Implementing a Histogram for Image Processing Applications

Use the Virtex™ or Spartan™-II True Dual-Port™ RAM and DLLs to create a real-time histogram.

by Edgard Garcia
Engineer, Multi Video Designs
edgard.garcia@mvd-fpga.com

Image processing is key for many automated industrial inspection applications. However, even the most sophisticated algorithms can't extract the right information if the image contents are not available in a convenient format. By using a histogram, you can ensure that the image content can be easily processed.

What is a Histogram?

For each possible pixel value, the histogram algorithm counts the number of times the value was encountered in the current image. For example, the histogram of an 8-bit-per-pixel image will contain 256 values (2^8), each one representing the number of pixels found at this value. This allows a microprocessor or DSP to quickly get the profile of the image, and take the appropriate decisions, by analyzing just those 256 precomputed values. You can do this easily, in real time and at low cost, in a Virtex or Spartan-II FPGA.

A Basic Hardware Implementation

For an 8-bit-per-pixel image, 256 different values are possible for each pixel, so 256 16-bit counters would be necessary to complete the real time histogram. However, only one of the 256 counters will be active at each valid pixel clock (only one value will be updated). Therefore, the registers of the 256 x 16 bit counters can be replaced by a memory array, such as a 4K-bit block SelectRAM™ organized as 256 x 16 (RAMB4_S16).

A 16-bit incrementer will allow you to update the RAM contents during a Read-Modify-Write operation, where the video data inputs are used as the address of the memory block. Figure 1 shows the block diagram of a basic hardware implementation.

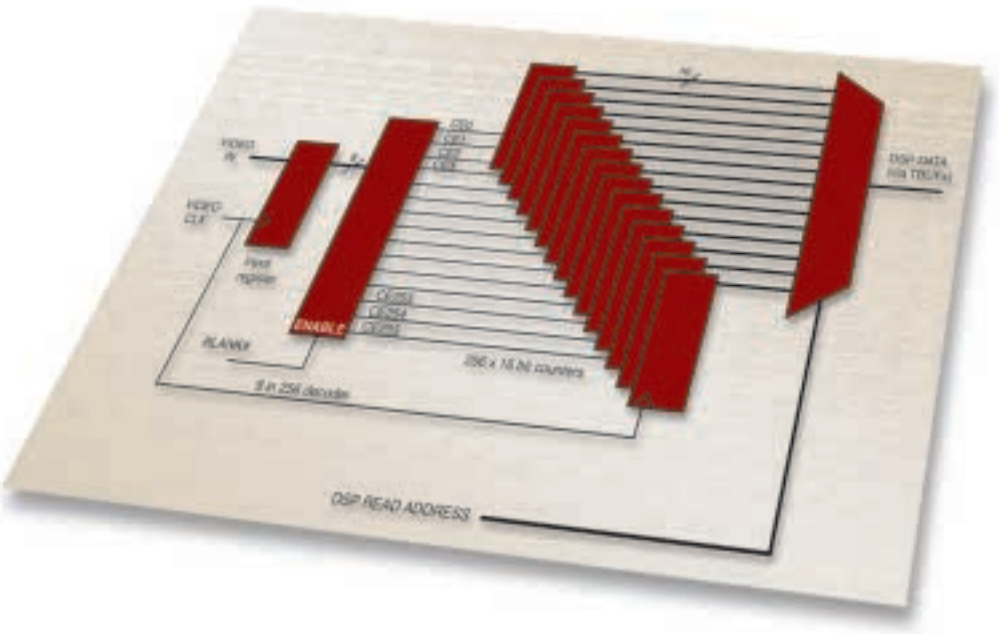

Figure 1 - Basic hardware implementation

Optimized Implementation

Each memory cycle can be either a Read or a Write, so we need to divide each pixel clock cycle in two sub-cycles: a Read cycle for getting the current value, and a Write cycle for updating (+1) the memory content. You can do this easily, using a CLKDLL to recover a clock at twice the frequency of the video clock (CLK2X), and to create an image of this clock shifted by 90° to validate Read and Write cycles. Figure 2 shows the detailed diagram of an optimized implementation.

During horizontal and vertical retrace, pixel values must be discarded. This is done with no additional logic, by connecting the BLANKING# signal to the ENA input of the memory block. Figure 3 illustrates the timing of the operations.

The DSP or microprocessor can directly read the result of the histogram by using the B port of the same block SelectRAM (configured as a RAMB4_S16_S16). A multiplexer is not needed because the two ports (A and B) each have dedicated inputs.

Resources and Performance

Here are the logic resources required for implementing the histogram algorithm:

- 1 x CLKDLL + BUFG
- 1 x RAMB4_S16_S16
- 1 x 16-bit INCREMENTER (8 slices)

For a Virtex -6 or Spartan-II -6 device, Fpix = 50 MHz.

Conclusion

By taking advantage of the high level features of the Virtex and Spartan-II FPGA architectures, you can greatly increase the speed and reduce the cost of your designs. For more information about how to implement the histogram algorithm, e-mail: edgard.garcia@mvd-fpga.com

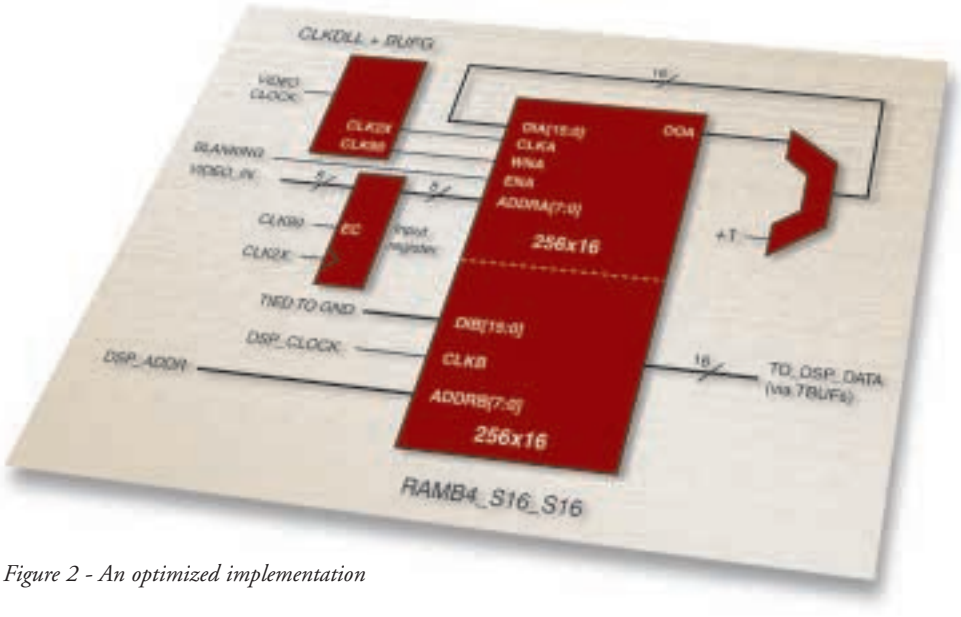

Figure 2 - An optimized implementation

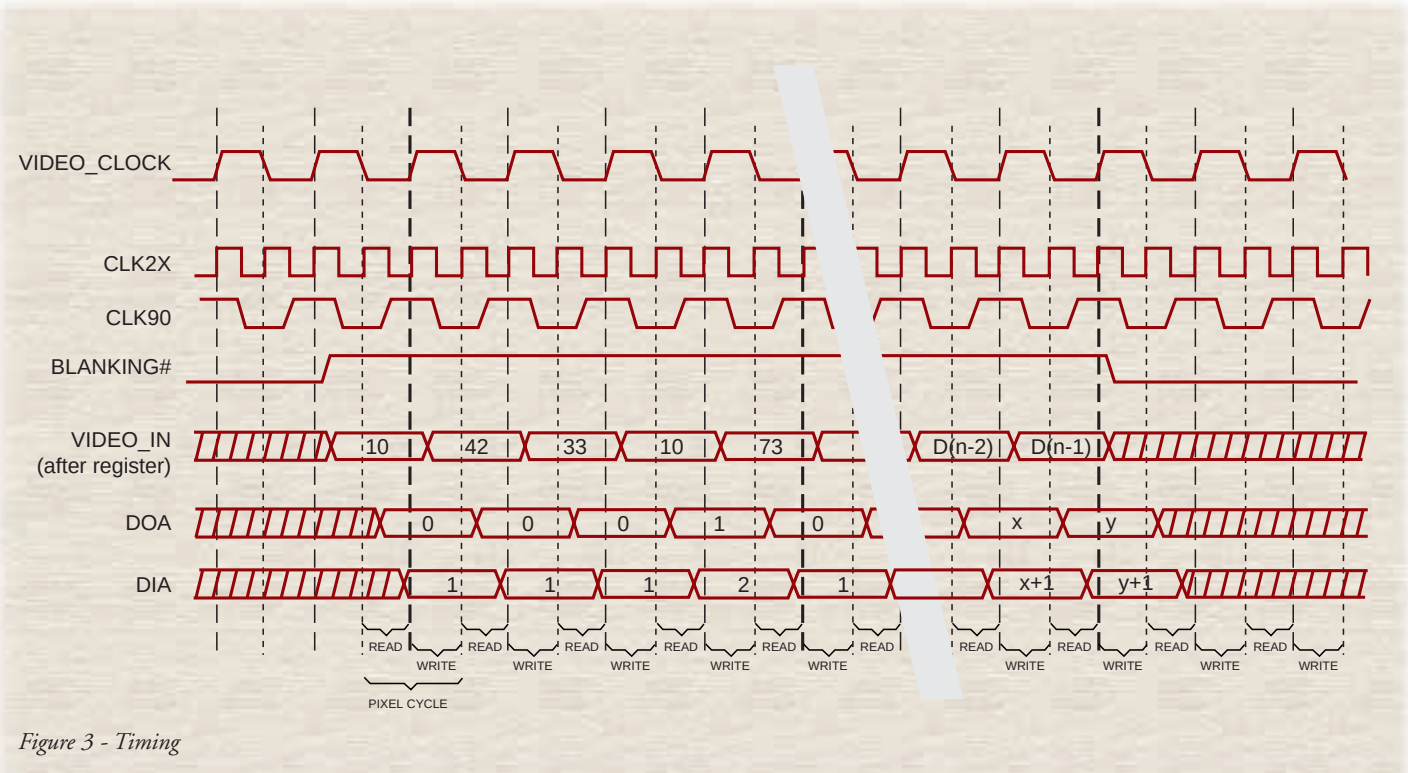

Figure 3 - Timing