Intel® PXA27x Processor Family Video Decoder Display Performance Optimization

Application Note

April, 2004

Order Number: 280006-001

Contents

Figures

Tables

Revision History

Date	Revision	Description
April 2004	1.0	Initial release

1.0 Introduction

The Intel® PXA27x Processor Family PXA27x processor is equipped with system level features which enable advanced video and multi-media applications. These features include:

- Intel XScale® microarchitecture enhanced with Intel® Wireless MMX™ technology
- Capability of the LCD controller to display the YUV 4:2:0 format, eliminating software conversion from YUV to RGB.
- On-chip SRAM large enough to be used for a frame buffer.

Depending on the application workload the system can be configured to make use of the above resources in many ways. During video decoding the hardware (HW) acceleration for color conversion coupled with internal SRAM can offer large performance improvements in a power efficient manner. This document presents experimental results using the color conversion HW engine and internal SRAM in multiple configurations displaying H.264 decoded video sequences. The document offers a set of recommendations, focusing on the optimal use of the HW color conversion and internal SRAM. The document assumes no impact by the Intel® Wireless MMX™ technology.

2.0 Feature Description and Use

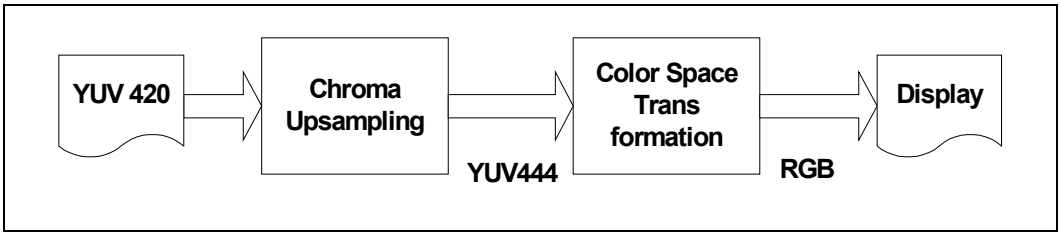
Display chain in video applications refer to the process of displaying the application generated data onto the screen. The PXA27x processor has an LCD controller which can drive a range of displays with desired set of content. There are two features that impacts the ease of use and performance of the display chain are as follows :

- HW Color Conversion
- Internal on-chip Memory

2.0.1 HW Color Conversion

The PXA27x processor LCD controller has a mode for displaying planar YUV 4:2:0 (among other formats such as YUV422, 444, and other RGB formats). Planar YUV 420 is a common output format for video decoders including H.264. Using this mode eliminates the need for CPU cycles to be spent in SW conversion so overall video decoding performance is improved. Figure 1 shows the functions accelerated by HW and flow of information between them (Details of the chroma upsampling and transformation is captured in the EAS [Ref] and are not described here):

Figure 1. Steps in HW Color Conversion



When HW color conversion is used, the video decoder produces the standard YUV 420 output, then makes the data available to the LCD controller for display; the LCD controller performs the required chroma upsampling and color space transformation while displaying the data.

2.0.2 Internal Frame Buffer

The area of memory containing the decoded image to be displayed is known as the frame buffer. The video decoder stores the decoded image to this buffer once per frame; the LCD controller reads from the buffer at the refresh rate. The LCD controller can become a significant bus bandwidth consumer in the system based on the LCD display size and refresh rate. (Refer to Figure 2.)

The PXA27x processor includes 256K-byte SRAM on-chip that can be used to contain the frame buffer, reducing off-chip SDRAM accesses by the LCD controller for refresh and benefiting the storing of data to be displayed because internal memory offers lower access latency and higher access throughput.

The HW color conversion mode requires using 2 frame buffers - the usual RGB base frame plus the YUV formatted frame, known as overlay 2. For the pixels in overlay 2 to be visible the corresponding pixels in the base frame must be marked as transparent.

Figure 2. Internal Frame Buffer

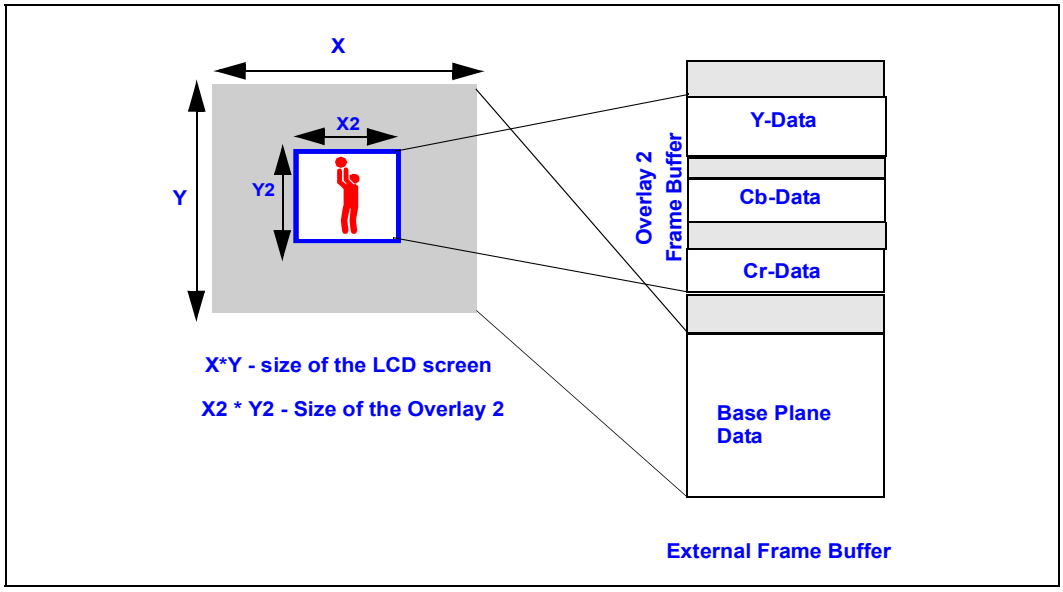


Table 1 shows the frame buffer requirements for each format for QVGA and for VGA.

Table 1. Frame Buffer Requirements (Bytes)

	VGA	QVGA
RGB16	614400	153600
YUV 4:2:0	460800	115200
4bpp	153600	38400
1bpp	38400	9600

The YUV 4:2:0 buffer is 3 independent buffers, one for each of Y, U and V. The Y buffer is 8 bpp, the U and V buffers are each 2 bpp; the table above shows the 12 bpp total memory required for all three buffers.

It can be noted (see Table 1) that the 256KByte of internal SRAM is not adequate for both of the frame buffers. Another issue is that when overlay 2 is full screen (typical for a video playback application), none of the pixels of the base frame are visible, yet the LCD controller is reading the base frame at the refresh rate, wasting bus cycles.

One option that addresses these issues is to change the format of the base frame from 16 bpp to something less, an option available if none of the base frame is visible. The LCD controller supports 4 bpp and 1 bpp. Switching to one of these formats dramatically reduces the frame buffer requirement for the base frame and the LCD refresh accesses.

The lower base frame buffer requirements using 1bpp enable consideration of placement of the base frame buffer and the overlay 2 frame buffer in SRAM (256K) or in SDRAM; experiments with several configurations for QVGA and VGA were performed using the H.264 decoder (see next section).

The user needs to choose which data objects will be placed in the internal memory space. Placement of the buffers in the memory sub-system is critical. Section 5 explores alternative configurations towards making the optimal choice.

An alternative that may be appropriate for some applications is to locate the LCD frame buffer in external SDRAM and use the internal SRAM for the local frame buffer that many applications use. The application can update the local frame buffer image faster (because the buffer is internal) then use system DMA to copy from the application's frame buffer to the LCD frame buffer for display. This also works well when the display is composed of more than one overlay.

3.0 Impact of Color Conversion HW on H.264 Decoder

Experiments were performed running the H.264 decoder using various frame buffer configurations (refer to Table 2), on a Mainstone I running at 403 MHz CPU, 201.5 MHz system bus, 100.75 MHz memory and LCD controller frequency. The video sequence is “stuart” (a 24 fps 30-second extract from a movie) compressed to 518 Kbps for 320x240 and to 1932 Kbps for 640x480; B frames are not used. For QVGA, a portrait mode display is used, the video sequence is not pre-rotated, so YUV copy for display includes image rotation.

Table 2. H.264 Decoder Experiment Results (Sheet 1 of 2)

Configu-ration	Display/Video	Base frame	Overlay 2 frame	Total Normalized FPS	Color Convert
1	QVGA	SD 16bpp	Not used	1.0	18%
2	QVGA	SD 16bpp	SD	1.11	7.5%
3	QVGA	SD 1bpp	SD	1.13	7.1%
4	QVGA	SD 16bpp	SR	1.13	6.1%
5	QVGA	SD 1bpp	SR	1.16	6.25%
6	QVGA	SR 1bpp	SR	1.16	6.25%
7	VGA	SD 16bpp	Not used	1.0	15.2%

Table 2. H.264 Decoder Experiment Results (Sheet 2 of 2)

Configu-ration	Display/Video	Base frame	Overlay 2 frame	Total Normalized FPS	Color Convert
8	VGA	SD 16bpp	SD	1.05	8.0%
9	VGA	SD 1bpp	SD	1.10	7.1%
10	VGA	SD 16bpp	SR (UV)	1.07	7.0%
11	VGA	SD 1bpp	SR (UV)	1.11	6.4%
12	VGA	SR 1bpp	SR (UV)	1.11	6.4%
	SD: SDRAM SR: on-chip SRAM (UV): Only U and V in SRAM, Y is in SDRAM CC: Portion of Total fps used for "color conversion", is YUV frame copy when Overlay 2 is used.				

Observations:

- The largest gain is obtained by using the LCD color conversion feature instead of SW color conversion.
- An additional 2-3% gain in overall playback performance is obtained by switching the base frame buffer format from 16 bpp to 1 bpp.
- An additional 2% gain is obtained by locating the overlay 2 frame buffer in SRAM. This is useful to know when considering other potential uses of the SRAM.

4.0 Recommendations

Based upon the best overall H.264 playback performance obtained the recommended frame buffer / SRAM configurations to use are described here. A primary assumption is video is played full screen and none of the base frame pixels need to be displayed.

- Use HW color conversion.
- Base frame format should be switched to 1 bpp to minimize base frame buffer size and base frame refresh bus cycles.
- The minimal 1 bpp base frame can be located in SRAM or SDRAM; because it is so small the choice has little impact upon performance.

The YUV frame buffer (overlay 2) should be located in SRAM (internal memory). For QVGA all 3 (YUV) buffers fit into the 256K SRAM; for VGA the U and V buffers fit, but the Y buffer must be located in SDRAM. Note that placement of the frame buffer in SRAM is a BSP-level decision on how to use the SRAM for a specific device.

For the H.264 decoder these choices improve overall playback performance (decode + display) over playback using SW color conversion by 16% for QVGA and 11% for VGA. For a less complex decoder such as MPEG-4 the percentage gains would be significantly larger.

5.0 Buffer Management

The capability of the PXA27x processor LCD controller to directly display YUV 4:2:0 raises questions regarding methods for getting the decoded video frame to the frame buffer to be displayed. The experimental results above show that using a frame copy is about 6% of the total playback frame time when using H.264 so it is strongly desired to reduce the number of frame copies required. This section explores the issue, trade-offs, and describes why the one frame copy is the choice used for the H.264 decoder, achieving the best performance while meeting video player requirements.

5.1 Background

A video bit stream is decoded by a video decoder to YUV frames that are used by a playback application (player) for display after color conversion and are used as reference frames for decoding future frames. The player and the decoder both require buffering of the decoded frames, but for different purposes.

A video decoder must maintain YUV buffers for decoding - one for the frame currently being decoded, plus one for the previous reference, plus one for the future reference if B frames are used. H.264 allows multiple previous and future reference frames, so the H.264 decoder may use more YUV buffers (up to a total of 16+1).

A video playback application (player) must maintain a pool of decoded frames for display, synchronously emptied in sync with audio at a constant frame rate, and asynchronously filled by the decoder (decode time can vary significantly from frame to frame). This pool is not required (number of buffers can be reduced to one) when the slowest decode time is a small percentage of overall frame time - an example would be decoding MPEG-4 QCIF video on the PXA27x processor at 520 MHz.

The memory buffer from which the LCD controller DMA reads must be a contiguous section of physical memory for which software must be able to obtain both the physical address (to program the DMA) and the virtual address (for normal access).

To ensure pixels to be displayed are in memory, the area needs to be configured as uncacheable (but should be buffered for best store performance).

If the OS being used does not provide the necessary support for dynamic allocation of buffers with these attributes it will be necessary to reserve physical memory space for the buffers (as is currently done for 1 or 2 frame buffers).

5.2 Exploring Different Configurations

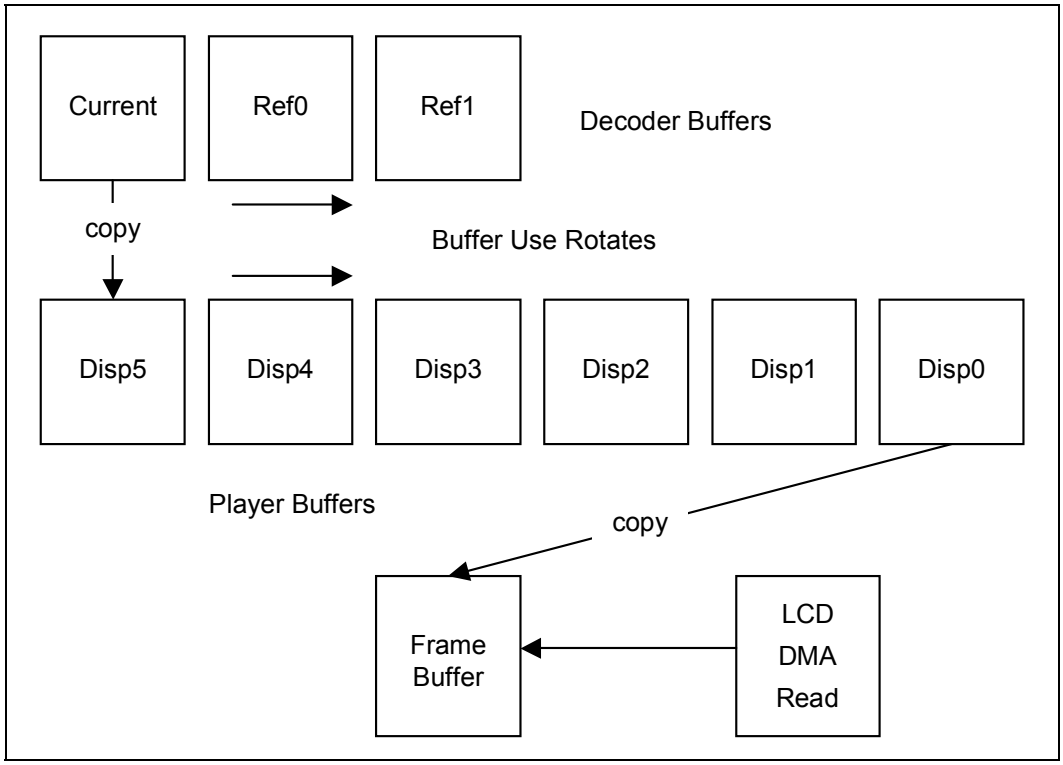
Figure 3, Figure 4, Figure 5, and Figure 6 show possible decoder/player/displayer buffer configurations, followed by a table of characteristics for each (shown in Table 3). The number of decoder and player buffers shown in the figures is an example only; specific decoder and player implementations will use different numbers of buffers. The configurations vary in terms of their data foot-print or memory requirements, number of copying operations that are performed and also the complexity of the buffer management from the software/application's point of view.

In a typical video decoding application, such as video viewer or movie watcher application can be decomposed into a set of components, they are a player (over all application), video decoder which will decode a video clip (decoder) etc.

5.2.1 Configuration 1: Fully Independent Buffers

Configuration 1 illustrates an architecture using separate buffers for the decoder, the player, and the displayer. The decoder and player use their buffers in a rotating fashion. For example, on the next frame the decoder buffer Ref1 will become the Current buffer, the decoder buffer Current becomes the Ref0 buffer, etc. This architecture has the best modularity, supports image rotation and frame buffer in SRAM, but has a performance penalty of requiring two frame copies each frame. For a device without LCD color conversion capability the Disp0 -> Frame Buffer copy is replaced with SW color conversion. This is also the point at which image rotation is performed if required. Refer to Figure 3 for an illustration of Configuration 1.

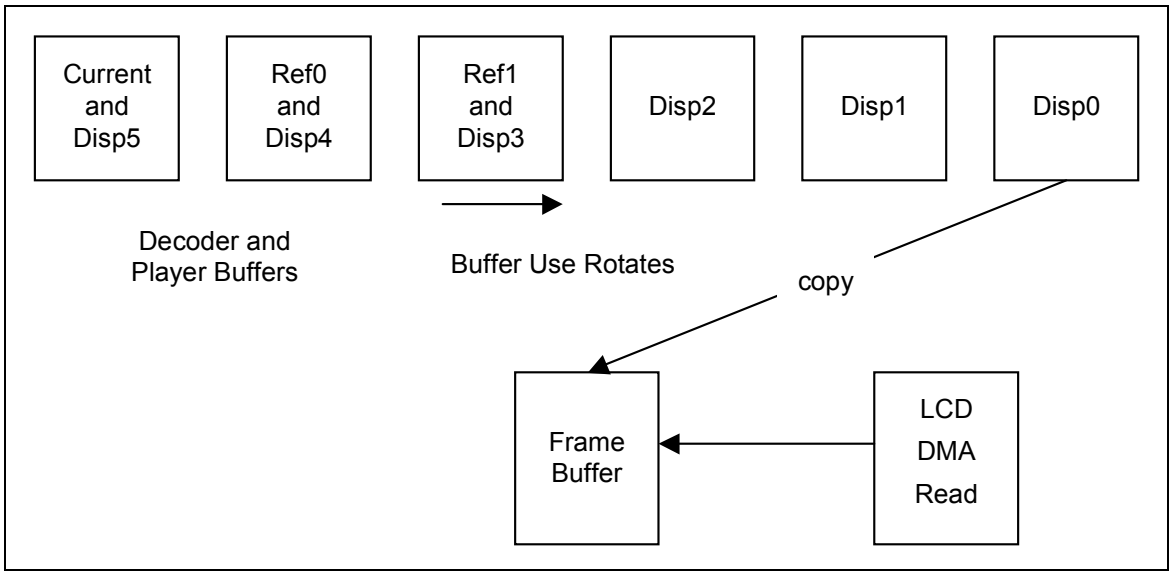
Figure 3. Fully Independent Buffers



5.2.2 Configuration 2: Decoder and Player Share Buffers

Configuration 2 illustrates an architecture using shared buffers for the decoder and the player, maintaining a separate buffer for the displayer. This architecture eliminates one of the frame copies of configuration 1 at the cost of increased coupling between the decoder and the player – the decoder can not store a new current frame into a buffer until after the previously decoded frame in that buffer has been displayed. Configuration 2 does maintain independence between the application software and the displayer and supports image rotation and frame buffer in SRAM. Refer to Figure 4 for an illustration of Configuration 2.

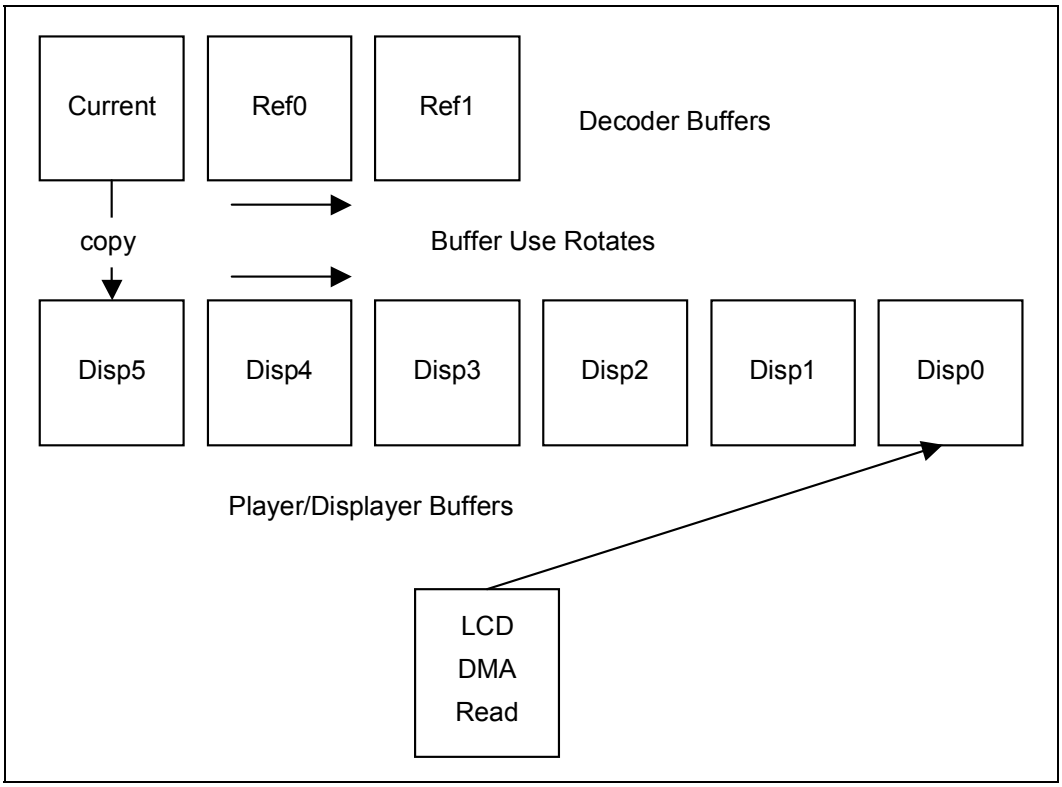
Figure 4. Decoder and Player Share Buffers



5.2.3 Configuration 3: Multiple Display Buffers

Configuration 3 illustrates an architecture using separate buffers for the decoder and the player while directly using the player's buffers for display – the LCD DMA is switched from buffer to buffer. This architecture maintains modularity between the decoder and player while closely coupling the player to the displayer. It has the benefit of eliminating one of the frame copies of configuration 1. It supports image rotation but does not support use of on-chip SRAM for the frame buffer. Refer to Figure 5 for an illustration of Configuration 3.

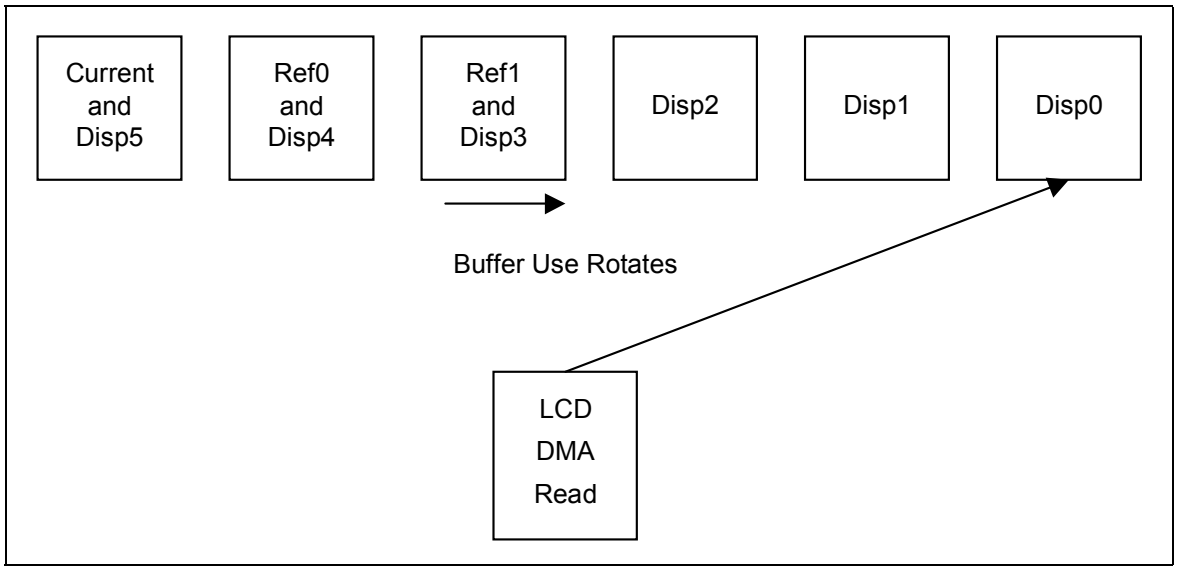
Figure 5. Multiple Display Buffers



5.2.4 Configuration 4: One Set of Buffers for All

In Configuration 4 both frame copies of configuration 1 have been eliminated at the cost of closely coupling the decoder to the player to the displayer. It does not support image rotation and use of on-chip SRAM for the frame buffer. This configuration also requires the buffer layouts to be as needed for DMA. For the H.264 decoder (and likely other decoders) the required layout is not met – the H.264 decoder buffers include extra bytes at the end of each row for decoder reference frame usage. Refer to Figure 6 for an illustration of Configuration 4.

Figure 6. One Set of Buffers for All



5.2.5 Configuration Summary

Table 3 summarizes the attributes of each of the configurations.

Table 3. Configuration Attribute Summary

Configuration	Frame Buffer in SRAM	Frame Copies per Frame	Image Rotation	Independent Decoder/Player Buffer Mgmt	Standard Buffer Memory Allocation
1	Yes	2	Yes	Yes	Yes
2	Yes	1	Yes	No	Yes
3	No	1	Yes	Yes	No
4	No	0	No	No	No

Configuration 3 and configuration 4 do not support placement of the frame buffer in SRAM due to the limited size of SRAM – there is not enough space for the multiple buffers used for display.

Image rotation is not supported in configuration 4 because the LCD DMA controller does not offer the capability to rotate while copying – the method used by software to obtain rotation at little to no performance cost in the other configurations.

The buffer memory allocation for the player/displayer buffers of configuration 3 and all buffers of Configuration 4 must address the physical memory requirements for frame buffers described above, introducing coupling and possibly overall system restrictions on memory usage.

Configuration 4 offers the best potential overall performance if the buffers can be configured as cacheable (an experiment using the H.264 decoder revealed that overall playback performance is about 80% slower when the decoder uses non-cacheable reference buffers). One method to use cacheable frame buffers is to use a large enough pool of buffers so it is almost certain, although not guaranteed, that by display time the processor has decoded additional frames and evicted all cache

lines for the picture to be displayed. Another method is to force a cache flush (without invalidation) at the completion of decoding a frame. In a H.264 decoder experiment the performance cost of such a data cache flush was so small there was no measurable impact.

Configuration 4 has disadvantages of tight coupling among decoder/player/displayer, memory allocation issues for the frame buffers, and lack of image rotation support. This makes it appropriate only for dedicated usage applications that demand the highest level of performance.

Configuration 1 is the practical choice for applications requiring independence among video decoder, player, and displayer. An example would be a video player intended to run on multiple types of devices using a variety of video compression formats.

Our implementation for the Personal Video Player (PVP) demos using the H.264 decoder uses configuration 2 with frame copy functions optimized for the PXA27x processor. We found the required coupling between the player and the decoder to be reasonable (a new interface plus buffer usage flags) and worth the performance gain of eliminating one frame copy. The player is implemented with the intent of supporting only the H.264 decoder so not having the flexibility of using other decoders is not a problem. Based upon the experimental measurements, configuration 1 would be about 6% slower for H.264 while configuration 4 would be about 6% faster.

6.0 Conclusion

This document has addressed the question of how a video playback application can optimally use the HW color conversion capability and the internal SRAM feature of the PXA27x processor. It explored several possible configurations, including management of decoded video buffers, and provided experimental results using an H.264 decoder. Using these processor features can significantly improve overall video playback performance on the PXA27x processor.


ARM
POWERED