Gary Stringham & Associates, LLC
12948 W Woodspring St, Boise, ID 83713
Phone: 208-939-6984 Cell: 208-850-1597 Fax: 208-939-9123
gary@garystringham.com www.garystringham.com

Using DMA Controllers to Troubleshoot Problems

Friday, November 30th, 2007 | Hw/Fw Best Practices | http://blog.garystringham.com/?p=152

[Note: Forensic investigation is very similar to debugging and troubleshooting products during development. Evidence can be gleaned by watching operations of various subsystems. This blog post discusses using DMA controllers to gather evidence into the operation of the product.]

Many blocks within chips process data read from or written to memory using direct memory access (DMA) controllers. Firmware initiates the data processing by setting up the blocks and DMA controllers. When problems occur, the first task is to figure out if the problem is in the block, in the firmware, or in the data. The current values of the address and byte count registers in the DMA controllers can provide clues to troubleshoot the problem. Temporary debug statements in firmware can retrieve those values for analysis.

Firmware sets up the DMA controllers by writing the starting address and byte count into their respective registers. After the DMA transfer has been initiated, the address and byte count values are changing as the DMA controller moves the data. The DMA controller should be designed to allow firmware to read both the starting values (that firmware originally wrote) and the current values (that the DMA controller is currently using) of the address and byte count registers. These starting and current values assist troubleshooting activities in different scenarios. I will first talk about using the current values for troubleshooting and then later in this article about using the starting values.

The following table lists DMA register states that can indicate problems during DMA transfers either from or to memory. While not comprehensive, this table shows how current state information in DMA registers can provide information useful for troubleshooting data processing problems.

DMA Register Status	Potential Problem	
	DMA from Memory	DMA to Memory
Both the address and byte count registers are unchanged from what firmware wrote.	The DMA transfer has not started, maybe due to an incorrect setup of the DMA registers.	The block has not yet given data to the DMA controller. The block may be set up incorrectly.
The address and byte count registers are one DMA burst size off.	One burst size from the beginning indicates that the DMA transfer has started but the block has not consumed the data. The block might not be set up correctly.	One burst size from the end of the buffer might indicate that a last byte is stuck somewhere in the block.
The address and byte count registers are	Maybe the data read in had a corrupted spot causing the block to generate an	This might indicate that the block terminated early, stopping it from

somewhere in the middle of the transfer.	error and quit. The address indicates the general vicinity where the corrupted data is in memory.	sending more data to the DMA controller. The byte count indicates how much was transferred to memory.
The address and byte count indicate a completed transfer but the block has not finished.	The block may be expecting more data than the DMA controller was programmed to transfer.	The block might have more data than the DMA controller was programmed to transfer.

Reading the current values in the DMA registers not only helps troubleshooting but also can be used to work around defects in the hardware. During one firmware development effort for a block with a faulty state machine, I could not abort the block unless I knew that the data flowing into the block was stalled. I did this by monitoring the DMA byte count register until it quit changing. As long as data was flowing, the byte count register would keep changing; but once the flow stalled, the byte count register would not show a change across several reads. Of course, I had to consider how often that register would change and make sure I sampled across enough time to catch a change if the data was still flowing. This technique became part of normal operation so we could ship without respinning the chip – it was a good thing that the DMA controller had this capability for this contingency.

> **Best Practice:** Design DMA controllers to provide current address and byte count values throughout the transfer.

Some DMA controllers are equipped with chaining (scatter-gather) capabilities that use linked lists to indicate to the controller the location and size of multiple buffers in memory. In addition to the current value of the address and byte count register, the DMA controller should make available the starting address, byte count, and next pointer for the buffer in the chain that the controller is currently working on. This information can help firmware engineers diagnose linked list and chaining problems, as it did once for one of my project teams: It helped us discover that we had not correctly translated the virtual addresses of the linked list pointers into the proper physical addresses needed by the DMA hardware.

> **Best Practice:** Design DMA controllers to provide the starting values of the address and byte count for the current buffer in the chain, and the pointer to the next buffer in the chain throughout each chaining (scatter-gather) operation.

Tags: Debugging Tricks, DMA