Projeto de Sistemas Digitais

Trabalho Laboratorial 3

DSP IP core for computation of real-time FIR filter

Rui Graça
Vinícius Ginja

Universidade do Porto
Faculdade de Engenharia
FEUP

1 Introduction

For this project, we set out to optimize a given digital audio IP able to compute digital FIR filters with up to 16384 coefficients, so that it meets timing requirements established by a 48 KHz input signal bandwidth. Both input and output are stereo with 18 bit per channel. To achieve this speed requirement, parallel and pipeline architectures were explored during the development. We also found different solutions that are fast enough but with unequal FPGA resource utilization. The results were verified with behavioural and post-synthesis, as well as FPGA implementation using a signal generator and an oscilloscope. Post-layout simulation failed, probably due to some error in the model of a RAM block, but tests done on the FPGA show that the behavior is correct. The input of IP, as well as the filter coefficients, are represented in two's complement with fixed point with only one non-fractional bit.

2 Design decisions

Since we must be able to compute the response of a filter with up to 16384 coefficients in $\frac{1}{48}$ ns = 20.8333 μs, if we set the clock frequency to 100 MHz (clock period of 0.01 μs), we must have $\frac{16384 \cdot 0.01}{20.8333} = 7.8$ MAC (Multiply and accumulate) blocks operating in parallel, computing one product and one accumulation per clock cycle. We conclude easily that it is not feasible to use the RAM blocks given, since they have a bandwidth of 4 coefficients/data samples per clock cycle, and, to use only 4 MAC blocks in parallel, we would have to use a clock frequency of 200 MHz in order to achieve the desired timing requirements, which exceeds the maximum clock frequency in the RAM specification (147 MHz). Our choice was to extend the RAM bandwidth to 8 coefficients/data samples and to use 8 MAC blocks in parallel. The MAC block used was the one given, with small modifications, since some of the bits in the accumulator output of the original block were not used. The MAC block, that is supposed to multiply an input of 18 bits with one other of 36 bits and then to add the product to an accumulator. The block used divides input of 36 bits in an upper part of 18 bits and in a lower part and executes the multiplication and the accumulation of both parts in parallel. Because the FGPA used has Hardware optimized for the computation of MACs with signed 18 bits input (DSP48 blocks), and the lower 18 bits of the 36 bit input should be used in the MAC in an unsigned fashion, operations with the lower 18 bits were further divided, being that the lower 17 bits were multiplied with the 18 bits input, being forced to signed by introducing 0 in the MSB. The 18th bit of the 36 bit input was dealt with individually, being that its influence in the result is given by a multiplexor that outputs 16b'0 if it 0, or

the 16 bit input if it is 1. This allows to take advantage of the usage of the DSP48 blocks, optimizing the performance of the MAC. This MAC block has 4 pipeline stages Since the inputs of the multiplier stage are signals of 18 and 36 bits, the output has 54 bits. The output of the MAC is the result of 16384 sums of 54 bit signals, which corresponds to a product of a 54 bit signal with a $log_2(16384) = 14$ bit signal, and so, it is a signal of 68 bits. However, because this implementations is supposed to support FIR filters without gain, and the output of the IP is a signal of 18 bits, the 15 most significant bits of the MAC output are truncated (as well as the lower 35 bits). Because of this, it is unnecessary to implement signals of 68 bits, because the occurrence of overflow in the intermediate computations will not affect the final result (unless overflow occurs in the output, due to filter coefficients that do not respect the no-gain restriction), and the internal signals of the MAC block were truncated to 53 bits.

As referred, both the circular buffer RAM and the RAM for the coefficients were extended to support a bandwidth of 8 coefficients/data samples per clock cycle. This was done by keeping the structure of the original blocks, but adjusting the parameters in order to change the bandwidth. In the circular buffer, it was not required to keep the two physical blocks for each logic RAM block required in the original circular buffer, causing the resulting design to be somewhat simplified in relation to the original.

The final IP is constituted by 8 MAC blocks operating in parallel, followed by the sum of the output of all the MAC blocks. We have tested three options for this sum: using only one pipeline stage, in which all the 8 signals from the MAC outputs are summed, using two pipeline stages, being that the MAC outputs are summed two by two in one stage, and the 4 signals that result from that stage are summed in the following stage, and using three pipeline stages, being that the output of the MAC blocks are summed two by two in the first stage, the 4 outputs of this stage are summed two by two in the second stage, and the two signals that result are summed in the last stage. These three options were compared relatively to time performance and resource usage. Moreover, we have implemented control mechanism for the reset of the MAC blocks in the end of the computation of one output sample and a shift register to set the output ready signal in the clock cycle that the right result outputs the last pipeline stage.