NAPLPS IMPLEMENTATION: HARDWARE AND SOFTWARE ISSUES

Leo Lax, Assistant Vice-President, Corporate Marketing, NORPAK Corporation

ABSTRACT

With the acceptance of the North American Presentation Level Protocol Syntax (NAPLPS) videotex standard by major manufacturers comes the problems inherent in implementation. The primary issue in decoder design is the partitioning of implementation between hardware and software. Here the tradeoffs that result when judging which features should remain in software, which designed in hardware, are examined. Measures for judgement include cost, performance and desired features. NAPLPS features particularly problematic include text generation, color mapping, blink process, geometric primitive generation, and processing of common or often-used code. The judicious choice of placement of these features in software or hardware can ensure a successful and efficient implementation of an NAPLPS decoder.

KEYWORDS: videotex, NAPLPS, protocol, display generator, decoder, designer.

INTRODUCTION

The North American Presentation Level Protocol (NAPLPS) standard for videotex coding has been tentatively accepted by both ANSI and the Canadian Standards Association. While deliberations continue, an announcement was made at SIGGRAPH in Boston last spring that DEC, Intel, Tektronix and ten other North American computer firms had, in fact, accepted NAPLPS as a standard. These firms have since re-affirmed their commitment to producing chips, boards or intelligent terminals which incorporate NAPLPS.

The endorsement given to NAPLPS at SIGGRAPH indicates that computer makers, even before the standards bodies, have recognized the inherent advantages of using NAPLPS as a communications protocol. It is efficient for data storage, transmission, and display. Most importantly, it is device-independent. And, as more manufacturers produce NAPLPS equipment, each compatible with the other, hardware costs will certainly be reduced for the customer.

Tradeoff Issues

The protocol implies a certain level of functionality in the videotex hardware and software. As we continue to refine and develop these systems, any tradeoffs between hardware and software must be carefully defined and evaluated to ensure the provision of cost-effective and useful devices. There are two basic measures for judging the tradeoff: unit price and performance. Unit price is determined by:

a) the number of components required, ie, level of integration;
b) the memory requirements - RAM for intermediate buffers and computation, ROM for program size;
c) processing power requirements for computation, housekeeping, and I/O.

Factors affecting performance include the speed of decoder processing provided by the manufacturer and the speed of the display generator/display function.

This paper will examine five principal features of NAPLPS and discuss the implementation options available for each.

TEXT GENERATION

Traditionally, text generation hardware incorporates CRT controllers and ROM-based text fonts. The use of the CRT controller assumes fixed character size and the division of the screen into fixed character cells, with locations only on character boundaries.

NAPLPS Requirements

The service reference model (SRM) for NAPLPS, however, demands multiple fonts (with regard to Dynamically Redefinable Character Sets - DRCS) as well as "standard ASCII", proportional spacing horizontally and vertically, and multiple character paths and character rotations. The text should also be able to scroll horizontally and vertically.

Implementation Options

Table-driver character generation may be implemented in software using ROM (ASCII) or RAM (DRCS) character set tables and "copying" characters from the table into video RAM. DMA support would offer speed enhancement and offloading of processor overhead.

Another option is to use a specialized processor to generate the character font in "realtime" to reduce ROM and RAM requirements. One approach is to encode the character using "chain encoding" techniques.

The manipulation of characters is required primarily in the scroll mode; this places constraints upon the display generator. Additional video RAM must be available to provide buffer space for scrolling and smooth-scrolling circuitry.

Three scrolling implementation options are possible. One is to create software which will "copy" the scrolling characters to their new position. Another is to use raster operation techniques to copy and move a "piece" of video memory content in the displayable video memory. Finally, the designer would implement smooth scrolling in the display generator by redefining the START and STOP locations (in video RAM) of the displayable page.

COLOR MAPPING

NAPLPS Requirements

For color mapping here, a mechanism is required for indirect color assignment based on entry into a fast programmable section of RAM.

Implementation Options

All options depend upon the size of the color map table and its interface to the external world. Minimum size is 16 x 12 x 4 bits per gun and 16 entries. All other functions using color mapping are then done by software and possibly additional circuitry.

One could increase the size to 16 x n, where n is greater than 12, to provide additional status information on that "color", such as transparency state or blink process state.

One could also increase the size to m x n where m is greater than 16, n greater than 12 to increase color resolution and provide more colors, or to implement multiple screens for simple screen switching.

Interface options are also threefold. Integrate the color mapping table onto the display generator, providing access to it through set-up registers in the display generator. Design a color mapping circuit as a stand-alone device accessed by the main processor at appropriate times. Or, integrate the color mapping table with the usable video RAM space through appropriate circuitry in the RAM refresh and addressing circuitry.

BLINK PROCESSES

NAPLPS Requirement

Blink processes provide the most effective animation mechanism for NAPLPS, and is also the least costly method. NAPLPS requires the implementation of 16 independently varying blink processes (independent on-timings as well as off-timings). The blink states or "on-colors" and "off-colors" are considered to be entries in the color mapping table.

Implementation Options

The blink processes may be implemented in software, where each process is set up and then the color mapping table is modified by the main processor as necessary.

Another option is to implement a programmable "blink processor" that can be programmed with the appropriate parameters. This processor then administers the blinking process. The hardware for the processor can be integrated either with the display generator or the color mapping circuitry, depending upon the partitioning strategy of the system as a whole.

GEOMETRIC PRIMITIVE GENERATION AND THE PROCESSING OF COMMON/OFTEN-USED CODE

NAPLPS Requirement

In order to reduce the required ROM and increase display speed for an NAPLPS decoder, a set of specialized routines can be defined that are unique to NAPLPS, such as coordinate conversion routines, generation of geometric primitives (line, arc, polygon, et al), and many others.

Implementation Options

Since these routines are known and fixed, a custom processor can be designed that implements these traditionally software-based routines in hardware, utilizing micro-coded processors. In order to provide the expected improvement, the processor must be tailor-designed to interface to the display generator, video RAM, the main processor, and the I/O circuitry. Careful consideration of such design criteria can reduce parts count and can significantly improve the performance.

Some hardware implementation of geometric primitive algorithms has already been attempted. However, the designer must weigh the implied danger of implementing only those algorithms most suited to a simple processor and ignoring others. This can result in an incomplete and ultimately cumbersome NAPLPS implementation.

Again, even though improvements in space and speed can be achieved through hardware implementation of geometric primitives and often-used code, it may also be shown that the code required for unpacking of the PDI protocol when placed into LSI hardware calls for more effort than is warranted by the results.

CONCLUSIONS

When considering hardware-software tradeoffs in implementing NAPLPS, the decoder designer must clearly understand not only the standard, the technology and the software, but also the goals of the designer. In general, improvement cost, decoder performance, implementation of features and functionality are all candidates for directing the tradeoff analysis.

Once a clear understanding of the primary consideration for the application has been achieved, the appropriate partitioning of hardware and software can take place.