



Video Framework Considerations for Image Processing on Blackfin® Processors

Contributed by Kunal Singh and Ramesh Babu

Rev 1 – September 8, 2005

Introduction

In any system, memory partitioning and data flow management are crucial elements for a successful multimedia framework design. Blackfin® processors have a hierarchal memory and non-intrusive DMA with the Parallel Peripheral Port interface. When used in your application, they can provide very high system efficiency.

This EE-Note discusses the following topics that should be considered for obtaining maximum performance on ADSP-BF533 and ADSP-BF561 Blackfin family processors in video processing applications:

- Memory considerations
 - Internal memory space
 - SDRAM memory space
 - Managing external data accesses
- DMA modes for PPI capture and display
 - Working with ITU-R-656 input modes
 - Outputting ITU-R-656 video frames
 - DMA prioritization and traffic control register

Memory Considerations

The Blackfin processor architecture supports a hierarchical memory that allows the programmer to access faster, smaller memories for code that runs the most often and larger memory for data

buffers associated with video applications. The Blackfin processor's memory has a unified address range, which includes the internal L1 memory, (in case of the ADSP-BF561 processor also L2 memory) SDRAM memory, and asynchronous memory spaces.

Internal Memory Space

The L1 memory operates at core clock frequency and hence, has lowest latency compared to the other memory spaces. Blackfin processors have separate Data and Instruction L1 memory.

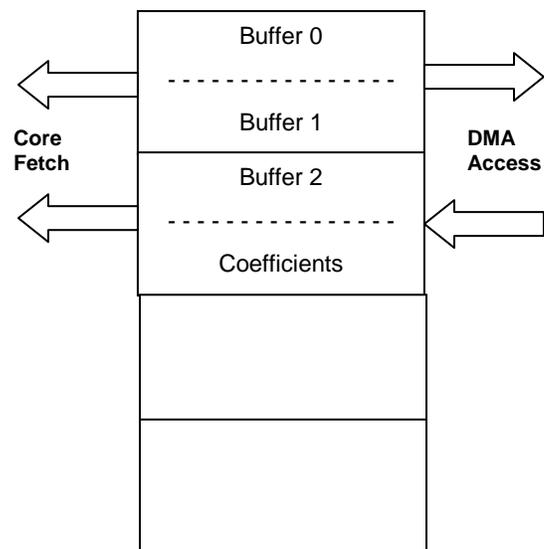


Figure 1. Un-Optimized L1 Memory Allocation

The L1 data SRAM is constructed from single-ported subsections, each subsection consisting of 4 Kbytes of memory. This organization results in multi-ported behavior when there are simultaneous access to different sub-banks or

accessing one even and one odd 32-bit word within the same 4K sub-bank.

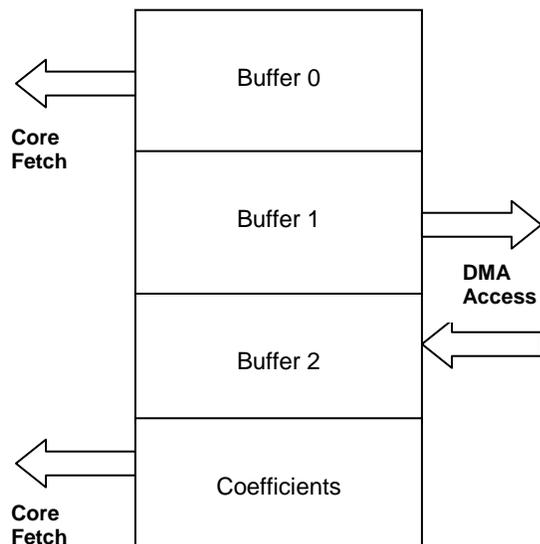


Figure 2. Optimized L1 Memory Allocation

Figure 1 shows the un-optimized allocation of memory for different buffers. Each block in the figure represents a 4 Kbyte sub-bank in internal data memory. Here the internal data buses are not used effectively, since the processor cannot fetch the two data words simultaneously.

Figure 2 shows the optimized memory allocation across internal 4 Kbyte data memory banks. This memory allocation allows simultaneous dual DAG and a DMA access, and hence, maximum throughput over data buses.

In video encoding and decoding applications, optimized memory allocation reduces the latency involved in accessing L1 data memory due to simultaneous access from core and DMA controller

SDRAM Memory Space

The SDRAM Controller (SDC) enables the processor to transfer data to and from Synchronous DRAM (SDRAM). The SDRAM controller supports a connection to four internal banks within the SDRAM. In end applications, by mapping the data buffers appropriately in different internal sub-banks, the latency involved in accessing data by core/DMA can be

minimized. The SDC can keep track of one row per bank (with up to four internal SDRAM banks) at a time, hence it can switch between four internal SDRAM banks without any stalls.

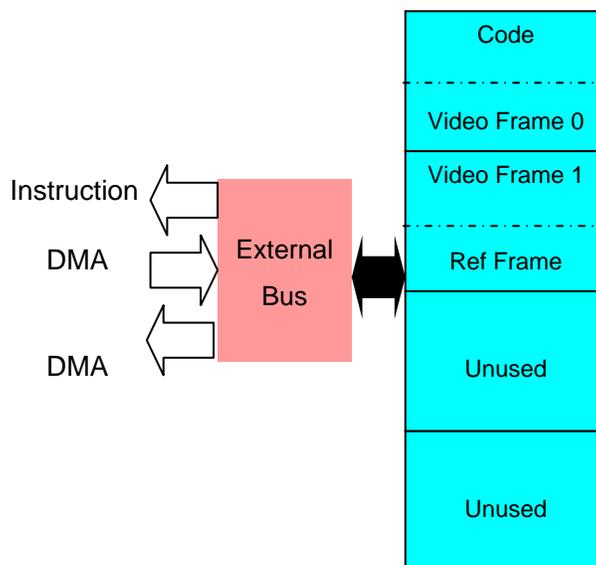


Figure 3. Un-Optimized SDRAM Memory Allocation

In image processing applications, the video frame is brought into the memory using a PPI DMA. Because of the image size (i.e., VGA, D-1 NTSC, D-1 PAL, 4CIF, 16CIF, etc.), each frame of the image must be captured in SDRAM memory using a PPI DMA channel. The algorithm can read the pixels block by block from SDRAM and process each block as it is brought in. The PPI captures the next frame into another buffer while core is processing the previous buffer. Since both core and DMAs are accessing the SDRAM memory simultaneously, it is necessary to map the code, video frame, and other buffers appropriately to minimize the latency involved in accessing SDRAM memory.

Figure 3 shows the un-optimized memory allocation in SDRAM internal sub-banks. In Figure 3, both the code and video frame buffer are mapped to SDRAM internal Bank 0. This allocation method causes more latency because SDRAM row activation cycles occur at almost every cycle. This is due to alternating core accesses (fetching the instructions) and DMA accesses to different pages within the same

SDRAM internal bank. The latency may cause a PPI FIFO overflow error (in the case of image capture) or underflow (in the case of image display). In order to increase throughput of external memory accesses, it is necessary to allocate video/reference buffers such that only one DMA accesses an SDRAM bank at any given time.

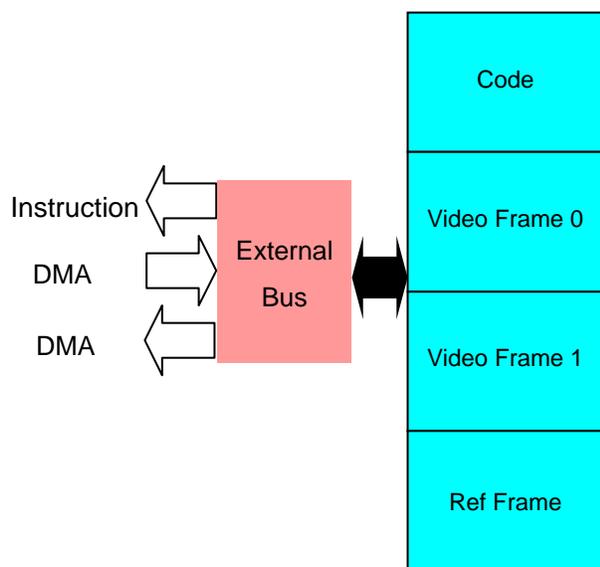


Figure 4. Optimized SDRAM Memory Allocation

Figure 4 shows the optimized memory allocation within SDRAM internal banks. In this memory allocation example, at any time, either the core or the DMA controller is accessing the particular internal bank of SDRAM memory. Hence, the latency is minimized, since row activation cycles are spread across hundreds of SDRAM accesses.

Managing External Data Accesses

Accessing the external memory is performed more efficiently when the transfer is made in the same direction. While accessing the SDRAM memory, performing group transfers in a single direction (avoiding frequent direction turn-arounds) can reduce the latency involved in the data transfer. Frequent turnarounds on the DMA controller can increase the latency due to a write followed by a read access.

DMA Modes for PPI Capture and Display

The Blackfin DMA controller can transfer data between its memory space and peripherals efficiently. The designer can choose appropriate DMA modes (for example: stop mode, auto buffer, or descriptor-based DMA) to transfer the data. Also, the programmer can choose the DMA priority for a particular peripheral by using appropriate DMA channel.

The Blackfin processor's PPI port supports the industry-standard ITU-R-656 mode and general-purpose I/O mode with various internal and external frame sync options. Images can be seamlessly captured or displayed using the PPI along with the appropriate DMA mode. The programmer has to choose the appropriate DMA mode such that images can be processed in real time without losing a frame.

In image encoding applications, the PPI can be programmed in descriptor chaining mode to capture images in two or more buffers. The core can process one buffer while DMA is filling the other buffer. You must ensure that the core and the DMA controller do not access the same SDRAM bank, as discussed in [SDRAM Memory Space](#) section.

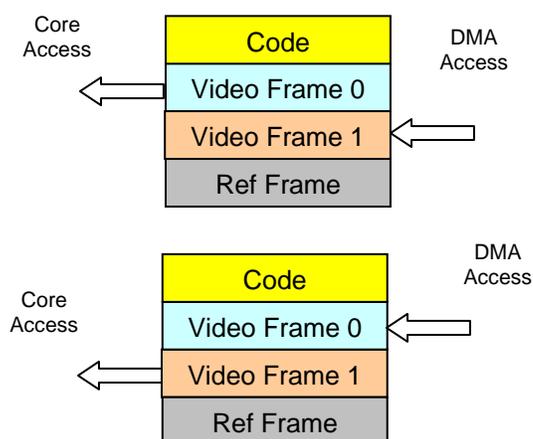


Figure 5. PPI DMA and Core Accesses Without Any Conflict in Accessing SDRAM Internal Sub-banks

In image encoding and decoding applications, the number of million cycles per cycles (MIPS)

consumed by the processor is not constant. The MIPS consumed varies from the compression ratio, captured image, and so on. In image decoding applications, if the decoded frame to be displayed is not yet ready, the PPI can transmit the recently decoded frame again. In order to achieve this functionality, the PPI can be programmed in stop mode DMA. This mode has more control over the data that needs to display as output.

In stop mode DMA, an interrupt is generated after each work unit and the DMA channel is in a paused state.

Hence, if the next frame to be displayed is not yet ready, the same frame can be re-transmitted. This can be achieved inside the PPI DMA interrupt subroutine.

Working with ITU-R-656 Input Modes

The PPI supports three input modes for ITU-R-656-framed data.

- Entire field mode
- Active field only
- Vertical blanking only

In video encoding applications, the video frame can be captured in active field only mode, so that only Field 1 and Field 2 are captured. Since the ITU-R-656 has interlaced video format, video algorithms may require the video data to be in de-interlaced format. Using Memory DMA, the programmer can de-interlace the video frame.

	BLANKING
	FIELD 1 ACTIVE VIDEO
	BLANKING
	FIELD 2 ACTIVE VIDEO
	BLANKING

Figure 6. Typical ITU-R-656 Video Frame Partitioning

In order to minimize processor overhead to de-interlace the frame, the PPI can capture the video frame by skipping one line after each active line (as depicted in Figure 7). Then memory DMA can de-interlace the Field 2 into the Field 1 by filling the blank lines.

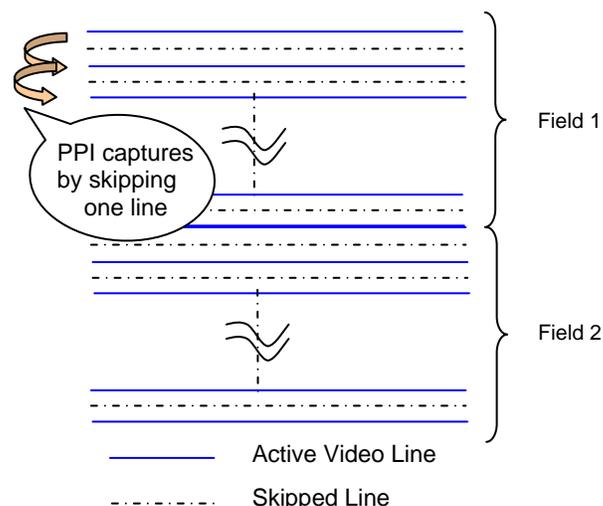


Figure 7. 2-D DMA Capture Alternating Lines

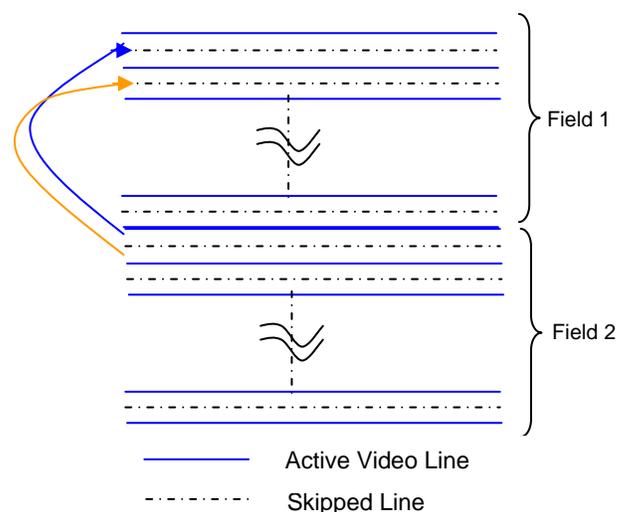


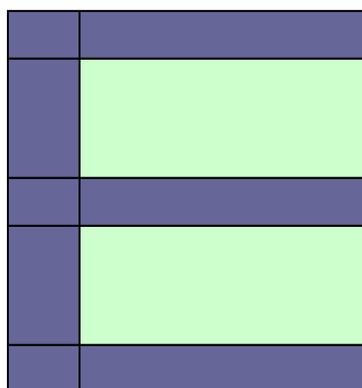
Figure 8. De-Interlacing Using Memory DMA

Figure 8 illustrates de-interlacing using memory DMA. The data in Field 2 must be de-interlaced with data in Field 1. Hence, the skipped line is replaced with the data in Field 2. Here, the MDMA source address should contain the address of the first line of Field 2, and the MDMA destination address should contain the

first skipped line. Both the source and destination `MDMAX_Y_MODIFY` should be configured for count to skip one line.

Outputting ITU-R-656 Video Frames

The PPI does not explicitly provide functionality for framing an ITU-R-656 output stream with proper preambles and blanking intervals. However, this is achieved by first creating a complete frame in the memory and then transmitting the same via the PPI in zero frame sync mode. The video data, blanking data, and control code can be set up in memory prior to sending out the video stream. The horizontal and vertical blanking information can be set up in memory (one time) and then only the active fields can be updated on a frame-by-frame basis.



- One-time initialization of horizontal, vertical, and control codes in memory
- Active field data updated for each frame

Figure 9. Blanking and Active Video in Memory

DMA Prioritization and Traffic Control Registers

In Blackfin processors, all peripherals are DMA capable. By default, each peripheral is linked to a particular DMA channel. Each DMA channel has its own priority to access the memory. DMA channels with the lowest number have the

highest priority. Programmers can assign a particular DMA channel to a peripheral and hence can change the priority of peripheral DMA.

By default, the PPI uses higher priority channels, compared to other peripherals. If an application has more than one DMA running in parallel, the peripherals with high data rates or low latency requirements can be assigned to lower numbered (higher priority) channels via the `DMA_PERIPHERAL_MAP` registers.

Using DMA Traffic Control registers, the programmer can influence the direction of data movement in internal DMA buses (DAB, DCB, and DEB). Traffic control provides a way to influence how often the transfer direction on the data buses may change, by automatically grouping same-direction transfers together.

The DAB, DCB, and DEB buses provide a means for DMA-capable peripherals to gain access to on-chip and off-chip memory with little or no degradation in core bandwidth to memory.

The DMA controller uses the DAB bus to access the DMA-capable peripherals. The DCB bus is used to access internal memory. Similarly, the DEB bus is used to access external memory through the EBIU.

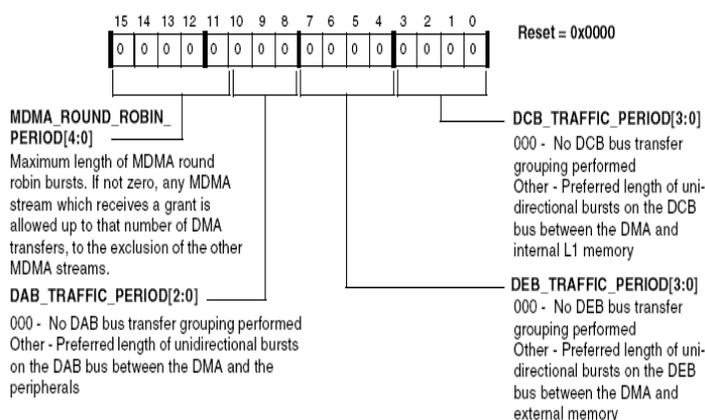


Figure 10. DMA Traffic Control Register

By using the traffic control register, the programmer can influence the direction of each

DMA bus independently, by grouping same direction transfers together.

Figure 10 shows the fields in the `DMA_TC_PER` register. For example, consider an application in which both the Memory DMA and PPI DMA are active and a traffic control register is used. If both the PPI DMA and Memory DMA request the DEB bus (with memory DMA going “with the traffic” and the PPI DMA going “against the traffic”), the memory DMA is granted the DEB bus even though PPI DMA has higher priority. Since the PPI DMA is “against the traffic” of the DEB bus, the effective priority of the PPI DMA is increased by 16, and hence, the memory DMA is granted access to the DEB bus. The PPI DMA gets access to the DEB bus when the traffic control counter times out (or until traffic stops or changes direction on its own).

For more information DMA Traffic Control registers, refer the ADSP-BF533 and ADSP-BF561 Hardware Reference manuals.

Bus Arbitration

The DMA bus has higher priority over the core bus while accessing internal L1 memory. By default, when accessing external memory, the core has higher priority over the DMA bus. By setting the `CDPRIO` bit in the `EBIU_AMGCTL`

register, all DEB transactions to the external bus have priority over core accesses to external memory. Programmers can use this bit, depending on their application requirement.

DMA and Cache Coherency

In an application, if both core and DMA access the shared buffer and cache is enabled, the software should provide cache coherency support by invalidating the data in the shared buffer. In Blackfin processors, the cache can be invalidated using memory-mapped registers. Also, the VisualDSP++® 4.0 development tools provide C-callable library functions to invalidate the individual cache banks. Software can invalidate the cache each time before accessing the shared “volatile” buffer.

Conclusion

By considering the various factors discussed in this EE-Note, you can obtain maximum performance from Blackfin processors in image processing applications.

References

- [1] *ADSP-BF533 Blackfin Processor Hardware Reference*. Revision 3.1 May 2005. Analog Devices, Inc.
- [2] *ADSP-BF561 Blackfin Processor Hardware Reference*. Revision 1.0, July 2005. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – September 08, 2005 by Kunal Singh	Initial Release