

Multi-Channel FIR Filters

This chapter illustrates the use of the advanced Virtex™-4 DSP features when implementing a widely used DSP function known as multi-channel FIR filtering. Multi-channel filters are used to filter multiple input sample streams in a variety of applications, including communications and multimedia.

The main advantage of using a multi-channel filter is leveraging very fast math elements across multiple input streams (i.e., channels) with much lower sample rates. This technique increases silicon efficiency by a factor almost equal to the number of channels.

The Virtex-4 DSP48 slice is one of the new and highly innovative diffused elements that form the basis of the Application Specific Modular BLock or ASMBL architecture. This modular architecture enables Xilinx to rapidly and cost-effectively build FPGA platforms by combining different elements, such as logic, memory, processors, I/O, and of course, DSP functionality targeting specific applications such as wireless or video DSP.

The Virtex-4 DSP48 slice contains the basic elements of classic FIR filters: a multiplier followed by an adder, delay or pipeline registers, plus the ability to cascade an input stream (B bus) and an output stream (P bus) without exiting to a general slice fabric.

The resulting DSP designs can have optional pipelining that permits aggregate multi-channel sample rates of up to 500 million samples per second, while minimizing power consumption and external slice logic. In the implementation described in this chapter, multi-channel filtering can be looked at as time-multiplexed, single-channel filters.

In a typical multi-channel filtering scenario, multiple input channels are filtered using a separate digital filter for each channel. Due to the high performance of the DSP48 block within the Virtex-4 device, a single digital filter can be used to filter all eight input channels by clocking the single filter with an 8x clock. This implementation uses 1/8th of the total FPGA resource as compared to implementing each channel separately.

Multi-Channel FIR Implementation Overview

Top Level

The implementation of a six-channel, eight-tap FIR filter using DSP48 elements is depicted in Figure 7-1. The design elements used in the implementation include the following:

- Six-to-one multiplexer that is implemented in slice logic as described in “Combining Separate Input Streams into an Interleaved Stream,” page 101
- Coefficient ROMs using SRL16Es connected in “head-to-tail” fashion
- Input sample “delay-by-seven” SRL16Es to hold the interleaved streams
- DSP48 slices for multiplication and additions

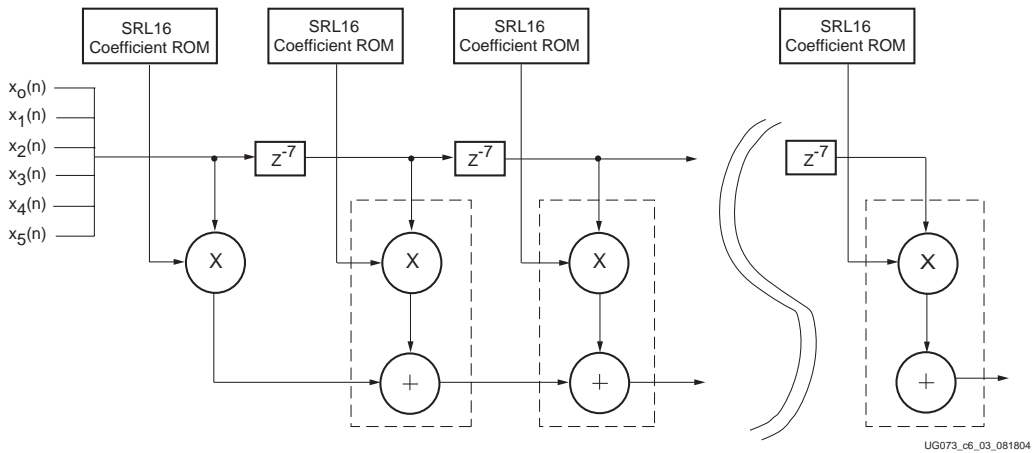


Figure 7-1: Block Diagram of a 6-Channel, 8-Tap FIR Filter

All datapaths and coefficient paths for this example are 8 bits wide. The coefficient ROMs and input sample delay elements are designed using SRL16Es. The SRL16E is a very compact and efficient memory element, running at the very high 6x clock rate. For adaptive filtering, where coefficients can be different depending upon their input signals, coefficient RAMs can be used to update the coefficient values.

The DSP48 slices and interconnects also run at the 6x clock rate, providing unparalleled performance for multiplication and additions in today’s FPGAs.

DSP48 Tile

The multi-channel filter block is a cascade implementation of the DSP48 tile. Each tile is implemented as shown in Figure 7-2. An SRL16E is used to shift the input from the six channels. The

product cascade path between two DSP48 slices within the tile can be used to bring the product output from one tap into the cascading input of the next tap for the final addition.

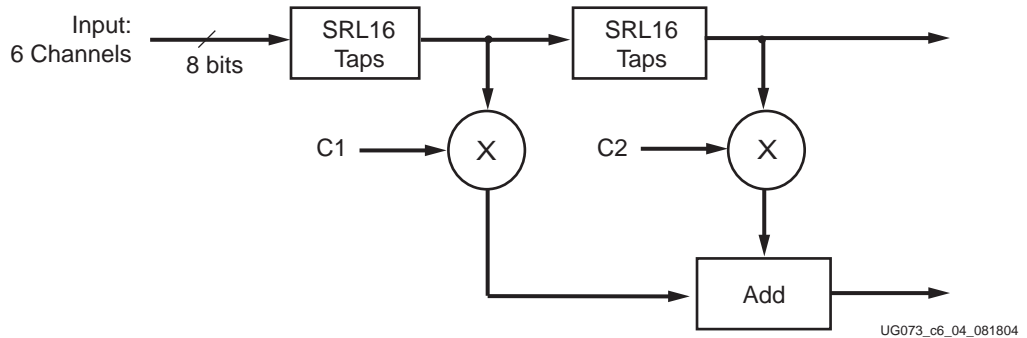


Figure 7-2: DSP48 Tile Cascading Diagram

Combining Separate Input Streams into an Interleaved Stream

As shown in Figure 7-3, six separate video input sample streams must be combined into one interleaved sample stream for this multi-channel FIR filter example. Conceptually, a high-speed, six-to-one multiplexer feeds a seven deep SRL16E shift register to accomplish this task. The SRL16E depth is the number of channels plus one.

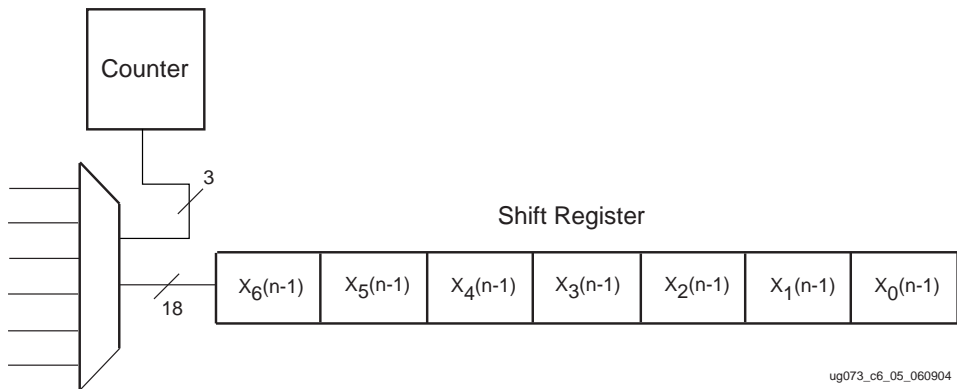


Figure 7-3: Converting Eight Input Streams to One Interleaved Input Stream

For each clock tick, the counter selects a different input stream (in order), and then supplies this value to the SRL16E shift register. After six clock ticks, the six input samples for a given time period are loaded sequentially, or interleaved into a single stream.

A six-to-one multiplexer must be designed carefully, as it is constructed with slice logic that must run at the 6x clock rate. At 446 MHz, good design practices dictate connections “point-to-point,” a maximum of one Look-Up Table (LUT) between flip-flops and RLOC techniques.

To reduce the high fanouts on the selected lines of the multiplexer, the conceptual multiplexer in Figure 7-3 is implemented as shown in Figure 7-4. This circuit is repeated for all eight bits of the input sample width.

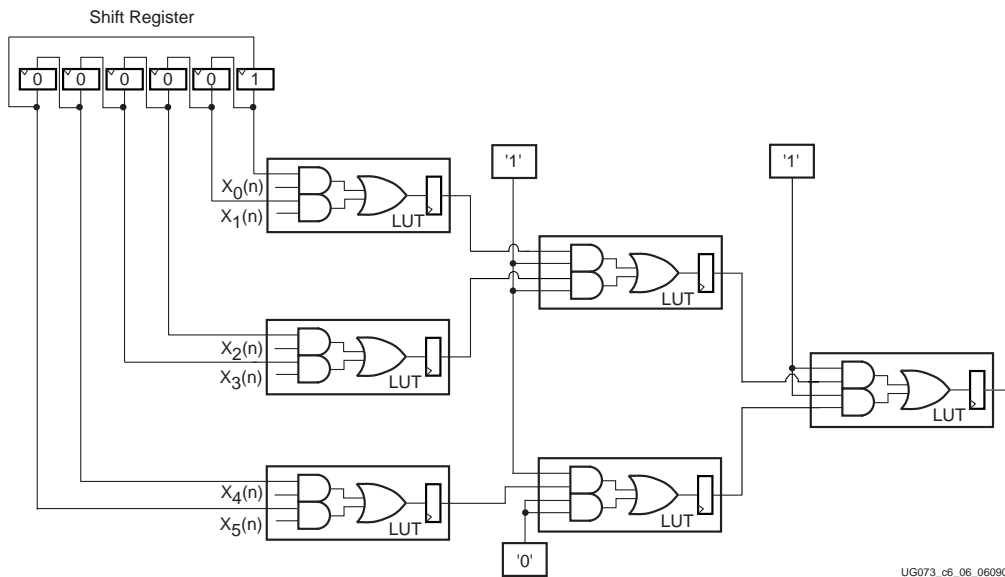


Figure 7-4: High-Speed 8-to-1 Multiplexer Used in the Filter

Coefficient RAM

The six coefficient sets are stored in the SRL16 memories. If the same coefficient set is used for all channels, then only a single set is stored in the SRL16. If the different channels use different coefficients, then six sets of SRL16s are used for each tap. (Six RAMs can be used instead, one for each channel.)

Each RAM is 8 bits wide and six deep, corresponding to the six taps. The optional Load input is used to change or load a new coefficient set. Six clock cycles are needed to load all six RAMs. Input C1 is used to load the eight locations of RAM1 which are used for Channel1. C8 is used to load the eight locations of RAM8 which are used for Channel8. At the eighth clock, all eight locations of the eight RAMs are loaded; the filter then becomes an adaptive filter. The speed of the overall filter will be reduced when the coefficients are stored in the RAM.

Control Logic

The control logic is used to ensure proper functioning of the different blocks. If the coefficient RAM block is used, the control logic ensures that the load signal is High for six clocks. Different tap-enabled signals are used to make sure that RAM values are read into the DSP48 correctly. For instance, clock1 reads in the first location from RAM1, but the first location of RAM2 is read only at the clock number equal to shift register length. The design assumes a clock is running at 6x that of the input

signals. The DCM can also be used to multiply the clock if the only available clock is running at the input channel frequency.

The control logic also takes care of the initial latency such that the final output is enabled only after the initial latency period is complete.

Implementation Results

The initial latency of the design is equal to the $\{(\text{number of channels} + 1) * \text{number of taps}\}$ plus three pipe stages within the DSP48. After placement and routing, the design uses 216 slices and eight DSP48 blocks. The design has a speed of 454 MHz.

Conclusion

The available arithmetic functions within the DSP48 block, combined with fine granularity and high speed, makes the Virtex-4 FPGA an ideal device to implement high-speed, multi-channel filter functions. The design shows the efficient implementation of a six-channel, eight-tap filter. Due to the high-performance capability within the DSP48 block, a single channel, eight-tap filter can be used to implement the six-channel, eight-tap filter, reducing the area utilization by 1/6th.

