DSP SEE-THROUGH: GOING BEYOND TALK-THROUGH

Adrian Rothenbuhler[†] *Cameron H.G. Wright*[‡] *Thad B. Welch*[⋆] *Michael G. Morrow*[◇]

[†]Hewlett-Packard, Boise, ID; Adrian.Rothenbuhler@HP.com
[‡]Dept. of Elec. and Comp. Engineering, University of Wyoming, WY; c.h.g.wright@ieee.org
[⋆]Dept. of Elec. and Comp. Engineering, Boise State University, ID; t.b.welch@ieee.org
[◇]Dept. of Elec. and Comp. Engineering, University of Wisconsin - Madison, WI; morrow@ieee.org

ABSTRACT

Engineering educators have found that students making the transition to real-time digital signal processing (DSP) from the more comfortable world of off-line processing using MATLAB must establish confidence in the hardware and software platform before significant learning can begin. In the audio realm, a talk-through project accomplishes this. For moving on to a more complicated signal such as video, the authors propose the use of a see-through project. A description of a see-through project on a high-performance real-time DSP platform, and how this can lead to better follow-on learning, is provided.

***Index Terms*—** digital signal processing, video processing, engineering education

1. INTRODUCTION

Competence in digital signal processing (DSP) topics is now expected by most employers of new electrical and computer engineering (ECE) graduates. While the subject may be taught various ways, it has been found that a solid understanding of many fundamental DSP topics is more fully realized by students when they attempt to implement various DSP algorithms in real-time (typically in C), as compared to non-real-time (i.e., off-line) implementations with tools such as MATLAB or LabVIEW [1]. Interactive learning, exercises, and demonstrations to students using off-line methods are very useful for helping them to build an initial mental model [2–6]. But making the transition to real-time DSP implementations cements a more complete understanding.

The authors of this paper have, over the last decade or so, reported on proven DSP teaching methodologies, hardware and software solutions, and various DSP tools that have helped motivate students and faculty to implement real-time DSP-based systems to improve education in signal processing and related topics [7–18]. This support to educators includes a textbook and a web site that specifically helps both professors and students (and working engineers) master real-time DSP concepts [19, 20].

There is an initial stumbling block that can greatly impede student progress. We have found that when students are first making the transition from the "comfortable" world of off-line signal processing (typically using MATLAB) to real-time DSP, they must quickly establish confidence in the hardware and software platform before significant learning can begin. Without such confidence, any errors or incorrect results from their DSP efforts are quickly blamed on the platform. The students will almost never investigate further to uncover other possible reasons for the flawed outcome, mainly because they are not yet comfortable with the new platform. To establish such confidence, a "stripped down" first exercise is used that tests the ability of the platform to correctly acquire data samples as input and provide unmodified samples as output. No signal processing algorithm is executed by the processor to modify the samples.

Correct output thus confirms proper initialization and configuration of all the hardware, a correct definition of the interrupt vector table, correct execution of the appropriate interrupt service routines (ISRs), proper operation of the ADC and DAC aspects of the associated codec, correct execution of the compile-link-load software development chain, and even correct connection of all the necessary cables and wires. A problem with any of these would cause the first exercise to fail, at which point the instructor can guide the student toward resolution of the problem. Once the first exercise is successfully completed, students are much more likely to take a more critical and investigative approach to any errors they may encounter in more sophisticated exercises. At this point, real learning can proceed. While such a "do nothing" program may seem to be a trivial exercise, we have been convinced by our own and our colleagues' experiences that skipping such a step impedes learning.

When dealing with signals in the audio range, this first exercise has been called *talk-through*. An analog audio signal is acquired by codec's ADC channels at a certain sample frequency, and (hopefully) the same audio signal is output from the codec's DAC channels (within the limits of realities such as quantization error, of course). Minimal modifications to the talk-through exercise can introduce concepts such as aliasing, quantization, left+right versus left–right channel combinations, and so forth. While the audio signal is often not speech, and is more often music, the name talk-through has stuck.

At some point, the professor may want to introduce the students to a more complicated signal, perhaps at a higher frequency and wider bandwidth. One such signal that is readily available using low-cost equipment and one that also seems to excite students even more than audio is a video signal. For moving up to video signals, the authors propose the use of a logical extrapolation of talk-through that we call a *see-through* project.

2. SEE-THROUGH FOR VIDEO SIGNALS

Our current choice of high-performance real-time DSP hardware for classroom use is the relatively new Texas Instruments OMAP-L138 Low Cost Development Kit (LCDK) [21, 22]. One of the many advantages of this platform is the plethora of I/O choices. Most germane to this discussion is the video input and video output shown in Fig. 1.

The video input is intended for a standard definition television quality composite video signal such as NTSC, PAL or SECAM. NTSC, for example, is an analog baseband signal with a nominal 6 MHz bandwidth [23, 24]. The video is interlaced at approximately 60 fields per second and 30 frames per second.[1] Each field is 262.5 scan lines; two fields per frame results in 525 scan lines per frame, of which only 483 lines can be visible due to factors such as synchronizations and vertical blanking/retrace. NTSC has subcarriers defined for luma, chroma, and audio. Obviously, this video signal is a significant step up in complexity from a two-channel, 20 kHz bandwidth audio signal sampled at 48 kHz, and the pitfalls for the student are that much greater.

On the LCDK, the analog video input is digitized, decoded, and formatted by the TVP5147M1 digital video decoder. This chip includes an ADC stage (at up to 30 Msps) and the necessary circuitry for extracting and sending as output the luma and chroma information as separate 10-bit data streams (via intermediate steps of YUV $\rightarrow$ YCbCr $\rightarrow$ luma (Y) and chroma (C) format). The chroma data stream provides 5 bits/sample for Cb and Cr, and is interleaved as [CbCrCbCr$\cdots$]; the luma data stream uses the full 10 bits/sample for Y. No audio demodulation is performed by this chip. While the LCDK implementation of this decoder chip only takes advantage of a small subset of the available modes, the user must still fully configure the chip via an I^{2}C bus to set the appropriate mode of operation.

The analog video output from the LCDK is produced by the THS8135 video DAC; this chip accepts digital input formats as either YCbCr or RGB and supplies a standard VGA (analog RGB) signal as output. On the LCDK, this output connects to a DB-15 connector.

[1]More precisely, NTSC is 59.94 fields/sec and 29.97 frames/sec.

On the surface, a see-through exercise seems straightforward. Simply connect an analog video camera to the LCDK's video input, and a VGA-compatible monitor to the DB-15 VGA output, write a bare-bones "do nothing" real-time DSP program to bring in the input and send out the output. . . but in reality it's not nearly that easy. The initialization must include extensive configuration code to set the proper I/O, interrupt enable, interrupt vector table, initial setup of the TVP5147M1 digital video decoder, DMA channels, and so forth, which is not trivial but is only done at startup. However, the actual real-time frame to frame operation isn't straightforward either. While the TVP5147M1 digital video decoder outputs YCbCr, and one of the input modes of the THS8135 video DAC accepts YCbCr, the LCDK board is constructed such that the configuration pins on the THS8135 chip are hardwired to set the mode to RGB input only. Thus at a minimum, a conversion from YCbCr to RGB must be part of the real-time see-through code.

Conversion from YCbCr to RGB is a simple linear relationship. Defined most basically, independent of the number of bits per sample, the conversion is

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.402 \\ 1.0 & -0.3441 & -0.7141 \\ 1.0 & 1.772 & 0.00015 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

where it is assumed that the values are normalized such that the range of the RGB values and the luma Y values is $[0, 1]$, the range of the chroma Cb and Cr values is $[-0.5, +0.5]$, and any head-room or toe-room has been removed by rescaling to full-range as needed. Note that there are minor variations on the conversion coefficients depending upon what version ITU standard is appropriate. The normalized values shown here are correct for ITU-R BT.601, which are the same ones used for the JPEG and MPEG standards.

Since some processing had to be performed inside the see-through ISR anyway, we considered it an opportunity to explore certain aspects of the OpenCV library and the TMS320C6748 SYS/BIOS Software Development Kit (SDK) provided by Texas Instruments (TI); see [25]. In particular, we used basic parts of the "Facedetect" and "VPIF Loopback" example projects from the SDK.

3. IMPLEMENTATION DEMO

This SDK take advantages of the full OpenCV library. OpenCV is an open source computer vision library started by Intel in 1999 and transferred to the non-profit OpenCV.org group in 2012 [26]. While see-through, by definition, should perform minimal processing, we wished to retain the ability to call OpenCV routines for follow-on projects that would be based on the see-through project.

The CCS example projects as supplied with the SDK required the full 2 GB SDK, but stripping this down to only the necessary device drivers and library files dropped this to

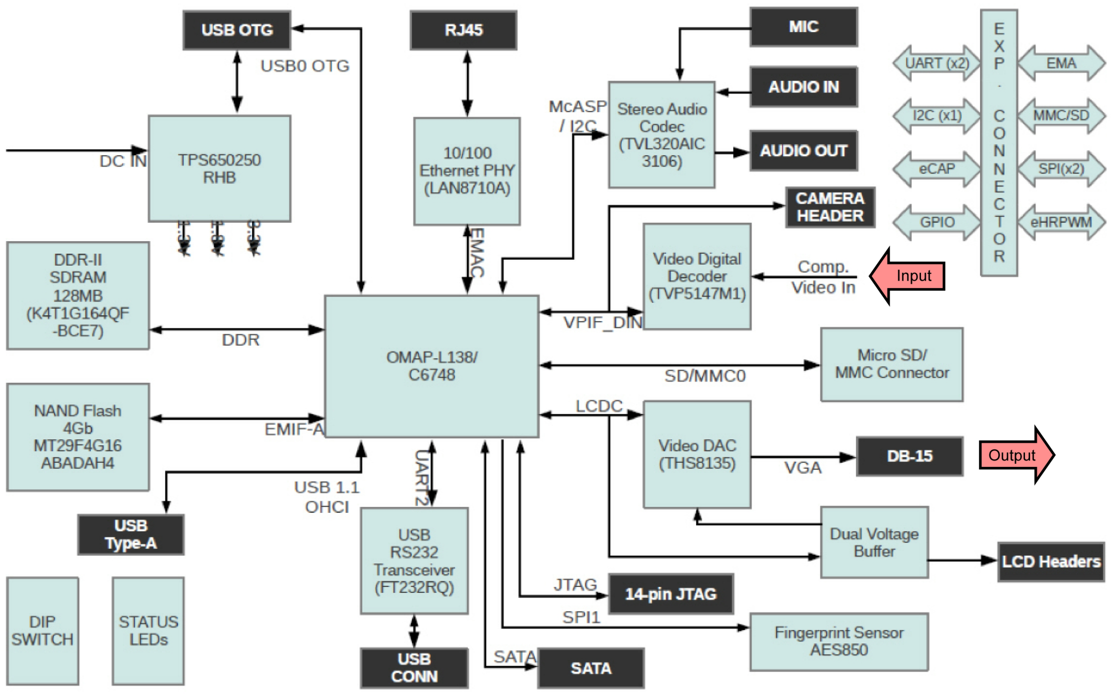

Fig. 1. Block diagram of the LCDK from Texas Instruments. Note the video input and output capability.

just under 20 MB (OpenCV itself, once precompiled by the user, is 18 MB of that total). The original example projects also used the SYS/BIOS real-time operating system (RTOS) from TI, which we have found is not a good choice for first exposure to students. The SYS/BIOS RTOS adds considerable complexity, overhead, and executable file size to support functionality we don't want or need for projects that are intended for real-time DSP students. Therefore, we further stripped down and borrowed from the SDK example projects, removing the dependence on SYS/BIOS, and created a real-time project in the manner recommended in [19].

In the project, `main.c` initializes the hardware and the camera, then calls `Process_Video` in an endless loop. The `Process_Video` function waits for a frame to be captured before proceeding. At the end of each incoming frame from the camera, an interrupt is generated.

There are two ISRs, `VPIFIsr` and `LCDIsr`. The `VPIFIsr` ISR brings in the video data and stores it using separate frame buffers for the luma and chroma; a double buffer (i.e., ping-pong) method is used for each. At this point, the `Process_Video` function resumes and performs the YCbCr $\rightarrow$ RGB conversion via a call to the `cbcr422sp_to_rgb565_c` TI utility function. Note this conversion does not occur inside an ISR. The `LCDIsr` ISR continuously points the appropriate frame buffer (now containing RGB values) to the output raster buffer for DMA transfer and display on the monitor; this ISR also has a placeholder where some real-time image processing algorithm can be executed. Since most standard image processing algorithms assume RGB values as the starting point (not YCbCr), this is the best location for such a placeholder. As a simple test, the `cvRectangle` OpenCV function is called here to place a small blue rectangle on the screen over the video data.

It should be noted that, for improved speed, the frame buffers for this project are created in the DDR RAM of the LCDK's L1 cache rather than in the regular external DDR RAM memory space.

The real-time see-through project provided an excellent demonstration of bringing in a video signal from a camera and sending it out for display on a monitor, as shown in Fig. 2. Note the blue rectangle displayed on top of the real-time video image. This "simple" real-time demo is highly motivating for students, and provides many opportunities to segue into various basic concepts. For example, sampling and aliasing can be discussed in terms of measuring the frame rate of the system by imaging a variable-speed rotating disk with a high-contrast marker on it. As the disk speed is steadily increased, it appears to start slowing down when the rotational rate exceeds half the frame rate (i.e., $F_s/2$) and aliasing occurs. The disk appears to be stationary when the rotational rate equals the frame rate (i.e., F_s). This is also a good time to show students that, while the fully optimized "Release Build" of the project can run at the full 29.97 frames/sec rate of NTSC, the non-optimized "Debug Build" can only run at approximately 6.5 frames/sec.

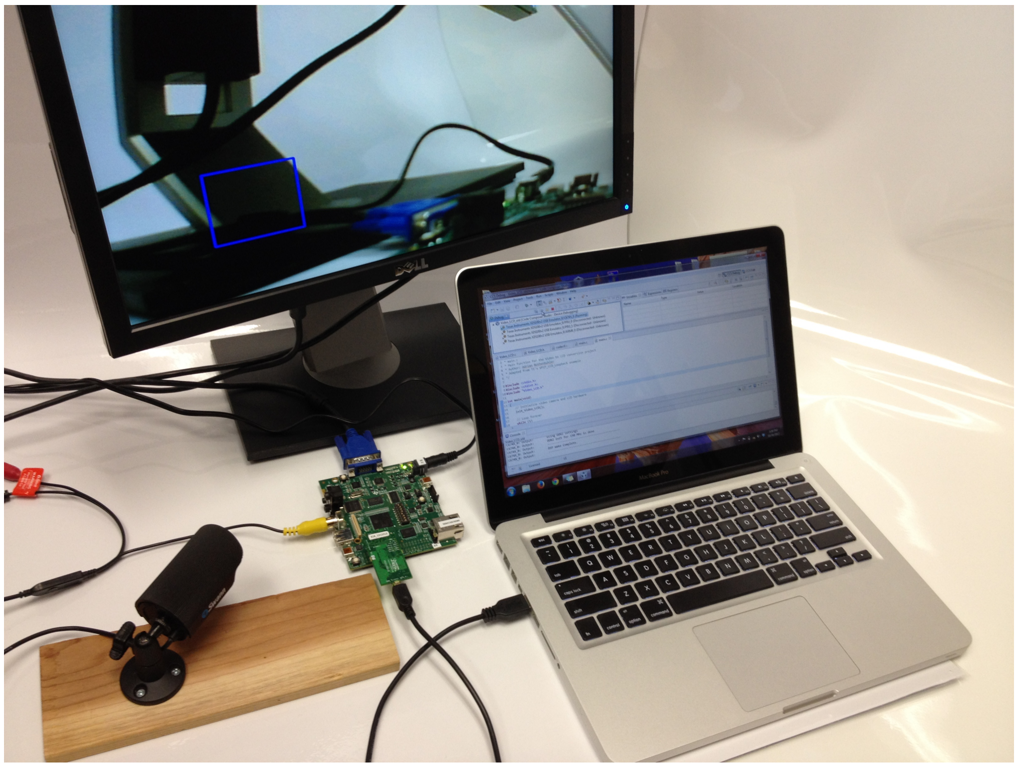

Fig. 2. A demonstration of the see-through real-time DSP exercise.

4. CONCLUSIONS

Including a bare-bones first project, such as talk-through or see-through, to build student confidence in the real-time platform is a valuable pedagogical approach. Skipping this step has been seen to greatly reduce student motivation to pursue the real cause of incorrect results from a real-time DSP exercise, as students are otherwise quick to "blame" the platform.

When making the change from talk-through to see-through, a host of additional considerations and complications must be addressed. Not only is the video signal itself more complicated, but the configuration and use of the input and output chips specific to video are more challenging to use than an audio codec. An additional requirement when using the LCDK for video see-through is the need for conversion from YCbCr to RGB.

We have built such a see-through project and successfully run it at the full frame rate of NTSC video using the LCDK. In the future, we plan to more completely strip away the reliance on pieces of the original example projects supplied with the TMS320C6748 SYS/BIOS Software Development Kit.

Faculty who teach DSP are strongly encouraged to incorporate demonstrations and hands-on experience with real-time hardware for their students, and to include talk-through and see-through as confidence-building projects for the students. We have made various resources widely available to help in this endeavor [20, 27].

5. REFERENCES

[1] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching DSP: Bridging the gap from theory to real-time hardware," *ASEE Comput. Educ. J.*, pp. 14–26, July–September 2003.

[2] C. S. Burrus, "Teaching filter design using MATLAB," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 20–30, Apr. 1993.

[3] R. F. Kubichek, "Using MATLAB in a speech and signal processing class," in *Proceedings of the 1994 ASEE Annual Conference*, pp. 1207–1210, June 1994.

[4] R. G. Jacquot, J. C. Hamann, J. W. Pierre, and R. F. Kubichek, "Teaching digital filter design using symbolic and numeric features of MATLAB," *ASEE Comput. Educ. J.*, pp. 8–11, January–March 1997.

[5] J. H. McClellan, C. S. Burrus, A. V. Oppenheim, T. W. Parks, R. W. Schafer, and S. W. Schuessler, *Computer-Based Exercises for Signal Processing Using* MATLAB 5. MATLAB Curriculum Series, Upper Saddle River, NJ (USA): Prentice Hall, 1998.

[6] J. W. Pierre, R. F. Kubichek, and J. C. Hamann, "Reinforcing the understanding of signal processing concepts

using audio exercises," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, pp. 3577–3580, Mar. 1999.

[7] C. H. G. Wright and T. B. Welch, "Teaching DSP concepts using MATLAB and the TMS320C31 DSK," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, pp. 3573–3576, Mar. 1999.

[8] M. G. Morrow and T. B. Welch, "winDSK: A windows-based DSP demonstration and debugging program," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, pp. 3510–3513, June 2000. (invited).

[9] M. G. Morrow, T. B. Welch, C. H. G. Wright, and G. W. P. York, "Demonstration platform for real-time beamforming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2693–2696, May 2001.

[10] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching hardware-based DSP: Theory to practice," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 4148–4151, May 2002.

[11] T. B. Welch, R. W. Ives, M. G. Morrow, and C. H. G. Wright, "Using DSP hardware to teach modem design and analysis techniques," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. III, pp. 769–772, Apr. 2003.

[12] T. B. Welch, M. G. Morrow, and C. H. G. Wright, "Using DSP hardware to control your world," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. V, pp. 1041–1044, May 2004. Paper 1146.

[13] T. B. Welch, C. H. G. Wright, and M. G. Morrow, "Caller ID: An opportunity to teach DSP-based demodulation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. V, pp. 569–572, Mar. 2005. Paper 2887.

[14] T. B. Welch, C. H. G. Wright, and M. G. Morrow, "Teaching rate conversion using hardware-based DSP," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. III, pp. 717–720, Apr. 2007.

[15] C. H. G. Wright, M. G. Morrow, M. C. Allie, and T. B. Welch, "Enhancing engineering education and outreach using real-time DSP," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. III, Apr. 2008.

[16] T. B. Welch, C. H. G. Wright, and M. G. Morrow, "Software defined radio: inexpensive hardware and software tools," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2934–2937, Mar. 2010.

[17] M. G. Morrow, C. H. G. Wright, and T. B. Welch, "winDSK8: A user interface for the OMAP-L138 DSP board," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2884–2887, May 2011.

[18] M. G. Morrow, C. H. G. Wright, and T. B. Welch, "Real-time DSP for adaptive filters: A teaching opportunity," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2013.

[19] T. B. Welch, C. H. G. Wright, and M. G. Morrow, *Real-Time Digital Signal Processing: From MATLAB to C with C6x DSPs*. Boca Raton, FL (USA): CRC Press, 2nd ed., 2012.

[20] "RT-DSP website." `http://www.rt-dsp.com`.

[21] M. G. Morrow, C. H. G. Wright, and T. B. Welch, "An inexpensive approach for teaching adaptive filters using real-time DSP on a new hardware platform," *ASEE Comput. Educ. J.*, pp. 72–78, October–December 2013.

[22] Texas Instruments, "L138/C6748 Development Kit (LCDK)," 2013. `http://processors.wiki.ti.com/index.php/LCDK_User_Guide`.

[23] K. B. Benson and J. Whitaker, *Television Engineering Handbook*. New York: McGraw-Hill, revised ed., 1992.

[24] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Prentice-Hall, 2002.

[25] "Texas Instruments SYS/BIOS real-time kernel," 2013. `http://www.ti.com/tool/sysbios`.

[26] "OpenCV website," 2013. `http://opencv.org/`.

[27] Educational DSP (eDSP), L.L.C., "DSP resources for TI DSKs." `http://www.educationaldsp.com/`.