How to Add Low-Power,
Multi-Codec, Digital Video
and Audio to Your Next
ASIC or SOC Design

Since the early 1990s, video compression has grown increasingly important for the
design of modern electronic products because it aids in the quest to deliver high-
quality video using limited transmission bandwidth and storage capacity. Many
consumer products now incorporate video-recording and video-playback functions.
ASIC, SOC and ASSP design teams developing chips for such products need ways to
quickly add proven, tested audio/video functions to their designs. It is now possible
to obtain tested, off-the-shelf IP blocks and firmware that implement multiple digital-
video and audio codecs. These products allow ASIC and SOC design teams to
literally drop digital audio and video into their design without the need for detailed
knowledge of the new video standards.

Video Basics

The first widely successful video-compression standard was MPEG-2. It is used in
DVD players, its first big success, and then grew into other applications such as
satellite and set-top boxes. The MPEG-4 and H.264/AVC video-compression
standards then followed some years later. With all video-compression standards, the
goal is to deliver high-quality video with minimum bandwidth. As compression
technology advances, codec (coder-decoder) developers can exploit increasing
processor performance to achieve higher compression ratios while delivering images
with much higher visual quality.

Digital video encoding starts with a series of still images (frames) captured at a
certain frame rate (usually 15, 25, or 30 frames/sec) by cameras use CCD or CMOS
sensors to capture the images. These sensors capture red, green, and blue (RGB)
light but the RGB images they produce do not directly correspond to the way the
human eye works. The human eye uses rods and cones to separately sense light
intensity (luma) and color (chroma). The eye is more sensitive to luma than chroma
because it contains more rods than cones. Consequently, most video-compression
systems start by transforming RGB pictures into luma and chroma (YUV) images.
To save bits in the video stream, compression schemes subsample the image’s
chroma portion. Thus most digital video compression schemes transform a series of
YUV images into a compressed video stream while video decompression streams
expand a compressed video stream into a series of still images coded in YUV format.
Digital video compression schemes use many of the lossy compression techniques originally developed to compress still images. Lossy compression techniques identify and discard portions of an image that cannot be perceived by the human eye. Digital video compression benefits even more from lossy compression schemes because any image imperfections produced by compression appear fleetingly in the video stream and are therefore even less perceptible.

Digital video-compression algorithms:

- Divide pictures into small pixel blocks
- Predict values for these pixel blocks
- Compute the differences (residuals) between the predictions and the actual values, and then
- Transform these residuals from the spatial domain into a series of coefficients in the frequency domain.

The DCT (discrete cosine transform) is commonly used to transform from the spatial to the frequency domain. DCT was first widely used for JPEG still-image compression. Most video-compression schemes prior to the H.264/AVC digital-video standard employ the DCT transform on 8x8-pixel blocks but H.264/AVC uses a simpler, integer-based transform on 4x4-pixel blocks.

Images tend to contain more lower-frequency image patterns. Quantizing a pixel blocks’ frequency-domain representation using low coefficient bit depths tends to remove higher-frequency patterns by driving the coefficients for these patterns to zero, which reduces the number of bits needed to represent that block. Encoding the frequency-domain coefficients takes two steps. First, the coefficients are quantized to discrete levels using perceptual weighting to limit the number of coefficient bits. The quantized coefficients are then compressed using a lossless variable-length-coding (VLC) coding technique that assigns fewer bits to frequently occurring coefficient numbers, which again reduces the size of the video bitstream.

**Wringing Out Redundant Video Information**

VLC compression is called entropy coding. Huffman coding was the most common entropy-coding method used for video compression prior to the introduction of H.264/AVC compression. The H.264/AVC standard uses two entropy-coding methods called CAVLC (context-adaptive, variable-length coding) and CABAC (context-adaptive, binary arithmetic coding). CABAC coding, which is more computationally demanding, uses a non-integer number of bits for VLC encoding. CABAC coding improves the bit-compression ratio by roughly 10% over CAVLC for a given quantization quality level. The trade off for improving the bit-compression ratio with CABAC coding is an increased need for processor performance, obtained through a higher processor clock rate, additional operation parallelism in the processor’s execution unit, or both.
Still-image compression methods can deliver compression ratios of 10:1 while still producing good quality images. (Higher compression ratios deliver visibly inferior images.) Video consists of a stream of still images and video frames compressed using still-image compression techniques are called “I frames” (intra frames). Because each video frame in the image stream tends to be related to the images that come before and after, there is correlation and redundancy among successive frames (except when there’s a jump cut or sudden frame blackout).

Digital-video compression schemes exploit temporal inter-frame redundancy to achieve much higher compression ratios, approaching 200:1, while still delivering good quality video. In the simplest case of a video stream that represents an unchanging scene, a video-compression scheme merely needs to tell the video decoder to repeat the last image, which requires very few bits and results in very high compression. Most successive video frames differ by at least a few pixels (by many pixels if the video includes a lot of motion or there’s a scene change), so video-compression schemes must use a variety of motion-dependent compression methods to accommodate different types of video.

One simple method involves subtracting one frame from a previous frame and then encoding the difference. Motion JPEG and Motion JPEG2000 video codecs use such a scheme. For video streams where there’s little movement, this approach can be very efficient. More advanced schemes employ a technique called motion compensation, which breaks frames into macroblocks and searches a previously encoded frame for a similar macroblock. A motion-estimating video encoder then encodes a motion vector that tells the video decoder where a previously decoded macroblock will appear within the new frame. This scheme requires matching video decoders to save macroblocks from the previous frame in a frame-buffer array for possible reuse.

A reference frame used for motion estimation need not be a previous frame. Video-compression techniques can encode frames out of temporal display order (especially for recorded video that’s not being encoded in real time), which means that intermediate image frames can be based on previous frames, future frames, or both. Frames based on single frames alone are called “P” frames or slices. “B” frames or slices are encoded using information from two frames. Motion estimation represents a very large search problem, which is computationally intensive and can consume billions of operations per second.

Figure 1 shows simplified block diagrams for a video-compression encoder and decoder based on the techniques just discussed. Video encoding is more complex than video decoding and a video encoder includes most of the blocks found in a corresponding decoder, which the encoder uses for motion estimation.
The two block diagrams in Figure 1 show the sequential nature of the steps used for video compression and decompression. All of these steps can be performed by a microprocessor. However, general-purpose processors must operate at very high clock speeds to achieve the required computational bandwidth for video compression and decompression. Multi-GHz processors draw substantial amounts of power and are inappropriate for portable and many other video applications. Multi-GHz processor cores for SOC designs don’t yet exist and they consume vast amounts of power. So there are tremendous benefits to finding more efficient video-codec processor hardware.

The block diagrams in Figure 1 also clearly show that the encoding and decoding algorithmic sequences contain substantial inherent parallelism. Frames are passed down an assembly line where separate operations are performed sequentially. A separate processor can perform the function of each box in the Figure 1 block diagrams, which would greatly reduce the required clock rate. The lower clock rate would substantially reduce operating power to levels compatible with fanless, line-powered products and portable, battery-powered devices.
A Hardware Video Codec

Figure 2 shows a block diagram of Tensilica’s 388VDO Video Engine, which uses separate microprocessor cores (a Stream Processor and a Pixel Processor) and a DMA controller to exploit the parallelism inherent in video-compression and video-decompression algorithms. The Stream and Pixel Processors inside the 388VDO core split the video-compression tasks while the DMA controller moves uncompressed images into and out of the core and between the two processors. Each processor inside the 388VDO Video Engine has its own local instruction and data RAMs.

The 388VDO Video Engine’s Stream and Pixel Processors are based on Tensilica’s configurable Xtensa processor architecture. The Stream Processor has been augmented with additional instructions to perform bitstream parsing and entropy coding. Some of these new instructions are based on Tensilica’s FLIX (flexible-length instruction extensions) and employ a VLIW instruction format with two independent operations per instruction. The 388VDO Video Engine’s Pixel Processor has been augmented with SIMD (single-instruction, multiple data) instructions that perform operations on multiple pixels simultaneously.
The instructions added to both processors allow the 388VDO Video Engine to efficiently implement the following codecs:

<table>
<thead>
<tr>
<th>Decoders</th>
<th>Encoders</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 Baseline Profile</td>
<td>H.264 Baseline Profile</td>
</tr>
<tr>
<td>H.264 Main Profile</td>
<td>JPEG</td>
</tr>
<tr>
<td>JPEG</td>
<td>MPEG-4 Simple Profile</td>
</tr>
<tr>
<td>MPEG-2 Main Profile</td>
<td>MPEG-4 Advanced Simple Profile</td>
</tr>
<tr>
<td>MPEG-4 Simple Profile</td>
<td></td>
</tr>
<tr>
<td>MPEG-4 Advanced Simple Profile</td>
<td></td>
</tr>
<tr>
<td>VC-1/WMV9 Simple Profile</td>
<td></td>
</tr>
<tr>
<td>VC-1/WMV9 Main Profile</td>
<td></td>
</tr>
<tr>
<td>RealVideo 9/10</td>
<td></td>
</tr>
</tbody>
</table>

The 388VDO Video Engine’s efficiency (in terms of both clock rate and memory bandwidth) is apparent from the following performance summary:

<table>
<thead>
<tr>
<th>Video Standard</th>
<th>Pixel Rate</th>
<th>Bit Rate</th>
<th>Maximum Required Clock Rate</th>
<th>Required DRAM Bandwidth</th>
<th>Power Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 Main Profile Decode</td>
<td>D1</td>
<td>5 Mbps</td>
<td>162 MHz</td>
<td>86.2 Mbytes/sec</td>
<td>59 mW</td>
</tr>
<tr>
<td>MPEG-4 Advanced Simple Profile Decode</td>
<td>D1</td>
<td>6 Mbps</td>
<td>167 MHz</td>
<td>59.8 Mbytes/sec</td>
<td>35 mW</td>
</tr>
<tr>
<td>VC-1/WMV9 Main Profile Decode</td>
<td>D1</td>
<td>6 Mbps</td>
<td>172 MHz</td>
<td>88.9 Mbytes/sec</td>
<td>50 mW</td>
</tr>
<tr>
<td>MPEG-2 Main Profile Decode</td>
<td>D1</td>
<td>8 Mbps</td>
<td>151 MHz</td>
<td>46.1 Mbytes/sec</td>
<td>38 mW</td>
</tr>
<tr>
<td>MPEG-4 Advanced Simple Profile Encode</td>
<td>D1</td>
<td>4 Mbps</td>
<td>188 MHz</td>
<td>148 Mbytes/sec</td>
<td>TBD</td>
</tr>
</tbody>
</table>

A key feature target that guided the development of the 388VDO Video Engine was the goal of decoding and encoding video bit streams at standard-definition (SD or D1) display resolution and 30 frames/sec while running at clock rates below 200 MHz. Low clock rates generally result in lower power requirements and the 200-MHz clock rate was selected as a goal so that the 388VDO Video Engine could be implemented in a generic, low-cost 130nm IC-fabrication process (TSMC 0.13G). As you can see in the table above, that goal was met. More advanced IC fabrication processes allow the 388VDO Video Engine to achieve even faster clock rates and to deliver even more performance while consuming less power.

Figure 3 shows the task allocation within the 388VDO Video Engine while decoding H.264/AVC video bit streams. The Stream Processor performs bit-stream parsing (separating the Network Abstraction Layer, the Picture Layer, and the Slice Layer) and entropy decoding. The Pixel Processor performs inverse quantization, inverse transform coding, intra-frame prediction, motion compensation, and image deblocking. The Stream Processor assists the Pixel Processor with motion compensation.
Figure 3: Task allocation within the 388VDO Video Engine for H.264/AVC decoding.

Note that it would have been possible to run all of these decoding tasks on one processor but at a much higher clock rate, which would require a more expensive process technology. Given the number of available transistors in nanometer IC lithographies and the need to minimize power dissipation in portable, battery-powered video products, the division of labor among the two processors and the DMA controller within the 388VDO Video Engine accomplishes the goal of minimizing power dissipation by keeping clock rates low even when decoding video at standard-definition and higher resolutions.

Frames In/Frames Out

Although video encoding and decoding are complex algorithms, the 388VDO Video Engine simplifies digital video for an SOC design team by acting as a black-box hardware video codec, as shown in Figure 4. To encode video, the system’s host control processor first configures the 388VDO Video Engine with a command and then passes unencoded video frames to the engine, which encodes the frames and passes back encoded images (called “video decodable entities” or VDEs) to the host processor. For video decoding, the host processor first configures the 388VDO Video Engine appropriately with a command and then passes VDEs to the 388VDO Video Engine, which decodes the images and passes decoded frames back to the host processor.
Figure 4: Passing encoded and unencoded video to and from the 388VDO Video Engine.

Figure 5 shows the host control processor running a media-player application. The host control processor uses predefined API calls to operate the 388VDO Video Engine. The host control processor uses two queues in main system memory to send command messages and data to the 388VDO Video Engine and two more queues to receive status messages and data from the 388VDO Video Engine. Firmware-driven interrupts from the host control processor to the 388VDO Video Engine initiate the queue-based message transactions.

Figure 5: The 388VDO Video Engine’s queue-based, message-passing API.
A simple design for a video SOC based on the 388VDO Video Engine appears in Figure 6. The 388VDO Video Engine, like the other system components, attaches to the main system bus in this SOC design. A host system-control processor directs the SOC’s operation and runs code from on-chip and off-chip memory while Tensilica’s 330HiFi Audio Engine core—a popular solution for on-chip SOC audio, used by companies such as Samsung, LG, ATI, and Nvidia—provides digital-audio decoding in this example. The 330HiFi Audio Engine, which is currently in volume production inside of mobile-phone SOCs, runs a wide and growing range of ready-to-run digital-audio codecs. Note that in many designs, the 330HiFi Audio Engine can also serve as the host control processor.

Both the 388VDO Video Engine and the 330HiFi Audio Engine illustrate how processors and ready-to-run firmware can form the core of a complex, high-performance, low-power block for complex chip designs. Using the block-oriented approach to design illustrated in Figure 6, development teams can rapidly assemble extremely complicated SOCs from complex, proven IP cores and then program these SOCs with application code to produce unique products for the market.
An Open Invitation

If your design team is interested in incorporating digital video and audio into your next SOC design, contact Tensilica for assistance. For more information on the unique abilities and features of Tensilica’s 388VDO Video Engine and 330HiFi Audio Engine, see www.tensilica.com, email sales@tensilica.com, or contact Tensilica directly at:

**US Sales Offices:**
Santa Clara, CA office:
3255-6 Scott Blvd.
Santa Clara, CA 95054
Tel: 408-986-8000
Fax: 408-986-8919
San Diego, CA office:
1902 Wright Place, Suite 200
Carlsbad, CA 92008
Tel: 760-918-5654
Fax: 760-918-5505
Boston, MA office:
25 Mall Road, Suite 300
Burlington, MA 01803
Tel: 781-238-6702 x8352
Fax: 781-820-7128

**International Sales Offices:**
Yokohama office (Japan):
Xte Shin-Yokohama Building 2F
3-12-4, Shin-Yokohama, Kohoku-ku,
Yokohama
222-0033, Japan
Tel: 045-477-3373 (+81-45-477-3373)
Fax: 045-477-3375 (+81-45-477-3375)

UK office (Europe HQ):
Asmec Centre
Eagle House
The Ring
Bracknell
Berkshire
RG12 1HB
Tel: +44 1344 38 20 41
Fax: +44 1344 30 31 92

Israel:
Amos Technologies
Moshe Stein
moshe@amost.co.il

Beijing office (China HQ):
Room 1109, B Building, Bo Tai Guo Ji,
122th Building of Nan Hu Dong Yuan, Wang Jing,
Chao Yang District, Beijing, PRC
Postcode: 100102
Tel: (86)-10-84714323
Fax: (86)-10-84724103

Taiwan office:
7F-6, No. 16, JiHe Road, ShihLin Dist,
Taipei 111, Taiwan ROC
Tel: 886-2-2772-2269
Fax: 886-2-66104328

Seoul, Korea office:
27th FL., Korea World Trade Center,
159-1, Samsung-dong, Kangnam-gu,
Seoul 135-729, Korea
Tel: 82-2-6007-2745
Fax: 82-2-6007-2746

If your design team is interested in incorporating digital video and audio into your next SOC design, contact Tensilica for assistance. For more information on the unique abilities and features of Tensilica’s 388VDO Video Engine and 330HiFi Audio Engine, see www.tensilica.com, email sales@tensilica.com, or contact Tensilica directly at: