Texture and Geometry Syncing

By George Bain @ SCEE Technology Group

Introduction

There has been much talk in the PlayStation®2 development community about which path is best to transfer texture data. In fact, there have been discussions in some video game magazines trying to describe this secret PATH3 that can help developers make their games run faster. Yes, there are benefits and this article will allow developers to gain a better understating of this path. I'll first give a brief introduction to the GIF, the applicable modes, texture swapping techniques, and finally operating conditions developers should be cautious about when implementing these techniques.

Data Paths to GS

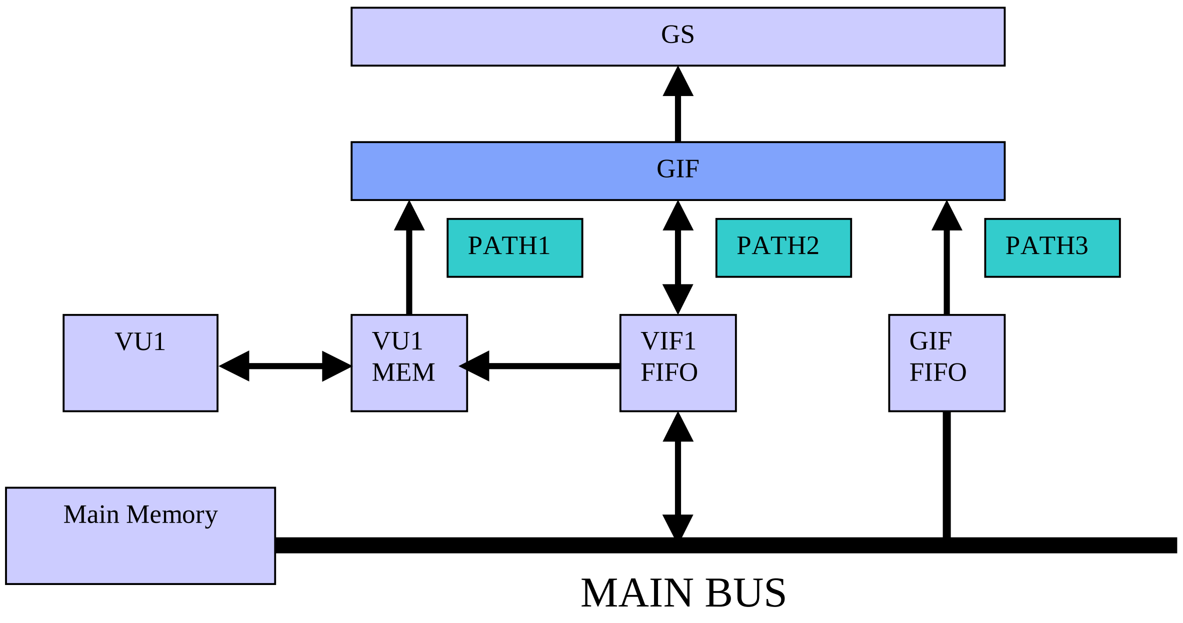
When transferring data to the GS the data must first enter the GS interface, which is known as the GIF. There are three data paths that lead to the GIF and they all serve various purposes. When a transfer of a GS packet has terminated as indicated by EOP flag in the GIF tag, the GIF will check if there is a request from the other paths.

Almost every developer uses a double buffer storage system in VU1 data memory to get the best performance. This double buffering allows the micro-program to process data and the VIF to unpack data in parallel. Once the micro-program has completed processing the input data, the output data is stored in another area of VU1 data memory. When the micro-program executes the XGKICK instruction a transfer is performed from a memory location in VU1 data memory to the GIF. This first transfer path to the GIF is called PATH1 and has the greatest priority.

Another transfer path to the GIF is PATH2. This path transfers a GS packet via the VIF1 DMA channel and through the VIF1 FIFO bypassing VU1 data memory. The DIRECT VIF code is used to indicate to the VIF that GS packet data is to follow and will be transferred to the GIF.

The last data path is PATH3 and is used to transfer a GS packet from main or scratchpad memory to the GIF.

Diagram of GS Data Paths



Texture Data Paths and GIF Operating Modes

There are two potential data paths to transfer textures to the GS but one path is exceptionally better then the other.

The second data path to the GIF is best used to set GS registers such as alpha blending and texture settings. Some developers use this path to also transfer texture data but don't realise the drawbacks that can occur. Since PATH1 and PATH2 use the same DMA channel to transfer data there is no parallelism and therefore stalls will most certainly occur. When using this path to transfer textures, you must ensure that the PATH2 transfer is complete before the micro-program executes the XGKICK instruction. This will depend on how fast your micro-program is and the size of the texture data. Nevertheless, this is tough to sync and from my experience analysing games with the Performance Analyser this is the worst possible way to transfer texture data.

Now the mysterious PATH3 and the intermittent mode! This is the best way to transfer texture data to video memory because the path can be used in two different modes when the data format specified in the GIF tag is IMAGE mode.

The default GIF mode is continuous mode whereby the GIF will check the other two paths for a request when PATH3 has finished transferring all its data as indicated by the EOP flag in the GIF tag. If there is a request from PATH1 or PATH2 they must both wait for the end of PATH3 transfer before they can start.

The other mode the GIF can run in is intermittent mode. This mode allows developers to transfer texture data via PATH3 while VIF1 and VU1 continue to process data. Instead of the GIF checking for a request from PATH1 and PATH2 at the end of PATH3 transfer, it will check every 8 quad words. When the XGKICK instruction or the DIRECT VIF code signal to the GIF that they wish to have access, PATH3 will stall and PATH1 or PATH2 will be granted access to the GIF. After either of the two PATHS has fully completed transferring their data then the GIF will allow PATH3 to regain access. The advantage gained from using this technique will of course rely on how often the other two paths are used. If developers use very fast micro-programs then PATH1 will continually occupy the GIF. This is something to keep in mind but not all micro-programs have 8 cycle rendering loops and therefore texture data will have enough time to get transferred through PATH3. The diagram below shows the data flow of the GIF running in intermittent mode.

VIF1 and VU1	PATH 1 to GIF	PATH 3 to GIF
Process (A)	Stall	Texture Transfer 8 quad words
Unpack (B)	XGKICK Packet (A) to GIF	Stall
Process (B)	Stall	Texture Transfer 8 quad words
Unpack (A)	XGKICK Packet (B) to GIF	Stall

VIF1 interrupts

The texture swapping method that I presented at last year’s developer conference was well received by many developers. The method involves using two DMA channels to achieve parallelism and using interrupts to start the texture transfer. The geometry data gets transferred to VU1 data memory via the VIF1 DMA channel and the texture data to video memory via the GIF DMA channel. When generating this geometry list in main memory, VIF codes are inserted with the high bit set to generate an interrupt. If VIF1 processes any VIF codes with this high bit set, it will enter the interrupt handler that we initialised beforehand. Inside this handler a texture transfer is started using the GIF DMA channel and therefore entering the GIF via PATH3.

After starting this texture transfer it is very important that we do not wait for the DMA transfer to complete inside the handler. The next FLUSHA VIF code in the VIF1 DMA list will wait for the texture transfer to complete. The reason why we use the FLUSHA VIF code is because it’s the only flush code that will wait for a PATH3 transfer to complete. After the texture transfer has started we simply restart VIF1 so it can continue to process the geometry list.

MSKPATH3

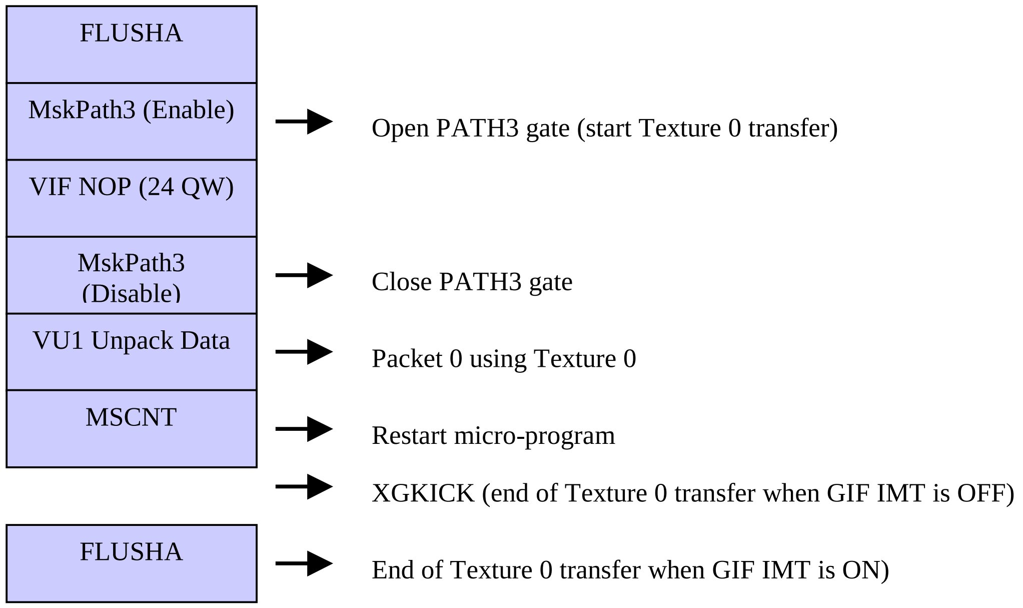
There’s an additional technique to swap textures that doesn’t involve interrupts. This method uses VIF codes MSKPATH3 to start and stop PATH3 transfer and FLUSHA to wait for PATH3 transfer in the geometry list. These VIF codes will synchronise both the geometry and texture DMA lists. The texture list is constructed with the end of packet bit set in the appropriate GIF tags so that the VIF code MSKPATH3 will know when to start and stop PATH3 transfer. The texture list can have several GIF tags with this bit set but it is important that it’s set in the GIF tag where you wish to stop transfer or both lists will get out of sync. After the texture list has been generated, the VIF codes must now be inserted into the geometry lists in strategic locations.

When the VIF processes a MSKPATH3 code without the mask bit set it will begin to transfer textures via PATH3. After processing this code you need to call the VIF code again but with the mask bit set in order to inform the GIF to stop transfer when an end of GS packet has been reached. You must make certain that you don’t open and close the PATH3 gate too quickly before the GIF has a chance to start receiving texture data. How you overcome this is by inserting a gap of 24 quad words between opening and closing a PATH3 transfer. The reason why we have such a gap is because the VIF1 FIFO is 16 quad words and a DMA slice is 8 quad words. The gap can easily be utilised with unpacked geometry data in the list so there is no need to use 24 quad words of empty data.

The FLUSHA VIF code is also inserted into the geometry list before the MSKPATH3 enable code. This ensures that the current PATH3 transfer has completed before another is started. After the texture and geometry lists have been assembled, you can start both DMA transfers and the VIF codes will sync both DMA lists.

Common mistakes

The most common mistake developers make when implementing either of these techniques is the location of the VIF1 interrupt or MSKPATH3 VIF code that starts the PATH 3 transfer. Finding the perfect location totally depends on when the application actually needs the texture. Most developers will at least fill their texture buffer in video memory before starting the geometry list. When filling the texture buffer during run-time it is crucial that the texture has been transferred to video memory before any PATH1 transfer. You can't start a PATH3 transfer and expect to use the texture immediately because the GIF operates differently when in IMT mode. For instance, if you have a fast rendering micro-program and the GS packet is very small, it's possible that the texture hasn't yet finished transferring into video memory before the PATH1 transfer. The GS packet from PATH1 could possibly use the previous texture stored at that location in video memory. The diagram below shows the data flow and possible hazard.



Conclusion

In conclusion, I wish for every PlayStation®2 developer to implement one of the two texture swapping techniques into their games. If you choose to implement the VIF1 interrupt method, it's important to limit the amount of interrupts in the DMA list to minimise slowdown. The MSKPATH3 method is faster because it doesn't use any form of interrupts but please download the sample code and follow all operating conditions before using this advance technique. If any developers have questions, please feel free to contact me through the PlayStation®2 Developer Support e-mail address.