# Hardware-based Image Retrieval and Classifier System

Jason Isaacs, Joe Petrone,
Geoffrey Wall, Faizal Iqbal,
Xiuwen Liu, and Simon Foo

Department of Electrical and Computer Engineering
Florida A&M - Florida State University College of Engineering

# Project Goal

- To develop and implement a standalone web-mining image classifier system using a Xilinx Virtex II Pro based PCI development card.

# Design Issues

- Develop an embedded Linux web-miner for the Power PC
- Design software for a PC-FPGA interface through PCI to allow fast data transfer.
- Code (VHDL) PCI bridge, SRAM controller, and Texture Analysis IP for development board.

# Outline

I. Multimedia Web-Mining
II. Texture Classification
III. Hardware Implementation
IV. Conclusions
V. Future Work

# Multimedia Web-Mining

•Images are an important class of data.
•The Web is presently regarded as the largest global multimedia data repository, encompassing different types of images in addition to other multimedia data types.
•To search the web for images, a crawler (also called a spider, mobile agent, or bot) is utilized.

src="home_page/images/**rover_spin.jpg**" alt=""
width="124" height="70"></a><a
href="images/home_page/pgt_in_use.jpg"><img
src="images/home_page/**pgt_in_use_small.jpg**"

•The agent searches the HTML documents for strings of type jpg, gif, tif, etc., stores the image, url, and texture class utilizing hardware-implemented texture analysis.

# Texture Classification Algorithm

- The Classification Problem
- Spectral Histogram Image Representation
- Classifier
- Classification Results (Software)

## The classification problem

- Have 40 texture classes stored in 32x32 gray scale images with 256 shades of gray (8 bits per pixel).
- ½ Image database used for training, ½ for testing.
- Textures classes and labels taken from http://www.Dbv.Cs.Unibonn.De/image/texture.Tar.Gz.
- Would like a classifier to be fast and accurate.

# The Spectral Histogram Representation

- Properties
  1. A spectral histogram is translation invariant.
  2. A spectral histogram is a nonlinear operator.
  3. With sufficient filters, a spectral histogram can uniquely represent any image up to a translation.
  4. All the images sharing a spectral histogram define an equivalence class.

- Preprocessing step in texture classification
  1. Choose *N* image filter kernels to convolve with the image.
  2. Perform the convolutions, generating *n* resultant responses.
  3. For each response, generate a response image histogram.
  4. Concatenate each of the histograms and send to the classifier.

# The Spectral Histogram Representation

- 1st step – choose *N* image filter kernels to convolve with the image.
  - Filter kernels chosen carefully from several image filter banks including intensity: *δ(x,y),* differencing or gradient filters, laplacian of gaussian filters:

$$LoG(x, y|T) = \left(x^2 + y^2 - T^2\right) e^{(-x^2+y^2)/T^2}$$

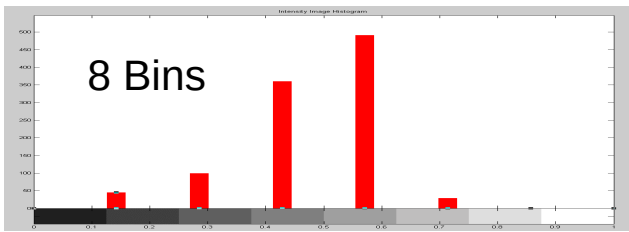
  - Where *t* determines the scale of the filter, and finally the gabor filter defined by sine and cosine components:

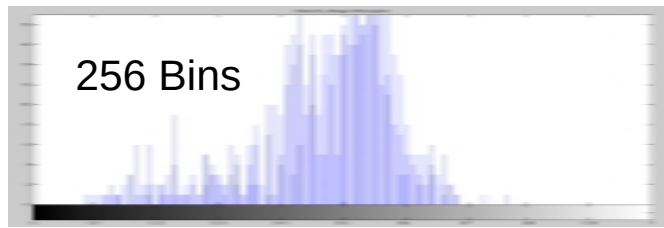
$$Gabor(x, y|T, \theta) = e^{-1/2T^2(4(x\cos\theta+y\sin\theta)^2+(-x\sin\theta+y\cos\theta)^2)} e^{-i2\pi/T(x\cos\theta+y\sin\theta)}$$

- 2nd step – perform the convolutions, generating *n* resultant responses.
  - To calculate each response pixel value, roughly *m x n* multiplies and adds must be performed, where m x n is the dimension of chosen kernel.
  - Thus a total of 32 * 32 * *n* * $m_k$ * $n_k$ multiplies and adds must be performed, where subscript *k* implies the $k^{th}$ filter.

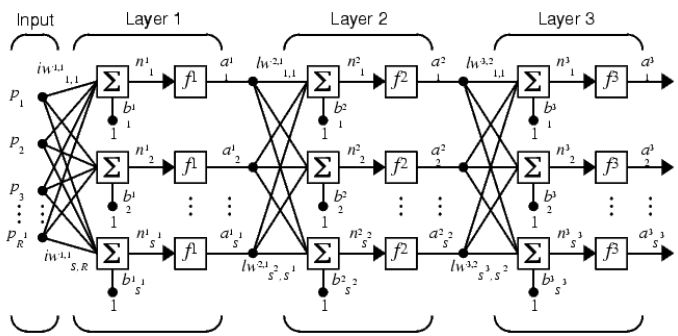
# The Spectral Histogram Representation

- 3rd step – for each response, generate a response image histogram.
  - A number of bins must be chosen for this step.
  - This determines the resolution of the response histogram.
  - The more bins, the better the resolution.



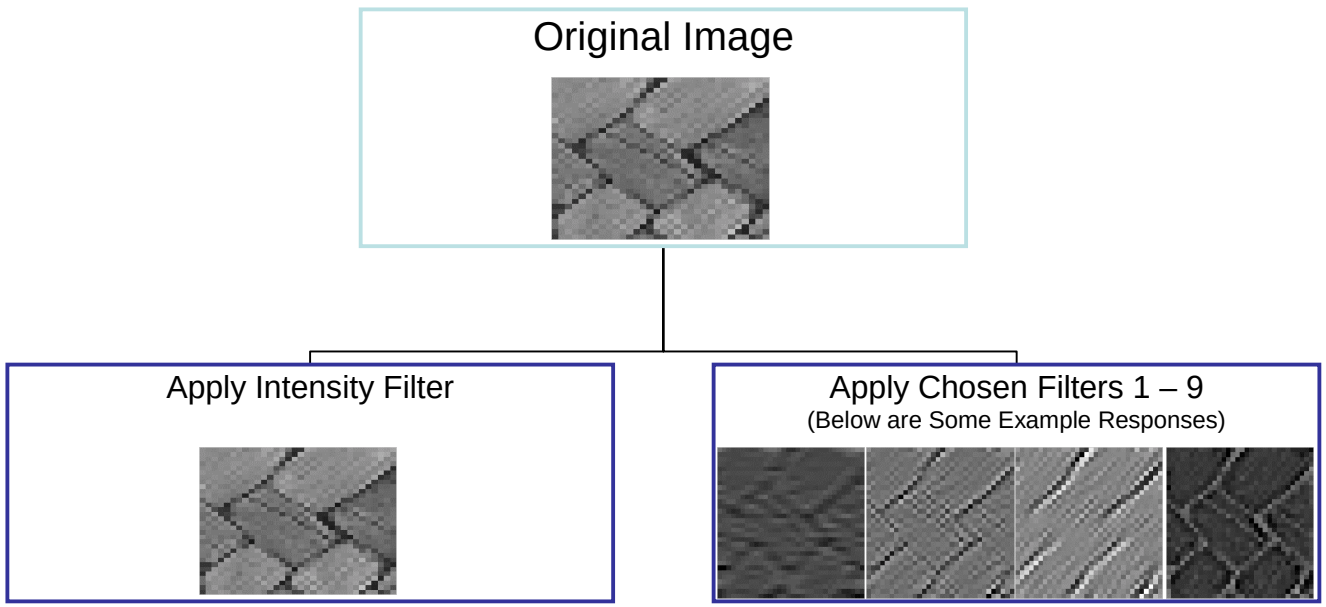




- 4th step – concatenate each of the histograms and send to the classifier
  - Histograms are concatenated with each other to form a new histogram vector.
  - This vector is then sent to the classifier (multilayer perceptron, k nearest neighbor, or other similar trained classifier).

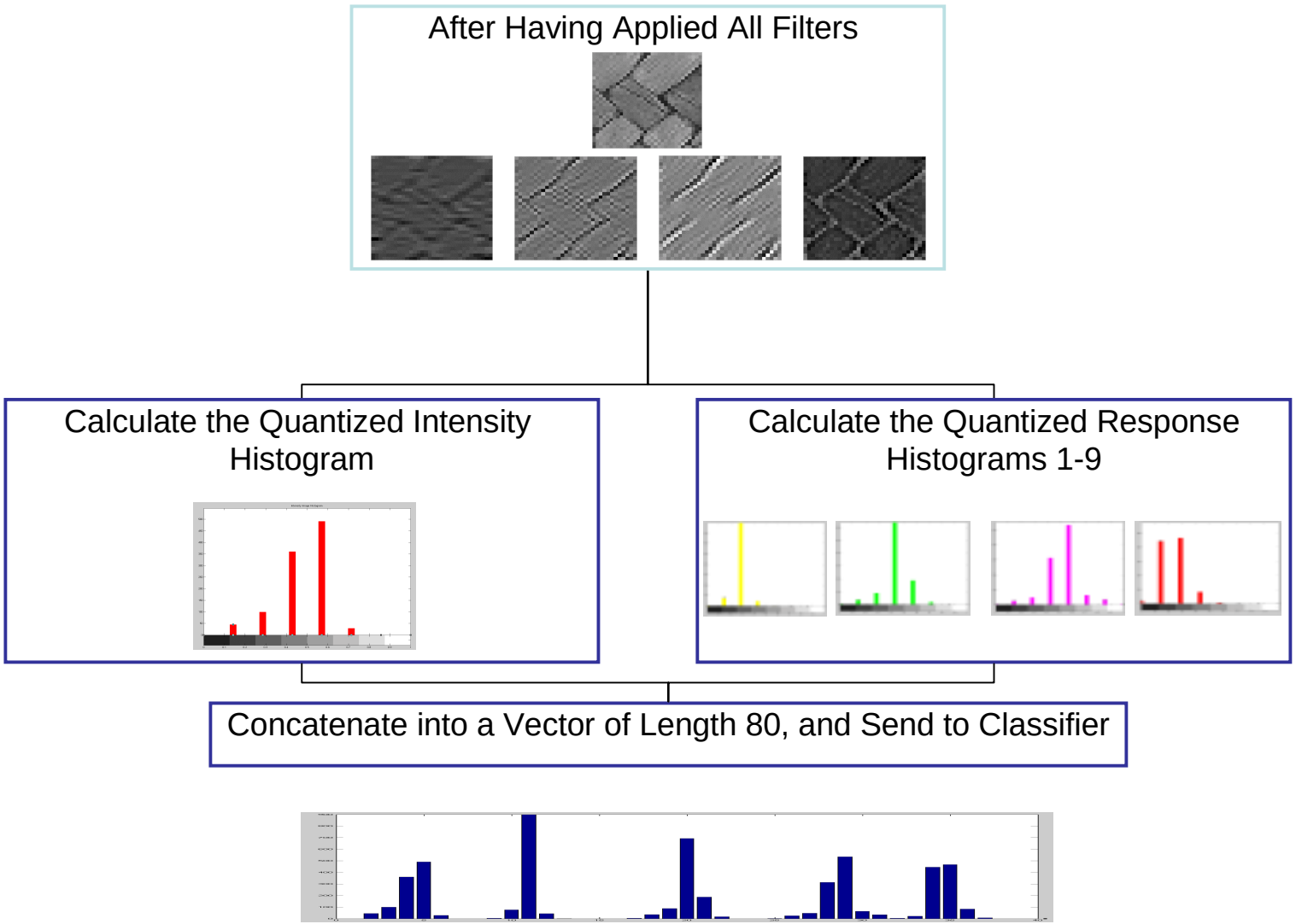


# Example Spectral Histogram Representation

- Suppose we choose 10 filters
  - Choose that each image histogram will have 8 bins.
  - Let the first filter be a simple intensity filter (thus the first response is simply the original image).
  - Calculate the original image's histogram.
  - Filter the image with the remaining 9 filter kernels.
  - Calculate each filter response's histogram.
  - Concatenate all the histogram vectors – will result in a pattern vector with 80 elements.
  - We have reduced the input pattern size from 32 x 32 (for a given texture image) bytes to 100 bytes (10 bits per bin) - this is a 10:1 reduction



# Example Spectral Histogram Representation



# Classifier

- Multilayer perceptron, nearest neighbor
  - Various sizes of input and hidden neurons were tested.
  - The number of bins chosen for histogram training altered the number of the inputs to the network (less bins = less inputs).
  - The number of hidden neurons was altered to try and find the best classification result.
  - The network was trained stochastically with over 1000 input patterns from 40 classes, to an error threshold of 0.001 average error, or until 5000 epochs had elapsed.

- Classification results
  - With 20 Hidden Neurons
    - 4 Bins/Response : 86.5625% Correct Recognition
    - 8 Bins/Response : 95.9375% Correct Recognition
    - 16 Bins/Response : 98.8281% Correct Recognition
  - With 30 Hidden Neurons
    - 4 Bins: 92.2656%
    - 8 Bins: 98.4375%
    - 16 Bins: 99.2188%
  - With 40 Hidden Neurons
    - 4 Bins: 93.8281%
    - 8 Bins: 98.5156%
    - 16 Bins: 99.6094%
  - With 50 Hidden Neurons
    - 4 Bins: 95.8594%
    - 8 Bins: 98.0469%
    - 16 Bins: 99.2188%

**Nearest Neighbor Performance**

- 4 Bins: 96.2500%
- 8 Bins: 96.9531%
- 16 Bins: 97.9688%

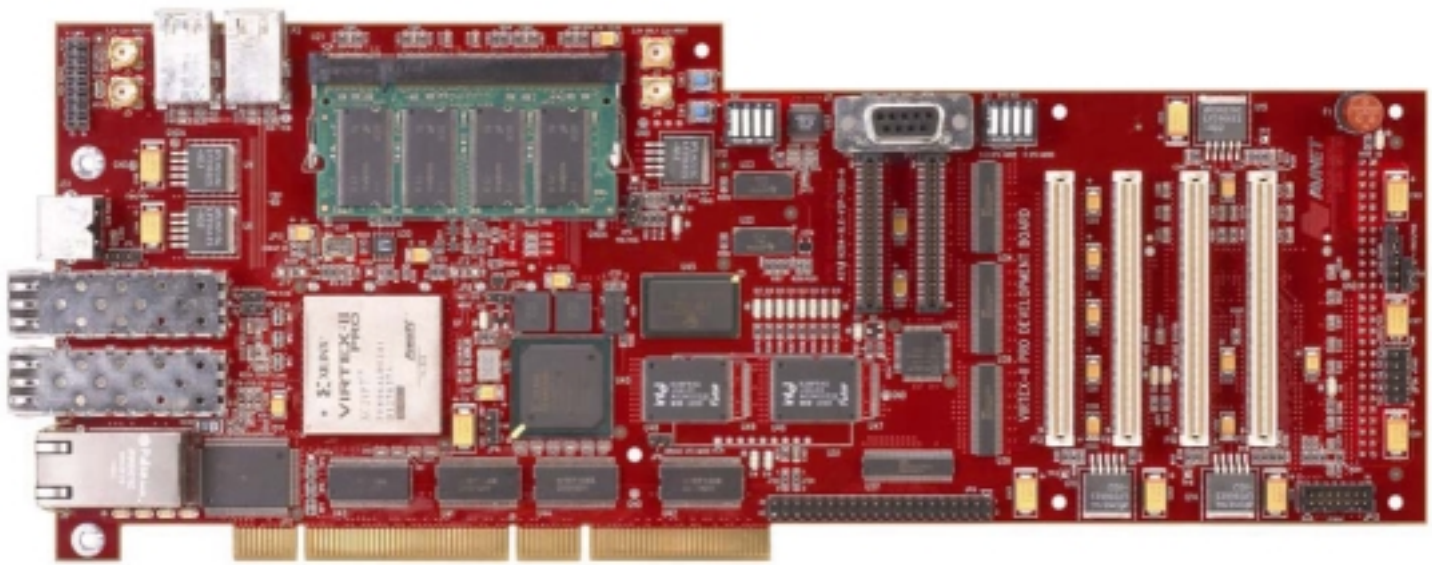
# Why Move to Hardware?

- For such a large dataset, containing texture classes which are visually very similar, our classifier performed remarkably well.
- Speed of classification is limited in software.
- For example, given a 32x32 8-bit gray scale image, the number of computations required to generate the spectral histogram for 9 5x5 filters + 1 intensity filter is roughly 256k multiplies and 256k adds.
- Therefore, we must improve this computational bottleneck.
- The general purpose microprocessor can only perform one or two multiply/adds simultaneously.
- PLDs allow for many simultaneous multiplies and adds to be performed in one or two clock cycles.
- The preprocessing algorithm is inherently parallelizable, therefore well suited for hardware implementation.

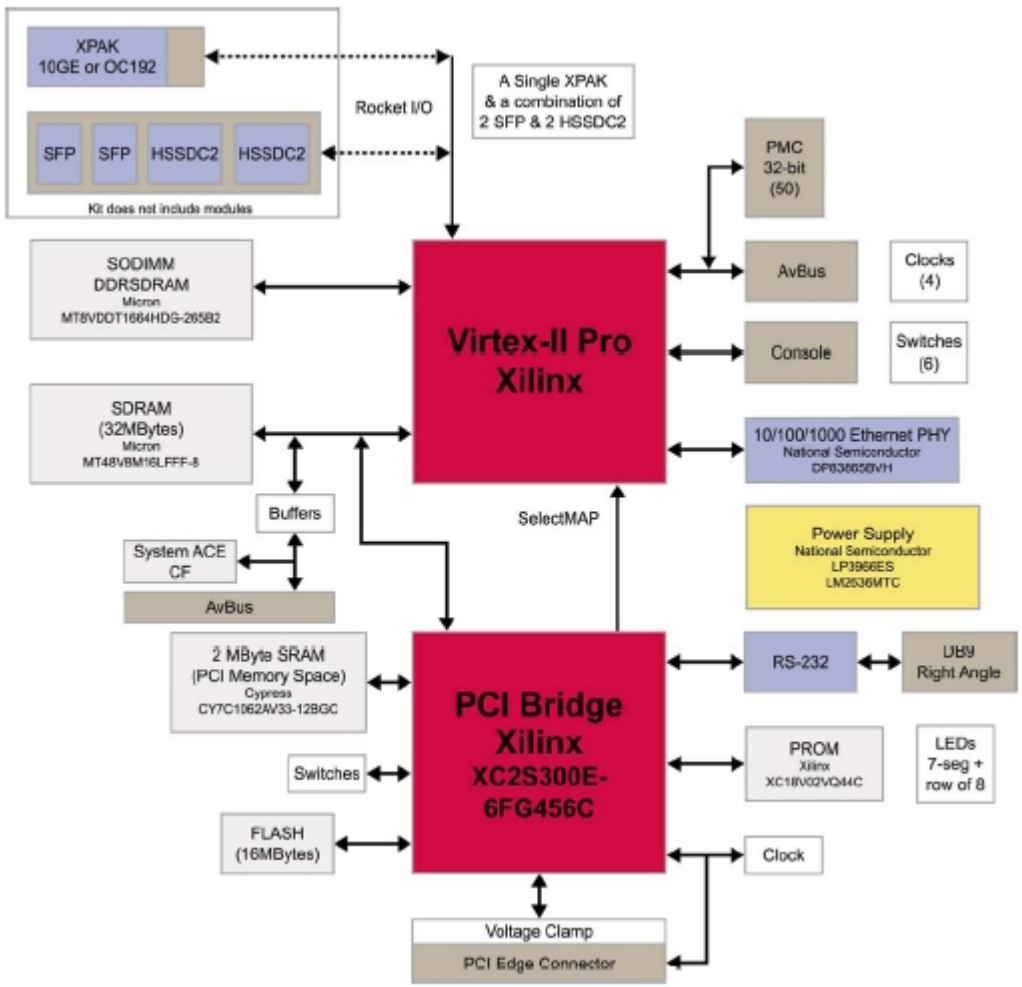
# Target Hardware:

## Avnet's Virtex II Pro Board

- Uses Virtex II Pro XC2VP20
- Many Options for I/O.
- 32 Bit PCI Bus has Data Throughput of Over 100 MB per Second.

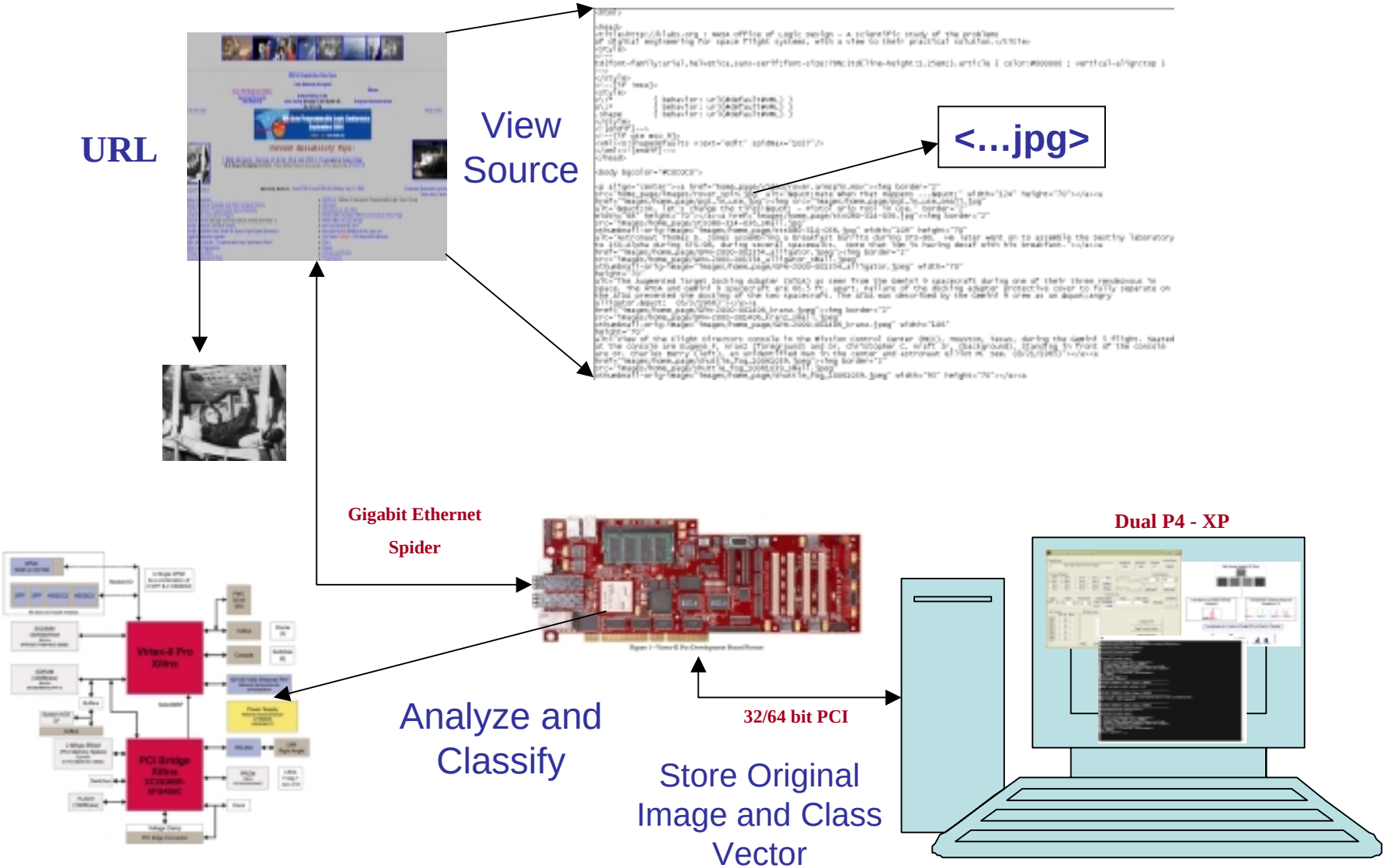


# Block Diagram




- Virtex-II Pro is focal point.
- Spartan acts as bridge to PCI
- On Board Memory
  - 32 MB SDRAM
  - 2 MB SRAM
  - 16 MB FLASH
  - 128 MB DDR SDRAM
  - 64 MB Compact Flash
- Ethernet
- RS232
- 4 AvBus Connectors
- 2 PMC Connectors

# System Layout



# Using Embedded Software

- Hard IP (μP, Mult., BRAM) takes up small percentage of die space
  - Can do traditional FPGA design.
  - Can incorporate embedded software for free.
    - Requires new tools (EDK)
  - Need to determine optimal hardware/software tradeoffs.
    - SoC → System on Chip

# Hardware vs. Software

- Custom Hardware is definitely faster
  - Harder to design
- Software is slower but easier
  - Use when speedup not a factor
- Example: Adaptive Filtering

Filter Equation:

$$y(n) = \sum_{k=0}^{L-1} x_k(n) \cdot h_k$$

- L: # taps
- Needs L multiplications and L-1 additions.

# Hardware vs. Software

- Software Implementation: $y(n) = \sum_{k=0}^{L-1} x_k(n) \cdot h_k$

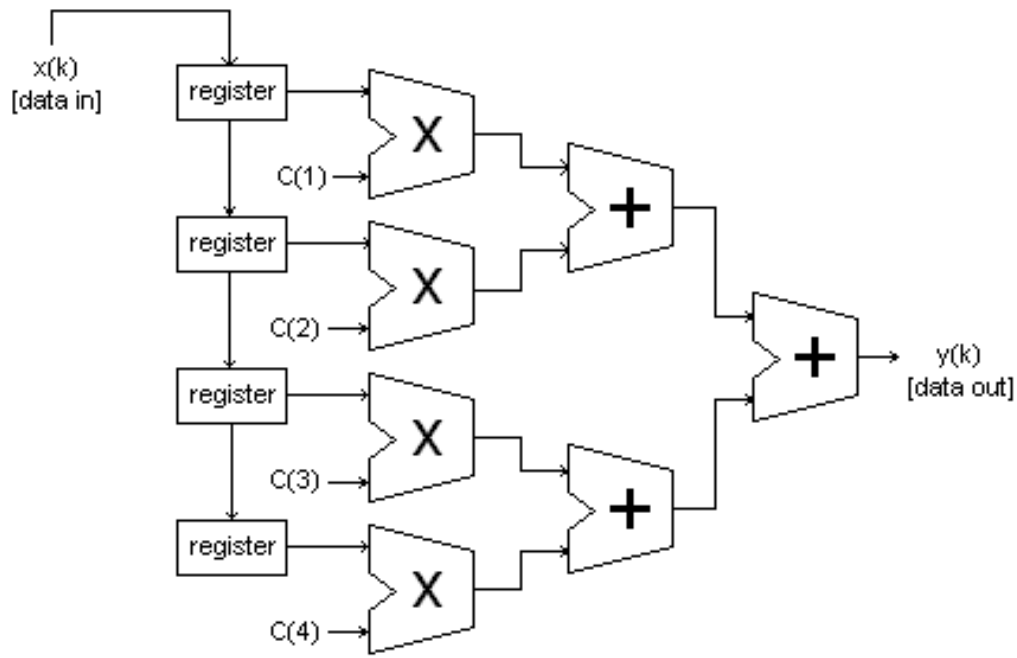
```
for i in range 0 to L loop
   multiply x(i) * h(i)
   add to result
 end loop
```

Loop takes 2*L processor cycles* to complete.

*A specialized DSPµP has a one cycle multiply-accumulate instruction. This would take at least L cycles.

# Hardware vs. Software

- A custom hardware implementation:




Takes 1 clock cycle to complete.

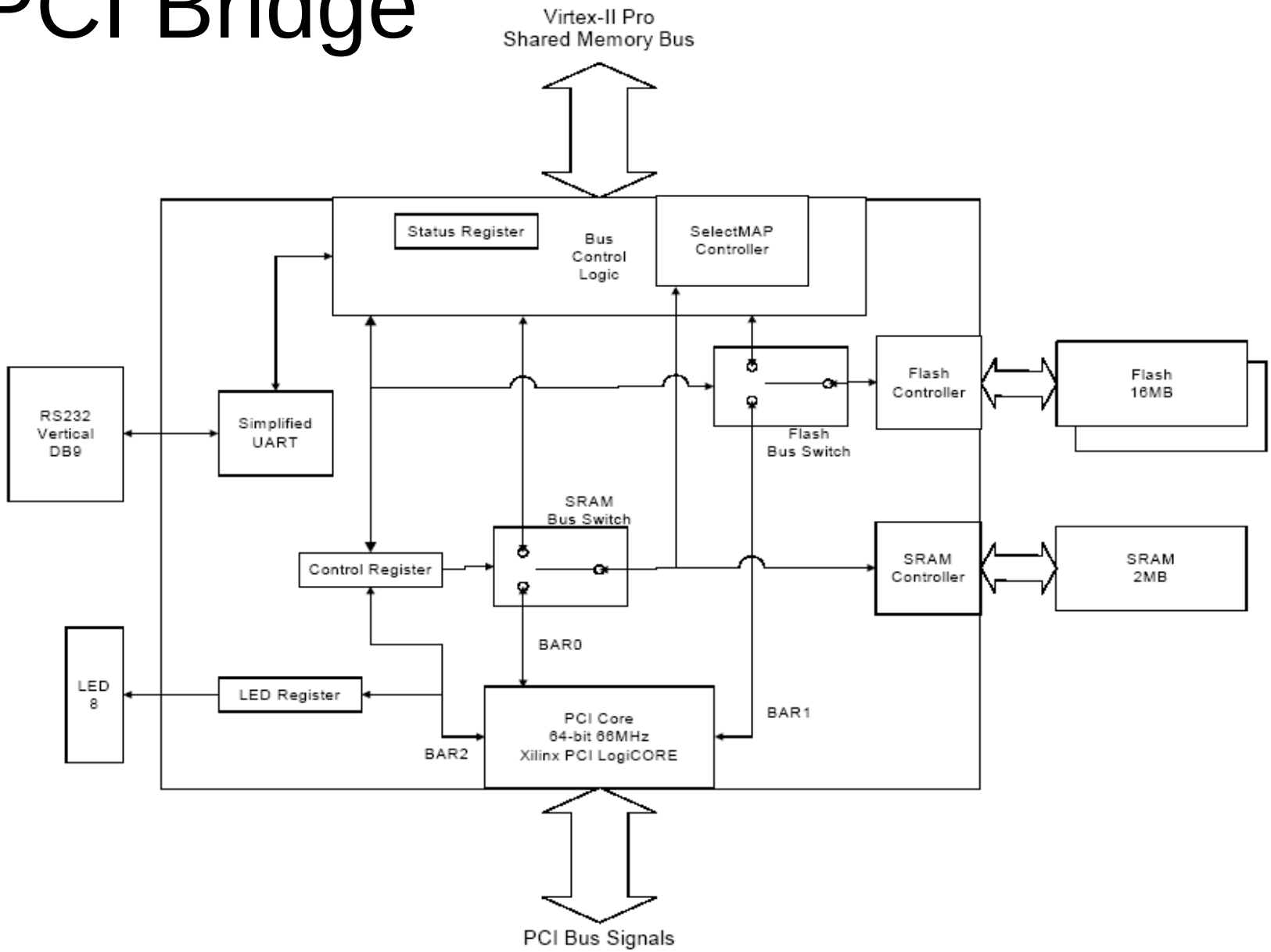
# Hardware vs. Software

- Not all tasks have such a drastic speedup in hardware.
  - Memory Accesses
    - Only one address per clock cycle can be read in SDRAM, Flash, or SRAM.
    - We require more than 32-bits per action, so we waist time reading data.
    - Possible to store more data in BRAM to create an initial data stack that would overcome future read times.
- Combine hardware and software for optimal ease of design and speed of execution.
  - Need to determine optimal compromise.

# PCI Bus

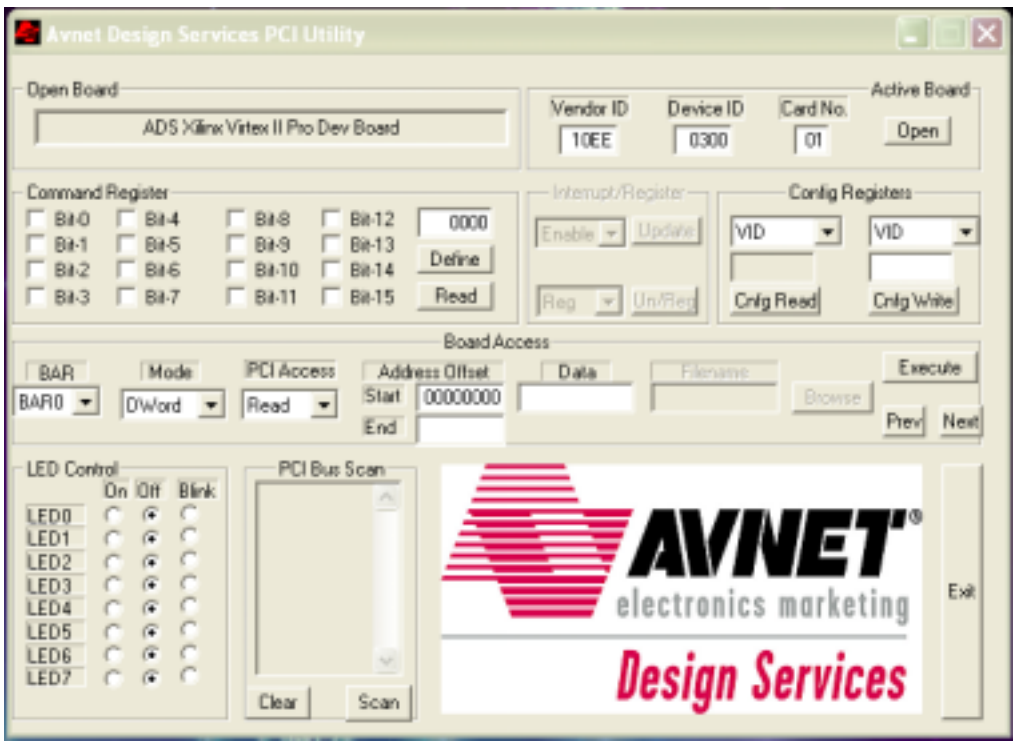
- The V2P does the spectral histogram calculation for each image and sends this classification vector representation to the host pc for classification via the PCI Bus.
- However, proprietary bridge (see figure on next slide) design is a black box and therefore difficult to optimize.
- Also, software interface is tedious at best.
- We must redesign software interface and PCI bridge.

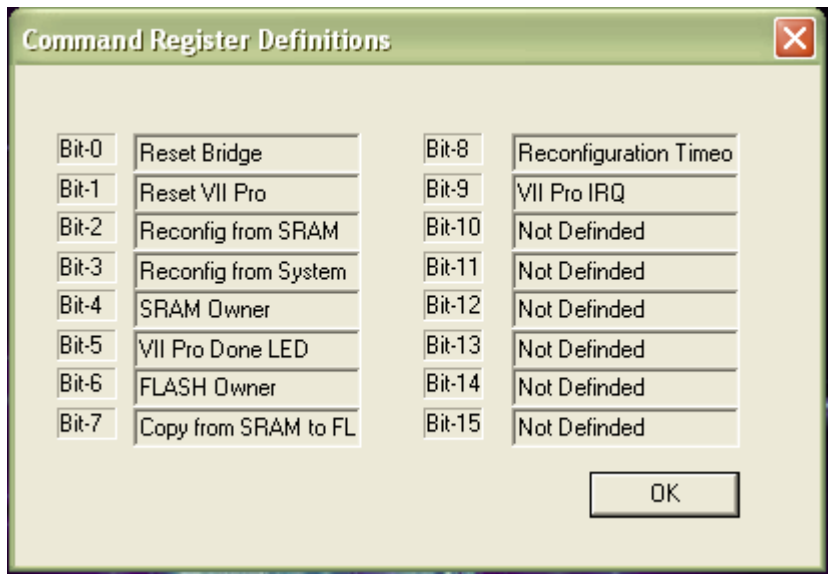
# PCI Bridge


Virtex-II Pro
Shared Memory Bus
Status Register
Bus
Control
Logic
SelectMAP
Controller
RS232
Vertical
DB9
Simplified
UART
Flash
Bus Switch
Flash
Controller
Flash
16MB
SRAM
Bus Switch
Control Register
SRAM
Controller
SRAM
2MB
BAR0
LED
8
LED Register
BAR2
PCI Core
64-bit 66MHz
Xilinx PCI LogiCORE
BAR1
PCI Bus Signals

# Avnet's PCI GUI Interface

- Very Simple Tool for Configuring the V2P or writing data to the SRAM, FLASH, and Command Registers via the PCI Bus. Much Faster than the RS232 (serial) Port.
- Downside is that it is not automated.

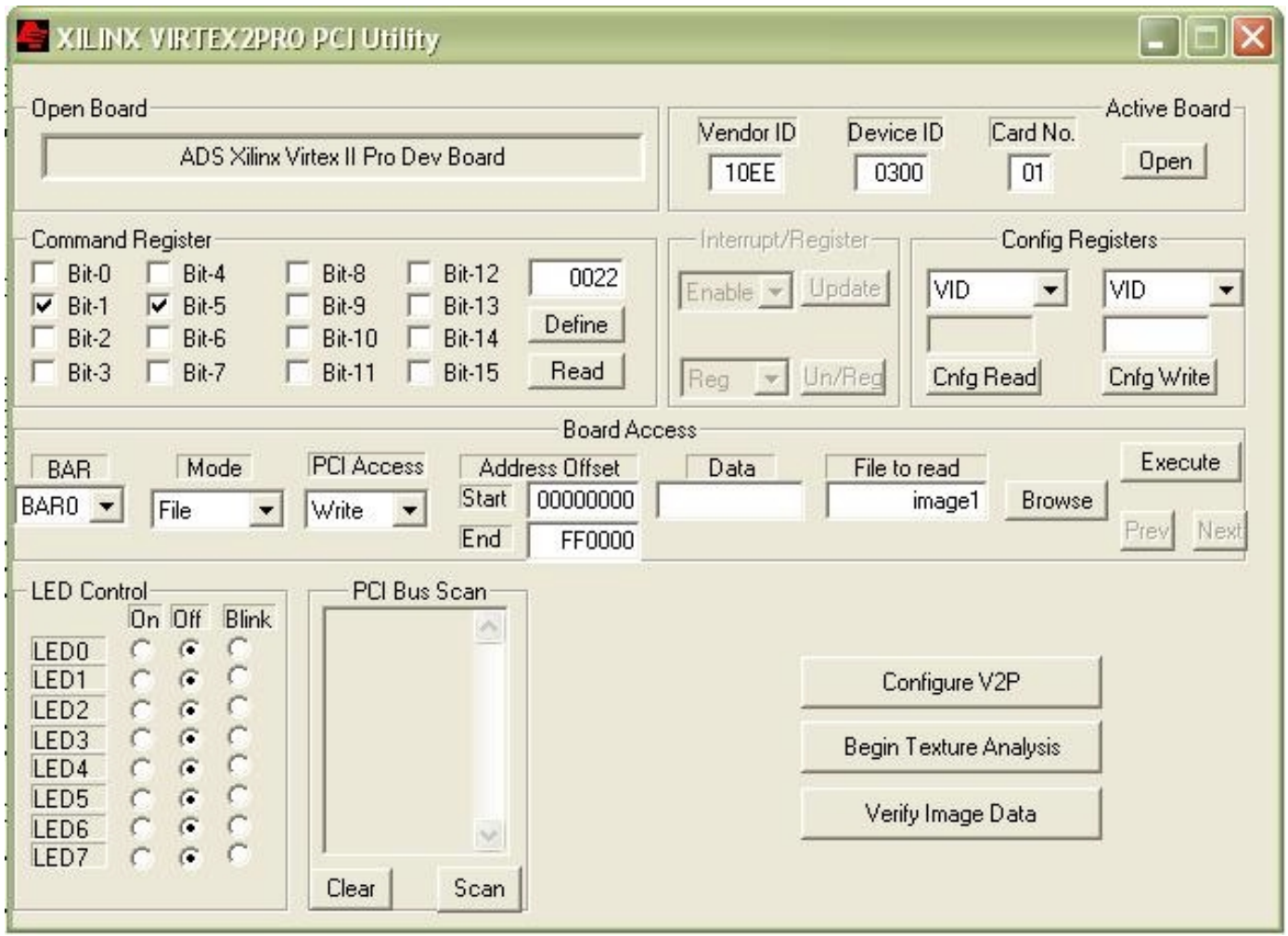


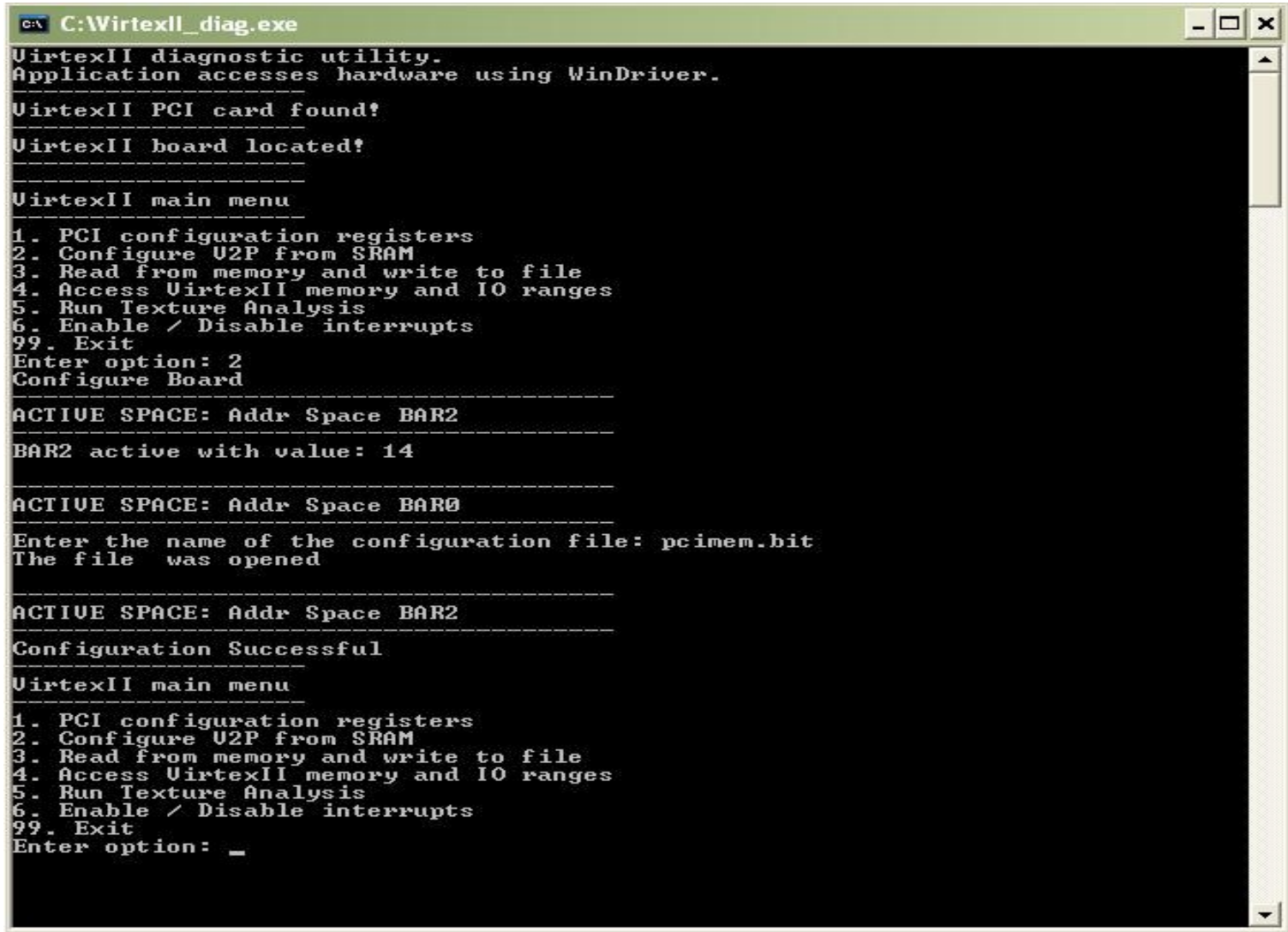
# Reverse Engineering the GUI

- GUI C/C++ source code included in avnet's documentation.
- Some source files missing, were found and downloaded at http://www.jungo.com.
- Used Visual C++ to reverse engineer the gui.
- Most of the functionality of the gui is provided by windriver, a device driver development tool written by Jungo
- We downloaded an evaluation copy of Jungo's windriver, which provided diagnostic tools for driver development.
- Using the tools, we generated a dos executable which mimicked the functionality of the gui, but could later be automated as part of the classification software.

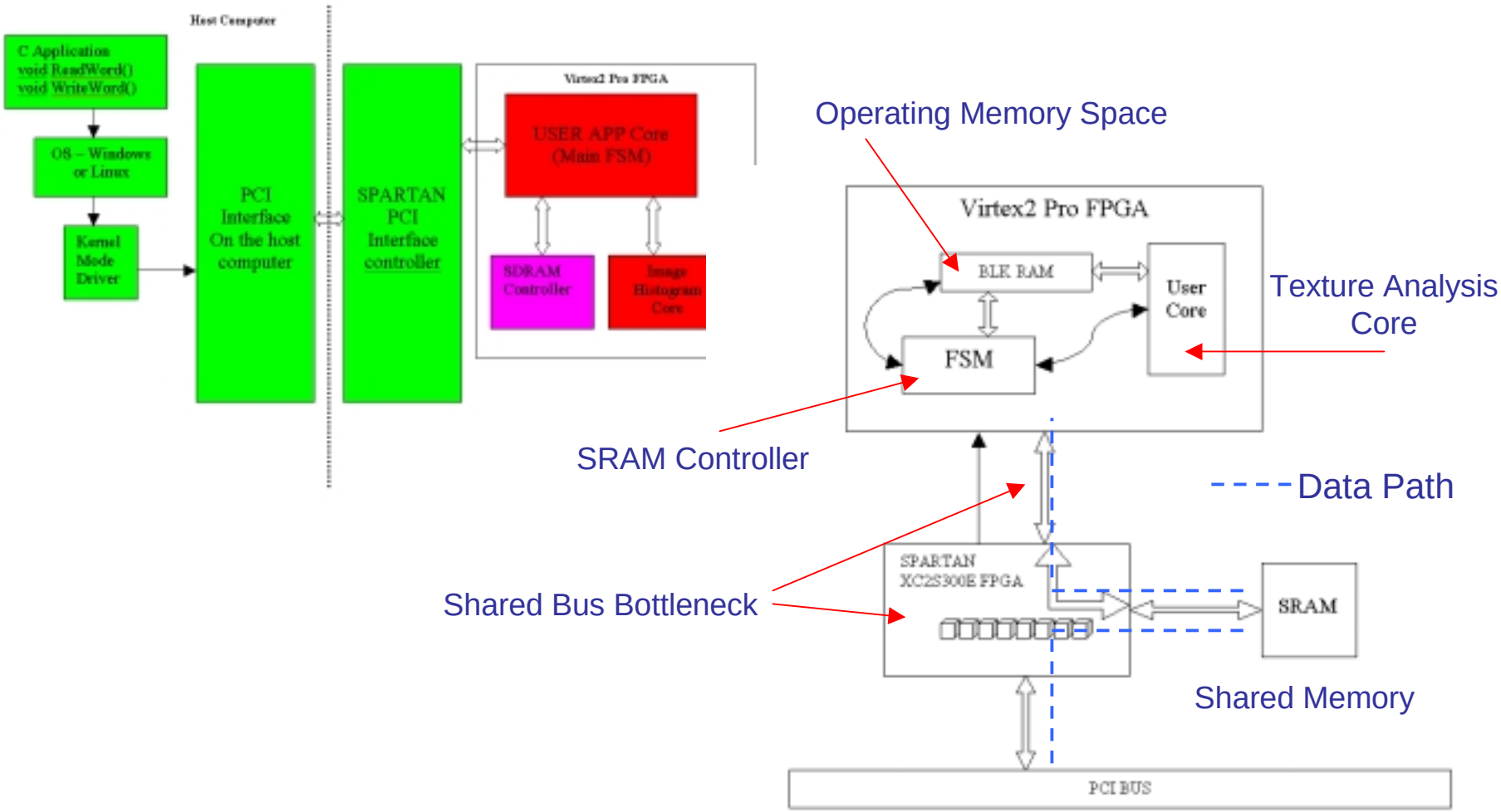
# Remanufactured GUI



# New Command Line Interface



# System Interface



# Conclusions/Future Work

- Optimize the hardware/software interfacing
- Time the results (how many images can we process per second); is it real-time?
- Possibly move to a board with better interfacing tools, as well as faster interfacing via PCI-X or PCI express, or DMA capabilities.
- Finally, optimize calculating efficiency of the texture analysis algorithm, i.e., consider a multi-stage pipeline with more efficient memory access algorithms.
- Moving the trained ANN classifier to the FPGA (really just another convolution core).
- The ultimate goal is to do real time object detection using a similar spectral histogram preprocessing stage.