CMPEN 417 – FPGA Design

Spring 2021

Lab 2

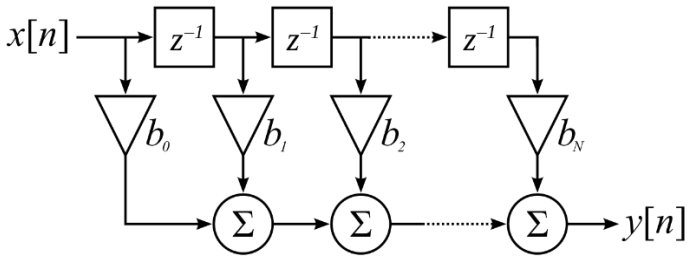
FIR Filter / 1D Convolution/Cross-Correlation

Overview

In this lab we constructed an FIR filter that receives a stream of values and multiplies a sliding window of them with an array of weights, summing the result. The weights are updated individually based on a supplied index. The interface of the module also includes control signals to indicate the ready and valid states of the value, weight, or output.

What is an FIR Filter?

In signal processing, a finite impulse response (**FIR**) **filter** is a **filter** whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. (https://en.wikipedia.org/wiki/Finite_impulse_response).



This diagram can be written algebraically as:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i \cdot x[n-i],$$

(both from https://en.wikipedia.org/wiki/Finite_impulse_response)

What is Pipelining and How is it Used?

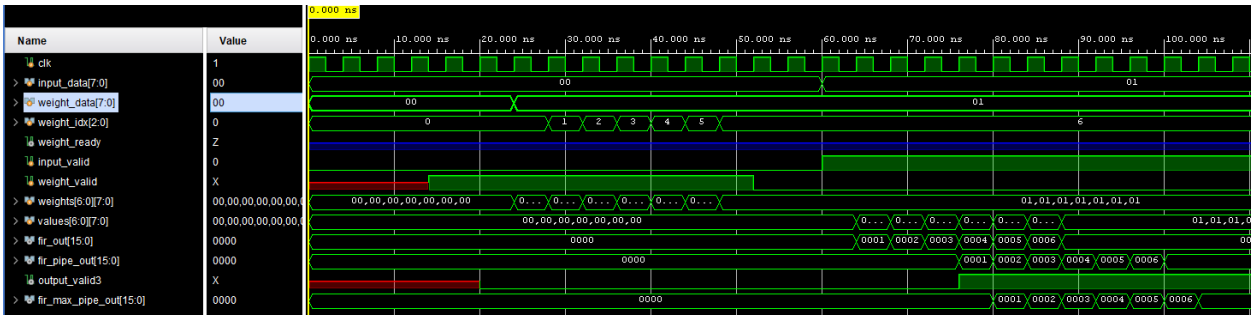
In computing, a **pipeline** is a set of data processing elements connected in series, where the output of one element is the input of the next one. The elements of a pipeline are often executed in parallel or in time-sliced fashion; in that case, some amount of buffer storage is often inserted between elements. (https://en.wikipedia.org/wiki/Pipeline_(computing)) Pipelining transformation leads to a reduction in the critical path, which can be exploited to either increase the clock speed or sample speed or to reduce

power consumption at same speed.(https://www.oreilly.com/library/view/vlsi-digital-signal/9780471241867/sec-3.1.html)

While pipelining reduces the critical path, it leads to a penalty in terms of an increase in latency. Latency essentially is the difference in the availability of the first output data in the pipelined system and the sequential system. (id) For example if latency is 1 clock cycle then the k-th output is available in (k + l)-th clock cycle in a 1-stage pipelined system.

Implementation in Verilog

There are three versions of the filter: a first that comprises only combinational logic for the multiplies and accumulate, a second that partially pipelines these operations - adding registers between each summation, and a third ("max") that pipelines every operation.



As shown in the trace, the "fir_pipe_out" trace lags the "fir_out" trace by 3 clock cycles corresponding to the three pipeline stages we added. The final "stage" is merely a buffer for the output that could be omitted. The "fir_max_out" lags by an additional cycle due to the additional pipeline stage added after the multiply step. Note the output_valid signal is activated as appropriate corresponding to the max version – this way the pipeline carries the valid flag through the operation. Pipelining in this situation does not affect total bandwidth as there is no variance between operations that may benefit from buffering and there is no out-of-order execution.

Conclusion

If we had greater understanding of the hardware target we would be using, we could discuss how increased pipelining can lower the time required for state changes in the module. As certain operations may take more time to complete, dividing up complex operations into multiple steps can decrease the time required. This can ultimately facilitate higher clock speeds – which will negate some, but not all, of the cycle lag incurred by the pipelining. Adding pipeline registers increases the footprint of the module as well as the total power requirements to charge and discharge these registers.