HYDRA Engine ASIC Flexibility

The implementation of the HYDRA engine is done in ASIC hardware with the support of a software driver.

The ASIC is located between the Northbridge and the GPUs, as shown in figure 4.

The HYDRA engine scales performance of multi-GPU configurations from any GPU vendor and will scale relative to each GPU's individual performance. In other words, the GPUs do not need to be identical. Due to today's operating system limitations, the HYDRA solution requires the GPUs to be from the same vendor. However, the HYDRA engine is designed to support mixing and matching of different brand GPUs within a single system. As the operating systems develop to support this functionality, the HYDRA system full implementation can be realized.

The HYDRA ASIC handles all connectivity between the CPU and the GPU and between the GPUs through a full-duplex wired speed implementation. As such, the solution is connector-free and does not require any GPU to GPU connector.

This freedom of choice allows motherboard and add-in board OEMs and ODMs to design the price-performance-power systems that are best targeted for their markets. The universal and flexible approach is especially important when it comes to mainstream mass market segment.

The HYDRA system is the first to create a real alternative to the identical multi-GPU specific solutions offered by current graphic cards vendors. By eliminating the bias to any specific GPU, the HYDRA allows system builders the flexibility to design a custom system.

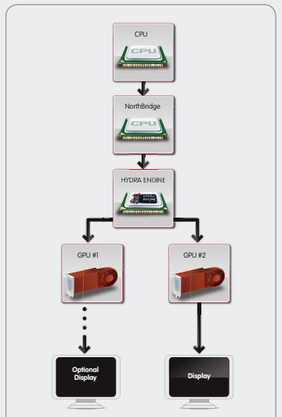

Figure 4: Lucid HYDRA Implementation

Additional Implementations

Lucid's HYDRA engine high connectivity performance and lower power consumption has been designed bottom up to load-balance GPUs for a variety of systems. The HYDRA engine is appropriate GP-GPU and GPU based applications, such as High Performance Computing (HPC), Broadcast & Film, Digital Signage, advanced Financial Simulations, Research, Life Sciences and Oil & Gas.

About Lucid

LucidLogix is a fabless semiconductor company that has developed the only universal multi-GPU solution. The LucidLogix HYDRA engine allows interoperability between different GPU solutions, dramatically simplifying the process of increasing graphics rendering power for consumer gaming and other 3D visual applications. The company's innovations are protected by more than 60 patents and patents pending. LucidLogix is a privately-owned company based in Israel and backed by Rho Ventures, Giza Venture Capital, Genesis Partners and Intel Capital.

For more information, visit www.lucidlogix.com.

CONTACTS

R&D Headquarters:

Kfar Netter Industrial Park
P.O.B. 3785 Kfar Netter 40593
Israel

T: +972-9-864-96-00
F: +972-9-885-77-85

www.lucidlogix.com

Sales & Marketing Headquarters:

5201 Great America Pkwy. Suite 32
Santa Clara, California 95054
USA

T: 408-850-7241
F: 408-850-7242

sales@lucidlogix.com

Lucid

Representations of true-to-life visual imagery is one of the most intriguing and challenging tasks in computer science. Within that filed, generating 3D scenes for computer gaming is one of the most demanding tasks on PCs and consoles today, as consumers demand richer and faster applications, in particular in the gaming field. The increased demand for rich applications leads to ever growing demand for increased processing power.

The technology's hardware core, the Graphics Processing Unit (GPUs), is one of the major factors determining computer performance in displaying 3D graphics scenes. However, GPU manufactures are still limited by die size, power and heat dissipation issues, as well as price/ performance limitations set by the market.

As a result, one of today's common solutions for upgrading performance is to use multiple GPUs to share the load of graphics performance. Not only does this maintain reasonable power consumption, it also allows consumers to upgrade existing cards by using Add-in Graphics Boards (AiBs).

The demand for scalable platforms and uncompromised visual quality is migrating from high end enthusiasts to the mainstream segments. This mass market segment is now demanding great graphics performance and at the same time expecting new solutions to be flexible, easy to deploy and maintain pricing levels.

Today's multi-GPU solutions have been developed by a select number of vendors, and require the consumer to use only identical GPUs from that particular vendor, which seriously limits consumer choice. Another obstacle is the requirement for special multi-GPU connectors. Furthermore, to install multiple GPUs, the consumer needs to be tech-savvy. For most consumers, getting the proper hardware and performing this kind of installation is beyond their technical abilities.

To overcome these obstacles, multi-GPU support needs to provide a smoother upgrade path and more flexibility for regular users. In order to enable this, a totally different approach is needed to interoperability between GPUs and to the multi-GPU enabling technology architecture.

This white paper discusses graphics processing architecture and presents a new technological approach that can enable multi-GPU processing independent of the vendor. Through this approach, consumers will be able to upgrade to multiple GPU processing and load balancing, with less complexity and without being locked into a particular vendor.

Graphics Processing Architecture Overview

The architecture of today's Graphics Processing Unit (GPU) includes the primary computing components shown in Figure 1 below

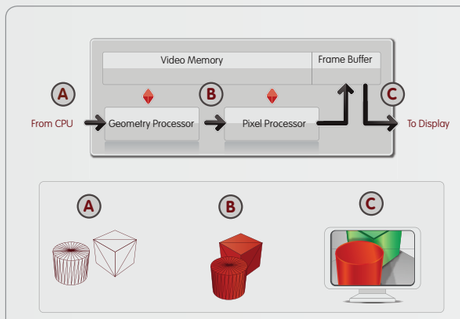

Figure 1: Standard GPU Architecture

The geometry processor (also known as the geometry shader) is responsible for processing polygons and creating the actual order between the objects, their location in the frame, perspective distortion, and partial removal of hidden polygons. The output of the processor/shader is a raster polygon. Other polygons that are in a hidden part of an object, or objects that are hidden behind other objects, are discarded.

The pixel processor (also known as the pixel shader in advanced architectures) fills each polygon with the correct texture, adding shades, lighting effects and color variations. The final output of the two processors is stored in a frame buffer memory and sent to the display.

This sequential processing of the frame creates three major potential bottlenecks:

- Geometry shader: bottleneck processing of frames where there are many changes like movements of objects or new objects appear
- Pixel shader: bottleneck of high "per-pixel-operations" such as high resolutions and anti-aliasing
- Memory capacity and access time: bottlenecks in memory capacity in major operations, for example, when large textures are being swapped

Different parallelization methods should be implemented in the various application scenarios to resolve each of the bottlenecks and allows better performance scaling. The flexibility to select the correct parallelization method in real-time that matches the application scenario is essential for getting optimized results. The selection should be such that the correct parallelization method is activated based on the current application scenario.

Parallel Graphics Processing Methods

To address the need for processing power, the GPU vendors have turned to the multi-GPU approach, similar to the multi-core approach of the general CPU and even network processors. Using this approach, any number of graphics cards can simultaneously process a single frame within an application or game. In this topology, GPUs are connected to the Northbridge via the PCIe slots and one of the graphics cards is connected to the display.

To load-balance between the GPUs, two common methods are used today:

- Split Frame/Tiling, where each GPU displays a portion of the screen space. This methodology has limited use today.
- Alternate Frame, where GPUs take turns displaying the entire screen, so each has longer to render. This is most commonly used in today's solutions

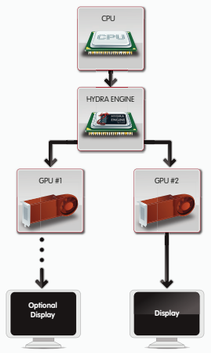

Figure 2: Basic Multi-GPU Architecture

Split Frame / Tiling

The split frame or tiling methodology is not commonly used today. When implemented, each GPU is configured to handle a specific part of the screen, for example, upper or lower part in a dual-GPU configuration. The exact positions of where the frame splits are determined dynamically according to the processing power required to process each part.

The split frame method reduces the number of pixels processed by each so that the pixel shading bottleneck is reduced. However, each GPU still needs to store in its memory the entire screen, so the geometry shader and memory bottlenecks are not affected. This memory storage activity slows down the system and constitutes a major drawback of this methodology.

Split frame/ tiling works best when there are no inter-frame dependencies and the per-pixel operation is the significant bottleneck, which is common to many of the games these days. However, it breaks down when there are other bottlenecks or inter-frame dependencies and render-to-texture techniques exist in the application.

Alternate Frame

The Alternate Frame method is the most commonly used for today's multiple-GPU solutions. In this method, each frame is assigned alternately to each GPU, such that each GPU performs the rerndering while the other GPU is rendering the previous frame. This provides more time for each GPU to render the frame. For example, in a two-GPU scenario, the first GPU handles the even (n) frames and the second GPU handles the odd (n+1) frames. The main drawback of this method is latency and scaling over two GPUs. With high frame rates, the latency is rarely noticeable.

Alternate Frame methodology performs best when each consecutive frame is well balanced, such that it takes approximately the same time to render each frame, and the GPUs are identical in their performance.

When the GPUs are not identical, or inter-frame dependencies exist in the application, this methodology tends to break down. Inter-frame dependencies are found in most of the game titles developed in the last few years.

Real Time Distributed Processing

A multi-GPU solution that will be accessible to most users should meet the following requirements:

- Allow users to choose their favorite GPU for the performance and price
- Enable choice for future system upgrades. Consumers should not have to scrap their current graphics technology or be locked-in to their existing vendor when they are looking for an add-on to their existing system.
- Eliminate the need for special or proprietary connectors
- Provides application scalability when more than one graphic card is installed.
- Allow non-identical GPUs to work in a system, thereby avoiding the need to replace both graphics cards when one is faulty or outdated.

With these requirements in mind, LucidLogix developed the Lucid HYDRA Engine. The Lucid HYDRA engine is the first dedicated silicon solution implementing real time distributed processing (RTDP) to deliver these requirements.

Load balancing in a frame & between frames

The HYDRA engine contains processes to analyze the frames before rendering and intelligently distribute the rendering tasks between the GPUs on board. The frame decision mechanism resolves bottlenecks and inter-frame dependencies prior to rendering, in real time, such that there is no additional latency.

The HYDRA engine contains a generic solution for different games, as well as rendering methods and an auto-correct load-balancing scheme for scaling. For GPUs that are not identical in performance or manufacturer; the HYDRA engine allocates the resources appropriately during processing for optimization of the GPU rendering power.

The following image (figure 3) shows the engine architecture:

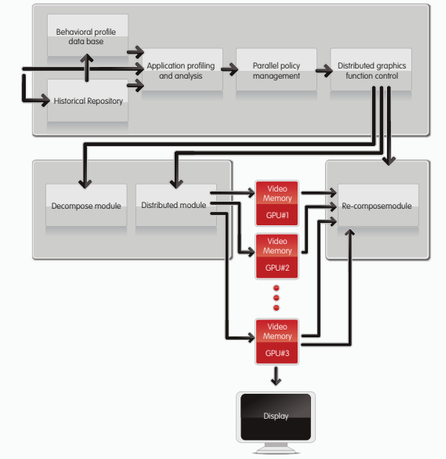

Figure 3: The Engine Architecture