Memory Utilization and Data Traffic Pattern in Fujitsu Graphics Display Controller ICs
**Introduction**

Fujitsu has a successful history of developing Graphic Display Controller (GDC) products for embedded applications such as automotive information and navigation display. Beginning with the introduction of the MB86290 in 1999, the GDC product family has evolved to include the low-end, feature-rich MB86276, which was designed for the North American markets, all the way to the high-end, power-efficient MB86297.

One important design consideration for using these GDCs is graphics memory utilization. This paper will examine the traffic pattern across the different interfaces in a typical Fujitsu GDC, then identify potential bottlenecks in the system. Finally, the paper will compare the dedicated and shared architectures for graphics memory, proposing a balanced approach between the two that also optimizes system performance, cost, and board space utilization.

**GDC Graphics Memory Usage**

The display frame is essentially a subset of the drawing or logical frame(s). One of the drawing frames is designated to be shown on the display panel, either wholly or partially. The drawing engine updates the frames after drawing processing, and the modified display frame is then shown on the panel. The memory occupied by these frames is directly proportional to the resolution (horizontal * vertical pixels) and color depth that is used.

The Z-buffer data contains depth information needed for 3D drawing. The buffer has the same number of bits per pixel as the drawing frame. Each data unit in this buffer indicates how deep the corresponding pixel is located in the 3D space. Note that Fujitsu GDCs use a 3D Cartesian coordinate system that is left-handed and negative (i.e., the direction of positive z-axis is away from the viewer). The data captured from the video input interface also has to be stored temporarily in the graphics memory. That area is the Video Capture Buffer. It is a 16-bits-per-pixel area and its size is typically 2.2 times one video frame.

The Polygon drawing flag buffer contains 1 bit per pixel of information that is required while drawing polygons. In addition, the graphics memory can also store display lists, texture maps and bitmaps.

**Prioritizing Graphics Memory Access**

With such a large variety of information stored in graphics memory, the tasks utilizing this information have to be prioritized for memory access. This has already been specified for the Fujitsu GDCs, as shown below:

1. Display Frame Refresh
2. Video Capture
3. Display Processing
4. Host CPU Access (for display lists, texture maps, and bitmaps)
5. Drawing Access (Z-buffer, polygon flag buffer, etc.)
The display frame refresh updates the display panel's contents 50 or 60 times a second, or at the vertical refresh frequency rate. This task requires that a significant amount of data be transferred, which directly affects the Human Machine interactivity. Therefore, the refresh task is assigned the highest priority.

The second task, the video capture, buffers the input video data in the graphics memory. The next task, display processing, includes different processes related to the display controller, e.g., display alpha blending, overlaying, and cursor pattern processing. Then comes the host CPU access, which includes transferring display lists, texture maps, and bitmaps to the memory. The last task is the drawing access, which involves updating the drawing frames, using the Z-buffer and polygon flag buffer.

**Data Traffic Pattern in a Typical Fujitsu GDC**

Large quantities of data travel across the graphics memory interface, so the bandwidth requirement for this interface is high. A typical Fujitsu GDC has a graphics memory bandwidth of 532 Mbytes/sec. However, given the fact that one memory read or write takes multiple clock cycles, the effective bandwidth may be between one half and two thirds of this value. This calculation is done assuming a memory clock frequency of 133MHz and a data bus width of 32. The highest end GDC, the MB86297, has a memory bandwidth twice this value, or almost 1Gbytes/second. As mentioned before, the effective bandwidth will be lower than this value. The increase in the memory bandwidth of the highest end GDCs is enabled by using DDR-SDRAM technology instead of SDR-SDRAM.

As Figure 1 shows, along with the memory interface, the GDC has a CPU, video capture, and video output interfaces. Let's look at the traffic pattern on these interfaces, especially as compared with the memory interface.

Data traffic across the CPU interface is marginal compared with the memory interface. Display lists, bitmaps, and texture maps are transferred from the CPU to the GDC. The CPU might also need to access the GDC registers or memory directly. However, all these tasks do not generate a significant volume of data. In the case of a PCI host interface (33MHz), the GDC typically has a bandwidth of 50 Mbytes/sec. On the other hand, an SRAM-style host interface has a bandwidth of more than 100 Mbytes/sec. The value depends on the bus clock frequency.

Similarly, the bandwidth requirements for the video-capture and video-output interfaces are low compared with the memory interface. These interfaces are dedicated to specific tasks, with data flowing into the video capture interface and out of the video output interface. So the memory interface is the biggest bottleneck in the entire GDC system. Its architecture depends on the target application of the GDC.

**Dedicated Memory Architecture: Optimized for Performance**

If graphics performance is the top priority, the memory interface has to be dedicated. Assigning non-graphics tasks to the memory, such as using a portion of it as the host CPU's work area, compromises the bandwidth and directly affects the GDC's performance. Such a compromise has to be avoided for boosting performance. This approach has been used in the Fujitsu GDCs. The host CPU's memory remains isolated from the graphics subsystem in this scenario.

While graphics performance is an upside for this approach, the overall system cost is a downside. The system requires a separate piece of memory (and BOM cost) for the GDC and more PCB space to accommodate that extra component.
Unified Memory Architecture: Optimized for Economy
The other approach to the memory architecture is to keep it unified, so the host CPU and GDC share a single unit of memory. If the system uses separate ICs for processing and graphics, then the memory interface must be implemented in either the CPU or the GDC. For the IC that does not have a memory interface, memory data traffic has to be routed through the interface connecting it to the second IC. The scenario is depicted in Figure 4(a). In any case, such a system has a very high bandwidth requirement for the interface connecting the host CPU to the GDC.

A better and more sophisticated way to implement this type of memory architecture is to use an SoC (System-on-Chip), combining the host CPU and the GDC into one chip. The high bandwidth demand for the CPU-GDC communication can be satisfied by using a high-speed data channel within the SoC. Such a high-speed interface can be much easier to implement inside a chip rather than outside it. Thus, it is possible to satisfy the bandwidth requirements and space constraints simultaneously without sacrificing GDC performance. With today’s processing techniques, it also is possible to develop such an SOC at reasonable cost. These ICs have started to appear in the market and provide a suitable approach for addressing price-sensitive applications.

Conclusion
In summary, graphics memory is like a backbone to the GDC system. It contains very vital information that has to be transported at high speed to insure that the GDC performs the way it is supposed to. Because of that, the interface to graphics memory is a bottleneck for the system. Fujitsu GDCs do a good job at assigning correct priorities to different tasks that use the memory data. While dedicated memory architecture for graphics is extremely important for high performance, a unified architecture is essential for a cost effective implementation. Standalone GDCs with dedicated memory interface will continue to be in demand, but SOCs will also be needed to address the cost sensitive applications.

Figure 4 – Different Approaches to Unified Memory

References