfiltered image

tiger.pnm

Sample Ideal Filtered Image[1]

(created with "*xv*")

The sample image and ideal filtered result can be viewed with "*xv*" on the Sun workstations.

1. Filtered image was created using "*xv*" for illustration only, Actual filtered image will undoubtedly vary from this ideal.

FINAL PROJECT
INTERFACE SPECIFICATIONS

- All external interfaces shall be *std_logic* or *std_logic_vector*
- The Edge Detection Module shall contain an external active-low reset (***resetLowIn***), and an external 1 MHz clock (***clkIn***)
- The Edge Detection Module shall contain a CPU programming interface consisting of:
  - a 12-bit address bus (***cpuAddrIn***)
  - a bidirectional 8-bit data bus (***cpuDataBi***)
  - a 1-bit active-low Write Enable (***cpuWriteEnLowIn***)
  - a 1-bit active-low Data Strobe (***cpuDataStrobeLowIn***)
- The Stimulus/Response Module(s) shall contain two Delay Element interfaces consisting of:
  - two 8-bit output data buses (***line1Out*** and ***line2Out***)
  - two 8-bit input data buses (***lineDelay1In*** and ***lineDelay2In***)
  - two 1-bit active-low Push and Pop Enables (***line1PushOut, line1PopOut, line2PushOut, line2PopOut,***)
- The Module shall accept 8-bit video data (***vidDataIn***), an active-high horizontal blank (***horizBlankIn***), and an active-high vertical blank (***vertBlankIn***)
- The Module shall generate 8-bit filtered data (***filterDataOut***), an active-high horizontal blank (***horizBlankOut***), and an active-high vertical blank (***vertBlankOut***)

FINAL PROJECT
FUNCTIONAL SPECIFICATIONS

All Students are responsible for the following:

- Input image size shall be a maximum of 128 x 128 pixels.
- The Edge Detect Module shall contain a CPU programmable Coefficient Array.
- The Testbench shall contain an instantiation of the Design Under Test (the Edge Detector module) and the Stimulus and Response module(s).
- The Stimulus and Response module(s) shall implement a CPU Source, a Clock Source, a Video Source, and a Video Formatter.
- The device to target in synthesis is the C375I.

***ECE496B** students are also responsible for the following:*

- The design of the Edge Detect module need not fit within the specified device, but must pass the analysis stage.

***ECE596B** students are also responsible for the following:*

- The design of the Edge Detect module must fit within the device(s) selected. That is, it may be necessary to partition the module into several smaller modules in order to synthesize and fit within the device(s).
- The Response module shall be self-checking, capable of determining the correctness of the filtered result, both in value and in pixel calculation sequence.

FINAL PROJECT

SAMPLE CPU INTERFACE TIMING