DVI2PCIe Align Tool

Download

The capture cards from Epiphan come with a development kit that allow to easily access the capture card with own programs. I had the idea to explore this SDK a bit and wrote a quick tool that does alignment automatic because I use the capture cards often with different graphics card as sources. In the last two weeks I had a few new ideas and they added up to some very interesting features.

Features

- automatic position alignment
- automatic finding of optimal dynamic range per component
- automatic finding of optimal sample phase
- suggests custom mode timings for aligned mode
- shows signal curve over two pixels per component
- shows analog noise levels per component
- shows timing stability of horizontal sync start
- estimates sample position on pixel
- estimates signal rise time as indicator for output quality

Requirements

The program requires a Java Runtime Environment as the SDK from Epiphan is built on Java. It also requires a Epiphan capture card. Usage is only senseful with analog inputs. The source has to show a test image.

Quick Start

The source has to show a test image. There are a few images for common resolutions in the ftp folder. Typically a image viewer as IrfanView is able to show the test image in fullscreen. You can move the mouse in case to the lower right corner so it will not interfere.
When you run the tool by starting run.bat it uses the first Epiphan capture card by default. However you can give the serial number of your card as parameter e.g. '**run.bat D2P85331**' to use a specific card.
First check that the PLL setting is correct as it is not automatically tuned. If it is not correct you will see in the area of the vertical black and white lines a horizontal moiree pattern. For sources following standard timings a PLL setting of 0 is typical. Press AUTO and wait until it finishes aligning.

Description

The test image consists of various patterns. The white pixel lines at the center of the left/right/top/bottom are used for the position alignment. The RGB colored rectangles are for the dynamic range adjustment. The horizontal gradient from black to white is used to generate a curve in the first graph. The alternating vertical black and white lines are a worst case scenario for the source output and are used to get information about the output quality.
The image can be moved with the cursor keys when the preview image window is focused.
The first graph shows as diagonal line or steps the black-white gradient separated for each component. If the output color depth is not True Color the curve will show steps. For example with a 15 bit HiColor mode each component of RGB has 5 bits. This can be seen as 32 steps from black to white. If the HiColor mode is 16 bit then the components are distributed by 565 bits. In this case the green component will have 64 steps. Also with TrueColor it is interesting as the curve shows how linear the components increase. After the dynamic range is optimized the curves for all three components should be over each other. If the Gain values for RGB show now different values the source output for each component is not identical. In case with the same gain for each component the white would change to a colored tint as one component gets more pronounced. As the curves update live one can adjust offset and gain manually and watch how the curves change.

There are also six dotted lines in the first graph. They show the intensity over the vertical black and white lines. Three curves for the white pixels and three for the black ones. As alternating black and white pixels are the worst case the levels usually do not reach the same intensity for white and black as the gradient curve shows. This is normal. The intensity of these curves is sensitive to the sample phase value. With tuning the sample phase it is possible to move the position on the pixels where the capture card samples the colors. With a large shift it can even be possible to move the position of sampling right between the pixels. In this case the pattern of the vertical lines would get grey and also noisy. The noise is related to the fact that the sample is taken in the edge of the signal and a slight jitter in time results in a large brightness change. As the curves update live you can try to change the sample phase and watch the curves change. Typically when moved away from the optimal value the curves for the white pixels get darker and the the curves for the black pixels brighter.

On the right side of the adjustment panel the mode timings are calculated according to the CVT standard. Below adjusted values for a custom mode are suggested that can be added to the epiphan tool. To make these values useful one has to start with the capture card syncing in a mode with the same CVT timings. This can be done the following way:
Add a custom mode with the resolution and vertical refresh rate as shown in the panel and check 'Standard VESA Timings'. Now uncheck all default and all custom modes (apply) so the capture card will loose sync. Check again the just added mode and let the card resync.
Now the suggested values for a custom mode from the align tool can be used to add a custom mode in the epiphan tool that is properly aligned with HShift/VShift = 0. This is most useful if the source switches modes. In this case one can add custom modes for each source mode. If only those are allowed then the driver will sync in all those modes with a properly aligned image.
On the left side below the checkboxes are the values for the black levels of each component shown. It shows the minimum, maximum values from a certain area and the standard deviation of the noise (all per component). It can be used to set the black level by changing the offset values. Usually for a video capture the black level is moved below the noise on black so that black areas from the source have zero intensity. The offset will change the voltage threshold for black.
Below the black level values the standard deviation for the noise on the 50% grey area is shown. With these values one can get an impression about the noise per component reaching the capture card. So it is possible to identify bad cables, bad VGA cards or other noise sources.

Getting a pixel scan curve:

So how can one get a pixel scan curve if the capture card is actually sampling at the pixel clock of video source? The trick is like this: I use the periodic pattern from the vertical black-white lines. If I change the PLL now the capture card will sample at a slightly different frequency than the pixel clock of the source. What happens is that the sample position on a pixel shifts slightly to the next one and so on. Let's look at an example. If I have a 800x600 resolution image the capture card samples with an optimal PLL of 0 exactly 800 times over 800 pixels. I change now the PLL to -1 and it samples 800 times over 799 pixels. For the black-white pattern this means that starting with a black pixel at sample one I end up with a white pixel at pixel position 799 and sample 800. The sample position shifts now each pixel by 799/800, a very tiny bit. Ok so far so good. If I set the PLL to -2 it will sample 800 times on 798 pixels and looking at all even samples at the former black pixels it will now change from black to white in the middle and back to black at the end. For the white pixels the shift will make the samples go from a white source pixel at the left start to a black in the middle and back to a white one at the right edge. As my pattern of black and white lines is periodic I can treat the first two b/w pixels the same as the next two b/w pixels and so on. This means I can treat the tiny position shift the same as it is happening on the same two pixels. So with a PLL of -2 it needs 400 shifts to reach the next pixel and another 400 to be back on the same pixel. I get 400 subsamples per pixel.
It shows that this is exactly what happens. You can watch the dotted lines that scan over the even sample positions and the odd sample positions change when you change the PLL. If you move the PLL out of the optimal value there will already be one crossover. By 2 you will see two intersections. I can not read back the values how much exactly a PLL change of -1 or -2 changes the sample clock. But I know from the dotted curves that from one intersection to the other intersection the sample position has shifted exactly one pixel. The upper curve in between corresponds to a shift over the white pixel and the lower curve to the shift over the black pixel. This is sufficient to reconstruct the signal.
The first two intersection positions are shown by vertical lines in the first graph. The second graph shows raw signal curves for each scanline and also an averaged curve. As the reconstructed curves might have different number of points due to jitter they are natural cubic spline interpolated to the same number of points and then averaged. The intersection point of the red component is used as time zero for the other components. The intersection points are roughly at 50% grey level. The scan over the b/w pixels is shown in a coordinate system where the time axis is estimated from the pixel clock calculated in the mode timings.
The third graph shows the jitter of the horizontal sync position per scanline. As the sync signal gives the start trigger for the sampling jitter will lead to movement of the intersection points. Here the position of the first intersection point is translated to a sync start time. The left edge in the graph corresponds to the current earliest hsync start. The full X-range is the same like in the graph above, exactly two pixels. Typically old VGA cards show a higher jitter than newer ones due to better defined signals. I might add some statistical number output here later. Sometimes the position offset shows patterns from some lower frequency signal offset.
I added a routine that gets the time that it takes the signal to go from 15% to 85% intensity. These levels are shown as horizontal yellow lines. Where the averaged pixel scan curve reaches these levels a vertical white line is shown. For the time also the frequency is given. It is typically a bit higher than the DAC frequency of the VGA card but in the same range. However the time for a rise from 15 % to 85 % is much faster than e.g. from 5 % to 95 %. The frequency is good indicator of the actual output quality of the source. Be aware that also the capture card itself imposes a limit here as it does not have infinite bandwidth.
The actual sample position on a pixel can be estimated from the intensity level the capture card sample on the white and on the black pixels. Where the pixel scan curve reaches this level with a distance of exactly one pixel between black and white is the actual sample position. To get the regular intensity levels set the PLL back to optimum (typically 0) and press the 'getLevels' button. Then change the PLL back to -2 and the estimated position gets shown. It differs slightly per component and the white vertical line gives the averaged position of all three components.

Own test images in other resolutions:

The used areas in the test images are based on percentage values of the resolution. Here are the numbers:

- vertical single pixel white lines at left/right for horizontal alignment, expects pixel at 50% height. Be aware that the image at start is shifted, thus lines
- horizontal single pixel white lines at top/bottom for vertical alignment, expects pixel at 50% width. Be aware that the image at start is shifted, thus lines
- Black: between 36% and 58% height, full range horizontal, minus 5 pixel on each side border
- White: between 11% and 34% height, 58.1% and 75% width
- alternating vertical black/white lines: between 61% and 80% height, full range horizontal
- black to white gradient: between 81% and 89% height, full range horizontal
- black area for noise: between 36% and 59% height, 5% and 95% width
- 50% grey area (128,128,128): between 12% and 34% height, 76% and 95% width

Screenshots (please view 1:1)

Signal from Matrox Millenium
Signal from CardExpert ET4000/W32i
Signal from S3 Trio3D/2x at 1280x1024 60Hz

I have not tested the tool in Linux, but might work as well.