MPMC9 – Multi-Port Memory Controller #9

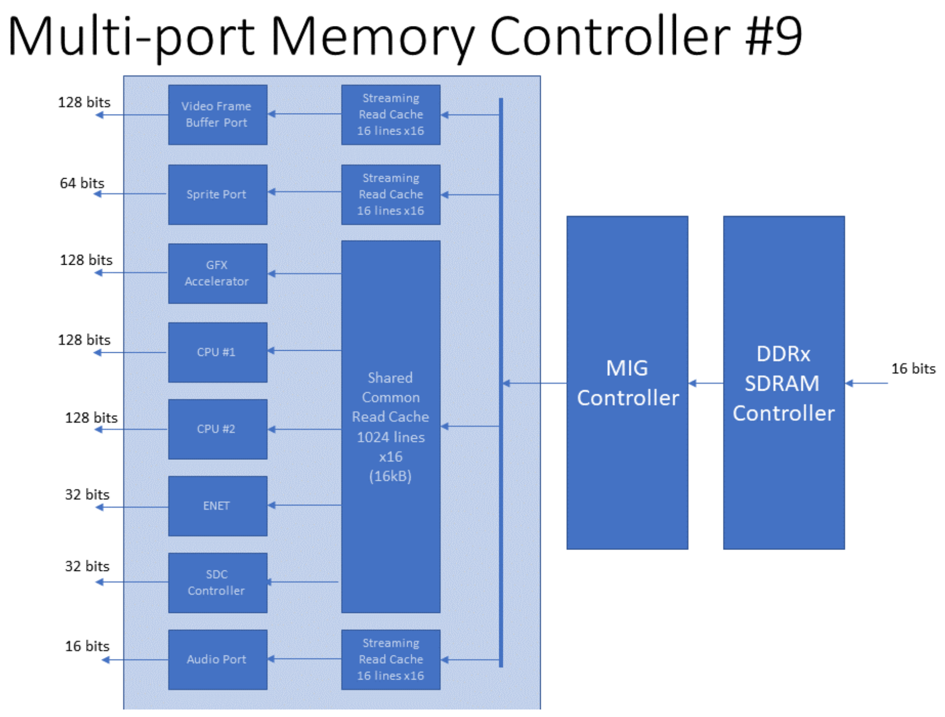
robfinch<remove>@finitron.ca

Overview

The multi-port memory controller provides eight access ports with either small streaming read caches or a 16kB shared read cache to the ddr3 ram. The multi-port memory controller interfaces between the SoC and a MIG controller. The DDR3 ram controller is outside of the scope of the MPMC.

The ports have suggested pre-designated usages. Ports may be specialized to their use. Data widths are configurable with parameters. Most devices are filling some sort of cache or buffer from memory or dumping data back to memory. A large number of memory ports are required because the system has only a single physical memory port and everything is in the system on chip.

MPMC9 increases the number of 16-byte memory strips that may be fetched in one access. MPMC8 allowed up to eight strips to be fetched, this has been increased to 64 for MPMC9. The larger number of strips fetched allows the controller to make much better use of memory bandwidth.



Port Suggested Usage

Port	Use	Port Bits	Common Cache	Stream Buffer	Access
0	Frame Buffer / Bitmap Controller	128		*	4
1	CPU #1	128	*		4
2	Ethernet controller	32	*		4
3	Audio controller	16		*	8
4	Graphics controller	128	*		2
5	Sprite controller (read only)	64		*	2
6	SD Card (disk) controller	32	*		4
7	CPU #2	128	*		4

Port Priorities

The ports have a fixed priority arrangement according to the port number. The lowest port number has the highest priority. Periodically, for one access cycle only, port priorities are inverted so that port 7 has the highest priority and port #0 the lowest.

Overall Organization

The controller may use separate small streaming read caches or a larger common shared cache for each of the ports. This allows multiple read accesses by different devices to occur in parallel, provided the data to be read is in the read cache. Some of the latency for memory reads can be hidden in this manner.

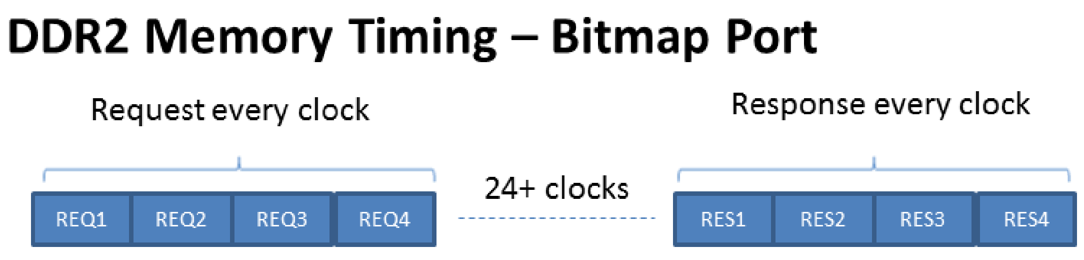
One may wonder why there are separate streaming read caches available when many systems often use a single cache. Using a common cache for all ports including streaming data ports is not a good idea. Streaming data, as for a frame buffer for instance, will fill the entire cache with data that used only once in a frame. Without special considerations, it will bump data from the cache causing a large number of cache reloads.

Port Descriptions

Frame Buffer Port #0

Bitmapped graphics can require a high memory bandwidth. In many systems the display memory is separate from the rest of the main memory of the system so that the bandwidth requirements of the display don't slow the rest of the system down. As an example of bandwidth requirements, the test system uses 800 x 600 x 16bpp color bitmapped graphics mode. The clock frequency for this mode is about 40MHz. About 80MB/s of data is required. (40MHz * 16-bit data). Each byte must be read in about 12ns.

Given that in the test system the main memory of the system is being used for display, it is desirable to make the best use of available bandwidth. To achieve better bandwidth memory access for the bitmap controller is organized into a number of strips. One key thing to note is that the bitmap controller basically always reads forwards through memory. It's addressing is predictable and not random. This makes it easier to achieve high bandwidth access. While the ddr3 controller already does burst access to fetch 128 bit (burst length of eight, two bytes at a time) at a time from the memory, performance can be maintained by performing larger burst accesses. Hence the use of memory strips. For the bitmap controller a burst of sixty-four 128-bit memory strip accesses or 8192 bits (1024 bytes) at a time. The vendor's controller does not need to wait for a read access to complete before starting a second (or more). However, there is about a 24-cycle latency for memory access. The following diagram shows memory access for the bitmap display.



Memory requests can be issued to the DDR2 controller every clock cycle at ¼ the DDR2 clock rate (75MHz in this case). After a latency of a number of clock cycles data is returned from the controller every clock cycle. Data is accessed in 128 bit parcels.

To perform accesses in this manner takes about 40 clock cycles. Note the display's effective dot rate is about 40MHz, while the memory controller clock is 100MHz. If performing sixty-four consecutive accesses to memory is good wouldn't it be better to perform even more? It would be except the issue here is that there are other devices in the system that need to access memory sometimes within a limited time-frame. Giving too many consecutive cycles to the bitmap controller would starve other devices. There is also an issue with the fact that part of the last access for a scan line is wasted. More data is fetched than needed because 128 bytes doesn't divide evenly into an 800-pixel scan line. (1600 bytes / 128 = 12.5 burst accesses).

The bitmap controller has a programmable access interval so that it does not continuously access the memory. This gives some time for other devices to access the memory. The interval has to be set carefully or there could be display problems with the bitmap display.

CPU#1 Port / CPU#2 Port

Memory access for a cpu is somewhat like the bitmap controller. Many accesses travel forwards through memory. However, there is a more random aspect to the cpu's accesses. To improve performance the CPU already has a cache. So, most of the read access required by the cpu is in order to fill a cache line. The CPU only fills one cache line at a time during a miss. Most of the time that's all that's required. Timing is similar to the bitmapped display, except that only four consecutive burst accesses are performed. Four burst access are enough to supply one cache line (512 bits / 64 bytes) of data.

Ethernet Controller

The Ethernet controller requires read / write access to memory for network transfers. Ethernet memory transfers are often done by a dedicated DMA controller and not the CPU. Hence, they require a memory port. The current system uses an Ethernet controller that supports only 32-bit accesses. So, the ethernet's port is only 32 bits wide. However, a read operation from memory always transfers 128 bits, so only those bits that the controller needs are passed back to it. The other bits are cached.

Graphics Controller

The graphics controller port services memory requests for that device. A graphics accelerator reads and manipulates pixels from memory then writes pixels back. Most of the time the accelerator is dealing with individual pixels. There's little benefit to fetching or storing large numbers of pixels in this case. Hence the port for the graphics controller uses only single burst accesses.

Sprite Controller

The sprite controller uses dedicated DMA triggered during the horizontal or vertical blanking interval. Sprite images have linear accesses to memory just like a bitmap display. To minimize the number of memory accesses required to display sprite data, each sprite has its own read cache. The sprite's read cache is enough to buffer about four scan-lines worth of display data. If the sprites were to share a common cache then the cache would be constantly dumped and reloaded since each sprite's display data address is different.

SD Card Controller

The SD card controller has only a 32-bit data path and hence a 32-bit port is used.

Write Ports

Everything mentioned so far has had to do with read ports. All ports except the sprite port are also capable of writing to memory. While the read ports are somewhat customized to the port owner, write ports all work in the same manner.

There is no write cache. Writes go directly to memory as soon as possible. Except for the bitmap controller, they have a higher priority than reads.

All write ports write perform a single 128 bit burst write to memory. Caches are not updated by writes. Instead, if there is a cache hit during a write cycle that cache line is invalidated. This is a simple means of keeping all the caches in the memory controller coherent.

Memory Reservations

The MPMC keeps track of up to two reserved memory addresses for the cpu ports. The number of address reservations is configurable. Memory reservations are used to implement atomic memory operations. A conditional store operation by the cpu will be done only if the store address is reserved. The cpu must output signals to indicate that an address is being reserved during a load operation, and to indicate that a conditional store is taking place.

Conclusion

It’s possible that more ports would be required in the system. For instance, the system may also be interfaced to an EPP parallel port for diagnostic purposes. More CPU cores might be present in the system.

Parameters

	Default	Description
NAR	2	Number of outstanding address reservations possible
AMSB	28	Most significant address bit
C0W	128	Data width for channel zero
C1W	128	Data width for channel one
C2W	32	Data width for channel two
C3W	16	Data width for channel three
C5W	64	Data width for channel five
C7W	128	Data width for channel seven
STREAM0	TRUE	Use streaming read cache instead of common cache for channel 0
STREAM1	FALSE	Use streaming read cache instead of common cache for channel 1
STREAM2	FALSE	Use streaming read cache instead of common cache for channel 2
STREAM3	TRUE	Use streaming read cache instead of common cache for channel 3
STREAM4	FALSE	Use streaming read cache instead of common cache for channel 4
STREAM5	TRUE	Use streaming read cache instead of common cache for channel 5
STREAM6	FALSE	Use streaming read cache instead of common cache for channel 6
STREAM7	FALSE	Use streaming read cache instead of common cache for channel 7