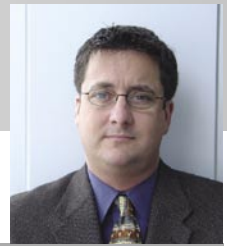


VME graphics board design considerations

By Grant Courville



After all the performance and interface issues of embedded computing are addressed, the success of a system design often comes down to the human element; data has to be displayed as pixels on a monitor that can be read by a human operator. The key question is how to get the bits on the screen.

There are a number of fundamental issues that designers need to consider when designing a graphics board including screen resolution, video mode support, graphics software support, and availability of source code. Additional issues arise from the traditional challenge of using PC market components for long-life cycle designs.

Graphics market factors

Major graphics device suppliers have increasingly looked beyond the desktop to grow revenues and mindshare, largely motivated by their shrinking presence on the PC desktop thanks to Intel's success with integrated graphics. The biggest market for high performance discrete graphics remains the gaming market where players demand the most compelling experience.

Graphics vendors are also turning to emerging markets such as mobile and cell phone displays, as phones and PDAs morph into multi-use devices. These users demand and expect quality displays for live video, web surfing, and games. The good news is that as graphics vendors search for new sources of revenue, the embedded market, with its relatively small volumes, has become increasingly attractive because it offers much higher margins, and expands their market reach.

Life cycle support

Graphics device obsolescence, always a risk when using commercial devices, is still a serious problem but it is being somewhat mitigated by an increased commitment from graphics device vendors to support the embedded market and its long-life cycle requirements. System designers need to understand how each vendor addresses this issue so they can predict if their vendor will remain a viable option for them five years later.

A key trade-off in selecting a graphics processor for an embedded application is performance versus support for long-life cycles. In recent years, graphics device manufacturers have recognized the requirement for longer product life cycles. The life cycle commitment that these vendors are willing to formalize ranges widely. All provide a last-time buy opportunity when a device has reached its End-Of-Life (EOL), but the length of time between the EOL Notice and the end of production varies significantly between vendors. But not all vendors provide die-banking services for an additional fee to ensure that a specific device architecture remains available over a long time period.

One aspect of life cycle support is the vendor's graphic device roadmap. Some vendors tout pin-to-pin compatibility over their entire graphics device line, while others may only guarantee

compatibility from one generation to the next. One approach to fighting graphics device obsolescence is to free the device design from a specific semiconductor process by moving the design to a FPGA, which have become increasingly affordable as their gate counts have increased.

Source and design availability

As graphics hardware has become standardized, source code and detailed design specification availability has become a key market differentiator. While many graphics applications in the embedded market require standard Windows drivers, a large percentage require drivers for RTOS applications such as VxWorks, INTEGRITY, LynxOS, Linux, and QNX. Development of such drivers can prove a real obstacle without access to the vendor's source code and detailed design specifications.

Another factor driving the need for source code is the requirement to meet software safety standards for critical applications. For example, RTCA DO-178B compliance is mandatory for critical civil and military aviation programs. DO-178B is the FAA safety critical software standard that establishes guidelines for software production certification. System designers must understand the device vendor's policy for source code and detailed design specification availability. They then must determine if it is adequate for their needs.

Performance requirements

The gaming market relentlessly drives graphics performance to deliver ever-advancing games to your teenager's desktop. Many embedded applications do not require this level of performance. High performance is often valued for non-technical reasons, but the consequences such as excessive heat and cost can be detrimental for embedded applications. Riding the PC market graphics wave can also expose the board vendor's design to the risks of accelerated obsolescence.

Hardware and software designers need to understand their application well enough to realistically determine if there is a requirement for high performance graphics. The requirements of many man-machine interface applications (such as console stations) are addressed with moderate performance graphics.

Graphics frameworks

Today, there are several commonly accepted graphics frameworks such as Microsoft DirectX, and open standards OpenGL and X Windows.

Originally introduced to compete against OpenGL and Silicon Graphics, DirectX has evolved into a popular and well known graphics framework which is limited to Microsoft Windows.

OpenGL was developed by Silicon Graphics as a standard 3D graphics language. Until the recent introduction of OpenGL ES, it had not been significantly changed in years. There is a public domain version called MesaGL that many feel is better than OpenGL, which, belatedly, is now open-sourced by SGI. Another development is that OpenGL common I/O device and text support has been supplemented with add-ons such as GLUT.

The use of OpenGL has greatly expanded in the embedded environment as a portable 2D graphics language as well, in which case, little if any use of the 3D capabilities are used. The goal when using OpenGL for 2D applications is portability. OpenGL allows users to develop code on their workstations, and then directly port the code to a supported embedded OS.

X Windows really is its own graphics environment. X Windows, while significantly more complex than OpenGL, offers the advantage of a much more comprehensive graphics framework. It is for this reason that X Windows was the preferred portable graphics environment early on. Now, embedded designers have now found the simpler solution of OpenGL, so X Windows has all but disappeared from use. OpenGL and X Windows also offer the advantage of developer familiarity, and as a result, complex application software time-to-market issues can be solved quicker than with a proprietary graphics language.

Now, embedded OS vendors are offering their own supported versions of OpenGL, OpenGL ES, and/or X Windows, because the distinction between the OS and the graphics/windowing system has become blurred in many user interface systems. For these systems, the graphics software is as critical as the underlying operating system because the embedded device requires a reliable user interface.

Custom libraries

Some graphics board vendors have developed their own solutions for developing graphics applications. Typically, their graphics subroutine library is intended to provide a simpler and more streamlined approach to solving basic graphics display requirements. Generally, a graphics subroutine library presents a trade-off by sacrificing device portability for simplicity. Embedded designers therefore need to consider whether the use of a custom graphics subroutine library is worth the loss in portability.

Multi-head versus Multi-display

Much confusion has resulted from attempts to differentiate multi-head graphic boards (which have two or more graphic devices), from multi-display graphics boards (which have two or more output channels).

An example of a multi-head graphics board, the Argus PMC from Curtiss-Wright Controls Embedded Computing, is shown in Figure 1 (the second Borealis graphics accelerator is on the other side). An example of a multi-display graphics board, the PMC-



Figure 1

704 Graphics PMC from Curtiss-Wright Controls Embedded Computing, is shown in Figure 2.

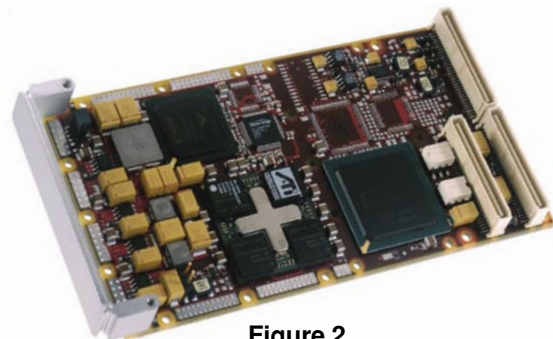


Figure 2

This is an important distinction for performance reasons. The throughput of a graphics device is often limited by its memory bandwidth. When you are driving a 1600 by 1200 by 32 bit display at 60Hz, you need 460 MBps of data access just to drive a single display. When you add a video input port, another graphics output channel, and bus accesses, you can easily exceed 1 GBps before you start your high performance graphics application.

If your graphics device uses DDR memory with a 128-bit or wider data bus, then you probably do not have a problem with multiple displays, because the systems should achieve a 3 GBps average throughput. However, if your hardware will not support such throughput, then you should scale back your expectations.

Alternatively, you can use a board that has a separate graphics engine for each display. While clearly more expensive and power hungry, this solution will provide a lot more flexibility as it obviates the concern for overtaxing the memory pipeline with too much display-related traffic. In addition, it will almost certainly give you enhanced graphics performance. Ω

Grant Courville is the Product Marketing Manager for Graphics, CompactCore, and Safety Certifiable Software for Curtiss-Wright Controls Embedded Computing (CWCEC). Based in Ottawa, Ontario, Mr. Courville has served in the embedded market for more than 18 years with experience in engineering, sales, and marketing.

For more information, contact Grant at:
Curtiss-Wright Controls Embedded Computing
333 Palladium Drive • Ottawa, ON K2V 1A6
Tel: 613-599-9199 (Ext. 5557) • Fax: 613-599-7777
E-mail: grant.courville@dy4.com
Website: www.cwcmbedded.com