A HIGH PERFORMANCE GRAPHICS SYSTEM FOR THE PERSONAL COMPUTER

BY

KEVIN GARTH LEE

B.S., University of Illinois, 1989

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 1992

Urbana, Illinois
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

GRADUATE COLLEGE DEPARTMENTAL FORMAT APPROVAL

THIS IS TO CERTIFY THAT THE FORMAT AND QUALITY OF PRESENTATION OF THE THESIS
SUBMITTED BY KEVIN GARTH LEE AS ONE OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE
ARE ACCEPTABLE TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Date of Approval

Departmental Representative
ACKNOWLEDGMENTS

My deepest gratitude goes to Doctor Ricardo Uribe for his support, patience and encouragement throughout the design and building of the graphics system and for allowing me to explore so many areas of digital design. The environment he has created in the Advanced Digital Systems Laboratory has allowed so many students, including myself, to express themselves and in the process get an education.

I would like to thank Matthew Klapman for his input into the design of the system and many hours of discussion about graphics systems.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION ............................................................</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Hardware Accelerators</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Software Applications</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Graphics System</td>
<td>3</td>
</tr>
<tr>
<td>2. SYSTEM OVERVIEW ..........................................................</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Graphics Pipeline</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Digital Signal Processor Board</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Graphics Processor Board</td>
<td>6</td>
</tr>
<tr>
<td>3. GRAPHICS PIPELINE .........................................................</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Input Devices</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Application Programs</td>
<td>8</td>
</tr>
<tr>
<td>3.3 Graphics System</td>
<td>8</td>
</tr>
<tr>
<td>3.4 Output Devices</td>
<td>9</td>
</tr>
<tr>
<td>4. DIGITAL SIGNAL PROCESSOR BOARD ......................................</td>
<td>10</td>
</tr>
<tr>
<td>4.1 Digital Signal Processor</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Input Device Interfaces</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Sound Processing Unit</td>
<td>12</td>
</tr>
<tr>
<td>4.4 Memory Subsystem</td>
<td>13</td>
</tr>
<tr>
<td>4.5 AT Bus Interface</td>
<td>13</td>
</tr>
<tr>
<td>5. GRAPHICS PROCESSOR BOARD ...............................................</td>
<td>15</td>
</tr>
<tr>
<td>5.1 TMS34020 Graphics Signal Processor</td>
<td>16</td>
</tr>
<tr>
<td>5.2 TMS34082 Graphics Co-Processor</td>
<td>17</td>
</tr>
<tr>
<td>5.3 Host Interface</td>
<td>17</td>
</tr>
<tr>
<td>5.4 GSP Memory Subsystem</td>
<td>18</td>
</tr>
<tr>
<td>5.5 Video Shift Registers</td>
<td>18</td>
</tr>
<tr>
<td>5.6 Video Digital-to-Analog Converter</td>
<td>20</td>
</tr>
<tr>
<td>6. FUTURE CONSIDERATIONS ...................................................</td>
<td>21</td>
</tr>
<tr>
<td>6.1 Hardware Additions</td>
<td>21</td>
</tr>
<tr>
<td>6.2 Software Improvements</td>
<td>22</td>
</tr>
<tr>
<td>6.3 Software Applications</td>
<td>22</td>
</tr>
<tr>
<td>LIST OF REFERENCES ............................................................</td>
<td>23</td>
</tr>
</tbody>
</table>
CHAPTER 1.

INTRODUCTION

Personal computers are very prevalent in society today. The use of computers is almost mandatory in business today as well as in the academic community. As computers become more powerful, people have come to expect and even demand more from them. One area in which users desire improvement is the way they interact with the computers such as the graphical user interface which models the computer environment displaying icons and pictures that people can easily identify. For example, when the user places a file in the "trash can" on a Macintosh screen, the file is deleted. However, the graphical user interface can require an enormous amount of computing time that can otherwise be spent on what the user really wants to accomplish. Another problem with the graphical user interface is the lack of available colors. On a typical computer system, the video display board will have a palette of only 256 colors to choose from. For most applications this might be adequate but as more sophisticated applications become available and the demands to improve the user interface increase accordingly, 256 colors will soon not be enough. The goals of this thesis were to design, build, and test a high performance graphics system (HGS) for the IBM Personal Computer AT bus compatible platform that alleviates the problems with current video display boards. These problems include a limited number of colors and a lack of processing power on the video boards.

Future projects in the Advanced Digital Systems Laboratory (ADSL) will also benefit from using HGS. Students in ADSL will be able to use HGS for developing high performance graphics applications. More importantly, the methods and techniques used in building the graphics system will be available to future students. This includes
interfacing to the AT bus and prototyping high speed circuits, as well as the various circuit designs in the graphics system.

1.1 Hardware Accelerators

Currently, hardware video accelerators exist for personal computers featuring 16.8 million colors and video processors on the circuit board. However, these circuit boards are usually very expensive and use custom chips to implement the graphics routines. One of the design goals was to use widely available parts to build the graphics system. In using widely available chips, the graphics system takes advantage of lower component cost and readily available tools. Other forms of hardware accelerators include dedicated computer systems for graphics applications such as the Silicon Graphics machines. Unfortunately, these systems are much more expensive than a personal computer system and are too expensive for most people. With a high performance graphics system for a personal computer, a person can obtain excellent graphics performance at an affordable price.

1.2 Software Applications

One of the main reasons for building the HGS is to be able to display as many colors as required without palette constraints. Reasons for having many colors include requirements inspired by ray tracing packages, visual simulations, data visualization, virtual reality 3-D worlds, and high performance graphics packages. Another reason for building the HGS is to unload some of the computation work from the main processor, e.g., Intel 80386 type chip, to other processors so that a higher throughput can be obtained. For example, the main processor can send a command to HGS to draw a rectangle at a certain region of the screen as opposed to actually drawing the rectangle by
itself. Therefore, while HGS is drawing the rectangle, the main processor can execute other instructions.

1.3 Graphics System

The HGS consists of two prototyping boards connected together by two ribbon cables. The two boards reside inside a PC chassis and communicate with the PC host through the AT bus interface. The HGS features include 16.8 million colors, a dedicated graphics processor, a dedicated digital signal processor, and various input/output device interfaces. A separate computer display monitor displays the output from the graphics system. The graphics processor and the digital signal processor are both completely programmable using on-board memory to store user programs. Optimizing C compilers are available for programming the processors as well as assemblers and code generation tools. The HGS includes custom software to download the code to the graphics system and diagnostic utilities to test the functionality of the hardware.
CHAPTER 2.

SYSTEM OVERVIEW

The graphics system consists of two major components, the graphics processor board and the digital signal processor board as depicted in Figure 1. Both boards reside on the host AT bus. The digital signal processor board is the master of the graphics system but the host processor has the ability to communicate directly with the graphics processor board. This allows for more flexibility in designing any software packages for the graphics system as well as for debugging purposes. The overall design philosophy is for the digital signal processor board to receive a majority of the commands sent to HGS and then generate the appropriate commands to the graphics processor board. The
graphics processor board in this case is a complete slave device to both the digital signal processor board and the host processor. At all times, the host processor can halt or put on hold either board and examine any registers and memory locations. Data transfer between the three processors is user-defined. There are only a few hardware registers between the host processor and the digital signal processor board. All other data transfers occur in user-determined memory locations.

2.1 Graphics Pipeline

A graphics pipeline consists of input devices, application programs, a graphics system and display devices. Today, most personal computers combine most and sometimes all of this functionality into a one-processor system. For example, most IBM compatible computers use an 8086, 80286, 80386, or 80486 for the main processor (CPU). The main CPU is therefore responsible for executing all code running in the computer including executing the main application program running on the machine, polling and executing interrupt service routines for all of the input devices in the system, computing the necessary calculations to display the output, and then sending the output data to a display controller. For graphics intensive application programs, this effectively means that the main CPU in the computer is spending a large portion of computing time processing graphics routines while the application programs are sitting idle. One method of solving this bottleneck is to relieve the main CPU of some of the peripheral workload so that it can concentrate on executing the application program. The HGS addresses this bottleneck problem and actually moves most of the peripheral workload from the main CPU to secondary processors.
2.2 Digital Signal Processor Board

The digital signal processor board is built around a TMS320C30 digital signal processor (DSP). The DSP board interfaces to the graphics processor board as well as to the host AT bus. The DSP board contains a DRAM (Dynamic Random Access Memory) controller as well as the DRAM for programs and data. Also included in the DSP board are two RS-232 asynchronous serial ports, a joystick port, a sound processing unit, and a data glove interface. The serial ports can interface to any device that adheres to the RS-232 asynchronous serial interface including pointing devices such as a mouse. The serial ports can also be used to communicate with other computers or microcontrollers containing RS-232 serial ports as well as simple dumb terminals. The joystick port interfaces to two independent analog joysticks and the data glove interface connects directly to a data glove.

2.3 Graphics Processor Board

The graphics processor board is built around the TMS34020 graphics system processor (GSP). The GSP board has a host interface to the DSP board as well as to the host computer. The board consists of a GSP, a TMS34082 graphics co-processor, VRAM (Video Random Access Memory), DRAM, a video serializer, and three video digital-to-analog converters. The co-processor is tightly coupled to the GSP and speeds up floating-point calculations done by the GSP. The video memory stores the frame buffer for the display screen while the DRAM stores any programs running on the GSP. The video serializer multiplexes the pixel data from the VRAM to the video digital-to-analog converters. The video digital-to-analog converters generate the red, green, and blue color component signals to the display device such as a computer display.
CHAPTER 3.

GRAPHICS PIPELINE

The process of drawing objects on a display device can be visualized as a graphics pipeline (Figure 2). Most application programs run on the central processing unit and generate requests or commands for the graphics system. The graphics system takes these high level commands and does the actual computation needed to complete the command. After doing the computation, the graphics system generates the results on a display device. In the case of the graphical user interface, input devices also play an important role in the graphics pipeline. The computations performed by the graphics system are directly linked to the status of the input devices. Therefore, processing the input devices and transferring the data to the graphics system as fast as possible are essential to any high performance graphics system.

Figure 2. Graphics pipeline
3.1 Input Devices

Input devices play a key role in a graphics pipeline. In a simple graphics system, the input devices could be a keyboard and a mouse. More complex systems use joysticks, drawing pads, optical sensors, and light pens to manipulate the graphical objects. The current HGS configuration includes device interfaces for a mouse, a dumb terminal with keyboard, a joystick, and a data glove.

3.2 Application Programs

Application programs run on the CPU in the computer and include word processing, graphical user interfaces, CAD programs and games. These programs are what the user directly interacts with. The applications that will benefit the most from HGS are graphical user interfaces, computer games, and CAD programs. These graphics intensive programs can benefit from the computational power available on the graphics system. As computer programs become more complex and user friendly, the user interface will also have to look and feel more like our usual sensory motor interactions. However, as the user interface becomes more sophisticated, the computing power needed to generate the user interface will also grow in proportion. It is true that processors of the future will be even more powerful than those of today but so will future applications. Therefore, the concept of a graphics processing system supporting the main processor will continue to be valid in future developments of hardware and software.

3.3 Graphics System

A graphics system is responsible for controlling the output display device. Simple graphics systems found in current personal computers have only a video controller and
memory for the frame buffer. The role of the video controller is to set the display video timings and provide basic cursor controls. The frame buffer has enough memory for only one frame and at most 8-bits/pixel or 256 simultaneous colors. More sophisticated graphics systems may have an actual microprocessor on-board to help relieve the CPU's workload as well as more frame buffer memory to store multiple frames and more colors (e.g., 24-bits/pixel or 16.8 million colors). The HGS has three microprocessors on board as well as 32-bits/pixel (up to 16.8 million colors and 256 levels of transparency) to help achieve a high performance graphics system for personal computers. The HGS also makes use of special video random access memory to achieve even higher throughput from the frame buffer to the computer display.

3.4 Output Devices

Output devices allow the data flowing through the graphics pipeline to be visualized. Many types of output devices exist including computer monitors, television screens, and virtual reality head sets. Most personal computers come with a standard computer monitor with screen resolutions of 640 pixels by 480 pixels and refreshing rates of 60 Hz. Television sets use an NTSC format which uses interlaced screens to obtain resolutions of up to 512 pixels by 512 pixels and refreshing rates of 30 Hz. Currently, HGS supports standard VGA computer monitors as well as higher resolution monitors. Support for NTSC is easily added with external circuitry.
CHAPTER 4.

DIGITAL SIGNAL PROCESSOR BOARD

The digital signal processor board is centered around the TMS320C30 digital signal processor. The DSP board is used as the computation unit in the overall graphics system and also as the input devices controller (Figure 3). The DSP board incorporates various input device interfaces, a dynamic memory system, a host interface, a graphics board interface, a TMS320C30, and a sound processing unit.

Figure 3. Digital signal processor board
4.1 Digital Signal Processor

The digital signal processor on the DSP board is a Texas Instrument's TMS320C30 ('C30) (Figure 4). The TMS320C30 Digital Signal Processor is a high performance CMOS 32-bit floating-point device operating at 33 MHz. With a 60-nsec instruction cycle time, the 'C30 achieves 16 million instructions per second (MIPS) or 33 million floating-point operations per second (MFLOPS). For high performance, the 'C30 has two on-chip RAM data blocks, a sixty-four word instruction cache, and two independent external data buses. The primary data bus is used to interface to the main memory in the DSP board while the secondary data bus is used to interface to the serial ports, joysticks, and various status registers. The sound processing interface connects to
the synchronous serial port of the 'C30. Due to the fast instruction cycles and a hardware floating-point unit, the 'C30 is optimized to compute matrix operations and various transformations needed in a high performance graphics system. Refer to the *TMS320C3x User's Guide* [1] for more information about the 'C30.

### 4.2 Input Device Interfaces

The input device interfaces on the DSP board include two serial ports, two joystick ports, and a data glove interface. The device interfaces connect to the TMS320C30 through the expansion bus interface. The serial ports consist of two Texas Instrument's Asynchronous Communications Elements (TL16C550A) [2]. The TL16C550A has a programmable baud rate generator varying from 300 to 115,200 baud. The TL16C550A also generates all of the signals necessary to interface to a modem. The EIA-232 (RS-232) voltage levels are achieved by using a max232 driver/receiver interface chip. The joystick port was implemented using an Analog Devices' 4-channel, 8-bit analog-to-digital converter (AD7824) for the x and y positions and a PALCE16V8 to latch the state of the buttons on the joysticks.

### 4.3 Sound Processing Unit

The sound processing unit on the digital signal processing board is very primitive. It consists of an audio digital-to-analog converter (DAC), some discrete components and various oscillators. The main purpose of the sound processing unit is to play back stored audio samples. However, since the 'C30 has the ability to synthesize sounds, the DAC can also be used to play computer synthesized sounds.
4.4 Memory Subsystem

The memory subsystem consists of four megabytes of dynamic random access memory (DRAM) arranged as one million 32-bit words and a DRAM controller. Since the 'C30 does not directly support a DRAM interface, a controller is required to interface the 'C30 to the DRAM. The DRAM is used due to cost considerations, as large amounts of static random access memory are prohibitively expensive. The DRAM controller takes the memory control signals from the 'C30 and produces the necessary control signals needed by the DRAM. Commercially available DRAM controllers are not used due to the size of the controllers. The DRAM controllers were built using two programmable devices (PLD).

The rest of the memory system is straightforward. Multiplexers are used to multiplex the linear address from the 'C30 into a row and a column address required by the DRAM. The refresh signal for the DRAM comes from the graphics processor which itself has a DRAM controller on-chip. Line drivers are used to buffer the control signals to the DRAM.

4.5 AT Bus Interface

The AT bus interface consists of data and address buffers between the graphics board, the 'C30, and the host computer. The bus interface contains all of the necessary logic to control the data flow between the graphics processor and the digital signal processor and the data flow between both processors and the host computer. The interface can be broken down into data bus transceivers, address bus drivers, control and status registers, address bank registers, and control logic.

The data bus transceivers buffer the data lines between the graphics system and the AT-bus allowing for bidirectional data flow. Likewise, the address bus drivers buffer
the address lines coming from the AT-bus into the graphics system. The control and status registers consist of two 8-bit registers. One register stores status and control signals from the PC to be used by HGS and the other register stores status and control signals from HGS to be used by the PC.
CHAPTER 5.

GRAPHICS PROCESSOR BOARD

The graphics processor board design is based around the TMS34020 graphics system processor (Figure 5). The TMS34020 does pixel operations, serves as a video controller, and also serves as a DRAM/VRAM controller. The other major components in the graphics processor board include the video memory, the program memory, the video generation circuit, the host interface, and the graphics co-processor.

![Diagram of Graphics Signal Processor Board]

Figure 5. Graphics signal processor board
5.1 TMS34020 Graphics Signal Processor

A Texas Instrument's TMS34020 graphics signal processor (GSP) is used on the graphics processor board (Figure 6). The GSP is a fully programmable 32-bit microprocessor designed for graphics display systems. It has a 512 megabyte linear address range, dedicated pixel manipulation instructions, on-chip video CRT control, and direct support for both DRAM and VRAM. The GSP has a 32-bit external data bus and a 32-bit address bus. In addition, the GSP has a 27-bit host interface address bus for interfacing with a host processor or system. By using the TMS34020, it is possible to design a very flexible system including multiple output devices, multiple memory configurations, and even multiple resolutions without changing the hardware. In the graphics processor board, the GSP is used to control the DRAM and the VRAM and to generate the video timing signals needed. The GSP can be programmed to execute code such as line drawing routines, clear screen routines, block moves, and other graphics operations. Refer to the TMS34020 User's Guide [3] for more information about the graphics signal processor.

Figure 6. Graphics signal processor
5.2 TMS34082 Graphics Co-Processor

The TMS34082 graphics floating-point processor is a high performance 64-bit IEEE floating-point RISC processor optimized for graphics operations. The major components of the TMS34082 include a 16-bit sequencer, a three operand 64-bit floating point unit, twenty-two 64-bit data registers, and an internal ROM (Read Only Memory). Vector operations, matrix operations, and 3-D graphics routines are built into the internal ROM. Moreover, it has the capability to execute micro-coded routines stored in external static memory. The TMS34082 has two external buses for transferring data to and from the processor. One external bus is used for interfacing to the GSP and the other external bus is connected to 16K x 32 bits of external static memory. The external static memory can be used to store micro-coded routines or only data. In the current graphics processor board design, the TMS34082 is tightly coupled to the TMS34020 GSP. The TMS34082 interfaces directly to the GSP with no additional logic. For further information about the TMS34082 refer to the TMS34082 Designer's Handbook [4].

5.3 Host Interface

The GSP has a built-in host interface that allows external systems to readily interface to it. The host interface requires one programmable logic device for control signals and a set of latching bus transceivers for the data lines. Address lines connect directly to the GSP and need no further control. The host interface connects to the DSP board as well as to the AT bus through the ribbon cables that go between the DSP board and the GSP board. The built-in host interface allows an external processor to access all memory locations accessible by the GSP including all I/O registers. The host interface allows these memory accesses to occur while the GSP is executing code.
5.4 GSP Memory Subsystem

The GSP memory subsystem consists of the program memory and the frame buffers. The program memory uses four megabytes of DRAM and the frame buffers use four megabytes of VRAM. The frame buffers store the images that are ultimately sent to the computer display. Currently, there are two frame buffers, each 640 x 480 pixels, available for graphics applications such as double buffering for animation. Aside from the actual memory, the memory subsystem consists of two programmable devices (PLD) and an 8-bit latch. One PLD decodes the current memory cycle and produces the correct row address strobe (RAS) signals while the other PLD generates the correct column address strobe (CAS) signals. The 8-bit latch stores the current memory cycle status.

5.5 Video Shift Registers

Once the image to be displayed is stored in the frame buffer, the data have to be transferred from the frame buffer to a display device. The transfer of data is accomplished using video random access memory (VRAM) and custom-built video shift registers (Figure 7).

![Video shift registers](image.png)

Figure 7. Video shift registers
The data are sent pixel by pixel for each row in the frame buffer from the VRAM to the output stage of the graphics processor board. The VRAM facilitates this operation through the use of a special serial port.

The video random access memory is a dual-port access memory in which one port is identical to a normal DRAM interface and the other port is a serial access port [5]. The serial access memory port (SAM) allows rows in the memory array to be shifted out serially. For example, if there are 512 rows in memory and 1024 elements per row, then one row of 1024 elements would be shifted out of the SAM one at a time - element 1, then element 2, ..., element 1024. An important characteristic of the SAM is the high speed at which it can shift the data out. This high speed allows it to be very useful in video environments. Unfortunately, even with this high speed, for pixel resolutions above 640 by 480 even faster speeds are necessary. Since the VRAM does not come in faster speeds, one solution is to interleave the pixels coming out of the SAM or effectively get four pixels of data every memory cycle instead of one. However, the output stage of the GSP board needs the data one pixel at a time. To solve this problem, a 24-bit four-to-one multiplexer was designed using Xilinx programmable gate arrays. Instead of following a strict four-to-one multiplexer design, the fact that the pixel data followed a certain predetermined order was used to build the 24-bit four-to-one multiplexer as twenty-four 4-bit shift registers with a parallel load. By clocking the shift registers with the pixel dot clock, four 24-bit pixels were effectively serialized down to one 24-bit pixel every clock cycle. The video shift registers were built using two XC3030 programmable gate arrays.
5.6 Video Digital-to-Analog Converter

The output stage of the graphics board is the Bt121 video DAC (Digital-to-Analog Converter) from Brooktree Corporation. The Bt121 is a triple 8-bit Video DAC designed for high performance, high resolution color graphics. The Bt121 generates RS-343A compatible video signals into a doubly terminated 75 Ω load and the Bt121 also generates RS-170 compatible video signals into a singly terminated 75 Ω load. One 24-bit pixel comes from the video shift registers every dot clock cycle and is converted by the video DAC into red, green, and blue analog video signals. Blanking data are also fed to the video DAC so that the analog signals can be driven to blanking levels at the appropriate times.
CHAPTER 6.

FUTURE CONSIDERATIONS

There are many possibilities for future expansion of the current graphics system. Additions include modifying and improving upon the hardware, optimizing the low level software running on the system, and writing applications that fully use the capabilities of the system.

6.1 Hardware Additions

Most of the hardware originally designed for the graphics system was built and tested. However, a few hardware design additions were added but not implemented. One such addition involves the addition of an NTSC converter to the graphics board. This NTSC converter circuit essentially uses a single IC to convert the RGB signal to a color composite signal. The pixel dot clock would also have to be changed to accommodate the NTSC timing specifications. Aside from the NTSC converter, the hardware on the graphics board is complete.

The sound generation unit on the digital signal processor board has to be completed and tested. The circuit consists of a single IC to do the audio digital-to-analog conversion and some discrete components. Again, the proper clock signals will have to be routed to the audio circuitry. Finally, the data glove interface has to be built and tested. This interface involves a simple external register and connecting control signals from the digital signal processor to the data glove.
6.2 Software Improvements

In a graphics system of this proportion, there is always room for software improvements. A serial mouse software interface has to be written so that a mouse can be connected to one of the asynchronous serial ports on the digital signal processor board. This software would include interrupt service routines to decode the data packets coming from the mouse. A complete input/output interface to a dumb terminal has to be written. This would allow software applications to use the terminal for keyboard input into HGS without going through the host system. The I/O interface would also facilitate debugging software running on the digital signal processor as well as software running on the graphics processor. An interrupt driven interface between the host and HGS could be written so that data transfers between the host system and HGS are better supported.

Optimized code is very important in getting high performance. Optimized assembly code should be written for 3-D transformations and matrix operations. Most graphics applications use these operations most of the time and would greatly benefit from faster routines. Likewise, optimized assembly code has to be written for the graphics processor and also for the graphics co-processor.

6.3 Software Applications

Many different graphics applications can be written or ported to HGS. Some possibilities include porting DKB Trace, fractal generation programs, and virtual reality software.
LIST OF REFERENCES


APPENDIX A.

COMPONENT LAYOUT

The following two figures depict the layout of the ICs on the two circuit boards. Figure 8 shows the component layout for the digital signal processor board and Figure 9 shows the component layout for the graphics signal processor board.
APPENDIX B.

DIGITAL SIGNAL PROCESSOR BOARD SCHEMATICS

The following pages contain the complete schematics for the digital signal processor board. The schematics were generated using Orcad Schematic Design Tools and the electronic design files can be found on disks in the Advanced Digital Systems Laboratory.
### Notes

HA31 = DA23

DA23 = 0 -> Writing to DSP Memory
DA23 = 1 -> Writing to DSP Memory
APPENDIX C.

GRAPHICS SIGNAL PROCESSOR BOARD SCHEMATICS

The following pages contain the complete schematics for the graphics signal processor board. The schematics were generated using Orcad Schematic Design Tools and the electronic design files can be found on disks in the Advanced Digital Systems Laboratory.
APPENDIX D.

PROGRAMMABLE LOGIC EQUATIONS

The following pages contain the full listing of the equations used to program all of the programmable devices in the system.
;PALASM Design Description

;------------------------ Declaration Segment ---------------------------
TITLE DSP DRAM Controller I
PATTERN CNTRL1.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/16/91

CHIP U10 PAL16R8

;------------------------ PINOUT ----------------------------------------

;PIN Declarations

PIN 1 CLOCK
PIN 2 /REFREQ
PIN 3 /DSTRB
PIN 4 MEMREAD
PIN 5 DA23
PIN 6 DA22
PIN 7 /PCSTRB
PIN 8 /DHOLDA
PIN 9 GSPRDY
PIN 10 GND
PIN 11 /OE
PIN 12 /Y3 REGISTERED ; Also doubles as REFCLR
PIN 13 /Y2 REGISTERED
PIN 14 /Y1 REGISTERED
PIN 15 /Y0 REGISTERED
PIN 16 /CAS REGISTERED
PIN 17 /REN REGISTERED
PIN 18 /RAS REGISTERED
PIN 19 /RDY REGISTERED

PAL16R8:
+------+
| CLOCK | 1 20 |
|-------|
| 2  | 19  | /RDY | REG |
| 3  | 18  | /RAS | REG |
| 4  | 17  | /REN | REG |
| 5  | 16  | /CAS | REG |
| 6  | 15  | /Y0  | REG |
| 7  | 14  | /Y1  | REG |
| 8  | 13  | /Y2  | REG |
| 9  | 12  | /Y3  | REG |
| 10 | 11  | /OE  | REG |
+------+
;---------------------- String Definitions ----------------------
STRING STRB ' ( DSTRB * /DHOLDA + PCSTRB * DHOLDA ) '
STRING IDLE ' ( /Y3 * /Y2 * /Y1 * /YO ) '
STRING RASO ' ( /Y3 * /Y2 * /Y1 * YO ) '
STRING CASO ' ( /Y3 * /Y2 * Y1 * /YO ) '
STRING CAS1 ' ( /Y3 * /Y2 * Y1 * YO ) '
STRING TRP ' ( /Y3 * Y2 * /Y1 * /YO ) '
STRING PCWAIT ' ( /Y3 * Y2 * /Y1 * YO ) '
STRING REF1 ' ( /Y3 * Y2 * Y1 * /YO ) '
STRING REF2 ' ( /Y3 * Y2 * Y1 * YO ) '
STRING REF3 ' ( Y3 * Y2 * Y1 * /YO ) '

;---------------------- Boolean Equation Segment ----------------------
EQUATIONS
;
; State Equations for DRAM controller state machine
;
; Y3 is also the refresh clear signal
Y3 = REF2;

Y2 =
    IDLE * REFREQ
    + CAS0 * (DHOLDA + /DHOLDA * /MEMREAD)
    + CAS1 * REFREQ
    + REF1
    + REF2
    + TRP * REFREQ
    + PCWAIT * PCSTRB
    + PCWAIT * /PCSTRB * REFREQ

Y1 =
    IDLE * REFREQ
    + RASO
    + CAS0 * /DHOLDA * MEMREAD;
    +. CAS1 * DSTRB * /REFREQ
    + CAS1 * REFREQ
    + PCWAIT * /PCSTRB * REFREQ
    + TRP * REFREQ
    + REF1
    + REF2

Y0 =
    IDLE * STRB * /DA23 * /DA22 * /REFREQ
    + CAS0 * (DHOLDA + /DHOLDA * MEMREAD)
    + PCWAIT * PCSTRB
    + REF1

; Write enable signal
REN = STRB * /MEMREAD

; Ras signal
RAS = IDLE * STRB * /DA23 * /DA22 * /REFREQ
    + RASO

64
; Cas signal
CAS = RAS0
+ CAS0 * DHOLDA
+ CAS1 * DSTRB * /REFREQ
+ PCWAIT * PCSTRB
+ PCWAIT * /PCSTRB * REFREQ
+ IDLE * REFREQ
+ CAS1 * REFREQ
+ TRP * REFREQ
+ REF1

; Ready signal to DSP
RDY = RAS0 * /DHOLDA
+ CAS1 * DSTRB * /REFREQ
+ PCWAIT * PCSTRB
+ DA23 * GSPRDY * STRB
+ /DA23 * DA22 * STRB

;------------------------ Simulation Segment ------------------------
SIMULATION

;------------------------ Simulation Segment ------------------------
; PALASM Design Description

; ------------------------ Declaration Segment ------------------------
TITLE DSP DRAM Controller II
PATTERN CNTRL2.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/16/91

CHIP U12 PAL16R8

; ------------------------ PINOUT ----------------------------------------

;   PAL16R8
; +--------+
; | CLOCK   +-> 1 ++ 20 --> VCC |
; | /RASEN  +-> 2  19 --> /PCSTRB REG |
; | /CSEN   +-> 3  18 --> /CAS1 REG |
; | HIWORD  +-> 4  17 --> /CAS0 REG |
; | /HOLDA  +-> 5  16 --> ROWSEL REG |
; | PCSTRBIN+-> 6  15 --> /REFREQ REG |
; | REFRESH +-> 7  14 --> NC |
; | /Y2     +-> 8  13 --> NC |
; | /Y1     +-> 9  12 --> NC |
; | GND     +-> 10 11 --> /OE |
; +--------+

; ------------------------ PIN Declarations ------------------------
PIN 1  CLOCK
PIN 2  /RASEN
PIN 3  /CASEN
PIN 4  HIWORD
PIN 5  /HOLDA
PIN 6  PCSTRBIN
PIN 7  REFRESH
PIN 8  /Y2
PIN 9  /Y1
PIN 10 GND
PIN 11 /OE
PIN 15 /REFREQ REGISTERED
PIN 16 ROWSEL REGISTERED
PIN 17 /CAS0 REGISTERED
PIN 18 /CAS1 REGISTERED
PIN 19 /PCSTRB REGISTERED
PIN 20 VCC

; ------------------------ Boolean Equation Segment ------------------------
EQUATIONS
; Used to sync PCSTRB
PCSTRB = PCSTRBIN
; Used to control address multiplexing
/ROWSEL = RASEN

; CAS control signals
CAS1 = CASEN*HIWORD*HOLDA + CASEN*/HOLDA + CASEN*REFRESH*Y2*Y1
CAS0 = CASEN*/HIWORD*HOLDA + CASEN*/HOLDA + CASEN*REFRESH*Y2*Y1

; Used to sync REFRESH
REFREQ = REFRESH

;----------------------------- Simulation Segment -----------------------------
SIMULATION

;----------------------------- Simulation Segment -----------------------------

67
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE DSP/PC Host Interface Control
PATTERN DSPHST.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/16/91

CHIP U19 PAL16V8

;------------------------ Boolean Equation Segment --------------------
EQUATIONS

PIN 1 CLOCK
PIN 2 /SMEMR
PIN 3 /SMEMW
PIN 4 PCS
PIN 5 HIWORDENB
PIN 6 /DHOLDA
PIN 7 /GHOLDA
PIN 8 DSPREAD
PIN 9 GSPMEMSEL
PIN 10 GND
PIN 12 PCMEMREAD
PIN 13 /HOLDA
PIN 14 DSPEN
PIN 15 /HOSTEN
PIN 16 /PREAD
PIN 17 /PREADLO
PIN 18 PWRITE
PIN 19 PCS
PIN 20 VCC

--- PINOUT ---

PAL16V8

+----++----+
| CLOCK ++ 1 ++ 20 |+-- VCC
| /SMEMR +-- 2 |+-- 19 |+-- PCS COM
| /SMEMW +-- 3 |+-- 18 |+-- PCWRITE COM
| PCS +-- 4 |+-- 17 |+-- /PREADLOCOM
| HIWORDENB +-- 5 |+-- 16 |+-- /PREADHICOM
| /DHOLDA +-- 6 |+-- 15 |+-- /HOSTENB COM
| /GHOLDA +-- 7 |+-- 14 |+-- DSPENB COM
| DSPREAD +-- 8 |+-- 13 |+-- /HOLDA COM
| GSPMEMSEL +-- 9 |+-- 12 |+-- PCMEMREACOM
| GND +-- 10 |+-- 11 |+-- NC

--- PIN Declarations ---
PIN 1 CLOCK
PIN 2 /SMEMR
PIN 3 /SMEMW
PIN 4 PCS
PIN 5 HIWORDENB
PIN 6 /DHOLDA
PIN 7 /GHOLDA
PIN 8 DSPREAD
PIN 9 GSPMEMSEL
PIN 10 GND
PIN 12 PCMEMREAD
PIN 13 /HOLDA
PIN 14 DSPEN
PIN 15 /HOSTEN
PIN 16 /PREAD
PIN 17 /PREADLO
PIN 18 PWRITE
PIN 19 PCS
PIN 20 VCC
; PC Active

/PCS = /((SMEMR + SMEMW) * PCSEL)

/PCWRITE = /(PCSEL * SMEMW * (DHALD + GHOLDA))

PCREADLO = PCSEL * SMEMR * (DHALD + GHOLDA) * /HIWREN
PCREADHI = PCSEL * SMEMR * (DHALD + GHOLDA) * HIWREN

HOSTENB = PCSEL * /GSPMEMSEL * SMEMW * DHALDA +
          GSPMEMSEL * DSPREAD * /DHALDA * /GHOLDA

/DSPENB = /( PCSEL * /GSPMEMSEL * SMEMR * DHALDA +
            GSPMEMSEL * /DSPREAD * /DHALDA * /GHOLDA )

; Used by GSPHST to determine if DTSRB is valid
HOLDA = /GHOLDA * /HOLDA

; Tied together with DR/W
/PCMEMREAD = /(SMEMR + /SMEMW)

PCMEMREAD.TRST = DHALDA

;------------------------ Simulation Segment ---------------------------

SIMULATION

;------------------------ Simulation Segment ---------------------------
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE DSP Expansion Address Decode
PATTERN DXADDR.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 12/09/91
CHIP U8 PAL16L8

;------------------------ PINOUT ----------------------------------------
PAL16L8
+--------+
|          |
| XA12     |
| XA11     |
| XA10     |
| XA9      |
| XA8      |
| XA7      |
| XA6      |
| XA5      |
| XA4      |
| GND      |
| XA3      |
+--------+

;------------------------ PIN Declarations -----------------------------
PIN 1 XA12
PIN 2 XA11
PIN 3 XA10
PIN 4 XA9
PIN 5 XA8
PIN 6 XA7
PIN 7 XA6
PIN 8 XA5
PIN 9 XA4
PIN 10 XA3
PIN 15 /ONECS COMBINATORIAL
PIN 16 /JCS COMBINATORIAL
PIN 17 /HOSTCS COMBINATORIAL
PIN 18 /SER1CS COMBINATORIAL
PIN 19 /SER0CS COMBINATORIAL

;------------------------ String Definitions ---------------------------
STRING IOSELA "(/XA12 * /XA11 * /XA10 * /XA9 * /XA8)"
STRING IOSELB "(/XA12 * /XA11 * /XA10 * /XA9 * /XA8)"
STRING IOSELC "(/XA12 * /XA11 * /XA10 * /XA9 * /XA8)"
STRING IOSELD "(/XA12 * /XA11 * /XA10 * /XA9 * /XA8)"
STRING IOSELE "(/XA7 * /XA6 * /XA5 * /XA4 * /XA3)"

70.
EQUATIONS

; Chip select for host interface and serial ports
ONECS = ( IOSELA + IOSELB + IOSELC ) * IOSELE

; Address 0x0100-0x0108
HOSTCS = IOSELA * IOSELE

; Address 0x0200-0x0208
SERCs = IOSELB * IOSELE

; Address 0x0300-0x0308
SERICS = IOSELC * IOSELE

; Address 0x0400-0x0408
JCS = IOSELD * IOSELE

SIMULATION

;-------------------------------------------------- Simulation Segment -------------------------------------

;-------------------------------------------------- Simulation Segment -------------------------------------
;PALASM Design Description.

;------------------------ Declaration Segment ------------------------
TITLE DSP Expansion Bus Control
PATTERN DXCNTRL.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 12/09/91

CHIP U42 PAL16L8

;--------------- PINOUT -------------------------------

; PAL16L8
; +-------+
; /IOSTRB ++ 1  ++ 20 ++ VCC
; DXREAD ++ 2  19 ++ /SER0WR COM
; /SER0CS ++ 3  18 ++ /SER0RD COM
; /SER1CS ++ 4  17 ++ /SER1RD COM
; /HOSTCS ++ 5  16 ++ /SER1RD COM
; /ONECS ++ 6  15 ++ RESET COM
; /RESETIN ++ 7  14 ++ /XRDY COM
; NC ++ 8  13 ++ /DIOREAD COM
; NC ++ 9  12 ++ DIOWRITE COM
; GND ++ 10 11 ++ NC
; +-------+

;--------------- PIN Declarations -------------------------------

PIN 1 /IOSTRB
PIN 2 DXREAD
PIN 3 /SER0CS
PIN 4 /SER1CS
PIN 5 /HOSTCS
PIN 6 /ONECS
PIN 7 /RESETIN
PIN 12 DIOWRITE COMBINATORIAL
PIN 13 /DIOREAD COMBINATORIAL
PIN 14 /XRDY COMBINATORIAL
PIN 15 RESET COMBINATORIAL
PIN 16 /SER1RD COMBINATORIAL
PIN 17 /SER1WR COMBINATORIAL
PIN 18 /SER0RD COMBINATORIAL
PIN 19 /SER0WR COMBINATORIAL

;--------------- Boolean Equation Segment -------------------------------
EQUATIONS

/RESET = /RESETIN

/DIOWRITE = (/HOSTCS * /IOSTRB * /DXREAD)
DIOREAD. = HOSTCS * IOSTRB * DXREAD

XRDY = 1
XRDY.TRST = ONECS * IOSTRB.

SERORD = SER0CS * IOSTRB * DXREAD
SER0WR = SER0CS * IOSTRB */DXREAD
SER1RD = SER1CS * IOSTRB * DXREAD
SER1WR = SER1CS * IOSTRB */DXREAD.

;------------------------- Simulation Segment -------------------------
SIMULATION

;-----------------------------------------------
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE GSP Memory Controller - CAS Signals
PATTERN GSPCAS.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory.
DATE 10/27/91

CHIP U10 PAL16L8.

;------------------------ PINOUT ----------------------------------------

;----------- PIN Declarations -------------------------------
PIN 1 /CAS3 ;
PIN 2 /CAS2 ;
PIN 3 /CAS1 ;
PIN 4 /CAS0 ;
PIN 5 STAT1 ;
PIN 6 STAT0 ;
PIN 7 REFRESH ;
PIN 8 LAD[6] ;
PIN 9 LAD[5] ;
PIN 10 GND ;
PIN 11 /VCAS3 ;
PIN 12 /VCAS2 ;
PIN 13 /VCAS1 ;
PIN 14 /VCAS0 ;
PIN 15 /VCAS3 ;
PIN 16 /VCAS2 ;
PIN 17 /VCAS1 ;
PIN 18 /VCAS0 ;
PIN 19 /VCAS3 ;
PIN 20 VCC ;

;------------------------ String Definitions ----------------------------
STRING WORDO '(/LAD[6] * /LAD[5])'
STRING  WORD1  '(/LAD[6] * LAD[5])'
STRING  WORD2  '(/LAD[6] * /LAD[5])'
STRING  WORD3  '(/LAD[6] * LAD[5])'
STRING  VRAM  '(/STAT1 * /STAT0)'
STRING  DRAM  '((/STAT1 * STAT0)'
STRING  SPECIAL  '(/STAT1 * STAT0)'

;-------------------------------- Boolean Equation Segment ---------------------------------
EQUATIONS

DCAS3  =  REFRESH*CAS3 + DRAM*CAS3
DCAS2  =  REFRESH*CAS2 + DRAM*CAS2
DCAS1  =  REFRESH*CAS1 + DRAM*CAS1
DCAS0  =  REFRESH*CAS0 + DRAM*CAS0

; RAS is used to determine which WORD is active
; This allows me to write and read 16 bits values from the VRAM
; In the future I might want to be able to selectively write to the upper 8 bits of the 32 bits / pixel.
VCAS3  =  (REFRESH+SPECIAL) * CAS2 + VRAM * (CAS2) * (WORD1 + WORD3)
VCAS2  =  (REFRESH+SPECIAL) * CAS0 + VRAM * (CAS0) * (WORD1 + WORD3)
VCAS1  =  (REFRESH+SPECIAL) * CAS2 + VRAM * (CAS2) * (WORD0 + WORD2)
VCAS0  =  (REFRESH+SPECIAL) * CAS0 + VRAM * (CAS0) * (WORD0 + WORD2)

;-------------------------------- Simulation Segment ----------------------------------------
SIMULATION

;------------------------------------------------------------------------------------------------
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE GSP Host Interface Control
PATTERN GSPHST.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 10/27/91

CHIP U15 PAL22V10

;------------------------ PINOUT ------------------------

;  PAL22V10
;  +---------+
;  | CLOCK  +-- 1 ++ 24 | -- VCC
;  | /DSTRB +-- 2   23 | -- HSTREAD COM
;  | DRD_WR +-- 3   22 | -- HSTWRITEREG
;  | GSPMEM +-- 4   21 | -- DELAY REG
;  | /HOLDA +-- 5   20 | -- NC
;  | PCWRITE +-- 6   19 | -- /HCS COM
;  | /PCREADLO +-- 7  18 | -- /HREAD COM
;  | /PCREADHI +-- 8  17 | -- /HWRITE COM
;  | HRDY     +-- 9   16 | -- GSPRDY COM
;  | HDST +-- 10  15 | -- NC
;  | /HCS +-- 11  14 | -- NC
;  | GND +-- 12  13 | -- NC
;  +---------+

;------------------------ PIN Declarations ------------------------
PIN 1   CLOCK
PIN 2   /DSTRB
PIN 3   DRD_WR
PIN 4   GSPMEM
PIN 5   /HOLDA
PIN 6   PCWRITE
PIN 7   /PCREADLO
PIN 8   /PCREADHI
PIN 9   HRDY
PIN 10  HDST
PIN 11  /HCS
PIN 12  GND
PIN 15  GSPMEMSEL  REGISTERED
PIN 16  GSPRDY  COMBINATORIAL
PIN 17  /HWRITE  REGISTERED
PIN 18  /HREAD  REGISTERED
PIN 19  /HCS  REGISTERED
PIN 21  DELAY  REGISTERED
PIN 22  HSTWRITE  REGISTERED
PIN 23  HSTREAD  COMBINATORIAL
PIN 24 VCC

;------------------------ String Definitions ------------------------
STRING GSPREAD '(DSTRB * DRD_WR * HOLDA + (PCREADLO+PCREADHI))'
STRING GSPWRITE '(DSTRB * /DRD_WR * HOLDA + PCWRITE)'
STRING FCSTRB '((PCWRITE + PCREADLO + PCREADHI)'

;------------------------ Boolean Equation Segment -------------------------------
EQUATIONS
/GSPMEMSEL = /GSPMEM

; Host Read equations (PC reading from the GSP)
HSTREAD = GSPMEMSEL * GSPREAD

; These equations latches the data into the buffers from the PC/DSP writes
DELAY = GSPMEMSEL * GSPWRITE * HRDY * /HOE
HSTWRITE = DELAY

; GSP Host Chip Select
HCS = GSPMEMSEL * (DSTRB * HOLDA + FCSTRB)

; PC/DSP Read to GSP
HREAD = GSPMEMSEL * GSPREAD

; PC/DSP Write to GSP
HWRITE = GSPMEMSEL * GSPWRITE

; Signal indicating that the GSP is ready to terminate a bus cycle
OSPRDY = HRDY * HDST * /HOE

;------------------------ Simulation Segment -------------------------------
SIMULATION

;---------------------------------------------
;PALASM Design Description

;------------------------------ Declaration Segment -----------------------------
TITLE  GSP Memory Controller - RAS Signals
PATTERN GSPRAS.PDS
REVISION Rev. 1.1
AUTHOR  Kevin Lee
COMPANY  Advanced Digital Systems Laboratory
DATE 10/27/91

CHIP U9 PAL16L8

;------------------------ PINOUT ----------------------------------------

;{ PAL16L8
;  +--------+
; /RAS ++ 9   12  
; GND ++ 10  11 
; DRAMSIZE
;       +--------+

;------------------------ PIN Declarations ------------------------------
PIN 1  LAD[29]
PIN 2  LAD[27]
PIN 3  LAD[6]
PIN 4  LAD[3]
PIN 5  LAD[2]
PIN 6  LAD[1]
PIN 7  LAD[0]
PIN 8  /RAS
PIN 9  GND
PIN 10 DRAMSIZE
PIN 11 CMD
PIN 12 REFRESH
PIN 13 STAT0
PIN 14 STAT1
PIN 15 /VRAS0
PIN 16 /VRAS1
PIN 17 /DRAS0
PIN 18 /DRAS1
PIN 19 VCC

;------------------------ String Definitions ------------------------------
STRING VRAM  

STRING DRAM ' ( LAD[29] * LAD[27])'

;------------------------ Boolean Equation Segment ------------------------

EQUATIONS

; Column Address Mode Equation - Set for 1Meg X 32 Memory System
; If DRAMSIZE is true then 1 Meg x 32 else 256K x 32 Memory System
/CAMD = /(DRAM * DRAMSIZE)

; Refresh Equation
/REFRESH = /REFR

; Memory Address Decode
/STAT0 = /(DRAM + SERIAL + MASK + BLOCKWRITE)
/STAT1 = /(VRAM + SERIAL + MASK + BLOCKWRITE)

; RAS Control Signals for display memory
VRAS1 = (REFR + SERIAL + MASK + BLOCKWRITE) * RAS + VRAM * RAS * LAD[6]
VRAS0 = (REFR + SERIAL + MASK + BLOCKWRITE) * RAS + VRAM * RAS * /LAD[6]
DRAS1 = (REFR * RAS + DRAM * RAS)
DRASC = (REFR * RAS + DRAM * RAS)

;------------------------ Simulation Segment ------------------------
SIMULATION

;------------------------ Simulation Segment ------------------------
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE DSP Interrupt Control
PATTERN INTCNTRL.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 12/09/91
CHIP U16 PAL16R8

;------------------------ PINOUT ---------------------------------------

; ----------- PIN Declarations ---------------------------------------
PIN 1 CLOCK
PIN 2 DINTO
PIN 3 DINT1
PIN 4 DINT2
PIN 5 DINT3
PIN 12 /INT3Q0 REGISTERED
PIN 13 /INT3Q1 REGISTERED
PIN 14 /INT2Q0 REGISTERED
PIN 15 /INT2Q1 REGISTERED
PIN 16 /INT1Q0 REGISTERED
PIN 17 /INT1Q1 REGISTERED
PIN 18 /INT0Q0 REGISTERED
PIN 19 /INT0Q1 REGISTERED

;------------------------ Boolean Equation Segment ---------------------

EQUATIONS

INT3Q0 = .DINT3 * /INT3Q0 + INT3Q1 * /INT3Q0
INT3Q1 = .DINT3 + INT3Q1 * INT3Q0
INT2Q0 = .DINT2 * /INT2Q0 + INT2Q1 * /INT2Q0
INT2Q1 = .DINT2 + INT2Q1 * INT2Q0

80
\[ INT1Q0 = DINT1 \times INT1Q0 + INT1Q1 \times INT1Q0 \]
\[ INT1Q1 = DINT1 + INT1Q1 \times INT1Q0 \]

\[ INTOQ0 = DINT0 \times INTOQ0 + INTOQ1 \times INTOQ0 \]
\[ INTOQ1 = DINT0 + INTOQ1 \times INTOQ0 \]
; PALASM Design Description

; ------------------------ Declaration Segment ------------------------
TITLE PC I/O Port Control
PATTERN IOPORT.PDS
REVISION Rev. 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/06/91

CHIP U30 PAL22V10

; ------------------------ PINOUT ------------------------

; PAL22V10
;
----------
;
;
SA[10] ++ 2 23 ADDWRITCOM
;
SA[9] ++ 3 22 /PCIIOREADCOM
;
SA[8] ++ 4 21 PCIOWRITCOM
;
SA[7] ++ 5 20 /IOCS16 COM
;
SA[6] ++ 6 19 NC
;
SA[5] ++ 7 18 NC
;
SA[4] ++ 8 17 NC
;
SA[3] ++ 9 16 AEN
;
SA[2] ++ 10 15 /IOR
;
SA[1] ++ 11 14 /IOR
;
GND ++ 12 13 SA[0]
;
----------

; ------------------------ PIN Declarations ------------------------
PIN 1..11,13,15 SA[11..0]
PIN 12 GND
PIN 14 /IOR
PIN 15 /IOW
PIN 16 AEN
PIN 20 IOCS16 COBMINATORIAL
PIN 21 PCIOWRITE COBMINATORIAL
PIN 22 /PCIIOREAD COBMINATORIAL
PIN 23 ADDRWRIITE COBMINATORIAL
PIN 24 VCC

; I/O Addresses 0310h - 0311h for Bank Addressing and
; 0312h - 0313h for Host Control

EQUATIONS

IOCS16 = /AEN * IOSEL0 + /AEN * IOSEL1
ADDRWRITE = /AEN * IOSEL0 * IOW
PCIOREAD = /AEN * IOSEL1 * IOR
PCIOWRITE = /AEN * IOSEL1 * IOW

SIMULATION
; PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE Joystick Controller
PATTERN JOYSTICK.PDS
REVISION Rev. 1.0
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 01/28/92

CHIP U47 PALCE16V8

;------------------------ PIN Declarations ---------------------------
PIN 1   H3               ;
PIN 2   B1               ;
PIN 3   B2               ;
PIN 4   B3               ;
PIN 5   B4               ;
PIN 6   /JCS             ;
PIN 7   /IOSTRB          ;
PIN 8   RDY              ;
PIN 12  DDX8             COMBINATORIAL ;
PIN 13  DDX9             COMBINATORIAL ;
PIN 14  DDX10            COMBINATORIAL ;
PIN 15  DDX11            COMBINATORIAL ;
PIN 17  DELAY            REGISTERED ;
PIN 18  /XRDY            REGISTERED ;
PIN 19  /CS              COMBINATORIAL ;

;------------------------ Boolean Equation Segment ------------------
EQUATIONS

DDX8.TRST = JCS * IOSTRB
DDX8  = B1
DDX9.TRST = JCS * IOSTRB
DDX9  = B2
DDX10.TRST = JCS * IOSTRB
DDX10 = B3
DDX11.TRST = JCS * IOSTRB
DDX11 = B4

DELAY = JCS * IOSTRB
XRDY.TRST = JCS * IOSTRB
XRDY = DELAY

;------------------------ Simulation Segment ------------------------
SIMULATION

;------------------------
PALASM Design Description

--- Declaration Segment ---

TITLE      PC/DSP Host Interface Control
PATTERN    PCHST.PDS
REVISION   Rev. 1.1
AUTHOR     Kevin Lee
COMPANY    Advanced Digital Systems Laboratory
DATE       11/16/91

CHIP U18  PAL16V8

--- PINOUT ---

;----------------------- string Definitions  --------------------------

85
STRING: PCMEMSEL ' (SA19 * SA18 * SA17 * SA16 * /AEN)'
STRING: DSPSEL ' (DHOLDA * GSPMEMSEL)'
STRING: GSPSEL ' (GHOLDA * GSPMEMSEL)'

;-------------------- Boolean Equation Segment --------------------

EQUATIONS

; Memory Chip Select for 16-bit data transfers
/MEMCS16 = /PCMEMSEL

; IO Channel Ready Signal for IBM AT Bus
/IOCHRDY = (PCSTRB*DSPSEL*DRDY + PCSTRB*GSPSEL*GSPRDY) + /PCSTRB

; Upper 4 bits SA[19:16] = 0xD
/PCSEL = /PCMEMSEL

; Reset signal generation
/RESET = SWRESET + RESETDRV

; Enables PC address PC onto DSP-GSP bus.
/PCADDRENB = PCMEMSEL * (GHOLDA + DHOLDA)

;-------------------- Simulation Segment --------------------

SIMULATION

;--------------------
;PALASM Design Description

;------------------------ Declaration Segment ------------------------
TITLE Serial Clock Controller
PATTERN SERIAL1.PDS
REVISION Rev 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/19/91

CHIP U27 PAL16R8

;------------------------ Pinout ----------------------.-----------------

PAL16R8

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CLOCK</td>
</tr>
<tr>
<td>2</td>
<td>/VBLANK</td>
</tr>
<tr>
<td>3</td>
<td>/HBLANK</td>
</tr>
<tr>
<td>7</td>
<td>SPEED 0=25MHz, 1=64MHz</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
</tr>
<tr>
<td>12</td>
<td>Y0 REGISTERED</td>
</tr>
<tr>
<td>13</td>
<td>Y1 REGISTERED</td>
</tr>
<tr>
<td>14</td>
<td>Z0 REGISTERED</td>
</tr>
<tr>
<td>15</td>
<td>Z1 REGISTERED</td>
</tr>
<tr>
<td>17</td>
<td>/VCLK REGISTERED</td>
</tr>
<tr>
<td>18</td>
<td>XLOAD REGISTERED</td>
</tr>
<tr>
<td>19</td>
<td>SLOAD REGISTERED</td>
</tr>
<tr>
<td>20</td>
<td>VCC</td>
</tr>
</tbody>
</table>

;----------------------- String Definitions ----------------------------
STRING BLANK IN ' (HBLANK + VBLANK) ' 

;------------------------ Boolean Equation Segment ----------------------

EQUATIONS

87
; State machine equations for a divide down counters
; Just a grey scale counter (00) - (01) - (11) - (10)
; Always running
/Y1 = /(Y1 * Y0 + Y1 * Y0)
/Y0 = /(Y1 * Y0 + Y1 * Y0)

; Equation for VCLK input to 34020
; Needs to be between 62.5ns and 100ns
; VCLK = /SPEED * (Y1 * Y0 + Y1 * Y0) + SPEED * (Y1 * Y0 + Y1 * Y0)

; Equation for serial shift clock on VRAMS and load signal for the
; video pixel shifters.
; Should be 4 times the input clock frequency (dot clock).
/Z1 = /(BLANKIN * Z1 * Z0 + Z1 * Z0 + Z1 * Z0)
/Z0 = /(Z1 * Z0 + Z1 * Z0)
/SLOAD = /(Z1 * Z0 * BLANKIN)
/XLOAD = /(Z1 * Z0)

;------------------------ Simulation Segment -------------------------------
SIMULATION

;------------------------ Simulation Segment -------------------------------

88
; PALASM Design Description

;---------------------------- Declaration Segment ---------------------------
TITLE Serial Clock Controller
PATTERN SERIAL2.PDS
REVISION Rev 1.1
AUTHOR Kevin Lee
COMPANY Advanced Digital Systems Laboratory
DATE 11/19/91

CHIP U29 PAL16R8

;------------------------ Pinout -------------------------------

PAL16R8

+--------++----++----+ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
CLOCK    /
/VBLANK   +--| 1  2  19  18  17  16  15  14  13  12  11  10  9  8  7  6  5  4  3  2
/HBLANK   +--| 3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
/VSYNC    +--| 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
/HSYNC    +--| 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
/RESETIN  +--| 6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
/BLANKIN  '(HBLANK + VBLANK)

;------------------------ string Declarations ---------------------------

STRING BLANKIN '(HBLANK + VBLANK)

;------------------------ Boolean Equation Segment -----------------------

EQUATIONS
; Equation for BLANK input to Video DAC
; Delayed by 4 cycles to clear and fill pipeline
BLANKA = BLANKIN
BLANKB = BLANKA
BLANKC = BLANKB
BLANKD = BLANKC
BLANK = BLANKD

; Sync equation for the Video DAC
; 
SYNC = HSYNC + VSYNC

/RESET = /RESETIN

;------------------------ Simulation Segment ------------------------
SIMULATION

;------------------------ Simulation Segment ------------------------