ANALOG DEVICES

AN-525
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • 781/329-4700 • World Wide Web Site: http://www.analog.com

ADV601 Video Codec Design Considerations

by David Starr

OVERVIEW

This Applications note is for hardware and software designers starting an ADV601 design. Using this note and the information in the ADV601 Video Codec data sheet you can do the following:

- Design ADV601-based video compression hardware.
- Write software drivers and hardware diagnostic programs.
- Integrate your hardware into the PCI bus and your software into Windows® 95.

The design examples in this application note refer to the ADV601-based Videolab demonstration board, but you can apply the techniques used in these examples to any ADV601-based design. The software source code and hardware schematics mentioned in this note are available on the Analog Devices computer products FTP site, whose Uniform Resource Locator (URL) is:

ftp://ftp.analog.com/pub/dsp/adv601/

VCLK (VIDEO CLOCK) FREQUENCY FOR SQUARE AND NONSQUARE PIXELS

The ADV601 uses the VCLK signal for internal processing, DRAM timing and strobing in video data. The ADV601's internal PLL multiplies VCLK up to generate the DRAM /CAS and /RAS timing. Use only the clock frequencies listed on the data sheet under "Clock Pins," even in nonreal-time applications. You must set the mode control register bits P/N (PAL/NTSC) and SPE (Square Pixel Enable) to match the selected VCLK frequency. For instance, if VCLK is 29.5 MHz, then set both P/N and SPE equal to one for the ADV601 to function properly. If you intend to switch square pixel enable on and off, you must also vary the clock frequency to match. Pulse-to-pulse jitter on VCLK should be less than 1 ns. The part is designed to function with VCLK phase locked to the horizontal sync. There is enough tolerance in the clock circuit to track the horizontal timing variations caused by tape speed variations (flutter and wow) on consumer grade VHS video cassette recorders (VCRs).

COMPRESSED VIDEO DATA INTERFACE DESIGN ISSUES

The compressed video data bus must support a high data rate. Raw video comes into the part at 12 to 14 Megapixels/sec. Video will come out of the part just as fast at low compression ratios. The compression ratio can vary from its programmed value, causing the video data rate to increase (or decrease, but the increase causes the difficulty). A slow compressed video bus will cause the ADV601's internal FIFO to underflow or overflow, resulting in lost frames on capture and torn frames on playback. Difficulty may occur if the compressed video bus is slower than 5 Megabytes/sec. The Analog Devices evaluation board uses a Bus Master PCI bus interface capable of 16 Megabytes/sec.

Many applications capture and play back video to/from hard disk. In this case the disk is the limiting factor in system throughput. However, if the disk and the ADV601 reside on the same bus (for example, a PCI bus system), bus bandwidth may also be a factor. If the video goes from the ADV601 card to main memory, and then from main memory to disk, bus traffic is double what it would be if the video went directly from the ADV601 to the disk

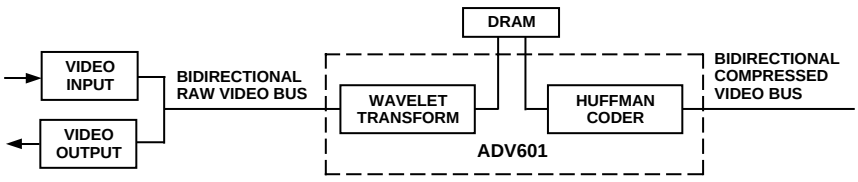

Figure 1. Video Signal Flow

controller with no halfway stop in main memory. Burst mode, where the hardware acquires the bus, asserts one address and transfers a block of data, will give best performance. The bus hardware may not be fast enough if it must acquire the bus and assert an address for each word transfer.

FIFO STATUS SIGNALS FIFO_SRQ, FIFO_STP, FIFO_ERR, FIFO_STP

FIFO_STP is a combined FULL and EMPTY pin. On encode it signals EMPTY, on decode it signals FULL. It means stop moving data into or out of the FIFO. FIFO_STP is asserted quite late and it can be difficult for hardware to see the FIFO_STP signal in time to halt the next FIFO transfer. In this case, an extra read will move invalid data, and an extra write will trash a word already inside the FIFO.

FIFO_SRQ

FIFO_SRQ is a combined NEARLY FULL and NEARLY EMPTY bit. On Encode it signals NEARLY FULL, and on DECODE is signals NEARLY EMPTY. NEARLY (the service request trigger point) is programmed by the FIFO Control Register over the range 32 to 480 long words. FIFO_SRQ is easier to use for data transfer control than FIFO_STP, because there is no penalty for moving one or two words after FIFO_SRQ goes away. FIFO_SRQ will go away at least 32 reads or writes before FULL or EMPTY occurs. The size of each data transfer can be controlled by programming NEARLY. Setting NEARLY to half full (256 words) will cause the hardware to move at least 256 words for each service request. This can be advantageous if there is significant overhead required to set up each bus transfer. Overhead might be arbitrating for the bus, entering host interrupt service or asserting the data address.

FIFO_SRQ can reoccur very rapidly. The host and the ADV601 are racing each other through the FIFO. It is possible for the host to transfer a single word that clears the FIFO service request and on the very next VCLK, the ADV601 can transfer a word that sets the FIFO service request again. FIFO_SRQ is asynchronous to the host port. Take care not to violate setup and hold time requirements of host port hardware.

FIFO_ERR

FIFO_ERR is a combined EMPTY and FULL pin. On decode it signals EMPTY and on encode it signals FULL. This is the reverse of FIFO_STP. When asserted, the host is falling behind.

BIN WIDTH CALCULATION BASICS

Off-chip computation, either by the host or a dedicated DSP, is required to control the compression ratio during encode. The Wavelet transformer output is 16 bits wide. To increase the compression ratio, some low order bits must be discarded before the run length and Huffman coders. This increases the length of the zero runs leading to more data compression. The ADV601's adaptive quantizer discards low order bits by multiplying every sample in the bin by a user-specified fraction, called the reciprocal bin width. On playback, the sub-bands are restored to proper size by multiplication by a user-specified coefficient called the bin width. Each of the 42 sub-bands has its own bin width and reciprocal Bin Width Register. The bin widths are embedded in the compressed data stream during the encoding process. On decode, the ADV601 extracts the bin widths from the compressed data stream and multiplies each sample by the bin width to bring it back up to proper size. Bin Width Registers are of concern on encode only; nothing need be put in the registers for decode.

Computation of a Bin Width Register is straightforward—merely take the reciprocal of the corresponding reciprocal Bin Width Register. Remember that the reciprocal Bin Width Registers are scaled 6.10 and the Bin Width Registers are scaled 8.8, and scale your reciprocal calculation accordingly.

The number of bits required to encode an image varies with the busyness of the image. A plain solid black field will encode very compactly since there is no high frequency energy in the picture. The higher sub-bands are all zero everywhere. On the other hand, something like a close-up of a plaid shirt has a lot of high frequency energy and will call for more bits to encode. As the picture gets busier, you need to use a smaller fraction in the reciprocal Bin Width Registers.

At the end of each field, the ADV601 supplies the bin width computer with the sum of the squares of each subband as a measure of the busyness. These (and a few other numbers) are referred to as "statistics." As the sum of the squares gets larger, the reciprocal bin widths need to get smaller.

This bin width computation works best if done quickly. The ADV601 will present the statistics just as vertical retrace is beginning. The bin width computer needs to read all the statistics, compute 42 reciprocal bin widths and 42 bin widths, and write the new setting back into the ADV601 before the next field starts. Next field starts in 20 horizontal line times or about 1.2 milliseconds. The computation needs to be repeated once per field, or every 16 milliseconds. The computation load will be about 1.2 milliseconds every 16 milliseconds or 7%. This assumes that the bin width calculation is actually completed within the 1.2 millisecond deadline. If not, the ADV601 will use the existing bin width setting on the new field. Since one field is much like another field, no great harm is done.

DIAGNOSTICS AND DEBUGGING STRATEGY

In testing out a new design it is important to get simple things working before testing more complex features.

For instance, the host interface has to successfully write the ADV601 mode control register before it is reasonable to expect video data transfer to work. Listed below, in order, are the diagnostic tests used to bring up the Analog Devices evaluation board.

16-Bit Data Interface (Bin Width Register) Test

First check the data interface into the part. If the data is not getting into and out of the part reliably, almost anything can go wrong. Data errors can cause the part's internal registers to program with unexpected values, causing unexpected operation. For instance, failure to set the encode/decode bit as intended will cause the part to drive an unexpected bus. Consider configuring the hardware for video capture. The video A/D will drive the raw video bus. If a data error causes the ADV601 to come up in decode mode by mistake, it too will drive the raw video data bus, causing both parts to get hot (or worse). The possibilities for unexpected behavior are broad, and defy analysis.

Test the data interface by writing data into the part and reading it back. Data read back shall match data written in. The 16-bit wide bin width and reciprocal Bin Width Registers are read/write, and hence testable, unlike other registers such as the mode control register. The bin width and reciprocal bin width registers, have addresses 100 to 153 (hex). The test should prove that each bit in each register can store both a one and a zero. It should prove that all 84 registers are unique by storing a different test value into each one and reading it back. The test should write all 84 registers and then read all of them back. It must not write one, read one and move to the next. It should take advantage of the auto incrementing feature of the Indirect Address Register to write and read all 84 registers in one swoop. Example code is in file wavetest.c (a part of 601test), and adv601.c (a part of 601cman). This code is on the Videolab CD-ROM and on the ADV601 FTP site.

As soon the Bin Width Register test runs once, put the program into a loop and test repeatedly. The new design should run overnight with no errors before the data interface can be considered free of timing errors.

A successful Bin Width Register test verifies the indirect address register, the indirect data register, the low 16 bits of the data bus , and the following pins, byte enables BE0 and BE1, addresses ADR0 and ADR1, CS, WR, RD and ACK. The ADV601 host interface handles all registers identically, so Bin Width Register test success gives confidence that writes to other read only registers function properly. Bin Width Register test failure suggests that other registers may not read or write properly, which throws doubt on the results of any other tests.

32-Bit Data Interface (FIFO) Test

A read/write test of the FIFO will verify all 32 bits of the data bus since the FIFO port is 32 bits wide. Use the same test data patterns as the Bin Width Register test uses, for the same reasons. The procedure for writing test data into the ADV601 FIFO is the same as for any video data. Reset the part, initialize it in decode mode, and write the FIFO data port until the FIFO is full. Write 512 long words and read them back. Read back requires the chip be placed into a special test mode. Do this by resetting the chip, and putting it into encode mode. Set the 100 bit in the FIFO control register. Example code is in 601 test module wavetest.c Then just read the FIFO port. Resetting the chip merely zeroes the FIFO read and write pointers. Data in the FIFO is not altered. Again, after the test runs once, by hand, loop it over night to detect intermittent errors. It took some time to resolve all the PCI bus interface problems on the Analog Devices Videolab design. As above, failure of the FIFO test suggests that the interface is dropping bits and corrupting the video. This casts doubt on the results of any other tests. The FIFO test will fail if the host cannot access the ADV601 control registers which the Bin Width Register test checks out.

Color Ramp Test

Color Ramp tests video decoding, which is easier than encoding. Color ramp is a simple ramp in both Luma and Chroma. This special case encodes VERY compactly, only 2440 bytes are required to encode a complete frame of video. This is small enough to fit into the memory of any microprocessor. Putting the test pattern into memory sidesteps all problems of disk speed, disk latency, seek time, thermal recalibration time, disk fragmentation, and bus contention. Your test program merely keeps loading the color ramp pattern into the ADV601 FIFO whenever the part asserts FIFO Service Request. It has a whole frame time (33 milliseconds) to load 2440 bytes, for a very modest data rate of 74 Kilobytes per second. The test passes when the color ramp pattern appears on the video monitor. The color ramp test should run forever (at least over night) to detect intermittent problems. Color ramp exercises the complete video decode path through the ADV601, plus the video decoder, if your design has one. If color ramp runs but captured video does not, the problem is either speed (host can't load FIFO fast enough) or encode. The color ramp data can be found in file c_ramp.c, a part of the 601test program available on the FTP site.

Bin Width and Reciprocal Bin Width Register Settings

To get the chip up and encoding initially, load the bin width and reciprocal Bin Width Registers with one "canned" set of values that will quantize each frame heavily (high compression). The high compression will reduce the data rate out the compressed data port (FIFO) and avoid FIFO overrun if the host is slow for some reason. Successful video capture means the captured video plays back properly. After successful capture, then it is time to enable the statistics ready interrupt, read the statistics, and compute a new set of bin widths for each frame. This interrupt service can be tricky to get right.

Errors can cause the part to output data furiously, causing a FIFO overrun. "Canned" bin width sets can be found on the FTP site.

AVOID BUS FIGHTS

Both the raw and compressed video buses (VDATA and DATA) are bidirectional buses, the ADV601 can either read them or write them. The design must insure that two chips do not drive the same bus at the same time, especially when coming out of reset or switching between encode and decode. The ADV601 comes out of reset reading the VDATA bus. If you rely upon software to initialize chips consider what will happen when the software fails. The ENC pin out of the ADV601 goes low when the part is driving the VDATA bus, and can be used as output enable for the other chips on the bus.

TEST POINTS AND LEDS

On a new design it will be necessary to observe each pin of the ADV601 with a logic analyzer. The first Analog Devices Videolab board had four 40-pin headers right on the board. If headers won't fit, Ironwood Electronics makes a series of "chip extenders" that give logic analyzer access. Placing LEDs on the following pins can greatly speed up hardware and software debug.

Pin Name	LED Indication
FIELD	Glows 1/2 bright when video is running.
FIFO_SRQ	Activity when video is running.
HIRQ	Activity when video is running. Full bright after software crash.
ENC	Verifies that software put the part into encode mode at the right time.
RESET	Signals unexpected reset.
FIFO_ERR	Should never happen.

HARDWARE VS. SOFTWARE RESET

The ADV601 has both a hardware reset pin, and a reset bit in the mode control register. Hardware reset must be asserted once at power up. Driver software will be easier to write if it too can assert hardware reset as a way of placing the chip into a known initial state. The hardware reset bit loads initial values into all the ADV601 registers that have defined initial values. Asserting the software reset bit in the mode control register makes the chip stop processing video and permits changes to the mode control register. It does NOT load initial values into registers. You must stop video processing with the software reset bit before changing video processing via the mode control register. You should assert software reset, then change the mode control register bits (and load all the other registers too) and then clear the software reset bit.

INTERRUPTS

The ADV601 asserts a single interrupt (HIRQ) to signal many different things. The host program reads the interrupt mask/status register to determine why the ADV601 required attention. There are many reasons (six to be exact). Reading the interrupt mask/status register clears and rearms all of the interrupts. In other words, reading the register acknowledges the interrupt to the ADV601 saying, in effect, "I've seen you, now go away and interrupt me again the next time it happens." This has two side effects on host software. First, reading the register clears the bits, so a second read of the register won't return the same data. Second, if the condition causing the interrupt reoccurs or another condition occurs, the chip will interrupt again. An interrupt routine might do the following things:

Insure that it won't be reentered if the interrupt stays "hot" or is reasserted during interrupt service.

Read the interrupt status register once, and service all the things that need service. For instance sooner or later both statistics ready and FIFO service request will occur at the same time.

When all servicing is done, read the interrupt status register one more time in case something came up during service time. If so, service the new condition.

Avoid becoming stuck in interrupt service if the interrupt stays "hot" no matter how much service the part is given.

CRUCIAL WIN 95 SETTINGS

You may be able to correct FIFO under runs or over runs (FIFO_ERR bit gets set) by turning off CD-ROM auto insert notification in Win 95. Go to "Settings" then "Control Panel". Click on "System" (a blue screen computer icon). Click on the "Device Manager" tab. This will display all the devices on the system. Click on the CD-ROM icon, which should then expand one level. Go down and click on "Properties." Pick the "Settings" tab. Clear the check mark in the "Auto Insert Notification" box.

It may also be necessary to remove the Win 95 "handicapped accessibility" option. While still on the "Control Panel" click on "Add/Remove Programs." Select the "Windows Setup" tab. Remove the "Accessibility Options" (icon of a wheelchair). These two changes cured FIFO overruns that occurred every ten seconds and every five minutes.

E3171–5–3/98

PRINTED IN U.S.A.