CS701 Individual Project Report

Ian Kuik

Introduction

In this report, the design of a Moving Average Filter component, or Data Processing Application Specific Processor (DP-ASP) is discussed and detailed. This will be described in detail through objectives as well as a comprehensive literature review regarding the history and background of this and other components to be integrated into a Heterogenous Multi-Processor on Chip (HMPSOC), followed by my method, results, and discussion of my design. Instructions on operating the testbench are included below.

Objectives

I have set out to explore various Digital Signal Processing applications to be implemented and eventually accelerated in hardware design, collaborating with others to each create a component to be integrated into a Heterogeneous Multi-Processor System on Chip (HMPSOC) to increase performance and throughput. I have been tasked with developing a moving Average Filter component. It is to be used in conjunction with other DP-ASPs via the TDMA-MIN Network on Chip (NOC) interface, where input power signal data is propagated through each component to be processed. ­

Background Information ­

Digital Signal Processors (DSPs) take real-world signals like voice, audio, video and even temperature that have been digitised and then mathematically manipulate them. These signals need to be processed so that the information they contain can be displayed or analysed. Converters such as Analog-to-Digital Converters (ADCs) are used to convert these external signals into digital representation. This is done through the process of discretisation, where samples of input signals are recorded and produced digitally.

Digital-to-Analog Converters (DACs) are then used to feed this information back into the world, for us to see, hear and use. DSPs can reduce noise, increase volume, equalise, and even eliminate signals [1].

Digital Signal Processors require high performance to quickly manipulate signals, large memory to store these signals, and wide data-width and parallelism to increase throughput. This is even more important, as demands for DSP have only increased in recent years [2].

The components of our Digital Signal Processor include an ADC, autocorrelation component, peak detector, and a moving average filter. This filter is designed to work in conjunction with a peak detector, correlation component and an ADC to form the HMPSOC.

A moving average filter is designed to smooth a signal from sudden fluctuations such as short-term noise. In its simplest form, a moving average filter, of length L, takes the average of L consecutive samples of a waveform, acting as a windowed average. If *L = 4*, then this window would only contain the last four inputted values, which are then averaged [4].

A black and white math equation

Description automatically generated

Fig 3. Rolling Average Filter Formula

If an input signal has a consistent value of x, and there is an impulse, the value of x will resist this change, mitigating the effect of this impulse. Thus, the bigger the window, *L*, the more effective this filter is on input signals. This is a simple and fast filter used in Digital Signal Processing when dealing with time-series data [5]. It is designed to reduce noise and increase the clarity of a signal, as seen below.

A graph of different colored lines

Description automatically generated with medium confidence

Fig 4. Effect of a Moving Average Filter [5]

One application of moving average filters is through power analysis. This was demonstrated in Zoran Salcic’s, ‘A new method of instantaneous power system frequency measurement using reference points detection’ [6], where a moving average filter was used to smooth the noise of an input power signal. This was done using a window size of 10.

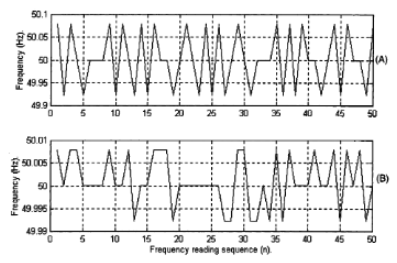


Fig 5. Effect of a moving average filter [6]

For our purposes, a digital processor is used to perform numerical calculations on sampled values of the signal, passed from the ADC [7]. In Field Programmable Gate Arrays (FPGAs), constant shifts and programmable shifts are used to ensure a low complexity in the implementation of this filter. These are able to be scalable and parallel, allowing for real time analysis and hardware acceleration. As such, a computationally simple moving average filter is desirable. It provides a satisfactory performance in time domain and effectively smooths signals out for autocorrelation.

Since averaging is done on a point-to-point basis, the more points that are included, each with equal weighting, the more accurate it will be.

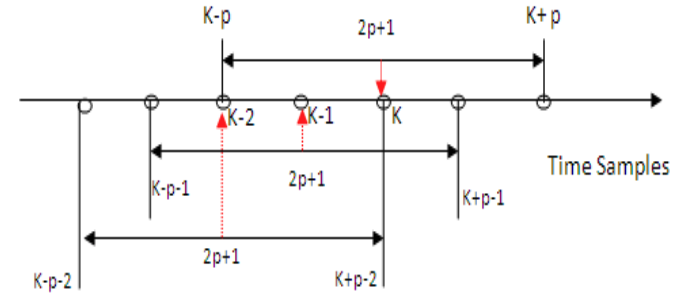


Fig 6. Calculation of average in data sequence in moving window of 2p + 1

As the window continues through the input signal, it follows this sequential logic;

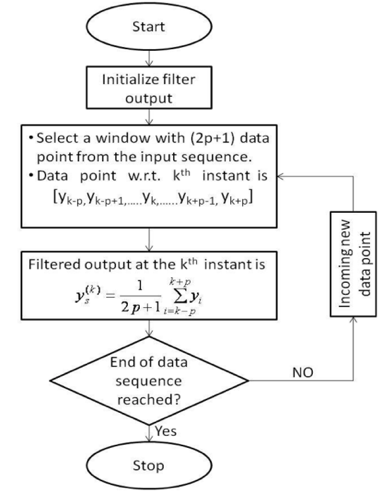


Fig 7. Sequential Logic of a Moving Average Filter

In this study [7], simulations were performed on MATLAB, with noise being generated via a Gaussian distribution with varying Signal to Noise ratios. This was put on a Xilinx XC2S200 FPGA and written in VHDL, with future improvements being toward improving throughput. However, its ideas and concepts introduced, allow for a strong foundation on how to implement this for our purposes.

Method

In order to implement a moving average filter, it was first important to understand our interface, and the interconnection between all of our different components. We decided to use the TDMA-MIN interface, for coherence and robustness our data.

A diagram of a network connection

Description automatically generated

Fig 8. TDMA-MIN Network-on-Chip interface

This allowed us to have a standard interface between all of our components, and give it the ability to communicate with a Nios interface.

This allows us to have multiple nodes, each with their control signals, such as enable and address. There will be two kinds of packets. Data and configuration packets. Data packets contain 16-bit data sections, which can be filled with zeros to accommodate for data packets of less bits.

In regard to the actual Moving Average Filter block, it needed to have the following inputs and outputs:

A diagram of a process

Description automatically generated

Fig 9. Moving Average Inputs and Outputs

* recv[32..0] is the output data of the previous component in our pipeline, in our case the ADC, or a configuration packet from our ReCOP, detailing window size.
* clock, is the input clock that synchronises this with data inputs.

Window size is configurable via a configuration packet from our ReCOP, with the lower 6 bits indicating the window size;



Fig 10. Input Configuration packet from ReCOP

Data will enter our block on the rising edge of every clock cycle. This input will be stored at the head of our array. The window size was required to be configurable up to 64, this was accomplished via a First In First Out (FIFO) array. Once sufficient new samples were recorded, they were all added together via a for loop, then averaged, giving us our averaged output. The simplified datapath is outlined below.

A diagram of a data flow

Description automatically generated

Fig 11. Simplified Datapath [8]

The averaged data will be output via the TDMA-MIN interface, with the following format.

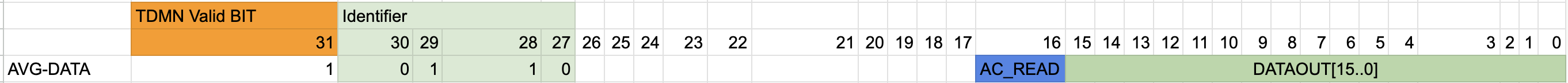


Fig 12. DP-ASP TDMA-MIN interface output

AC\_READ, will indicate to the Autocorrelator that new data is being inputted to store it in its memory.

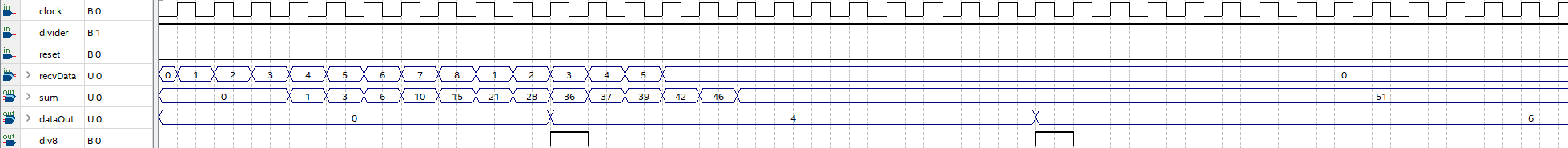
Results

To test my design, I initially generated a testbench on Quartus. The results are shown below.

A white background with black and blue dots

Description automatically generated with medium confidence

Fig 13. Quartus Testbench using window size of 4. L = 4.

Fig 14. Quartus Testbench using window size of 8. L = 8.

I then applied this block to an input signal generated from an ADC component, in a more realistic application. This gave me a varied signal, that could demonstrate actual application of this filter. I was able to successfully negate this impulse, using both window sizes, seen below.

A screen shot of a black screen

Description automatically generated

Fig 15. Input signal from ADC, window size of 8.

A green line on a black background

Description automatically generated

Fig 16. Input signal from ADC, window size of 16

A screen shot of a black screen

Description automatically generated

Fig 17. Input signal from ADC, window size of 32

A green line on a black background

Description automatically generated

Fig 18. Input signal from ADC, window size of 64.

Below is the compilation report of the DP-ASP.

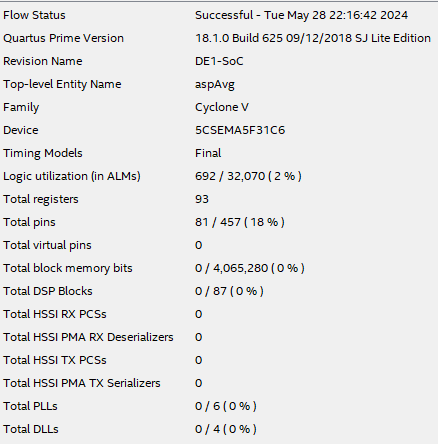


Fig 19. Compilation report of the Averaging Filter.

The component was tested through the NOC TDMA-MIN interface, where timing analysis reports a max frequency of 15.04 MHz.



Fig 20. Fmax summary

Discussion

This component is designed to take in any input signal and average it with a window size determined by a configuration packet from our ReCOP, configurable up to 64. It takes in input signals on the rising edge of the clock, and therefore its throughput is bottlenecked by the clock as well as the output of the ADC. However, this being a dedicated component allows it to be modularly used in conjunction with other DP-ASPs, contributing to a hardware accelerated HMPSOC.

The use of ReCOP configuration packets allows this DP-ASP to be modular, configurable and robust for our purposes.

Conclusion

In conclusion, this paper discusses the design of a Moving Average Data Processing Application-Specific Processor component. The processor can operate at a maximum clock frequency of 15.04 MHz and features a configurable window size set via a ReCOP processor configuration packet. It is engineered for direct integration into an autocorrelator through the TDMA-MIN interface, receiving input from an ADC. Its primary function is to clean up the signal before it reaches a Peak Detector, contributing to the formation of a Heterogenous Multi-Processor System.

# References

[1] “A Beginner’s Guide to Digital Signal Processing (DSP) | Analog Devices.” Accessed: May 20, 2024. [Online]. Available: https://www.analog.com/en/lp/001/beginners-guide-to-dsp.html

[2] D. Ibrahim and A. Davies, “The Evolution of Digital Signal Processors,” in *2019 6th IEEE History of Electrotechnology Conference (HIS℡CON)*, Sep. 2019, pp. 25–29. doi: 10.1109/HIS℡CON47851.2019.9040130.

[3] “Peak Detectors - How does it work and How to build one using Op-Amp?,” Components101. Accessed: May 20, 2024. [Online]. Available: https://components101.com/articles/peak-detector-circuit-type-working-explanation-using-opamp

[4] “Signal Smoothing - MATLAB & Simulink Example - MathWorks Deutschland.” Accessed: May 20, 2024. [Online]. Available: https://de.mathworks.com/help/signal/ug/signal-smoothing.html

[5] “Moving Average Filter: Towards Signal Noise Reduction,” Codemonk Blog. Accessed: May 20, 2024. [Online]. Available: https://codemonk.io/blog/moving-average-filter/

[6] “A-new-method-for-instantaneous-power-system-frequency-\_2000\_Electric-Power-S.pdf: COMPSYS 701: Advanced Digital Systems Design.” Accessed: May 23, 2024. [Online]. Available: https://canvas.auckland.ac.nz/courses/104293/files/12478626?module\_item\_id=2143242

[7] S. K. Shome, S. R. K. Vadali, U. Datta, S. Sen, and A. Mukherjee, “Performance Evaluation of Different Averaging based Filter Designs Using Digital Signal Processor and its Synthesis on FPGA,” *Image Process. Pattern Recognit.*, vol. 5, no. 3, 2012.

[8] “Additional frequency analysis notes.pdf: COMPSYS 701: Advanced Digital Systems Design.” Accessed: May 24, 2024. [Online]. Available: https://canvas.auckland.ac.nz/courses/104293/files/12982502?module\_item\_id=2211384

[9] University of Appalachian, Correlation and Auto-correlation Lecture Slides

http://www.appstate.edu/~grayro/comphys/lecture10c\_11.pdf