AN EMBEDDED IMAGE GENERATION SYSTEM FOR FIELD TRAINING

Charles Wakeland
Senior Systems Engineer
Rediffusion Simulation Incorporated
2200 Arlington Downs Road
Arlington, Texas 76011

Recent advances in VLSI (Very Large Scale Integration) logic have reduced the size and improved the performance of units for the Engineering Workstation and CAD/CAM market to the point where their performance is comparable to first generation Real Time Image Generation Systems. Within narrow confines, these units may be adapted to low-end training applications which require real-time imagery with moderate update and polygon requirements. This paper describes a turnkey dual processor image generation system with payload simulation capability which is packaged in a 10 slot Multibus chassis. The modular, systems oriented, design approach which yields a 10 board unit for field use is equally applicable to other low-end simulation/training system requirements.

Introduction

The capability for training in the field is an elusive goal for any large scale system. With many large systems, the priority of creating a system which fulfills the requirements of the primary (often changing) mission(s) tends to shrink the capacity available for the training mission. Our turnkey dual processor image generation system (IG) that five years ago would fill several dedicated trucks is packed into a single chassis.

Why Field Training?

The economics and desirability of field training using the embedded simulation concept can be motivated very simply by noting that every system has a complex payload that if lost or destroyed represents a significant dollar loss. Each time a bird is flown for other than the primary mission, some "risk" is involved. This risk is greatest when personnel are making the transition between classroom and field. Integration of new crew members into existing crews represents another risk area. Lack of opportunity to train (due to bad weather for example) represents a third risk category. Field training using an embedded simulation device directly attacks this risk factor by providing a weather independent, risk free training opportunity to allow teams to integrate, to allow members to mature, and allow testing of field readiness.

Package Goals

In addition to an unusual space constraint, our IG had to be portable, rugged, powered from existing power, provide compatible video, and interface to an existing (9,600 baud) RS-232 communications link. Furthermore, the system had to meet EMI (Electro Magnetic Interference), Predicted Mean Time Between Failure and Repairability criteria. Total package weight under 100 pounds was considered ideal since that implied that it could be transported by two men.

Turnkey Solution

Based on the above criteria, Rediffusion Simulation Incorporated initially designed an eight board system housed in a double high Multibus card cage which would provide approximately 500 polygon scene capacity with studio quality RS170A black and white video output. The RS-232 link would download the system software and then provide information blocks five times per second. Based on further discussions with the end users, this proposal was modified to provide floating point hardware support and ROM (Read Only Memory) storage of the program and data bases bringing the total board count to ten boards. All of the system elements were existing, commercially available hardware. The final, ruggedized packaging is shown in Figure 1. The "turnkey" nature of the solution is such that the only external control required for the ISC is a power switch (Figure 2).

Figure 1. Ruggedized Packaging

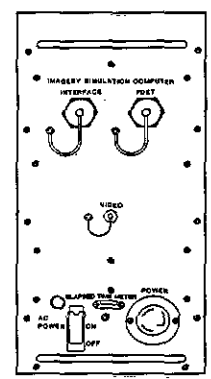

Figure 2. ISC Front Panel

System Performance

The coupled Payload Positioning System (PPS) and visual provide approximately 500 polygon instantaneous capacity with a total database in excess of 4,000 polygons. Several scenarios are supported which provides the ability to vary target position and control moving targets. The system supports target acquisition and tracking, standard search patterns, and allows complete six degree of freedom for the eyepoint.

Figure 3 is a view of the gaming area from approximately 10,000 ft. Note that the terrain includes roads, streams and hills. Some forested terrain is also included in the database.

Figure 3. Gaming Area from 10,000 ft.

Typical three dimensional targets are of 20 to 50 polygon complexity. The formation of tanks shown in figure 5 represent an enemy threat. Figure 5 is an enhanced photograph showing low level of detail to high level of detail for part of the formation. This transition is dependent upon field of view and range so that under normal operating conditions, the transition is not detectable.

System Architecture

Figure 4 is a system block diagram of the system as delivered. This dual M68000 configuration is just one example of how multi-micro systems can be configured to meet specific needs. The front-end or PPS is a dedicated simulation of the payload and uses a 10 Mega-Hertz clock speed 68010 processor and a floating point accelerator board to compute the position and orientation of the eyepoint. This information is passed to the second processor via shared memory located in the Multibus (TM) RAM card.

The second 68000 processor starts what would traditionally be the visual system. This is a modified version of the IRIS (TM) 1000 series graphics terminal from Silicon Graphics, Inc. This particular configuration uses 2 bit plane boards, a CPU, 1 Megabyte RAM Card, Update and Display controller boards and the GF1 board which contains a AMD2901 bit slice implementation of a FBC (Frame Buffer Controller) and 14 custom VLSI chips known as Geometry Engines (TM).

Figure 5. Tank Formation

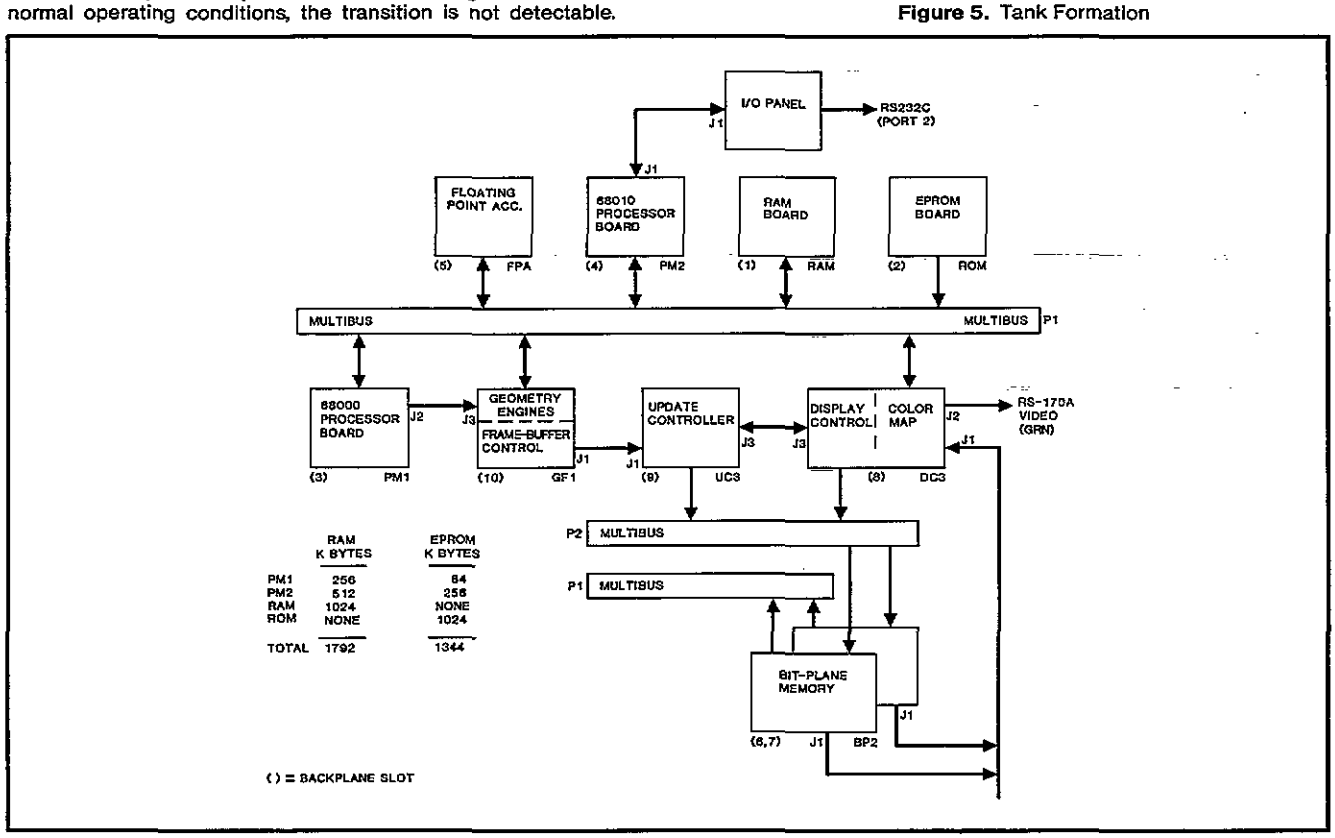

Figure 4. System Block Diagram

PPS CPU

The CPU from the Payload Positioning System is an SGI 68010 based processor board with 512K bytes of onboard RAM, sockets for up to 256K of ROM (128K used in this application), 4 serial ports and a parallel port. The RAM has on-board parity checking and is mappable using SGI's memory management hardware to appear anywhere in the processor's address space. Although the hardware supports virtual memory, this feature was not used since there is no "mass storage device" in the system.

SKYFFP (TM) – floating point accelerator board

The floating point accelerator board improves the speed of floating point operations from approximately 25 microseconds per operation to about 5 microseconds for single precision add, subtract and multiply. This is used strictly to support the PPS software.

Visual CPU

This 68000 based cpu board is used primarily as a controller for the graphics. At 500 polygons/frame or 2,500 polygons per second, 2,500 times 4 points/poly times 3 dimensions per point times 4 bytes or over 100K bytes of data must be pumped thru the pipeline every second. The board has 256K bytes of on-board RAM and up to 64K of ROM, two serial ports and a mouse-port.

GF1 – Geometry Engines

A ribbon cable connects the processor board to the start of the Geometry Pipeline (TM) on the GF1 board. The 14 GE's (figure 6) translate, rotate, scale and clip world coordinates (x, y, z, w) to determine screen coordinates. The frame buffer controller (FBC) located on this board interprets strings of coordinates and commands as either point, line or polygon primitives. The FBC then translates these graphical primitives to commands which Fill Processor (UC3) interprets. Polygons may be broken into several quadrilaterals by the FBC software to take advantage of an optimal rectangular filling algorithm.

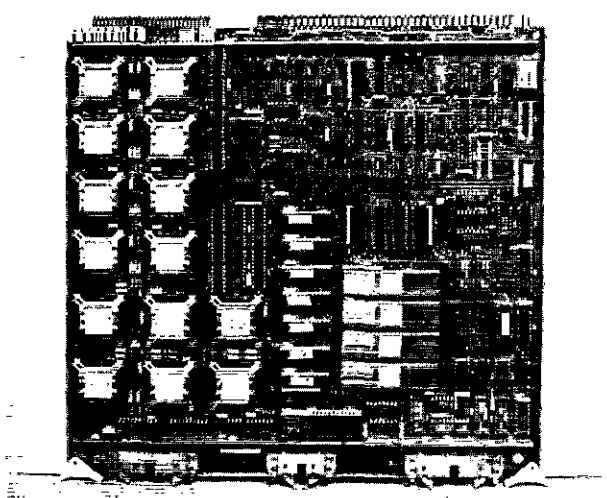

Figure 6. GF1 Board with 14 Geometry Engines

UC3 – Fill Processor

The Fill Processor implements the filling of polygons in very fast hardware, accessing up to 16 pixels in parallel. In a large area fill, 44 million pixels per second may be written. Line draw speed and filling at the edges of polygons is very orientation dependent due to the architecture of this device. Worst case, when the fill processor is hitting only 1 pixel per cycle, the 44 megapixel fill rate decays to less than 3 million per second.

BP1 and BP2 – Bitplanes

Bitplane memory comes in increments of 4 bits by 1,024 pixels by 1,024 lines per board. Since this application is black and white and typical monitor response is only 4 bits of greyscale (16 shades of gray), only 2 bitplane boards are used. They are applied by drawing an image into one board while the other board provides the data for the image seen on the screen. By swapping the role each board plays during the vertical retrace interval no glitches or flickering is apparent. In this application, the bitplanes are "ping-ponged" approximately five times per second (ie. the image update rate).

DC3 – Display Controller

One of the functions of this board is to provide all of the synchronization and timing information for the system. Numerous digital clocks criss-cross the system, and most of them are generated by this board. Since high speed state controller logic determines the raster format, the board is, in a sense, programmable. Digital to analog converters on the board provide the video output signals.

Software Architecture

All of the software developed for this application is written in C. The Silicon Graphics IRIS (TM) 1400 Workstation and Mostek Matrix 68K Development System running UNIX (TM) were used to implement and debug the code. Much of the PPS software existed as a Pascal implementation and was translated into C.

This is a multi-processor application where two independent processors execute their respective programs. These programs communicate using a block of shared memory which includes simple flags and structures to arbitrate access. The programs are independently compiled, linked and copied into EPROM. Changes to one processor's code do not necessarily imply re-compilation for both systems. In one processor, the entire application package is treated as a single program which starts with a physical reset of the processor board. In the other, a more traditional approach with an off-the-shelf operating system, the partitioning of real time and off-line software and the partitioning of tools versus application is evident. These two very different orientations co-exist on boards plugged into a common chassis and bus structure.

PPS

The PPS software is compiled as a set of independent modules, and all of the modules including bootstrap code and device drivers are linked into one executable file. The core image file is then blown into I27256A EPROMS. No operating system as such is involved in the PPS side of the system. Bootstrap code in EPROM on the PPS processor board is executed in response to a physical reset of the system. The system is initialized from code in the same ROM's, including downloading 4k of microcode to the SKYFFP (TM). The applications code (the PPS proper) is called as a subroutine from the monitor section of the bootstrap code. Note that all of the code executed by the PPS processor is contained in 4 read only memory devices (ROMs).

Two interrupt driven functions run on the PPS processor. An interrupt driven real time clock is included in this code and the serial device driver is interrupt driven. The device driver is application specific in that it detects the end of the information blocks from the host and sets both flag and pointer to end of block. The incorporation of this type of code into the ROM's dictated that the application run in a supervisory state on the 68010 microprocessor. Literally, there is no distinction between support and application code on this board.

Visual Software

Visual Software must be addressed as four separate areas: Real time software, real-time support, bootstrap, and modeling tools.

Real-Time Software One of the design goals for the overall program was to minimize the volume of real-time software. As a general rule of thumb less code implies less time implies better overall performance. The proprietary code for this application processes and displays three-dimensional objects on a two-dimensional terrain plane in correct visual priority. Part of the software calculates correct priority solutions in real time and passes the object descriptor and three space location to the code which displays the object. The system uses a "painter's algorithm" and processes objects and polygons in a farthest away first fashion. Obviously, the ground (terrain) is processed first.

All of this code makes extensive use of the IRIS (TM) Graphics Library. Code was generated to manage and prioritize objects, but the polygon display and object display routines are used directly without modification from the Graphics Library.

Real-Time Support All of the real-time graphics support software like transformation and display of three space polygons was available as part of the IRIS (TM) Terminal Programming Environment. Since library calls are the same in the remote applications (terminal) environment, standalone applications environment and UNIX (TM) System V, graphics code can easily be tested in several environments without modification.

Bootstrap As noted, an operating system (B kernel) is used to support graphics operation. Since this is a non-standard environment, special bootstrap code was created. The (up to) 1 Megabyte ROM card is treated as a mass storage device. A function which down-loads code from the ROM card is called as system initialization. After download, checksum verification and stack alignment, control is transferred to the downloaded program (B kernel). The ROM copy is created from an Ethernet downloadable file, therefore, the program may be tested before 28 EPROMS, almost 380K bytes of code, are created.

Modeling Tools

A comprehensive set of tools was created by R. Adams, of Rediffusion to bridge the gap between the modeler's point-poly-object orientation and the system implementation as "a C program". Collectively, they are referred to as the ETRAN modeling system.

Source preparation

The modeling process begins with the standard UNIX (TM) operating system utilities. The modeler creates a source file containing point and poly definitions. The last few lines of the file gather the polygons into structures called sub-objects and the sub-objects are gathered into objects, but the priority of the sub-objects is determined by a relational operator whereas the polygons are fix-listed within a sub-object. The modeler may at this time, or much later define a high and low level of detail version for the object.

The ETRAN source file is then compiled which checks all polygons for planarity, generates any points which are implied by translation, scaling, mirroring or rotation of the defined polygons. This level of compiler checking is independent of the graphical output device and may be done on a simple UNIX (TM) based system with or without graphics.

Preview

The goal of the modeling system is to provide feedback to the modeler as early as possible. Using the IRIS (TM) 1400 Workstation the object, sub-object or simply the raw polygons may be examined in 3D perspective from any postion and orientation. In Preview, the modeler "flys" around the object using keypad commands.

Once several objects are ready, and the terrain built, a version of the database is built for the workstation. All priority issues and transition range questions may be resolved at this time, completely independent of the ISC environment. While the ISC system is designed for 5 Hz. update rate operation, no constraints are placed on the operation speed of the workstation version. Its goal is to provide feedback concerning how the database looks and the priority solution.

ISC testbed

A special configuration of the ISC was used during the development process (figure 7). Packaged in a 20 slot rather than a 10 slot chassis, the testbed allowed use of additional resources like additional RAM, Ethernet support, a Multibus bus analyzer, and a floppy disk during the development.

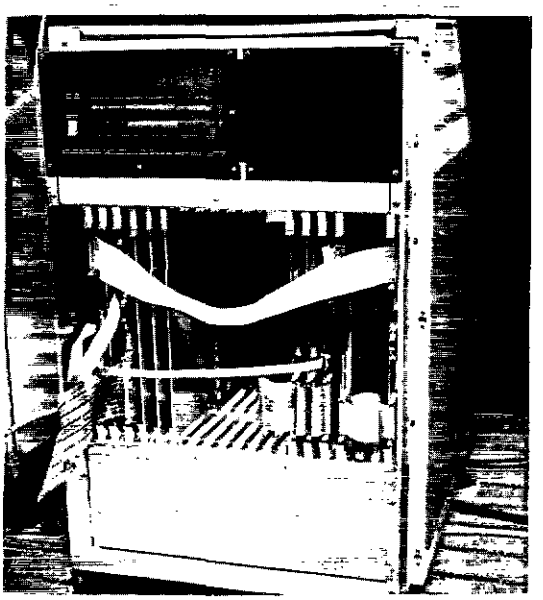

Figure 7. ISC Testbed

Once the content of the database is established, a version of the visual software is compiled for the ISC, including an object module which does inter-processor communication. This executable file is then concatenated with "B kernel" to form a "downloadable file". The visual program and database and operating system are downloaded via Ethernet as a single file. The loader in the visual processor board ROM then transfers control to the downloaded program. At this point, debug of the database timing, system performance, interface issues and all of the other elements of a system with two processors and over 512K bytes of information begins.

With Ethernet, downloading and initialization of a database takes less than 10 minutes. Programming a single I27256 EPROM takes 10 minutes on the average.

Debug

Isolated from the tools and support of the standard operating system, debug was the most difficult phase of the development. Such simple things as getting output from the program are non-trivial in the testbed environment. As an example, normally programs wait until a print statement has completed before continuing execution of the user code. In the real-time environment, assume

a 200 millisecond frame time and a 9,600 baud serial line as the output device. Even with interrupt driven I/O and good buffering, it is an error, perhaps fatal, when the user attempts to print more than approximately 200 characters in his code. The time to print 200 characters is longer than the frame time and the number of characters to print linearly increases with time. Ergo, at some point, the system will either stop processing until the characters all print, or lose output or some unpredictable combination of the two. Needless to say, status messages became short, cryptic and sporadic.

The ability to halt the system, examine core locations and restart the system proved invaluable in checking out the I/O subsystem. Examination of structures in core independently from visual and PPS processors proved the key to smoothly integrating the system. In-circuit emulation proved to be useful only in getting the debug monitor and software tools running. This applied to only the first 100 or so lines of code and certain aspects of the prom programming process.

The most consistently valuable technique in debugging the system was to simply sit down with the source code and go to work. The transition to high level languages significantly impacted the way the system was integrated. This was reinforced by the effort to prove code independently of the target environment on a module by module basis.

ROM Checkout

The final media for all programs and database information is EPROM. The file which was tested using Ethernet download is now burned into 28 I27256's. A final test of the integrated ten board system is performed.

Conclusions

Although initial skepticism concerning the approach was very high, this program has conclusively demonstrated the feasibility of tight, modular, turnkey systems for selected training tasks. The adaptability and performance of the M68000 micro when coupled to dedicated, high performance CAD/CAM oriented graphics encourages the pursuit of real-time graphics problems using similar techniques. The next generation of VLSI hardware, wherever it may come from, promises incredible advances in system capability. However, full exploitation of this increased capacity will depend upon the success of its coupling with the correct, systems oriented, design approach.

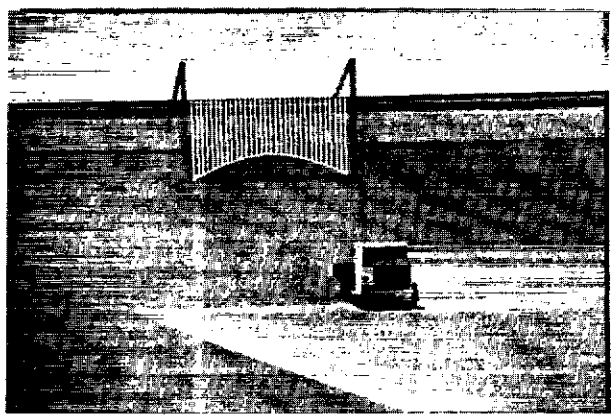

Figure 8. Recovery Truck

Acknowledgements:

The following people and organizations have contributed substantially to the technical success of this program:

- Ronald B. Adams II, Senior Systems Engineer, Rediffusion Simulation, Inc.
- Mitchell Summars, Consultant
- Bruce Borden, Member of Technical Staff, Silicon Graphics, Inc.
- Peter Broadwell, Member of Technical Staff, Silicon Graphics, Inc.