Data Aggregation Model in Verilog

Philip Tracton

July 4, 2018

Contents

1 Theory

This module is the link between the ADC or other data source and the FIFO module. The data from an ADC can come in a lot of different widths. This module supports data width inputs from 8 to 32 bits. This is managed by a parameter for the module. As the data comes in, it is placed into a 32 bit word as it fits. Once the word is complete, this module will push the data to the FIFO module.

For 8 bit data, it takes 4 samples and arranges them from low to high. It does not matter if this data is signed or unsigned since it fits into the specified size. For 16 bit data, it takes 2 samples and arranges them from low to high. It does not matter if this data is signed or unsigned since it fits into the specified size. For 32 bit data, it takes 1 samples. It does not matter if this data is signed or unsigned since it fits into the specified size.

For data all otehr sizes, the data pre-pends enough bits to make it the next largest half word aligned size. So for 9 to 15 bit data it becomes 16 bits. For 17 to 31 bits it becomes 32 bit data. If the data is unsigned, it is always prepended with 0. For signed data, it is sign extended.

There is an input indicating if the data is expected to be signed. When asserted the data is treated as signed data. When de-asserted the data is treated as unsigned.

2 Operations

2.1 Parameters

- There is a single parameter in this module, **ADC_DATA_WIDTH**. It defaults to 8 bits. It can go up to 32. Above 32 will trigger a simulation error and termination. This values controls the width of the expected data from the data souce. It is assumed that the source will be an ADC but that is not a requirement.

2.2 Signals

Table 1: wb_daq_data_aggregation Port Signals

Signal Name	Direction	Size	Behavior
wb_clk	Input	1	Clock for synchronous behavior
wb_rst	Input	1	Synchronous reset

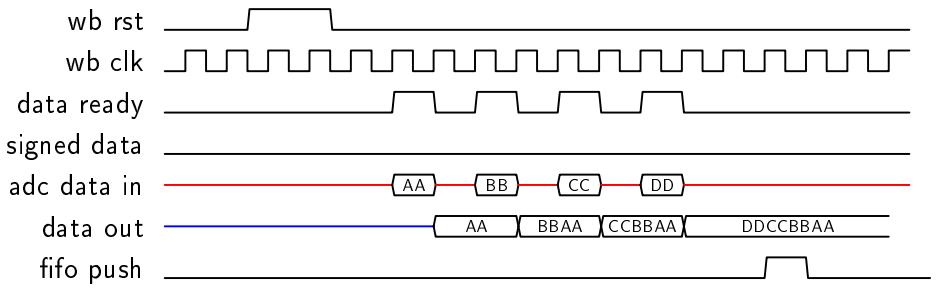
Continued on next page

Continued from previous page

Signal Name	Direction	Size	Behavior
data_ready	Input	1	Capture the adc_data_in as a sample
signed_data	Input	1	When asserted the adc_data_in is signed
adc_data_in	Input	ADC_DATA_WIDTH	The data to capture
data_out	Output	32	adc_data_in data aggregated and going to FIFO
fifo_push	Output	1	Push the data_out into the FIFO

2.3 Block Diagram

2.4 Bus Cycles



3 Testing

3.1 Run Simulations

Several different simulators were used to verify the RTL. The test bench is self checking and indicates pass/fail for both individual tests and the overal test run.

3.1.1 Icarus Verilog

- This is the primary tool since it is free and runs everywhere. To run the simulation with this tool use ***make sim***. This will produce a dump.vcd that can be viewed with GtkWave and a wb_daq_data_aggregation_iverilog.log

3.1.2 Modelsim

- Use ***make modelsim*** to execute the simulation via the modelsim command line options
- If the modelsim GUI is started, the modelsim.do file will run and produce the waveforms

3.1.3 Xcelium

- Use ***make xrun*** run the simulation with the Cadence Xcelium tools if you have access to them. This will produce a dump.vcd that van be viewed with Simvision.

3.2 Cleaning Up

- Use ***make clean*** to remove all produced output from any of the simulations or documentation tools.

3.3 Linting

- Use ***make lint*** to use verilator in it's lint-only mode on wb_daq_data_aggregation.v. It will pass silently. There is only feedback if there is a problem.

4 Synthesis

4.1 Yosys

- Use the command ***make synthesis*** to synthesize the wb_daq_data_aggregation.v file into a wb_daq_data_aggregation_synth.v for Xilinx technology.
  This is a new tool that is being learned as this is developed. It will silently run and produce both a wb_daq_data_aggregation_xilinx_synthesis.v and a wb_daq_data_aggregation_yosys.log.