XAPP241 (v1.1) October 3, 2000

Virtex-EM FIR Filter for Video Applications

Author: Ralf Kreuger

Summary

Virtex™-E Extended Memory (Virtex-EM) FPGA devices offer over a million bits of block RAM and up to 300 Kb of distributed RAM in a single high-performance device. This is ideal for high-bandwidth video applications where complex digital filtering logic can operate on several lines of pixel data on-chip. The reconfigurable nature of Virtex-EM devices offers designers a flexible platform for optimizing Digital Signal Processor (DSP) parameters and algorithms throughout the design and preproduction cycle, as well as when the devices are in the field. This reprogrammability allows periodic optimization of proprietary algorithms in such applications as MPEG compression.

High-end DSP applications running on Virtex-EM FPGA devices can also employ algorithms that share filter implementations. The ability to share logic resources combined with increased overall chip performance means that more logic can be integrated on the same chip. Extended on-chip memory enables designers to tightly integrate components that traditionally have been external to the FPGA.

Introduction

A commonly used filter in video applications such as image enhancement is a Finite Impulse Response (FIR) filter (see Figure 1). Every value applied to the input begins a finite-length output sequence, called the impulse response. When impulses are received consecutively, the outputs are multiplied and added linearly. This process is called convolution. Most two-dimensional (2D) filters are based on a square convolution mask (or kernel), which is simultaneously superimposed on a group of pixels. Typical kernels range from 3 x 3 pixels to 15 x 15 pixels. In high-end applications such as medical imaging, larger kernels (up to 63 x 63 pixels) provide improved results.

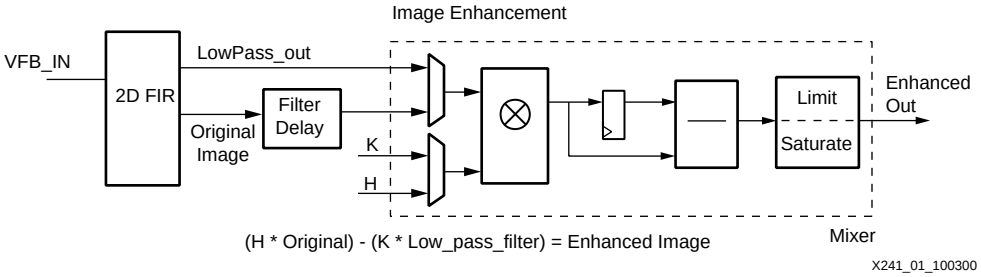

Figure 1: **Image Enhancement**

One-dimensional DSP operations are widely used in imaging or video systems to perform down-conversion, resampling, or filtering. In video, one-dimensional operations are typically done in the horizontal dimension, because video is sampled horizontally in real time. Two-dimensional operations are used primarily to filter in two dimensions, either for various graphic effects or to alter the size and shape of the image.

Such filter operations include high-pass filters, which sharpen edges in all directions, and low-pass filters, which limit high-frequency noise or deliberately soften edges. An important example is graphic and image enhancement where the input image is filtered and then fed to a mixer for overall image enhancement.

To ensure video quality in a high-performance system, line buffers are typically integrated with a filter on a single chip. In such cases, memory requirements can be significant. In high-end

applications such as medical imaging, large kernel masks of up to 63 x 63 pixels are often used. Assuming a 1024 x 1024 x 12-bit video frame, up to 756 Kb of RAM might be required. Traditionally, only very large FPGAs with extensive logic capability had sufficient memory for such applications, and this inefficient logic-to-RAM ratio increased the cost unnecessarily. With the introduction of Virtex-EM devices, the favorable ratio of block RAM to logic makes implementing such applications more feasible.

Imaging/Video Systems with Separable Dimensions

Most DSP applications involving filter operations on two-dimensional data rely on the assumption that the vertical and horizontal filtering operations can be separated. The assumption is that the image data can be filtered first in one dimension (the vertical or column dimension) and then filtered in the other dimension (the horizontal or row dimension). (See Figure 2.) This separation, where acceptable, greatly simplifies filter design.

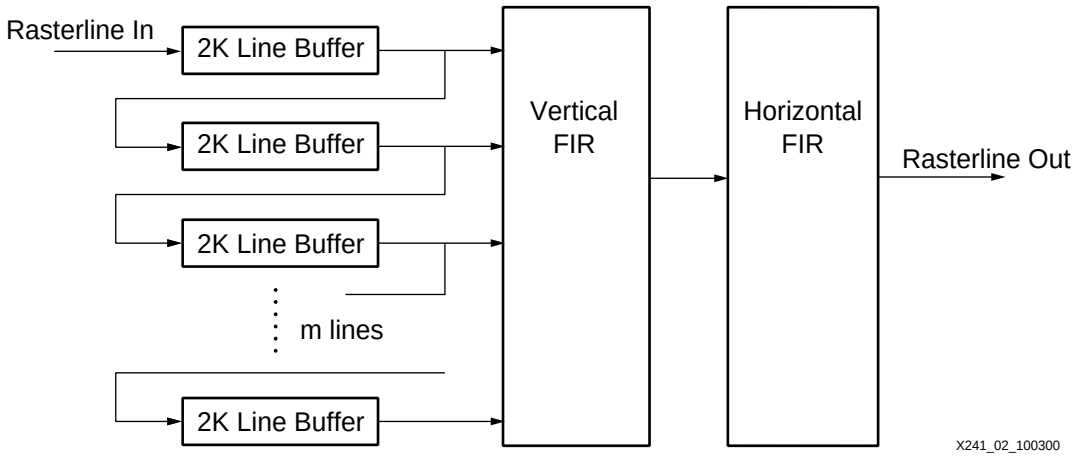

Figure 2: **2D FIR Filter With Built-in Line Buffers**

In terms of algorithms, no basic differences exist between horizontal and vertical FIR filters that are symmetrical in both directions. In fact, with high-performance Virtex-EM devices, it is possible to share the same filter implementation for column and row processing.

With an all-zero structure, a FIR filter has a linear phase response when the filter coefficients are symmetric, as is the case in most standard video filtering applications. When all coefficients are symmetrical around a center point (see Figure 3), this symmetry allows tabs to be pre-added before they are multiplied by the coefficients.

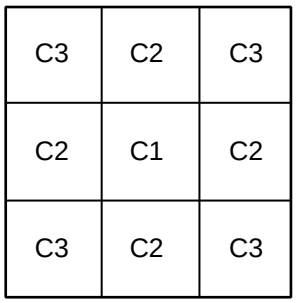

Figure 3: **Symmetrical Kernel**

Basic Operation

In image enhancement, a FIR filter is often used to achieve the following effects:

- Eliminating high/medium/low frequencies, as desired
- Smoothing images to suppress sharp edges or gradients, as well as random “noise” (Low-pass filtering)
- Enhancing fine detail and edges (High-pass filtering)

A square kernel mask is moved across an image pixel-by-pixel and line-by-line. The filter examines the center pixel to be enhanced and modifies it according to the surrounding pixels (neighborhood). Each pixel in the array is assigned a weight or coefficient. Each pixel is multiplied by an appropriate coefficient (weight) and the products of all pixels are summed. The derived value might be forwarded to other components of the overall system or applied to the center cell of the array. The filter then moves one pixel to the right, and the operation is repeated until the entire image is processed (see Figure 4).

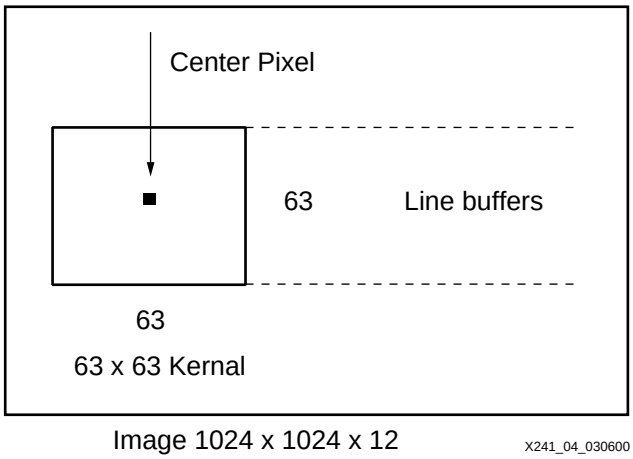

Figure 4: **Image Processing**

Algorithm and Implementation

For the purpose of this discussion, a 1024 x 1024 x 12-bit pixel image at a refresh rate of 30 frames per second is assumed. Clock rates of 80/40 MHz are needed to achieve this goal. The I/O performance, logic block performance, and routing speed of Virtex-EM devices easily accommodates such clock rates. In fact, Virtex-EM architecture accommodates other common image frequencies, including 54 MHz and 74 MHz clock rates.

As mentioned earlier, the 2D FIR algorithm described here (shown in Figure 5) is designed for linear phase, and thus a separable symmetric algorithm is implied. This allows for pre-adds in both the vertical and horizontal filtering sections, reducing the number of multiplies per clock. Additionally, this particular algorithm has two identical 1D FIR filters designed in a cascade structure, with vertical filtering occurring first, followed by horizontal filtering.

One solution to this problem is to simply employ two separate filters. However, a more efficient design is possible using an architecture that shares logic resources between horizontal and vertical data. Since the coefficients are the same for vertical and horizontal dimensions, the same constant coefficient multiplier can be shared for filtering. This sharing of constant multipliers also leads to a sharing of the multiplier/adder trees at a 2x clock rate, while the pixel data moves at 1x clock rate.

Using two Virtex-EM Delay-Locked Loops (DLLs), a 2x clock is easily synthesized from the system clock, and because both clocks are phase locked, setup and hold requirements are minimized. A tapped delay line holds and aligns the filtered vertical data. Thus, the filter works on either the vertical data (first stage) or the result of the vertical filter (second stage), alternately.

A typical problem with image filtering in FPGAs is the need for external line buffering and associated control logic. Virtex-EM devices eliminate the need for external buffering, even for the largest kernel masks, because extensive on-chip block RAM allows storage of sufficient

amounts of image lines close to where the data is processed. This significantly reduces the number of I/Os required and removes a performance bottleneck.

The on-chip line buffers in Figure 4 are 12 bits wide by 2 K deep, and each contains two image lines, due to the multiplexed nature of the architecture.

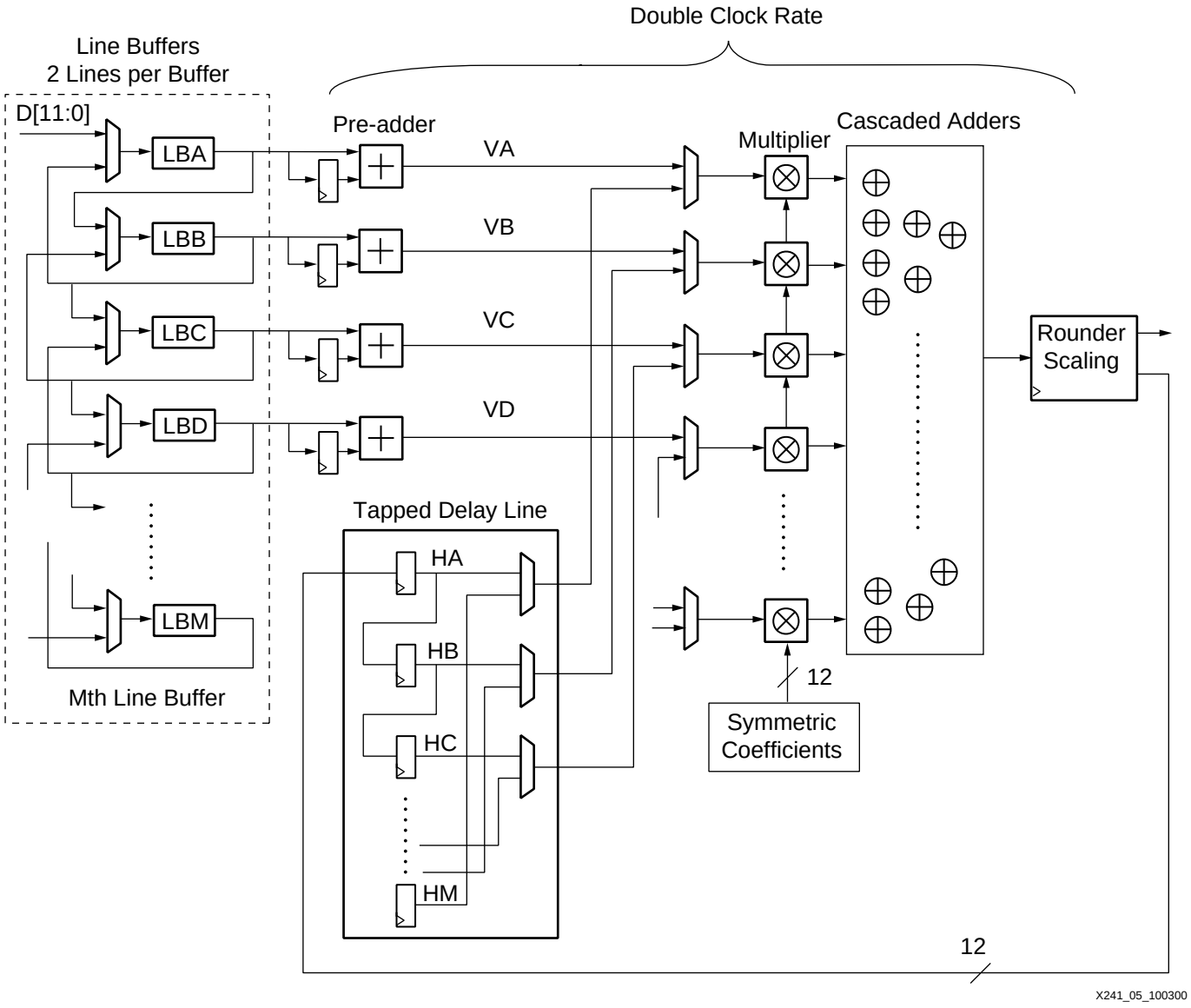

Figure 5: **2D FIR Algorithm**

For the physical implementation, designers can choose from an IP core library that contains the basic building blocks for the add and multiply functions. A parameterized adder/subtractor with sizeable bit width is available. For the multiplier, either a generic or special purpose multiplier can be used.

If no reloading of coefficients is required, a Constant Coefficient Multiplier (KCM) can be selected. Otherwise, a dynamic KCM is available. These modules create multipliers that multiply an n-bit wide variable by an n-bit fixed coefficient and produce a 2 n-bit result. The coefficient multiplication tables are stored in distributed ROM-based lookup tables (LUTs), taking advantage of the FPGA look-up table architecture. This is an efficient, high-speed, parallel implementation.

Virtex-EM series devices provide an unparalleled amount of block RAM for on-chip line buffers, eliminating the need for external buffers and moving the data closer to where it is processed. This significantly lowers the amount of I/O pins required. While external buffering would require

one set of pins for each image line, now only one set of pins is required, regardless of kernel size. Cost and additional board real estate associated with discrete line buffers are eliminated.

The Virtex-EM family delivers 560 Kb (140 blocks) of RAM on the XVC405E product and 1120 Kb (280 blocks) of RAM on the XCV812E product. Each block contains 4096 bits and can be configured with a width of 1, 2, 4, 8, or 16 bits. This flexibility can be used to adjust to various line buffer organizations, such as n-bit wide 1024 and 1920 (HDTV) lines. In the above example, two image lines of 1024 bits each are stored in a 2K buffer. Selecting a 2-bit wide block RAM yields the desired depth of 2048, with six blocks of RAM needed for a 12-bit per pixel image. In this configuration, a 63 x 63 kernel requires 192 blocks of RAM.

Depending on the application, the amount of RAM per line and number of lines required to be loaded can vary dramatically. For example, in the high-end HDTV arena, a single line of pixels can consume 37.5 Kb of RAM (for example, 1920 pixels/line, 10-bit luma, and 10-bit chroma). Up to 29 lines can be simultaneously loaded and processed in a XCV812E device, and up to 14 lines in a XCV405E device. Table 1 illustrates the wide range of RAM requirements for various pixel bit sizes and line lengths. The two columns to the right list how many lines of 12-bit pixels can be stored in the block RAM for each device.

Table 1: **RAM Requirements**

Pixels per Line	Bits per Pixel			Number of Lines	
	8	10	12	XCV405E	XCV812E
512	4096	5120	6144	93	186
768	6144	7680	9216	62	124
1024	8192	10240	12288	46	93
1280	10240	12800	15360	37	74
1920	15360	19200	23040	24	49
2048	16484	20480	24576	23	46

Virtex devices provide a macro called SRL (Shift Register LUT) that implements very efficient shift registers varying in length from 1 to 16 bits. A logic block (CLB) can contain four 16-bit shift registers with an optional flip-flop (17th stage) at the end of each register. Unused CLBs offer additional resources available for delay lines or further line storage. The XCV405E has 2400 CLBs, and the XCV812E has 4704 CLBs. Assuming an additional line buffer or delay of a 1024 x 12 size is required, only 181 CLBs (3.85% of the CLBs on a XCV812E) are used.

Conclusion

With high-performance arithmetic functions and large memory blocks, Virtex-EM devices are ideally suited for high-end image processing applications such as filtering and compression. For example, 2D Wavelet and 2D Discrete Cosine Transformation (DCT) in MPEG and JPEG applications can take advantage of increased memory in between computational engines. Other areas that can benefit from large on-chip memory include image histogramming.

A further advantage is that components traditionally external to the FPGA can now be tightly integrated on the same chip. Increased overall chip performance allows designers to share logic resources, providing room for even more integrated functions without other FPGAs or additional discrete components.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
3/14/00	1.0	Initial Xilinx release.
10/03/00	1.1	Revised video frame size to 756 Kb on page 2.