Implementing JPEG2000 compression

David Dashefsky, Software Architect Digital Video Products, Analog Devices - May 12, 2006

As High Definition video becomes more prevalent in the market, compression has become an important, mandatory step in its transmission due to the sheer quantity of data involved. Most of us are familiar at a high level, if not with the details, of MPEG which is the standard for digital distribution of video. Many may also be familiar with H.264 which aims to replace MPEG in that role. So down in the cubical trenches, when marketing has decided our next big video thing has to ship tomorrow, naturally MPEG (or H.264) is the first place we’ll look. In the case of SD (standard definition), it may be sufficient, but for HD this may be an issue, as real-time, high quality HD MPEG encoding with very low latency is still the province of the high-end rack mount. If that scenario sounds all too real, there is hope. JPEG2000 offers affordable, real-time HD encoding and decoding with very low latency. Below, we’ll try and demystify the issue and then provide some simple examples to get your product out the door faster and with minimum fuss.

About JPEG2000

JPEG2000 is a wavelet-based compression standard. Despite the name, it’s only come into wide use recently. It has no temporal element; it compresses images on a frame by frame basis. This is what gives it the advantage of very low latency without the quality sacrifice you would make with I-frame only MPEG as well as the drawback of poorer performance at ultra-low bandwidths. Precise rate control is also an advantage for constrained channels, as JPEG2000 can take a specific rate target and not exceed it by truncating the data at an optimal point. So if your channel is 10Mbit/s, you can guarantee that you will not exceed that bandwidth.

![Figure 1: wavelength decomposition process](image)

JPEG2000 also has interesting features applicable to the transmission of video which are derived from the wavelet transform. The wavelet decomposition process (Figure 1) breaks the image down into resolutions, each half the size of the one above it. Each resolution is broken down into high and low frequency components in both the horizontal and vertical direction. The low pass vertical and low pass horizontal, or "LL" sub band, is the one which is then transformed again to a lower resolution. At the end of the transforms, the LL sub band is basically a thumbnail of the original image which contains all the most important information. The multiple resolutions of the transform give the user the ability to extract and decode a smaller resolution image from the code stream without decompression and scaling. It’s simply a matter of only decoding (or transmitting) the sub bands up to the desired resolution. The fact that the most important information is in the LL sub
band implies that in a lossy transmission environment, errors in the other 80% of the code stream will have little visual impact. This reduces the need to sacrifice bandwidth on further error correction, as well as the extra development effort needed to implement it.

In addition to the wavelet transform, the organization of the data in the final code stream of JPEG2000 places data in packets based on sub-bands for each resolution, precinct (location in the image), quality layer, and color component. You can selectively transmit or decode these packets to achieve whatever combination of quality, resolution, number of components, or image area you want all from the same original code stream with no need for decompression before you can work with it. Clearly, there are many video applications that could benefit from these features... but bridging the gap from problem to solution requires more than just a specification on paper.

**About the ADV202**

![ADV202, JPEG2000 codec](Figure 2: ADV202, JPEG2000 codec)

The ADV202, a System-On-A-Chip JPEG2000 codec (Figure 2) can provide the bridge to this gap. With hardware to do the most compute intensive aspects of the standard, and software to put it all together, the ADV202 provides a complete JPEG2000 codec solution in one chip for SD, or two chips for HD. The ADV202 requires no external memory; just a video interface on one side and a host to program and source/receive compressed data.

For High Definition, two chips are required because the maximum throughput of the video interface on the ADV202 is 65 Msamples/s. 720p and 1080i exceed that, so the components are split with Luminance data going to one ADV202, and Chrominance data going to the other, with both parts running on the 74.25Mhz video clock. One ADV202 can handle a standard definition stream in standard 656 at 27Mhz with the components interleaved.

In a nutshell, the ADV202 produces a JPEG2000 compliant code stream from each part. In the case of HD, there will be one code stream with only luminance, and one with only chrominance. This code stream is produced at full frame rate with slightly over one frame of latency for encode. The decisions an engineer will need to make to get off the blank page include: How to get video in/out of the part? How to get compressed data in/out of the part?
**Getting video in and out**

The ADV202 has one video input port. Though there are several modes available, two may be the most applicable. In both cases, they can be connected to a video encoder/decoder with no glue logic. The first mode uses embedded synchs- EAV/SAV to allow the part to synchronize with the data. The second mode uses separate H, V, and F synchs. Most video decoders have both of these modes so which one your design uses is just a matter of personal preference.

If you are doing decode, there is one extra decision to make. As an HD decoder the ADV202 can operate in either a Slave/Slave or a Master/Slave mode. In Master/Slave mode, one ADV202 drives the syncs for the other which is the slave. In Slave/Slave, both ADV202s are driven from an outside clock generator. Master/Slave is easier to implement in that you now only need to provide a pixel clock. Slave/Slave provides you with some more control to fine tune the video if your application requires that.

**Getting compressed data in/out**

The host bus of the ADV202 is a 32-bit asynchronous, SRAM-style interface. It can be run at 16-bits as well. There are other modes available, but for most applications this may be the most straightforward to implement. The control signals are standard: read/write strobe, 4 address pins, an IRQ, and a chip select. There is also an ACK signal which is good to use if you can. The “Direct” registers, of which there are 16, have consistent timing but there are "Indirect" registers as well which can take longer—the ACK signal lets the host know when the data is ready. Generally register access of any kind is only done at start up when you are configuring the part. After that, the compressed data is all handled with DMA.

For data transfer, there is a burst mode DMA using DREQ/DACK signaling that provides plenty of bandwidth (up to 50MHz). The user provides a DMA burst length during configuration and the internal firmware will make sure that all compressed data frames are padded out to end cleanly on a burst boundary to simplify the interface. An additional pin, "LCODE", is provided which will pulse on the last burst of a given frame. In this way, the host can know when a frame is done without parsing the headers to determine its size.

**Encode**
Now that we’ve gone over the basic interfaces of the part, we can show a simplified JPEG2000 HD Encoder. Figure 3 shows the essentials of an HD Encoder for 1080i or 720p. Both formats operate with a VCLK of 74.25Mhz and the Luminance goes to one ADV202 and the Chrominance goes to the second ADV202. The “host” that is shown could be a CPU/DSP or it could be an FPGA or some combination. There are several CPU/DSPs out there which are capable of interfacing with a small amount of glue logic. Using an FPGA, if one has the cost structure to allow it, may be your safest bet as it offers opportunities to resolve issues that may come up that you would be stuck with using a fixed function CPU/DSP.

**Next: Setting JPEG 2000 parameters** There are a dizzying array of parameters that can be set for JPEG2000 encoding; however, for most applications the only ones that matter are rate control and video format. The ADV202 has 2 basic types of rate control: Target Quality and Target Size.

For those looking for a specific bit rate, Target Size is your mode. It allows you to set a specific byte target for each frame/field that the ADV202 will not exceed within 5%. Taken together with the FPS, this gives you a Mb/s that can be easily understood. A positive side effect of the two-chip configuration is that you can set the bit rates of the Luma and Chroma components differently. As it turns out, image quality perception is far more heavily influenced by Luma. Knowing this, you can assign most of the bytes to Luma and achieve good image quality for an overall lower bit rate—in practice this ratio can be as high as 10:1 for normal video without being noticeable to the average viewer.

Once you have completed the startup procedure for encode (documented on our website), the ADV202s should run with little intervention. The main problem that one can run into involves being too slow to pull data out and overflowing the ADV202’s internal memory. The part will hang if this happens and require a reset.

**Decode**
Figure 4: Master/Slave HD Decoder

Figure 4 shows a Master/Slave HD Decoder. This is the simplest to get working and may be the best for many designs. In this situation, an external 74.25Mhz Oscillator is necessary to set up the VCLK but the Master ADV202 will then output the synchs. In order to ensure that the Luma and Chroma chips are outputting the same frame at the same time, the ADV202 has a hardware output enable on SCOMM[5]. The procedure for this is that the host starts feeding in the code stream and when each chip is ready, it will assert IRQ with the SWIRQ1 flag on. When both chips are ready, the host asserts SCOMM[5] and the data will then start coming out.

For decode, there isn’t much in the way of parameters since they have already been set in the encode process. The JPEG2000 code stream contains most of the data needed by the ADV202 to understand how to decode it. The user will have to set the video format and code stream type, as well as the clocking configuration. Beyond that, the code stream itself will provide the configuration.

As with encode, once the startup procedure is completed the ADV202s should run with little need for intervention. If the host is slow to put data in and video data isn’t ready when the syncs come, then the ADV202 will output a black field/frame.

More JPEG2000 applications
Though it may appear daunting, getting basic High Definition JPEG2000 encode or decode doesn’t have to be too painful. The ADV202 provides a quick way to add this functionality to your product with as small a learning curve as you can get given the complicated technologies involved. HD video is a fairly well understood application with the ADV202 but there are many other modes and applications to which JPEG2000 and the ADV202 can be applied. Still imagery, streaming media, other video formats, medical and security imagery all are possible with varying levels of effort. The details and documentation are available from our ftp site at ftp.analog.com\pub\Digital Imaging. As users demand higher quality imagery and MPEG starts to top out, JPEG2000 can provide your product with a fast way to elevate your system’s image quality above the fray.

About the author
David Dashefsky is a Software Architect in the Digital Imaging Systems Group at Analog Devices where he has been for 2 years. He holds a BSEE and an MS in Business from the University of Texas at Austin. He has also worked on projection electronics at 3M for 8 years. He can be reached at...