Development of a Graphics Display Controller

Spring 1982

David R. Hudnall

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/634

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
DEVELOPMENT OF A GRAPHICS
DISPLAY CONTROLLER

BY

DAVID RAY HUDNALL
B.S., WEST VIRGINIA INSTITUTE OF TECHNOLOGY, 1975

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program
of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1982
DEVELOPMENT OF A
GRAPHICS DISPLAY CONTROLLER

BY
DAVID R. HUDNALL

ABSTRACT
The goal of this research is to define and develop a computer graphics display controller that will provide a high performance to cost ratio by employing state-of-the-art components and techniques. This report briefly defines the phrase "computer graphics display system" and the related technologies. The system requirements for a target display controller are developed and their hardware implications are presented. An architectural overview precedes a detailed explanation of all functional areas of the target system. This includes an overview of the VLSI graphic figure generator, which is the heart of the target system. Finally, the concept of hardware independent graphic software is presented.

Dr. Fred O. Simons, Jr.
Director of Research Report
ACKNOWLEDGEMENT

This research paper was inspired by Mr. Robert G. Willis and greatly aided by his depth of knowledge in the area of display graphics. The fabrication and debug of the prototype board were facilitated by the skill and support of Mr. James Register. I deeply thank both of these men. Finally, I owe special thanks to Dr. Fred O. Simons, Jr. for his continued support and enthusiasm.
# TABLE OF CONTENTS

INTRODUCTION ........................................... 1  

Chapter  

I. GRAPHIC BASICS AND PROJECT GOALS ............. 2  
   Graphic System Definition ........................... 2  
   Relevant Technologies ............................... 3  
   Development of Project Goals .................... 6  

II. HARDWARE DESCRIPTION ......................... 8  
   Architectural Overview ............................ 8  
   Functional Description ............................ 11  
   GDC Description .................................. 24  

III. SOFTWARE ASPECTS ............................. 32  

IV. CONCLUSION ...................................... 39  

REFERENCES .......................................... 41
INTRODUCTION

This paper deals with the generation and display of graphic figures on raster scan display systems. The goal of this research project was to develop a state-of-the-art, low cost graphics display controller. The basic principles of computer generated graphics systems are presented along with the goals of this project in Chapter 1. In Chapter 2, an architectural overview and a detailed functional description of the project's target system are given. The software aspects of computer generated graphics are examined in Chapter 3, and Chapter 4 presents the project conclusions.
CHAPTER I

GRAPHIC BASICS AND PROJECT GOALS

Graphic System Definition

The process of generating geometric figures and alphanumerical text by a sequence of computer calculations is loosely referred to as computer graphics. The area of computer graphics is one of the fastest growing market segments of the rapidly expanding computer field. The accelerating evolution of computer graphics is a result of major advances in semiconductor technology and the development of highly structured, transportable software.

Graphics display systems typically consist of four main components:

1. A host computer
2. A graphics display controller
3. A display device
4. A graphics software package

This report is focused on the design of a graphics display controller. Generally included in this function are the tasks of transforming basic descriptive
commands into digital graphic representations, processing those digital images for display, and generating all required control and video signals. Also, the close relationship of graphics display hardware and software is examined.

**Relevant Technologies**

A graphics display system may follow one of two major architectural paths. One approach is to combine a relatively simple display refresh memory/display controller hardware system with a high performance general purpose computer system executing very sophisticated software. In this approach, all calculations and image generation is performed by software and the pixel level information is downloaded into the display refresh memory. The display controller would only be responsible for reading the digital images out of display refresh memory and generating the required video signals. The alternate path is to remove the pixel by pixel calculations from the general purpose computer and provide special dedicated hardware processing in the display controller. By taking the latter approach, the performance requirements of the general purpose computer can be greatly reduced as the special, repetitive pixel by pixel calculations are efficiently
handled by hardware. A simplified block diagram of system functions is presented in Figure 1.

Graphic display systems may also be categorized by the type of display system they employ. Today, four main technologies are used: stroke writing (vector), plasma, storage tube, and raster scan. The high degree of development in the field of CRT's and their related electronic (deflection) technologies, combined with the highly ordered pixel by pixel processing techniques, have caused raster scan to become the dominant technology. The controller being considered in this paper is of the raster scan type. A basic characteristic of raster scan systems is that every display element (normally referred to as a pixel) must be refreshed every frame time which normally varies from 16 to 33 milliseconds. The resolution of the display is determined by the number of pixels per frame. A typical medium resolution display could be made up of 512 pixels per scan line with 512 scan lines per frame. Another important characteristic is that the display refresh memory (or more simply, the display memory) must contain at least one bit for each pixel location; this requires relatively large memories since typical systems range from 262,144 to 1,048,576 display pixels.
Figure 1. Typical graphics display system.
with from 4 to 24 bits per pixel (128 Kbytes to 3 Megabyte of display refresh memory).

**Development of Project Goals**

The goal of this research project is to develop a commercially desirable graphics display controller with a high performance to cost ratio. Normally considered as the minimal resolution is 4 bits per pixel and the ability to support up to a 512 by 512 display format. To be competitive from the standpoint of the state-of-the-art performance, the graphics display controller will have to contain a dedicated pixel processor to aid in graphic figure generation. From the standpoint of user acceptability, it must provide easy interface to common microprocessors and standard CRT monitors. To obtain a minimal cost factor, LSI and VLSI components must be employed and all circuitry should fit on a single, reasonably sized circuit board. Refer to Table 1 for target characteristics.
TABLE I

TARGET CONTROLLER CHARACTERISTICS

* Programmable Display format
* Support Display formats up to 262,144 pixels
* Four bits per pixel color/intensity resolution
* Support RGB Color and B&W standard monitors
* Dynamically programmable color lookup table
* Hardware blink capability with single pixel resolution
* Hardware, Zoom and Pan capability
* Hardware graphic figure processor
* CPU independent
* Single board
* Direct CPU to Display Memory Access
CHAPTER II

HARDWARE DESCRIPTION

Architectural Overview

The target graphics display controller, to be referred to as the TGDC, is a programmable hardware function capable of generating graphic characters and figures for display on raster scan CRT video monitors. Figure 2 provides a functional block diagram of the TGDC. The major TGDC functions are a Graphic Figure Generator, a one megabit Display Refresh Memory, and circuitry to generate red, green, and blue (RGB) video and all required control signals.

The TGDC is configured to interface with any CPU capable of supplying eight parallel I/O data lines and eight parallel address lines. In addition, a directly compatible Intel* Multibus* interface is provided for direct CPU access to the Display Refresh Memory. The Graphic Figure Generator, the Zoom Control register, the Color Lookup Table, and the Memory Control register are directly CPU programmable via the eight bit parallel I/O interface.

*Intel* and *Multibus* are registered trademarks of the Intel Corporation.
Figure 2. Functional block diagram of the TGDC.
Under program control the Graphics Figure Generator can generate graphic text and figures and load them into the Display Refresh Memory. The Display Refresh Memory is configured as four single bit memory planes each with 262,144 locations. The letters R, G, B and E are assigned to identify each of the four memory planes. The Plane Control Register allows multiple plane writing. The E Plane may be jumpered on board to provide a hardware blinking attribute. The video generation is performed by reading one 16 bit (pixel) word out of each of the four memory planes and parallel loading them into four 16 bit shift registers. The output of the shift registers provides a four bit address to access the Color Look Up Table. The Color Look Up Table provides three, four bit values to the three Digital to Analog Convertors which generate the red, green, and blue video outputs which can be used to drive a standard monitor. The TGDC is implemented on a single Intel SBC format circuit board.
Functional Description

The TGDC can be divided into nine major functional areas:

1. Host Interface
2. Zoom Control
3. Page Control
4. Plane Control
5. Display Refresh Control
6. Address Path Control
7. Data Path Control
8. Video Control
9. Graphic Figure Generator

Each of these functional areas will be discussed in the following paragraphs.

The TGDC provides two alternate interfaces to the host system. One of the interfaces is configured to be host independent. This universal interface is configured as eight address lines and eight input/output (I/O) data lines. The address lines could be connected to the host system's I/O address bus or simply an eight bit I/O port. The only control lines required are an I/O Read and I/O Write, which could be provided by actual host system control lines or two additional programmable I/O port lines.

The second interface is configured to be Intel Multibus compatible. The Multibus interface provides host access to all of the TGDC control functions, and, in addition, provides direct host access to the Display Memory. The Multibus interface is compatible with both
8 and 16 bit host systems. The control function's I/O addresses are jumper programmable to avoid possible contention problems in the host system's I/O address space. Similarly, the Display Memory may be mapped as an 8 or 16 Kbyte page or a contiguous 128 Kbyte segment in the host system's memory address space. The direct Display Memory access path is provided to allow rapid downloading of precompiled display scenes. Refer to Figure 3 for typical I/O space address assignment and to Figure 4 for typical memory address space assignment. The Zoom Control register, the Page Control register, and the Plane Control register are programmed by the host as I/O ports. Refer to Figure 3 for complete H/W - S/W interface definition.

The Zoom Control register in conjunction with the Graphic Figure Generator (GFG) Zoom control function enables the display to be magnified during display by a factor of one to sixteen. This is accomplished by dividing the pixel shift rate by a factor of one to sixteen to achieve horizontal zoom, and by displaying the same scan line from one to sixteen times to achieve vertical zoom. Refer to Figure 5 for typical zoom display effect.

The Page Control function provides a method for mapping the Display Memory into the host system's
Figure 3. I/O address assignment.
Figure 4. Memory address space assignment.
Figure 5. Typical display zoom effect.
memory address space. The 128 Kbytes of Display Memory can be divided into eight 16 Kbyte pages or sixteen 8 Kbyte pages. The desired section of Display Memory can be accessed by the host by loading the corresponding page number into the Page Control register. The selected page is thereby mapped into the upper 8 Kbytes or 16 Kbytes of a 64 Kbyte host memory address space depending on the page size. This feature allows CPU's which can directly access only 64 Kbytes (a common characteristic of 8 bit microprocessors) of memory to access the full 128 Kbyte Display Memory.

For systems capable of directly accessing up to 1 Megabyte of memory, the page function may be deactivated and the entire 128 Kbytes of Display Memory can be mapped into any one of the eight 128 Kbyte memory segments.

The Plane Control function allows the GFG or the host to write into more than one of the four memory planes which make up the Display Memory at the same time. This effectively allows the host system to draw or down load objects in Display Memory with any combination color and intensity in a single command sequence.

The Display Refresh Memory, which is more often simply referred to as the Display Memory, contains
1,048,576 bits (128 Kbytes) of memory. The memory is configured into four memory planes each containing 262,144 bits of memory. Each pixel location on the display frame corresponds to one bit in each of the four memory planes; therefore, together the four memory planes contain four bits of information about each pixel. The Display Memory is viewed in two different ways. During accesses to write into memory, the Display Memory is viewed as a single plane configured as 16,384 words of sixteen bits each (both the host and the GFG see only one memory space which all four planes occupy in parallel). The Plane Control register allows the host system to control which of the four actual memory planes are written into. Refer to Figure 6 for Display Memory Configuration. During accesses to read the Display Memory, it is viewed as one contiguous 64 Kword (128 Kbyte) memory with each of the four planes occupying its own unique 16 Kword (32 Kbyte) segment. During display refresh, the Video Control function must be fed one set (one pixel from each of the four planes) of pixel information every pixel time. The TGDC pixel rate is 100nS. The TGDC requirements on physical size and cost dictate the use of very dense dynamic rams: 64 Kbitrams represent the state-of-the-art today. In order to satisfy the 100nS
pixel data rate, the Display Memory is accessed by words every 1.6µS, and the output is fed into parallel to serial converters which have 100nS shift rates. One parallel to serial converter is provided for each of the four planes. Normal accessing techniques would require four memories of 16 Kwords by 16 bits each (4 × 16 = 64 16 Kbit memories). By using page mode addressing techniques, the four 16 Kword planes can be mapped into only sixteen 64 Kbit memories. During one display word time (1.6µS) one word from each of the four planes is sequentially read out of Display Memory and loaded into its respective parallel to serial converter. The skew in the four serial data streams is corrected by delay registers so as to provide four parallel bits of information per pixel to the Video Control section. The Display Memory is implemented with 64 Kbit dynamic rams. The ram refresh task is handled by the GFG during the horizontal retrace time.

The Address Path Control, APC, allows either the GFG or the Multibus interface to address the Display Memory. During any GFG or Multibus access, all four planes of the Display Memory are accessed. The APC determines from the selected address source which plane is desired and outputs only that plane or writes into each plane specified by the Plane Control register.
Figure 6. Display memory configuration.
The APC also multiplexes the selected sixteen address lines into eight lines as required by the memory IC's.

The Data Path Control, DPC, function directs the flow of data between the Display Memory and the GFG or the Multibus Interface. The DPC supports byte or word transfers and both 8 and 16 bit Multibus systems. Also included in this function are the four parallel to serial converters.

The Video Control, VC, function is responsible for converting digital pixel information into analog video and for generating the composite video sync signal. A functional diagram of the VC is given by Figure 7. The one bit of digital pixel information output from each of the four Display Memory plane's parallel to serial converters are combined to provide a four bit address (pixel code) to address the Color Look Up (CLU) table. The Color Look Up table consists of three 16 word by 4 bit high speed static rams. During display refresh, the four bit pixel code is used to address each of the rams. Each four bit ram output represents an intensity value which is used to control a digital to analog converter (DAC). One DAC is provided for each of the three colors: Red, Green, and Blue. This CLU table arrangement allows up to 16 unique color/intensity values to be displayed at the
Figure 7. Video control function.
same time out of a total of 4096 possible color/intensity values. The DAC's employed have eight bits of resolution. When the system is used to drive a monochrome (black and white) display, two CLU table rams can be used to provide eight bits of intensity information to one DAC by simply changing board jumpers. This arrangement allows up to 16 gray shade values to be displayed out of a possible range of 256 values. The Color Look Up table is initialized via the eight bit I/O interface. Each word of each of the three rams is mapped into the I/O space as an unique I/O address. Refer to Figure 8 for a typical CLU table initialization pattern.

By changing board jumpers, the E memory plane can be made to control the hardware blink function. This function provides the TGDC with a pixel level blinking attribute. When activated, the function will blink all display pixels which have their corresponding bit in the E plane set to a logic one at a one-half hertz rate.

The Graphic Figure Generator, GFG, is the heart of the TGDC. It is accessible to the host system via the eight bit I/O interface. The GFG generates control signals for DPC, APC, VC, and Display Memory. At the
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>DECIMAL</th>
<th>BINARY</th>
<th>RED</th>
<th>GREEN</th>
<th>BLUE</th>
<th>VIDEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>BLACK</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0001</td>
<td>1000</td>
<td>0000</td>
<td>0000</td>
<td>HALF INTENSITY RED</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0010</td>
<td>0000</td>
<td>1000</td>
<td>0000</td>
<td>&quot; &quot; GREEN</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0011</td>
<td>1000</td>
<td>1000</td>
<td>0000</td>
<td>&quot; &quot; YELLOW</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0100</td>
<td>0000</td>
<td>0000</td>
<td>1000</td>
<td>&quot; &quot; BLUE</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>0101</td>
<td>1000</td>
<td>0000</td>
<td>1000</td>
<td>&quot; &quot; MAGENTA</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>0110</td>
<td>0000</td>
<td>1000</td>
<td>1000</td>
<td>&quot; &quot; CYAN</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>0111</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>&quot; &quot; WHITE</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>&quot; &quot; BLACK</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1001</td>
<td>1111</td>
<td>0000</td>
<td>0000</td>
<td>FULL INTENSITY RED</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>1010</td>
<td>0000</td>
<td>1111</td>
<td>0000</td>
<td>&quot; &quot; GREEN</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>1011</td>
<td>1111</td>
<td>1111</td>
<td>0000</td>
<td>&quot; &quot; YELLOW</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>1100</td>
<td>0000</td>
<td>0000</td>
<td>1111</td>
<td>&quot; &quot; BLUE</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>1101</td>
<td>1111</td>
<td>0000</td>
<td>1111</td>
<td>&quot; &quot; MAGENTA</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>1110</td>
<td>0000</td>
<td>1111</td>
<td>1111</td>
<td>&quot; &quot; CYAN</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>&quot; &quot; WHITE</td>
</tr>
</tbody>
</table>

Figure 8. Typical Color Look Up table assignment.
heart of the GFG is a VLSI GDC (graphics display controller), the NEC µP7220.

**GDC Description**

The Graphics Display Controller, GDC, is a monolithic VLSI IC specifically designed to be the heart of high performance raster scan graphics display systems. The GDC is manufactured by Nippon Electric Company of Japan and is formally designated the µP7220. The GDC was sampled in very limited quantities in late 1981 with full production scheduled for late 1982. No other commercially available IC includes all of the features found in the GDC. The feature which makes the GDC truly represent the state-of-the-art is its graphics drawing processor which will be described later. The GDC is intended to work with any general purpose microprocessor to provide a highly efficient system by providing a dedicated hardware graphics processor.

The GDC can be divided into functional blocks which are controlled by an internal command processor and are connected by an eight bit internal data bus. Refer to Figure 9 for a block diagram of the GDC. The GDC is controlled by the host microprocessor via an eight bit bi-directional interface. An eight bit
Figure 9. GDC block diagram.
status register is provided to coordinate transfers. The GDC is assigned to two locations in the host system's I/O space; one address for command transfers and one address for parameter and status transfers. A sixteen byte command and parameter FIFO is provided to increase GDC performance. The Command Processor accepts command bytes from the FIFO and routes parameter bytes to their related functions.

The GDC instruction set consists of eighteen commands. Each single byte command is followed by a varying number of parameter bytes. The command set can be logically divided into four groups: video control, display control, drawing control, and data transfer commands. Refer to Table 2 for a complete list of commands.

The GDC draws graphics figures into Display Memory at a rate of one pixel per memory cycle. All modifications to the Display Memory are performed by a four clock period read-modify-write cycle. The TGDC clock period is 800nS which produces a pixel drawing rate of one pixel per 3.2µS. When drawing a graphics figure, the drawing processor must calculate the value and position of each pixel required for that figure. The drawing processor must be initialized by the FIGS command and its associated parameter bytes. Typical
Table 2

GDC Instruction Set

Video Control Commands
1. RESET: Resets the GDC to its idle state and specifies the video display format.
2. VSYNC: Selects master or slave video synchronization mode.
3. CCHAR: Specifies the cursor and character row heights.

Display Control Commands
1. START: Starts the display scanning process.
2. ZOOM: Specifies zoom factors for the display and graphics character writing.
3. CURS: Sets the position of the cursor in display memory.
4. PRAM: Defines starting addresses and lengths of the display areas and specifies the eight bytes for the graphics character.
5. PITCH: Specifies the width of the X dimension of display memory.

Drawing Control Commands
1. WDAT: Writes data words or bytes into display memory.
2. MASK: Sets the mask register contents.
3. FIGS: Specifies the parameters for the drawing processor.
4. FIGD: Draws the figure as specified above.
5. GCHRD: Draws the graphics character into display memory.

Data Read Commands
1. RDAT: Reads data words or bytes from display memory.
2. CURD: Reads the cursor position.
3. LPRD: Reads the light pen address.

DMA Control Commands
1. DMAR: Requests a DMA read transfer.
2. DMAW: Requests a DMA write transfer.
Figure 10. Typical FIGS command for a vector.
Figure 11. Typical PRAM command to define a graphic "R".
parameters include figure type, such as arc, circle, line, or rectangle; drawing direction; and figure dimensions such as angle, radius, or x-y lengths. Refer to Figure 10 for typical FIGS command sequence. The drawing processor algorithms and architecture are designed to optimize drawing speed.

The GDC can draw any graphic character into Display Memory which can be defined by an eight by eight pixel pattern. The host system controls the character set by initializing the GDC with a PRAM command before drawing each character. The PRAM command sequence contains eight bytes of pixel information which defines the character to be drawn. Refer to Figure 11 for typical PRAM command sequence. The GDC supports hardware Zoom of the display as previously described. It can also generate magnified graphic characters during character drawing operations. Graphic characters can be rotated in any number of 45° angle segments. Graphic figures can be rotated to any angle.

The GDC generates all basic video and memory timing and control signals which the GFG uses in generating controls for the TGDC. The host system can program the GDC to generate any display format desired (example: 640 pixels by 480 lines). The TGDC will
support both interlaced and non-interlaced displays. The GDC generates all required timing and address signals needed to handle refreshing the dynamic rams used in the Display Memory.
CHAPTER III

SOFTWARE ASPECTS

The development of graphics software which is independent of the characteristics of the processing hardware has been greatly aided by ACM Special Interest Group on Computer Graphics (SIGGRAPH) development of the CORE standards. Standardization is particularly crucial in graphics because the software which handles the interactive events and computes geometric relations is often more complicated to develop and maintain than non-graphics software. This is in part due to the fact that effective use of high performance graphics hardware often intertwines device dependent characteristics with the particular application. Without standardization, graphics software often requires complete redesign for each new application or each time new graphics hardware is used.

Basing the design of application graphics display software on the CORE standard assures development of modules which can be applied to a wide variety of applications. Partitioning software according to the CORE standard is useful for isolating and identifying
application or device specific areas requiring new design. It is also useful for determining allocation of graphic functions between hardware and software in designing environments for special devices such as the Nippon Electric Company µP7220 GDC chip. In addition to its utility in the design phase, a CORE based system can be easily interfaced with the growing number of CORE based commercial packages and public domain software appearing in graphic journals.

The ultimate output of any graphic system is a drawing or its equivalent. The input is geometric data and specifications of geometric transformations on that data. The CORE system achieves the important goal of device independence by clearly separating domains of geometric data, each characterized by the type of operation which can be performed in it. Figure 12 illustrates these domains.

At the lowest level is the physical screen upon which graphics are displayed. This is the level where the inputs are device independent primitives. The CORE system primitives include lines, characters, and graphic figures such as circles. While these are the basic building blocks from which a graphic device constructs a picture, the CORE system carefully defines them in world coordinates. This assures device
Figure 12. Data domains in the CORE system.
independence by allowing the device driver software to transform coordinates to the particular display device. Primitives may include attributes which may be regarded as flags which are set when the primitive is defined. Certain attributes might not be supported on certain display systems. In such cases, the device driver would ignore those attributes. The CORE system primitives and their associated attributes are listed in Table 3. The device driver for a CRT monitor could make direct use of the GDC instruction set to handle all of the CORE primitives. It is also at this level that graphics systems which have hardware pixel processing show their performance advantage. Device drivers which are not supported by hardware pixel processing must perform the pixel by pixel calculations via software routines.

The concept of graphics primitives identifies the basic operations of graphics software with operations which are natural to graphic devices while preserving device independence. This permits the structured design of software in a hierarchy which separates distinct functions and thereby maximizes portability, maintainability, and clarity. Figure 13 illustrates the levels of this hierarchy. The keyed borders of the symbols represent permitted interfaces
Figure 13. CORE graphic software partitioning.
between adjacent levels. At the highest level, the applications program determines the flow of interactive events, evoking the standard graphics package at the next level down. Below this level are the device drivers which are software modules which take inputs from the standard graphics package. These inputs are the standard device independent primitives. These devices are responsible for determining how to interpret attributes and for generating inputs to the specific device supported. The important concept here is that the same standard graphics package output (primitives and attributes) can be input to any device driver module.
TABLE 3
GRAPHICS PRIMITIVES AND ATTRIBUTES

<table>
<thead>
<tr>
<th>PRIMITIVE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>Color, Intensity, Thickness, Style: Solid, dashed, dotted</td>
</tr>
<tr>
<td>Character String</td>
<td>Quality (Precision): High, Medium, Low Font, Color, Intensity, Size, Path, Justification, Base, Plane, Margin</td>
</tr>
<tr>
<td>Marker (Symbol)</td>
<td>Symbol number, Color, Intensity, Thickness, Style: Solid, dashed, dotted</td>
</tr>
</tbody>
</table>

GRAPHICS PRIMITIVES:
Polyline, Circle, Arc, Ellipse

(Same attributes as for Line)
CHAPTER IV

CONCLUSION

The goal of this research project was to define and develop a graphics display system which would exhibit a high performance to cost ratio. To achieve the high level of performance desired, required the use of a VLSI graphics figure generator to perform the pixel-by-pixel calculation. This single VLSI component allows the TGDC to achieve a pixel calculation speed which could previously only be found in systems employing dedicated bit slice type processors. The 800nS per pixel calculation time is at least an order of magnitude better than normally achieved via software.

The use of the VLSI processor and the use of 64 Kbit ram in conjunction with a novel Display Memory design allows the TGDC to be implemented on a single board. This reduced component count and the single board implementation allow the TGDC to achieve a relatively low cost figure.

The TGDC compares very favorably when compared to commercially available system in both cost and size.
Comparable systems require from two to five similarly sized boards with a correspondingly higher cost.

The final point made is that a truly efficient overall system requires both efficient hardware and software. The use of a highly structured approach to software design such as that suggested by the CORE standard greatly aids the generation of efficient software.
REFERENCES


