Group 17 Lab 2 report - Wilson Wu, wwu681, Joey Back, jbac208

Task2: Creating and simulating DP-ASP

We designed the DP-ASP data-path around a three-sample sliding window per channel. On each rising clock edge, if a valid TDMA-MIN packet arrives (header "1000"), we: shift the new sample into our 4-deep FIFO, compute the four-point average, double the result, clip it to the range ±4096, then repackage the 16-bit clipped value into a new TDMA-MIN frame. All of this lives in one synchronous, clock-driven process.

In our top-level testbench, we extended the existing TDMA-MIN framework by instantiating our new data-processing ASP alongside the provided TestAdc and TestDac modules. Specifically, the AspDp entity was wired to receive from TDMA port 0, which is the TestAdc source, and to send its output into TDMA port 1, which feeds the TestDac sink. This routing allowed us in QuestaSim to view both the raw ADC samples (channel 0) and our processed output (channel 1) on the DAC waveform, confirming that our moving-average, scaling and clipping logic operated correctly in the full system context.

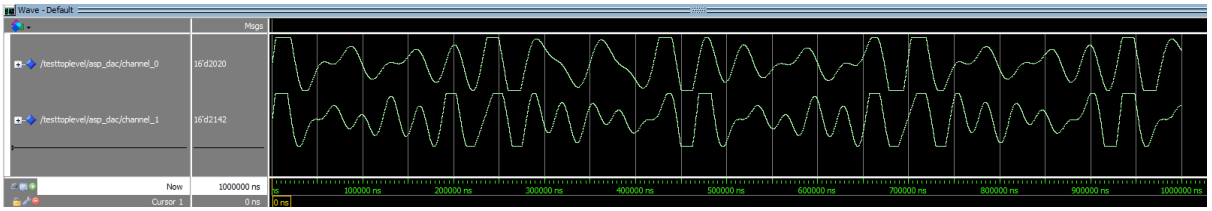

Fig 1. QuestaSim simulation of asp_dac

Task3: Integrating into hardware

For hardware deployment, TopLevel.vhd was modified to accommodate four NoC ports, specifically by updating the generic parameter from 3 to 4. Next, the new AspDp component was instantiated and connected to send_port(3) and recv_port(3). Additionally, a review of the existing AspExample.vhd was required in order to be familiar with how the I/O was to interact with the audio output. Pressing the different keys issued 40-bit TDMA-MIN frames for the left (channel 0) and right (channel 1).After synthesising and programming the FPGA, pressing the push-buttons triggered the configuration sequence. Live audio passed through the board, averaged by the DP-ASP, then clipped and routed to the DAC-ASP. The result matched our simulation, demonstrating successful end-to-end integration.