Implementation of Hardware for Image Display and Processing

Final Report

Wesley Snyder, Mustafa Dagtekin, Rajeev Ramanath
North Carolina State University
Center for Advanced Computing and Communication

November 7, 1999
# TABLE OF CONTENTS

1.0 Introduction ..............................................................................................................2  
2.0 Background ..............................................................................................................2  
   2.1 Video fundamentals ..................................................................................................2  
   2.2 Indicial Mapping ....................................................................................................3  
3.0 System Architecture .................................................................................................4  
   3.1 Analog-to-digital conversion ..................................................................................5  
   3.2 Frame Buffers (Video RAMs) .................................................................................6  
   3.3 Pointer Memory .....................................................................................................8  
   3.4 Lookup Table .......................................................................................................8  
   3.5 Video DAC ..........................................................................................................9  
   3.6 FPGA unit ...........................................................................................................10  
       3.6.1 Sync Generator ............................................................................................12  
       3.6.2 Controller and address decoder ...................................................................14  
       3.6.3 Video Mode Registers ..................................................................................17  
       3.6.4 Address generator .......................................................................................19  
3.7 PCI Interface ............................................................................................................20  
4.0 Software extensions ...............................................................................................20  
5.0 Implementation ......................................................................................................22  
6.0 Schematics and Layout ..........................................................................................23  
   6.1 Analog interface ...................................................................................................23  
   6.2 Power and Clock Generation ..............................................................................25  
   6.3 Frame Buffer # 1 and Interface Buffers ...............................................................26  
   6.4 Frame Buffer # 2 and interface flip flops .............................................................28  
   6.5 Lookup Table and Video DAC ............................................................................30  
   6.6 Pointer Memory and Interface Flip Flips .............................................................32  
   6.7 Interface between PCI and board .........................................................................33  
   6.8 Interface Flip Flops .............................................................................................36  
   6.9 FPGA module .....................................................................................................38  
   6.10 Interface between FPGA address generator and memories .............................40  
7.0 Layout ....................................................................................................................43  
8.0 Results ....................................................................................................................44  
9.0 Conclusions and Future Work .............................................................................44  
10.0 References ............................................................................................................44  
11.0 Appendices ............................................................................................................44
Abstract

In this project, a video image display system is implemented as a PCB board. The PCB for the system is described, as well as the software which drives the unit, including both control and application software.

1.0 Introduction

In a conventional raster scan image display system, the frames for the video signal are stored into a memory sequentially, and displayed in the same way. The electron beam sweeps both the camera and the monitor from left to right and top to bottom, and it goes to other line after finished with one line.

In this system, we propose another way to display frames. We use a second memory which contains the addresses of the pixels in the frame buffers. This allows us to process the image in real-time. The additional memory, “pointer memory”, is scanned in a raster-scan fashion and the data output of this memory provides the address input to the frame buffer. The pointer memory is written by the user, so it provides the capability to display the frame contents in an arbitrary fashion.

In this paper, the background is presented in Section 2.0. The system architecture is presented in Section 3.0, and the following section presents implementation details and results are presented in the last two sections.

2.0 Background

2.1 Video fundamentals

A video system has several components. It includes capturing, storing and displaying images. A video system includes a receiver part, the input of which could be a camera output, TV decoder etc. and storage part, simply a memory and a display part, the output of which is usually a monitor.

There are a lot of video standards based on the usage of the video signal. There are TV standards such as NTSC, PAL, SECAM. These standards are used in TV coders and decoders. and etc. There are computer video standards, such as VGA, XGA etc. These are used in computer monitors.

No matter which standard is used, the system works the same way. There is a circuitry which supplies the CRT 1 a analog signal which includes the video information and some extra. A CRT display unit is shown in Figure 1. Simply, an electron beam sweeps the monitor from left to right and top to bottom. Each time the beam makes a pass across the screen, it lights up phosphor dots on the inside of the glass tube, thereby illuminating the active portions of the screen.

1. CRT stands for “cathode ray tube”.

Implementation of Hardware for Image Display and Processing 3
A NTSC video signal is shown in Figure 3 on page 4. This is an analog input which determines the signal levels for each pixel, when a line starts or when a frame starts. The lines are separated by a “horizontal blank” period and the frames are separated by a “vertical blank” period. During that time the electron beam shuts off to synchronize with the analog signal.

The horizontal blanking period is made up of the front porch, the sync pulse, and the back porch. In electron-beam devices, such as most monitors and older TV cameras, the blanking period was provided as the time during which the electron beam is switched off (blanked), so that the beam may be repositioned from the right side of the screen to the left. A similar but much longer blanking period is provided at the end of each frame to
allow time for the electron beam to be repositioned from the lower right to the upper left corner of the screen.

In conventional television, a total of thirty frames are displayed each second, with each frame consisting of 525 lines. Each line is thus

\[
\frac{1}{30(525)} = 63.5\mu s
\]  

(EQ 1)

long. Of this, by the NTSC standard, 14% or 8.89 µs, is blanking, leaving 54.61µs of active video on each line. We are using a 12.288 MHz sampling clock, which thus produces 640 samples of active video on each line. Similar timing of the vertical blanking pulse produces 480 lines of video per frame.

The timing is somewhat complicated by the fact that the conventional television standards use *interlaced scanning*.

### 2.2 Indicial Mapping

In conventional video systems the frames are scanned in an orderly fashion when displaying. If we can scan stored video in an arbitrary way, we could implement any image transformation we want. The idea is illustrated in Figure 3 on page 5 The paper in Attachment 1 explains the idea in more detail [1].

![FIGURE 3. Illustration of indicial mapping](image)

The system simply works like this. A pointer memory is stored with the addresses of the stored image according to the transformation that we want to implement. After that the pointer memory is scanned sequentially, in a raster scan fashion, but it will lookup to video memory in the way that the transformation requires. An example is shown in Figure 4 on page 6.
3.0 System Architecture

The basic block diagram for the system is given in Figure 5 on page 6. The system consists of three major units:

- Analog Input Video -to- Digital Data Conversion
- Digital Data -to- Analog Video Conversion
- PCI interface operations
- FPGA modules

FIGURE 5. System’s Block Diagram
3.1 Analog-to-digital conversion

In order to use and manipulate the video data that is supplied from the camera, the analog video must be converted to digital data. Although the original plan called for 12 bit dynamic range, only 10-bits could be used, since no monolithic 12-bit A/D converters were available which would run as fast as needed. The resolution used is 10-bit for this conversion. A Burr-Brown ADS824 module, which is a 10-bit, 70 MSPS CMOS analog-to-digital converter, is used for this purpose. It is a 10-bit, 70 MSPS CMOS analog-to-digital converter. The module is shown in Figure 6 on page 7.

The waveform of a NTSC video signal is shown in Figure 2 on page 4. The output waveform of the camera is same as this waveform, if it an NTSC camera. The data that is supplied from the camera is sampled by the rising edge of the system clock, and stored into the appropriate registers and then transferred to one of the Frame buffers.

FIGURE 6. ADC and associated interface flip-flop
3.2 Frame Buffers (Video RAMs)

The memories which are used to store video frames are called “frame buffers” or “video RAM’s”. Motorola MCM6949 memory chip is selected to be used as Video RAMs. There are two frame buffers in this system. One of them is used to store the input frame from the ADC, while the other one provides output to the display in real time. These buffers are swapped at the end of the each frame. The modules are shown in Figure 7 on page 8 and Figure 8 on page 9.
3.3 Pointer Memory

This SRAM is used to implement indicial mapping [1]. The same module, Motorola MCM6949 is used to implement pointer memory. The module is shown in Figure 9 on page 9.
3.4 Lookup Table

This memory unit is used for color mapping. It accepts 10-bit addresses and puts out 8-bit red, 8-green and 8-bit blue values of the pixels. A Motorola 6926A SRAM module is used. The module is shown in Figure 10 on page 10.

![Figure 10. Lookup Table](image)

3.5 Video DAC

This is used to convert the digital RGB values to analog signal. A Analog Devices ADV7123 is used to convert digital data to analog video. It is 3-channel Video DAC, which accepts composite sync and blank signals externally. Each channel is 10-bits, but we used only 8-bits of this unit. It generates a waveform shown in Figure 2 on page 4. Most monitors will accept this output as a video signal. This module is shown in Figure 11 on page 11.
3.6 FPGA unit

The controller for the system, address generator, sync generator, video mode registers and address decoder are implemented in a FPGA. An Altera MAX7000S module is used for FPGA. This is a relatively small FPGA which is chosen to satisfy the speed requirements. The block diagram for the FPGA modules is shown Figure 12 on page 12. The controller generates the enable signals for the tri-state buffers, flip flops and memories. The address generator generates sequential addresses for raster scan. The sync generator generates “blank” and “sync” signals for the system. The video mode registers determine the video mode and are written by PCI interface. The address decoder decodes the address created by PCI interface to assert the enable signals.

FIGURE 11. Video DAC unit
Inputs.

pci_data_in[19:0]: This is a 20-bit data bus which is connected to least significant 20-bits of the data bus of the PCI interface. Basically, the video mode registers are loaded using this bus.

pc_vmr_select[1:0]: These 2 pins are connected to 20th and 21st bits of the PCI data bus. These are used to decode which video mode registers are being written.

la[24:22]: These three pins are connected to relevant pins of the PCI address bus. These are used to decode address spaces for the memories.

pci_read: This is connected to PCI9050’s RD# pin. It is the read strobe.

pci_write: This is connected to PCI9050’s WR# pin. It is the write strobe.

pci_cs: This is connected to PCI9050’s CS# pin. It is the chip select pin.

Outputs.

ff_enable[18:0]: These are the output enable signals for the all interface flip-flops and tri-state buffers on the board.

oe_mem[3:0]: These are output enable (read-enable) signals for the memories.

we_mem[3:0]: These are write enable signals for the memories.

FIGURE 12. The toplevel of the FPGA module
cs_mem[3:0]: These are chip select signals for the memories.

address[19:0]: This is generated by address generator. It is used to sequentially read the pointer memory contents for ‘display’ and one of the frame buffer contents for ‘write’.

HSync_out: This is a test signal that is ran to a header for extension purposes. It is an output from ‘horizontal sync’ signal. In this implementation, it is “negative true”.

VSync_out: This is a test signal that is ran to a header for extension purposes. It is an output from ‘vertical sync’ signal. In this implementation, it is “negative true”.

sync_out: This is a test signal that is ran to a header for extension purposes. It is an output from ‘composite sync’ signal. In this implementation, it is “negative true”.

HBlank: This is a test signal that is ran to a header for extension purposes. It is an output from ‘horizontal blank’ signal. In this implementation, it is “positive true”.

VBlank: This is a test signal that is ran to a header for extension purposes. It is an output from ‘vertical blank’ signal. In this implementation, it is “positive true”.

d_blank: This is a ‘composite blank’ signal that is wired to BLANK input of the DAC. It is ‘negative true’.

d_sync: This is a ‘composite sync’ signal that is wired to SYNC input of the DAC. It is ‘negative true’.

Control Signals.

clock: This is the pixel clock that is used in the whole system.

reset: This is the reset pin for the system. It is wired to PCI9050’s LRESET# pin.

3.6.1 Sync Generator

One of the major parts of the FPGA modules is the sync-generator. This module accepts the outputs from “Video Mode Registers” unit and an enable signal from “Controller” as inputs and put out the synchronization signals. The high level block diagram is shown in Figure 13 on page 14.
Description of the pins:

- **clock**: This is the global clock signal.

- **reset_sync**: This signal is generated by the controller. It is not the same as the global reset signal.

- **load_vmr**: This is actually a signal that goes to video mode registers. It is generated by the controller and it signals the time when video mode registers are allowed to be updated. It is monitored by Sync Generator so that it won’t create any synchronization signals when writing to the video mode registers.

- **HSon[4:0]**: This comes from the video mode registers module. Basically sync generator turns the “sync” on when its “pixel counter” reaches this value.

- **HSoff[7:0]**: This comes from the video mode registers module. Basically sync generator turns the “horizontal sync” off when its “pixel counter” reaches this value.

- **LineLength[10:0]**: This comes from the video mode registers module. Basically sync generator turns the “horizontal blank” off when its “pixel counter” reaches this value. (It starts with ‘on’ by default).

- **VBon[9:0]**: This comes from the video mode registers module. Basically sync generator turns the “vertical blank” on when its “line counter” reaches this value.

- **VSoff[9:0]**: This comes from the video mode registers module. Sync generator turns the “vertical sync” off when its “line counter” reaches this value.

- **VSoff[9:0]**: This comes from the video mode registers module. Sync generator turns the “vertical sync” off when its “line counter” reaches this value.
FrameSize[9:0]: This comes from the video mode registers module. Sync generator turns the “vertical blank” off when its “line counter” reaches this value.

Outputs:

HSync: This is the “Horizontal Sync” signal. It is an active high signal, it simply enabled when horizontal sync needs to be generated.

HBlank: This is the “Horizontal Blank” signal. It is an active high signal. It is enabled when horizontal blank needs to be generated.

VSync: This is the “Vertical Sync” signal. It is an active high signal which is enabled when vertical sync needs to be generated.

VBlank: This is the “Vertical Blank” signal. It is an active high signal which is enabled when vertical blank needs to be generated.

blank: This is “composite blank” signal. It is simply output of an OR gate where inputs are HBlank and VBlank.

csync: This is “composite sync” signal. It is simply output of an OR gate where inputs are HSync and VSync.

buffer_no: This is a flag which is used by the controller to determine which VRAM is going to be written and which one is going to be read. After each frame cycle\(^1\), the sync generator is being reset and invert whatever the value is stored in “buffer_no” register and stores back to the same register. So, if buffer_no = ‘0’, VRAM1 is being written and VRAM2 is being read, whereas if buffer_no = ‘1’, VRAM2 is being written and VRAM1 is being read.

reset_address: This is signal that goes to address generator module. It simply initializes the address generator module to initial value, which would be ‘0’.

3.6.2 Controller and address decoder

Controller. The controller module of the system is implemented in the FPGA. It is also embedded with an address decoder, which is used to control access from the PCI bus to the board modules. The block diagram is shown in Figure 14 on page 16.

The controller of this system has five states. These states determine all of the enable signals and some other signals that are required, except the PCI access. The controller works only when accesses are local, on other words “basic operation”. The accesses from the PCI bus will always win the bus, but even during that time the controller will work as if it is running in normal mode.

The states of the controller are reset_state, mode2, blank_state, mode3, freeze_frame. The state diagram is shown in Figure 15 on page 18.

reset_state: This is the beginning state for the controller. All of the enable signals are disabled and registers are initialized in this state.

\(^1\) Frame cycle is defined as “displaying one whole frame including the blank periods”.

mode2, mode3: These states are called “normal operation mode”. In ‘mode2’, the VRAM1 is being written and VRAM2 is being read, whereas in ‘mode2’, the VRAM2 is being written and VRAM1 is being read. It is called ‘normal mode’ because if nothing kicks in, the system will run between these two states and blank_state in between.

**blank_state:** This is when a ‘composite blank’ signal is received from sync generator. We don’t want to ‘read’ or ‘write’ anything during this time, because the data in this period will not be valid.

**freeze_frame:** This state is used to create a special mode. During this time VRAM1 is displayed continuously without being updated. (The choice of VRAM1 is somewhat arbitrary. VRAM2 can also be chosen)

During every one of these states, the controller puts out enable, chip select, read & write enable signals for the appropriate flip-flops, and memories.

But, for PCI accesses another address decoder is used to grant accesses.

![FIGURE 14. Controller and address generator](image)

Pin descriptions (Controller):

**Inputs:**

**blank:** This is generated by the Sync Generator. When it is “1”, the controller changes state to “blank state”. It is a composite blank signal.
buffer_no: This is also described in Section 3.6.1, “Sync Generator,” on page 13. This signal tells the controller which VRAM is being written and which one is being read.

freeze_on: This signal comes from the Video Mode Registers. It is a user writable bit which puts the system into freeze frame mode or out of this mode. If it is “1”, the next state would be “freeze_frame” state, if not, the next state would be one of the normal operation modes.

Outputs:

cont_ffen[8:0]: These are the enable signals for the flip-flops in the system. These wires are merged with the outputs of the address decoder within the multiplexing unit section.

cont_oe[3:0]: These are the output enable signals for the memories that are generated by the controller.

cont_cs[3:0]: These are the chip select signals for the memories that are generated by the controller.

con_we[3:0]: These are the write enable signals for the memories that are generated by the controller.

Address Decoder and Multiplexing Unit. This address decoder unit decodes addresses from the PCI accesses and generates necessary enable signals. The multiplexing unit will decide either to enable controller access or PCI access.

The address is decoded by using 24th to 22nd bits of the PCI address bus.

<table>
<thead>
<tr>
<th>la[24:22]</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Lookup Table</td>
</tr>
<tr>
<td>010</td>
<td>Pointer Memory</td>
</tr>
<tr>
<td>011</td>
<td>VRAM1</td>
</tr>
<tr>
<td>100</td>
<td>VRAM2</td>
</tr>
<tr>
<td>101</td>
<td>Video Mode Registers</td>
</tr>
</tbody>
</table>

TABLE 1. Mapping of addresses

Pin description:

la[24:22]: The 24th to 22nd bits of the PCI address bus. This is used to decode which memory is being accessed. The address generator will generate the appropriate enable signals for the memories.

pci_cs: This is the CS# pin of the PLX9050. Whenever a PCI access happens this signal goes low. Address decoder uses this to enable chip select signals and multiplexing unit uses this to grant PCI access.
*pci_write:* This is the WR# of the PLX9050. Whenever a PCI “write” happens, this will go low. Address decoder uses this enable write enable signals.

*pci_read:* This is the RD# signal of PLX9050. Whenever a PCI “read” happens, this pin will go low. Address decoder uses this to enable reads.

*load_vmr:* This is the intermediate signal to grant access for video mode registers. But this signal must be used with pci_cs and pci_wr to make sense. The combination of these signals (refer to VHDL code) is tagged as “load_vmr_out”. This is the actual pin that goes out.

3.6.3 Video Mode Registers

Video mode registers are used to be able to set and change the video modes. Basically these are 72-bit registers to hold the values for the sync timers. This information is decoded by the Sync Generator unit. The outputs from the VMR are shown in Table 1 on page 17.

![State diagram of the controller](image-url)
Also, the 74th bit of the this unit is used to store a “freeze frame” request by the user. If it is set to ‘1’, the system works in “freeze frame” mode, otherwise it will operate normally. (the 73rd bit is reserved to be used as “pci_on” signal, but is not used in the final version).

Pin descriptions:

*clock:* This is the global clock signal.

*reset:* This is the global “reset” signal. (It is tied to LRESET# signal of the PLX9050.)
**load vmr:** This enables the video mode registers for write access.

**pc_vmr_select[1:0]:** This is 20th and 21st bits of the PCI data bus. It determines which part of the VMR is being written. This map is shown in Table 3 on page 20. As the PCI updates VMR, it is being written in four cycles to be able to write all of the VMR.

<table>
<thead>
<tr>
<th>pci_vmr_select[1:0]</th>
<th>vmr</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>vmr[19:0]</td>
</tr>
<tr>
<td>01</td>
<td>vmr[39:0]</td>
</tr>
<tr>
<td>10</td>
<td>vmr[59:0]</td>
</tr>
<tr>
<td>11</td>
<td>vmr[74:0]</td>
</tr>
</tbody>
</table>

**TABLE 3. The decoding of video mode registers**

**pci_data_in[19:0]:** The PCI updates VMR in portions of 20-bits. The mapping is shown in Table 3 on page 20.

### 3.6.4 Address generator

This module is used to generate sequential addresses for accessing the pointer memory, and during write cycle, for addressing one of the frame buffers for writing. The block diagram is shown in Figure 17, “Address generator,” on page 20.

**FIGURE 17. Address generator**

Pin descriptions:

**clock:** The global clock for the system

**reset_address:** This is generated by the sync generator. It resets the address back to ‘0’.

**disable_address:** This is used to hold the address counter.
address[19:0]: This is the 20-bit address output.

3.7 PCI Interface

In order to access memories and video mode registers in the FPGA, a PLX PCI9050RDK PCI Interface Development Kit is used. Details of operation of the PLX Development kit are included in the documentation of that kit, appendix 11. Since this module is designed as prototype, the signals are provided from the headers of this kit to the headers of our board. In a redesign, we recommend that the PCI9050 chip itself should be incorporated onto the video controller board.

4.0 Implementation

These design steps were taken to implement the system:

- Design modeling and simulation. The system has been modeled in Verilog(TM) Hardware Description Language and verified for its functionality.

- Implementation of FPGA unit. The controller, sync generator, address decoder and address generator has been implemented by using Altera Maxplus-II software to be mapped into a Altera Max7000S FPGA device. VHDL is used to implement the FPGA part.

- Selection of the actual parts. This steps involved in selecting the components that satisfy the requirements.

- Board Design. The PCB board for the design is manufactured using Orcad tool suite. The design has been drawn by Orcad Capture schematics editor, converted to Layout format. For routing and placing the actual layout, Orcad LayoutPlus is used.

- Debugging. After the design is manufactured, hardware, FPGA and software bugs have been detected and corrected.

- Software implementation. Since the system is connected to a computer system through a PCI bus, a software implementation has been implemented. It includes the device drivers for the system and the application.

5.0 Schematics and Layout

The system is laid out to PCB board by using Orcad Capture and Orcad LayoutPlus tools.

5.1 Analog interface

Analog-to-Digital Converter: This is the Burr-Brown 824E ADC on board. The pin description can be found in Appendix 5. The operation is briefly described here.
Sampling of the analog signal occurs on the rising edge of the clock, pin 14 of the A/D chip.

The points that had to be mentioned are:

- The data bits are connected in reverse order, i.e. D1 pin is the “most significant” bit of the ADC, but it is connected to the “least significant” bit of the remaining system. That has to be corrected in the next design.
- The OE pin is tied to ground, but it would be better make it FPGA controlled, so that it won’t consume power when it is not working.
- For improved signal to noise, the bypass capacitors need to be placed physically closer to power pins of the A/D. In addition, bypass capacitors need to be added to pins 1, 16, and 26.
- The analog run from the BNC connector to the A/D input should be wired using on-board coaxial connection.

**FIGURE 18. The ADC and relevant connections**
JP1: This jumper is to control PD pin of the ADC, which is “Power Down” pin. If [1-2], the ADC will work, if [2-3], the ADC will shut down.

JP2: This is a 20-pin header which is used to extend the analog pins of the ADC available for an extra circuit. Basically any pin that is going to be used should be extended by fly-wires.

JP3: This jumper selects the input range for the ADC. If it is [1-2], then input range is 2Vpp, if it is [2-3], then the input range is 1Vpp.

JP4: This jumper selects the source for the voltage reference.(Please refer to the data sheet). If [1-2], then the reference is external, if [2-3], the reference is internal.

U55: BNC connector for the input video.

U2: This is 2X10 bit flip flop which has tri-state outputs. The first 10 bit data, which starts as 1D connects to data bus of the VRAM1, the second 10 bit data, which starts as 2D, connects to the data bus of the VRAM2. (D10 is the most significant bit). The output enable of the first set is controlled by FF_ENABLE10 signal, which is connected to $\overline{OE_1}$, the second set is controlled by FF_ENABLE13 signal, which is connected to $\overline{OE_2}$. These enable signals are generated by FPGA. Do not forget to pull these enable signals to VCC through a resistor. 4.7K is sufficient.

For remove noise on the power supply, 0.1uF bypass capacitors are used.

5.2 Power and Clock Generation

U3: In our design, the power was intended to be supplied from PCI bus. Since all of the PCI signals are supplied using test headers, PCI edge is only used to get the power. U3 only shows the power and ground connections.

U4: This is a voltage regulator which is used to get 3.3 volts from 5V. (please refer to data sheet). This regulator has only a 2 amp capacity, and the 3 volt circuit requires about a maximum of one amp. In a redesign, the traces on the circuit board from the card edge connector to the regulator need to be larger.

U5: This is the socket into which the clock oscillator is plugged.

JP7: This is used to enable/disable the clock. If [1-2], the clock is disabled, if [2-3], the clock is enabled.

U6: This is a clock driver to supply 3.3V clock signals to the other parts in the system.
5.3 Frame Buffer # 1 and Interface Buffers

U8, U9, U10: These are the memory chips of VRAM1. Each module has 4-bit data, 20-bit address buses. Since VRAM1 is only 10-bits, 2 bits of one of the modules are not used and are left floating. The enable signals, CS_MEM0, OE_MEM0, and WE_MEM0 are generated by the FPGA. $\overline{CS}$, $\overline{OE}$, $\overline{WE}$ are the pins that are relevant to these signals. There is an error that has been made in design, $\overline{CS}$ is connected to WE_MEM0, and $\overline{WE}$ is connected to CS_MEM0. This should be taken into account while designing the new one. In our design, we corrected it by swapping the pin assignments in the FPGA unit.
U7: This is one of the flip-flop modules that are used for pipelining. It is a 2X10-bit wide flip-flop. The first set, 1D, is used to connect the data bus of the VRAM1 to the address bus of the lookup table, the second set is used to connect the data bus of the VRAM2 to the address bus of the lookup table. The enable signals, FF_ENABLE17 for the first set, and FF_ENABLE18 for the second set, are generated by the FPGA.

FIGURE 20. Frame Buffer #1
5.4 Frame Buffer # 2 and interface flip flops

U12, U13, U14: These modules are the memory chips for VRAM2. The enable signals are supplied from the FPGA. These signals are CS_MEM1, OE_MEM1, WE_MEM1.
U41: This is 2x10 tristate-buffer which connects the PCI interface data bus to the data bus of the VRAM1 and VRAM2. The enable signals are FF_ENABLE9, for PCI data - VRAM1 data; FF_ENABLE8 is for PCI data - VRAM2 data.

FIGURE 22. Frame Buffer # 2

FIGURE 23. Interface Flip Flop for PCI data-VRAM1 data and PCI data - VRAM2 data.
5.5 Lookup Table and Video DAC

U17, U18, U19: These are the memory chips for the lookup table. Although the address bus of the modules is 16 bits, we need only 10-bits of it. The other address pins are tied to ground. In a new design, these pins can be connected to the FPGA to be used in some other purposes. One purpose can be using these address bits to store several brightness values. This means to change the brightness, we won’t need to update the LUT again, just point to another pre-stored lookup table will be sufficient. The data bus is 8-bit for red channel, 8-bit for green channel, 8-bit for blue channel, total of 24-bits.

U16: This is the Video Digital-to-Analog converter that we used to convert the digital data into the analog video signal. It accepts composite sync and blank signals, and puts out the appropriate video signal with synchronization signals on it. (Figure 2 on page 4). It is a 3x10-bit device, but we only need 3x8 bit of it. In our design, we made a mistake and tied the most significant 2-bits to ground instead of the least significant ones. This should be considered when designing the new board. In prototype, the green channel is corrected like this. The connection between ground and G8 and G9 pins is removed by pulling the pins up, and then G8 pin is connected to G0 pin and G9 pin is connected to G1. So when loading to lookup table, saying D7 is the most significant bit, D0 is the least, this format should be followed:

FIGURE 24. Lookup Table
TABLE 4. Storing Lookup Table

In this table, the bits starts with D shows order of the bit, the bits starts with G, are the actual order of data values that must be stored.

FIGURE 25. Video DAC and external circuitry
JP5: This jumper is used to control PSAVE pin of the DAC. If it is low, the DAC shuts down, if it is high, the DAC operates normally.

5.6 Pointer Memory and Interface Flip Flops

U24, U25, U26, U27, U28: These are the memory chips for the pointer memory. This memory has a 20-bits wide address bus and 20-bits wide data bus. The enable signals are supplied from the FPGA, CS_MEM2, OE_MEM2, WE_MEM2.

U22: This is the 1x20-bit flip-flop used to pipeline the data bus of the pointer memory (PM_D) and the data bus of the VRAM1 (V1_D). The enable signal for this module, FF_Enable14, is supplied by the FPGA.

U23: This is the 1x20 bit flip-flop used to pipeline the data bus of the pointer memory (PM_D) and the data bus of the VRAM2 (V2_D). The enable signal for this module, FF_Enable11, is supplied by the FPGA.
5.7 Interface between PCI and board

U42: This is the 1x20 tri-state buffer between PCI address bus and VRAM1 address bus. The enable signal, FF_ENABLE1, is supplied from the FPGA.

U48: This is the 1x20 tri-state buffer between PCI address bus and VRAM2 address bus. The enable signal, FF_ENABLE0, is supplied from the FPGA.
FIGURE 28. Some of the interface between PCI and board. VRAM1 address-PCI address interface(up), VRAM2 address-PCI address interface(down)
U45: This is the 2x10 bit tri-state buffer between PCI data bus and VRAM1 and VRAM2 data buses. The enable signals are created by the FPGA. FF_ENABLE5 controls the connection between PCI data bus and VRAM1 data bus; FF_ENABLE4 controls the connection between PCI data bus and VRAM2 data bus.

FIGURE 29. Some of the interface between PCI and board. PCI data-VRAM1 data and PCI data-VRAM2 data interface.

TEST JUMPERS: These are the headers that PCI signals are supplied from. In new design these signals should be supplied directly from the PCI9050 chip.
5.8 Interface Flip Flops

U59: This is the 1x20 tri-state buffer between PCI address bus and pointer memory address bus. The enable signal, FF_ENABLE2, is supplied from the FPGA.

U51: This is the 1x20 tri-state buffer between PCI data bus and pointer memory data bus. The enable signal, FF_ENABLE6, is supplied from the FPGA.
U49: This is the 2x10 tri-state buffer used for two purposes. First 10-bits is used between PCI address bus and look-up table address bus. The enable signal, FF_ENABLE3, is supplied from the FPGA. The first 4 bits of the second 10-bits are used between PCI data bus, bits 20 to 23, and lookup table data bus, bits 20 to 23. The enable signal, FF_ENABLE7, which is also enable signal of the U50 module, is generated by the FPGA.

FIGURE 31. Some of the interface buffers between PCI and board PCI address-PM address(up), PCI data-PM data(down)
U50: This is the 1x20 tri-state buffer between PCI data bus, lowest 20 bits, and lookup table data bus, lowest 20 bits. The enable signal, FF_ENABLE6, is supplied from the FPGA. The enable signal, FF_ENABLE7, is generated by the FPGA.

FIGURE 32. Some of the interface between PCI bus and board. PCI address-LUT address(up), PCI data-LUT data(down)

5.9 FPGA module

U16: This is the FPGA module. The pins of the FPGA will be explained here:

PCI_DATA_IN[19:0]: This is connected to the lowest 20-bits of the PCI data bus.
PCI_VMR_SELECT[1:0]: This 2 pins are connected to 20th and 21st bits of the PCI data bus. These are used to decode video mode registers.

ADDRESS[19:0]: This is the outputs of the address generator which is programmed in the FPGA.

LA[24:22]: These are connected to the address bus of the PCI address bus, 24th to 22nd bits. These are used to decode memory chips and enable the appropriate enable signals.

PCI_CS: This is the CS# pin of the PCI9050 RDK. It is an input to the FPGA.

PCI_WR: This pin is LW/R pin of the PCI9050 RDK. It is an input to the FPGA.

Also, two extra pins are added using fly-wires:

PCI_RD: RD# signal of the PCI9050 RDK. It is an input to the FPGA.

PCI_WR: WR# signal of the PCI9050 RDK. It is an input to the FPGA.

FF_ENABLE[0:18]: These are enable signals for the pipeline registers and tri-state buffers on the board.

CS_MEM[0:3]: These are chip select signals for the memories. See Section 3.6.2.

OE_MEM[0:3]: These are “output enable” signals for the memories. See Section 3.6.2.

WE_MEM[0:3]: These are “write enable” signals for the memories. See Section 3.6.2.

HSYNC: “Horizontal sync” output. This goes to “0” whenever the system is in horizontal sync.

HBLANK: Horizontal blank output. This goes to “1” whenever the system is in horizontal blank. (Warning! It is not “0”!)

VSYNC: Vertical Sync output. This goes to “0” whenever the system is in vertical sync.

VBLANK: Vertical blank output. This goes to “1” whenever the system is in vertical blank.

BLANK: Composite blank. This goes to “1” when the system is either in vertical or horizontal blank.

SYNC: Composite sync. This goes to “0” when the system is either in vertical or horizontal sync.

D_BLANK: Inverted and 2-clock cycle delayed version of the BLANK signal. This directly goes to video DAC. 2-clock cycle delay is implemented to handle the pipelining.

D_SYNC: 2-clock cycle delayed version of the SYNC signal. This directly goes to video DAC. 2-clock cycle delay is implemented to handle the pipelining.

LRESET: Local Reset Signal. This signal is generated by the PCI9050 RDK. Whenever it is “0” a “reset” is issued.

Clock_f: Input clock.

TCK, TMS, TDI, TDO: These are reserved pins for programming the FPGA. Please refer to the “Altera ByteBlaster Data Sheet”, Appendix 7, for using these pins. An extra circuit
is implemented to pull these pins to VCC level. The new design should have these pins pulled up through a resistor.

VCCINT: +5V power supply.

VCCIO: +3.3V power supply.

JP6: The header for the “ByteBlaster” cable. This is used for programming the FPGA. Warning: Use pull-up resistors to prevent TCK, TMS, TDI, and TDO signals to be left floating. (Please refer to “ByteBlaster” data sheet, Appendix 7)

JP4: The synchronization signals are also supplied to this header for wire-up purposes. (In fact, SYNC is used to synchronize the camera!)

U53 and U54: These are connected to test headers of PCI9050RDK.

JP9: This is used to enable/disable local clock connection to the PCI9050RDK.

5.10 Interface between FPGA address generator and memories

These three modules are used to pipeline the address generator of the FPGA and the address buses of the memories.

U38: This is the interface flip flop between ADDRESS[19:0] output of the FPGA and VRAM1 address bus.
FIGURE 33. Interface between FPGA address output and VRAM2 address bus.

U39: This is the interface flip flop between ADDRESS[19:0] output of the FPGA and VRAM2 address bus.
U40: This is the interface flip flop between ADDRESS[19:0] output of the FPGA and pointer memory address bus.

FIGURE 34. Interface between FPGA’s address output and VRAM2 address bus.
6.0 Layout

The design was laid out as a 6-layer circuit board by using Orcad LayoutPlus version 7.1. Two of the inner layers are +5V power and ground. An attempt was made to run busses on inner layers and control signals on outer layers since if errors were detected, they would be more likely to be in the control signals and it would be easier to change these if they were accessible. The netlist obtained by using Orcad Capture’s netlist generator and transferred into Layout view. The design steps for the layout generations is as follows:

- The board outlines and other constraints was entered.

- The footprints of the components were selected from the footprint library. For the components that did not have a footprint in the library a custom footprint library entry was created.

- The components are placed on to the board manually by considering the required density within the nets.

- The board was routed by using an autorouter tool, SmartRoute Ver. 7.1. But, since the density of the nets were too large the autorouter could not route the board in first
attempts. Some of the major constraints that prevent board passing the autorouting pass were via size, track width and track-to-track spacing. Since most manufacturers can only support 35-mil via diameter, 8-mil track width and 8-mil track-to-track spacing, the design was autorouted to satisfy these constraints, but it failed. After finding a manufacturer that supports smaller via sizes, track widths and track-to-track spacing, the constraints were modified to be 30-mil via size, 6-mil track size and 6-mil track-to-track spacing for signal nets. After these modifications, the autorouter routed 99% of the board and the rest of the board was routed manually.

- After passing all of the design constraints and DRC checks, design is converted into a Gerber format file and sent to a board manufacturer, Teknicircuits\(^1\), for fabrication.

### 7.0 Results

The design is manufactured and currently being tested. The whole system is verified to be working correctly except some minor noise problems.

### 8.0 Conclusions and Future Work

This circuit board is a demonstration that indicial mapping approach can be implemented to perform image processing. But a lot of improvements are required. First, another board has to made to perform at high clock cycles, possibly at XGA mode. Second, the configurability of the circuit should be improved by using a more dense FPGA and some extra circuitry.

### 9.0 References


---

\(^1\) Teknicircuits Inc., Address: 84 Shelter Rock Road, Danbury 06810-7096 CT, Phone: (203) 748-0564, Web: http://www.teknicircuits.com/
10.0 Appendices

1. VHDL Code for the FPGA units.
2. Board Schematics
3. Board Layout
4. Clock Driver QS53805 (70 k)
5. Analog to Digital Convertor ADS824 (190 k)
6. Video Digital to Analog Convertor ADV7123 (373 k)
7. Altera ByteBlaster Cable (253 k)
8. Max 7000 (2164 k)
9. SRAM MCM6926 (127 k)
10. SRAM MCM6949 (125 k)
11. PCI Interface 9050 (845 k)
12. 1x20 bit Flip Flops SN74ALVCH162721 (128 k)
13. 1x20 bit Tristate buffers SN74ALVCH162827 (126 k)
14. 2x10 bit Flip Flops SN74ALVCH16821 (137 k)
15. Voltage Regulator (555 k)
16. PCI9050-RDK data sheet
17. Windriver Data Sheet and User Guide