High-Density Programmable FIFO Memory For Use in Video and Imaging Applications

By Sivashankar M, Applications Engineer Staff and Harsha Venkatesh, Product Mktg Engineer Sr, Cypress Semiconductor Corp.

Many market segments, including video broadcasting, military, medical imaging, and base stations, can benefit from the use of high-density FIFO devices solutions have programmable features. In addition to providing significant cost savings and improved video quality compared to SDRAM + FPGA architectures, high-density FIFOs design complexity and cost can be further mitigated using system-level programmability.

In this article, we will first consider a few video applications to have an understanding of the data path and nature of data handling required. As a next step, we will try to estimate the complexity of handling data in any video processing pipeline. A programmable high-density FIFO is then introduced with its capabilities and how it can act as a more efficient alternative to the current conventional implementation of frame buffers using SDRAM and FPGAs.

Overview of video applications:

Figure 1 shows the system block diagram of an IPTV. The input transport streams in any encoded format like DVB –ASI, MPEG2 or SDI are passed through a multi-format CODEC to be transcoded (i.e., decoded and re-encoded) into an H.264 transport stream. The encoded transport stream is encapsulated with channel info and sent over Ethernet. On the receiving path, the incoming transport stream is decoded and post-processing like noise reduction, color enhancement, scaling, de-interlacing etc., are done before display.

Figure 2 shows a system block diagram of professional HD (high-definition) camera used in film making and studios. The image captured is passed through an image processing unit that does color processing, brightness enhancement, digital zoom, frame rate conversion, etc. This image processing unit is usually an FPGA-based design as most of the image processing is proprietary and often changing. The application processor manages communication with other equipment and compresses and stores the captured content on to mass storage (HDD). The application processor also has a graphics engine to generate an on-screen display (OSD) that is blended with incoming video to be displayed.
From the above examples, we can observe that there are two types of data handling involved:

**Frame synchronization:**
Frame synchronization is required for tasks such as transmission and reception over Ethernet where the bit-rate keeps varying whereas the decoder requires a constant bit-rate transport stream. Though the memory required for synchronization may seem small, it can be significant when multiple streams are involved. This synchronization is achieved with an asynchronous FIFO.

**Frame storage:**
Frame storage is required wherever any temporal processing like frame rate conversion, digital zoom (scaling), or de-interlacing is performed. The number of frames to be stored increases with the amount of temporal information required. As video data is sequential in nature, the frame buffer has to be essentially a FIFO.

From the above discussion, we can say that all the storage and synchronization can be achieved using FIFOs. To give an idea about the required size of these FIFOs, a typical 1080p frame in 10-bit 4:2:2 format would require a memory size of 39.55Mbit (number of pixel per line * number of lines per frame * number of bits per pixel = 1920*1080*20). The total size can be estimated by multiplying this figure by the number of frames that need to be stored. Typical video processing algorithms require 2 to 3 frames to be stored, which means the total size can be up to 120Mbit. As it is not possible to have such large on-chip SRAM based FIFO memory, the general approach is to use a DRAM to buffer this data.
High density FIFOs – conventional implementation and complexity:
Frame buffers are nothing but high-density FIFOs and conventionally they are implemented using external DDR SDRAMs. Consider a typical video processing application and how these FIFOs are implemented.

Figure 3 shows the data path for a typical scenario where four video streams from different sources are to be displayed on a single display. Four high-definition cameras capturing video with 1080p60 (24-bit RGB) resolution are connected to the system using a cameralink interface. After color space conversion (from RGB to YCbCr) and chroma down sampling (4:4:4 to 4:2:2), the frames are down scaled by a ratio of two both horizontally and vertically and stored in the DDR2 SDRAM memory. The stored frames are read back and positioned as required, and the resulting frame with merged frames is then up-sampled and color space converted to drive the panel over an LVDS link.

Let us look at the memory size and bandwidth requirements:

Size requirements
Though there is no temporal processing involved, to avoid tearing effects two frames of each source are stored so that when one frame is being written, the other can be read back. The size of two frames is \([(1920*1080*16)/4]*2 \approx 63.3\text{Mbit}.

Bandwidth requirements:
As the read and write path is multiplexed, the bandwidth required is the sum of the write and read path bandwidth.
Write path frequency = (frequency of each client)*(number of clients) = (148.5/4)*4 =148.5\text{MHz}
Read path frequency = the output frame resolution frequency = 148.5\text{MHz}.
The actual operating frequency would be \((\text{read frequency} + \text{write frequency})/2 + \text{overhead})\) as the interface is operating at double data rate and there is overhead such as DRAM memory refresh cycles, bank address switching, and so on. Assuming 80% efficiency, the frequency of operation would be around 185\text{MHz}. 

Published in EE Times Design (http://www.eetimes.com) April 2011
**Memory interface size and I/O requirements:**

As the frames are stored in 16-bit 4:2:2 format, a 16-bit interface is sufficient. The total number of I/Os required from the FPGA would be 46 pins as calculated below:

- Clock pins (2 for differential clock, 1 for clock enable) = 3 pins
- Command pins (chip select, RAS, CAS, WE) = 4 pins
- Address pins (14 address lines, 3 bank address lines) = 17 pins
- Data lines (X16 interface) = 16 pins
- Data strobe & mask (4 pins for 2 differential DQS, 2 pins for data mask) = 6 pins

**High density FIFO – as a discrete memory:**

Now let us look at an implementation using discrete programmable high-density FIFOs and define a feature set so that the DDR2 SDRAM memory can be replaced to simplify data storage.

**Multi-queue feature:**

It is impossible to write multiple video streams if the FIFO memory is defined as a single chunk of memory. Therefore, the FIFO must be able to be configured and divided into multiple queues. In the example scenario, there are four different frames to write and simultaneously four frames have to read from different queues. Thus, our application requires a minimum of eight queues.

**Mark and retransmit:**

Data once read from a standard FIFO is lost from the FIFO. The availability of a FIFO pointer that can be reprogrammable allows any frame to be read as many times as required.

*Figure 4* shows the block diagram of cypress CYFX072VXXX HD-FIFO.
Using high-density programmable FIFOs in video and imaging applications

Published in EE Times Design (http://www.eetimes.com) April 2011

*Figure 5* shows the example application with an CYPRESS HDFIFO replacing the DDR2 chip.
Let us look at the memory size and bandwidth requirements:

**Size requirements:**

The Storage size will remain the same as that of an DDR2 SDRAM memory which is equal to two frames 
\((1920*1080*16)/4)*2 \approx 63.3\text{Mbit}.

**Bandwidth requirement:**

As the read and write paths are separate, the operating frequency of read and write can be different. This offers a big advantage over DDR2 SDRAM memory.

- Write path frequency = (frequency of each client)*(number of clients) = (148.5/4)*4 =148.5MHz
- Read path frequency = the output frame resolution frequency = 148.5MHz.

The actual operating frequency would be 148.5MHz single data rate for both the read and write paths as there is no overhead such as DRAM memory refresh cycles and memory bank switching latencies involved.

**Memory interface size and I/O requirements:**

As the frames are stored in 16-bit 4:2:2 format, a 16-bit interface is sufficient. The total number of I/Os required from the FPGA would be 48 pins as calculated below:

- Clock pins (1 pin for write clock, 1 pin for read clock) = 2 pins
- Command pins (write enable, read enable, input enable, output enable, 3 pins to select which of the 8-queue to write, 3 pins to select which of the 8-queue to read, 1 pin for Mark, 1 pin for retransmit) = 12 pins
- Data pins (16 pins for write data, 16 pins for read data) = 32 pins
- Flags (1 pin for empty flag, 1 pin for full flag) = 2 pins

**Advantages of discrete HD-FIFO over conventional implementation:**

**Architectural advantages:**

(i) As the read and write paths are separate and there is no operating overhead, the operating frequency is reduced by more than half, offering a significant advantage.

(ii) The FPGA internal logic becomes simpler as the SDRAM controller and arbiter are eliminated.

(iii) The signal switching frequency is reduced more than half in turn, allowing increased setup time margins and relaxed clock-to-output constraints when compared to a DDR2 interface.

(iv) The number of clock domains in the design is reduced, in turn reducing handoff and cross clock domain related timing issues.

**Electrical advantages:**

(i) Reduced signal switching frequency reduces the amount of switching noise on the board.

(ii) The IO logic with an HD-FIFO can be any LVCMOS interface which has more noise margin compared to the SSTL2 logic of DDR2 SDRAM.

**Cost Savings:**

The FPGA resources on high-end FPGA solution which can be saved by using an HD FIFO comes from reduction of the following features:

(i) SDRAM Controller, which reduces the required memory, I/Os, and logic

(ii) Video processing features which can be implemented using the multi-queue feature on the HD FIFO, such as:

   a. Interlacing /De-interlacing of Video Signal
b. PIP implementation
c. Processing Interweaved Signals

The logic savings on logic elements, registers, memory, and I/O which are obtained from using a high-density FIFO enable developers to move from a higher-end FPGA to a smaller FPGA for cost savings ranging from 20 to 30%.

Figure 6. Block Diagram Comparing Systems With and Without High-Density FIFOs

Based on SRAM technology, high-density FIFO offers high data reliability and low latency. The easy-to-use bus interface reduces implementation and debugging efforts. With densities up to 144 Mb and speeds up to 150 MHz with segment-specific, value-added features such as multi-queue and selectable memory organizations, high-density FIFOs help developers design faster and more efficiently, making it ideal for a wide range of applications. It is an off-the-shelf solution that accelerates time-to-market and reduces associated engineering efforts. The device also offers width expansion options to suit the video broadcasting, military, medical imaging, and base station (networking) segments and caters to a host of applications such as:

• Frame buffers for common HD formats (720p, 1080i, 1080p): stores up to four frames of 1080p resolutions
• HDTV/SDTV frame synchronization
• Switcher or format converter box
• High-end digital video camera
• High-density buffering in military radars
• Medical imaging
• Base stations - 3G, 4G, and networking
Using high-density programmable FIFOs in video and imaging applications

Published in EE Times Design (http://www.eetimes.com)  
April 2011