Individual Project: ADC-DP

Author: Rishi Shukla

Introduction

This individual project presents one of the many components which will be used to optimise the digital signal processing algorithms to process a noisy power signal and derive the frequency by using an integrated set of Application Specific Processors (ASP) in a Heterogeneous Multiprocessor System-on-Chip (HMPSoC).

The ADC-ASP design is the foundation of this HMPSoC, and the following report outlines a brief background of the context, the ADC-ASP's design and features, and instructions on the usage of this ASP.

Background

The frequency of a noisy power signal can be determined through the method called the reference point detection, which in our case would be a max amplitudes of the wave.[1] The brief method of finding this frequency in the case of our microprocessor is to fetch and sample a power signal via the ADC, average the values using a moving average filter for the reduction of the noise, which is then sent to a correlator which later on is then sent to the peak-to-peak detector that aims to find the maximum values to finally calculate the time and ergo the frequency of the signal.

The Network on Chip (NoC) based on the Time Division Multiple Access - Multistage Interconnect Network (TDMA-MIN) allows a transfer of data between sources and destination nodes amongst many simultaneous transfers occurring without any conflicts. A plus side of using the NoC is that it is fully time-predictable and allows us to process multiple processes at the same time.

Our main focus is designing a time-predictable HMPSoC which is able to combine the software and hardware components and develop hardware/software design. We would want to build this as we can perform time-critical executions and there are two main mechanisms that can allow for this, the RE-CoP processor that was built before that will be integrated alongside the Nios to perform control-driven tasks, and the NoC which can allow interconnected data transfers without any collisions and provides a predictable latency between any two nodes in the NoC which can help us deliver high performing frequency determination.[3]

ADC-ASP Component Features and Objectives

The high-level overview of the ADC-ASP Component can be seen as follows:

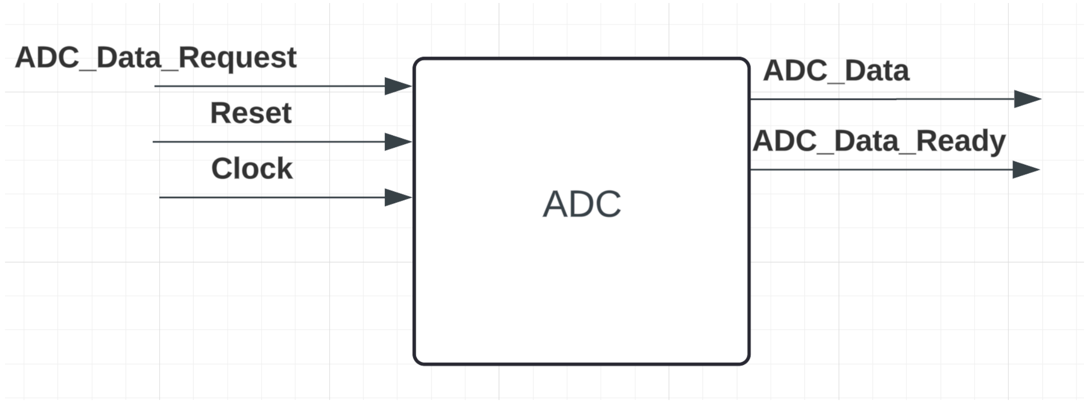


Figure 1: High-Level ADC-ASP Component

Inputs include the clock, reset, and an ADC_data_request, which is a pulse signal stating to ADC-ASP that new data is to be delivered. The new data which would be generated would be then sent to the output ADC_Data. As data transmission occurs, an ADC_Data_ready flag is enabled letting the next component know(in this case the averaging ASP) that the data is prepared for use.

The pulse signal's frequency is dependent on the ADC sampling delay value supplied by the user, and the sampling rate can be derived as:

$$Sampling\ Rate \ = \frac{100MHz}{ADC\ Sampling\ Delay + 1}$$

Equation 1: Sampling Rate Equation.

Since the sampling frequency for the wave is 16KHz, the ADC Sampling delay is calculated as:

$$16KHz \ = \frac{100MHz}{ADC\ Samping\ Delay + 1}$$
$$ADC\ Samping\ Delay \ = 6249$$

Equation 2: Calculation of ADC Sampling Delay

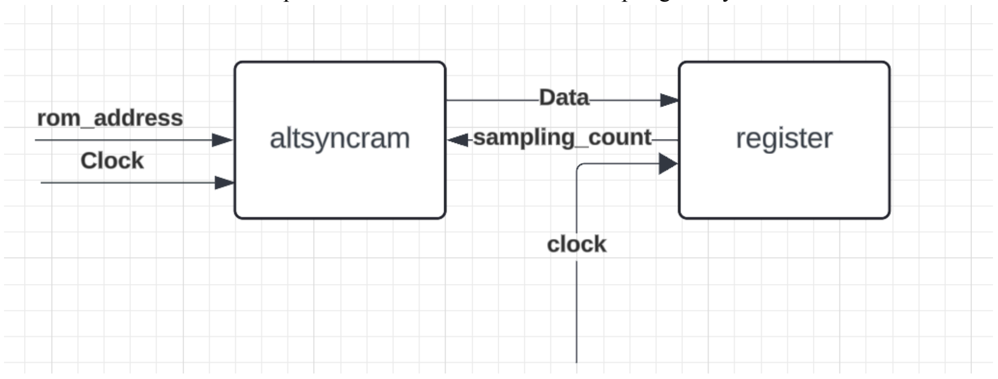
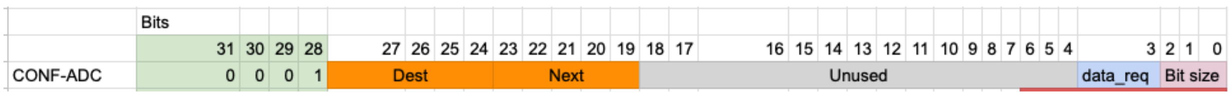


Figure 2: ADC-ASP component overview

Inside the ADC-ASP component, we have a generic component *altsyncram* which acts as our read only memory, with an input of the .mif file which emulates a quantized signal which would either be of 8, 10, or 12 bits. This memory is then accessed every clock cycle and is only outputted when the adc data is requested.

The following component would also interface with the NoC, which is TDMA-MIN, which would tell the ASPs the configurations necessary from the user. In the case for the ADC-ASP, the TDMA-MIN can send configuration messages which would change the output of the ADC component, namely configuring the number of bits utilized by the ASP.

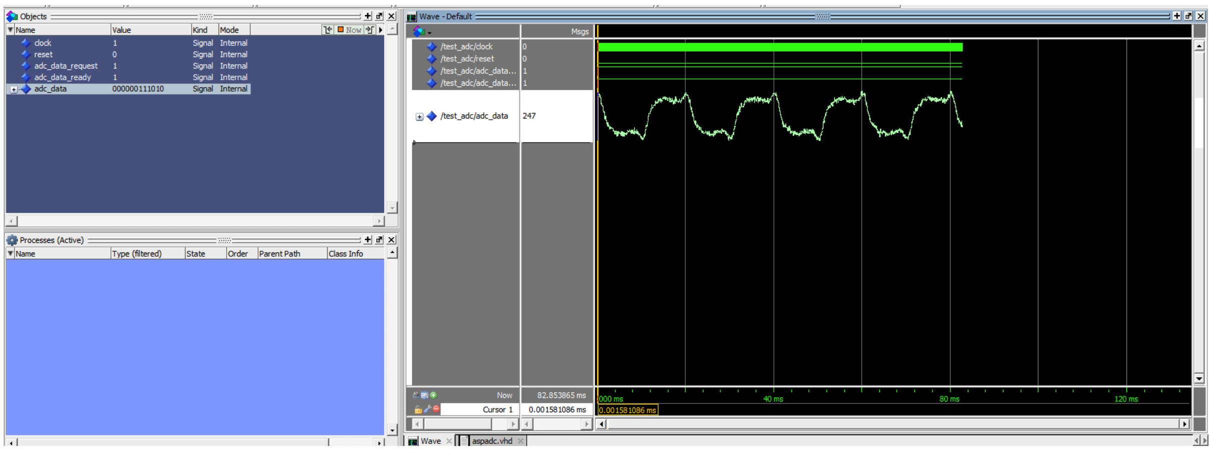


	Bits																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONF-ADC	0	0	0	1	Dest				Next					Unused															data_req	Bit size		

Figure 2: TDMA-MIN recv configure message

As shown above in figure 2, the NoC would let the system know that the ADC-ASP is being used with the bits highlighted 31 down to 28, with the bits following mapping out the destination and possible address the results could be forwarded to. Bits 23 to 19 being especially important as it would map out to the next component and give the data (which in our case would be to the AVG-ASP to average the results). Finally the final four bits would be the configuration bits indicating to the ASP what the size of the data is, whether it be 8, 10 or 12 bits.

Results and Discussion



Name	Value	Kind	Mode
clock	1	Signal	Internal
reset	0	Signal	Internal
adc_data_request	1	Signal	Internal
adc_data_ready	1	Signal	Internal
adc_data	000000111010	Signal	Internal

	Msgs
/test_adc/clock	0
/test_adc/reset	0
/test_adc/adc_data...	1
/test_adc/adc_data...	1
/test_adc/adc_data	247
Now	82.853865 ms
Cursor 1	0.001581086 ms

Figure 3: ADC-ASP output in ModelSim

As for the ADC-ASP, it was found that whilst as its own component it worked to retrieve from a .mif file and to be able to generate a quantized waveform, there are difficulties for it to be integrated with the TDMA-MIN and it currently is in a work of progress, as the issue is the ROM_address isn't moving ergo we cannot retrieve the next following values.

Conclusion

In conclusion, above is a current implementation of the ADC-ASP which is currently undergoing development and further improvements to interface with the NoC TDMA-MIN. Future work would also undergo calculating the timing analysis and also completing the component to include a reset functionality and bit configuration, however it is hopeful to see a correct result that provides a foundation to build upon and later on integrate with the other components to aid the greater context of a high performance frequency relay system.

References

1. Z Salcic, R Mikhael, 2000. A new method for instantaneous power system frequency measurement using reference points detection, Electric Power Systems Research, Volume 55, Issue 2, 1 August 2000, Pages 97-102, https://doi.org/10.1016/S0378-7796(99)00102-9
2. Salcic, Z., Nadeem, M. and Striebing, B, 2016, A Time Predictable Heterogeneous Multicore Processor for Hard Real-time GALS Programs. ARCS 2016
3. Salcic, Z., Park, H., Biglari-Abhari, M., and Teich, J., 2019, SystemGALS – A language for the design of GALS software systems, Embedded Systems Research Group, University of Auckland, Internal document, Embedded Systems Research Group, available to C701 class