FIFO to SRAM State Machine in Verilog

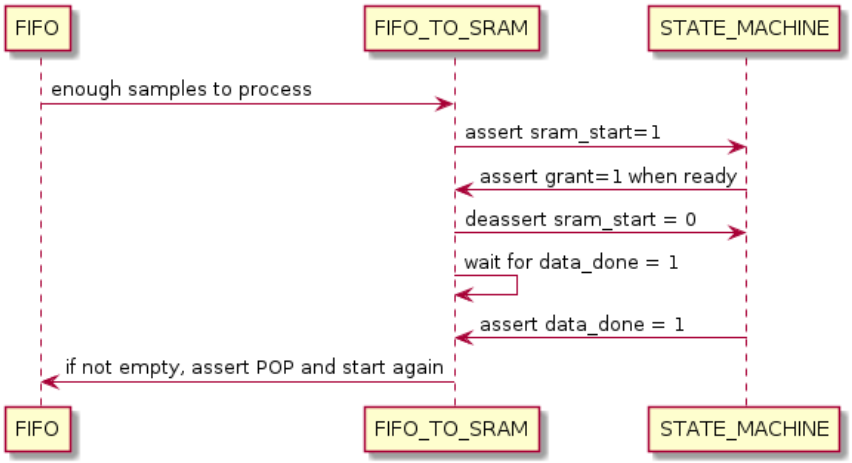
Philip Tracton

July 5, 2018

Contents

1 Theory

This module reads and pops the data from the FIFO and presents it to a state machine that will write it out to SRAM. Once there are enough samples in the FIFO, this makes burst operataions viable, this module will read the value from the FIFO and attempt to pass it to a state machine to write it out. This module will handshake with the state machine. This is started by asserting ***sram_start*** to signal that we have data from the FIFO. When the ***grant*** signal is asserted by the state machine, ***sram_start*** is lowered. This indicates that the data has been moved on to the state machine. The state machine will, when ready, assert the ***data_done*** signal to indicate that the data is finished writing and this module should get the next sample if there is one. As long as the FIFO is not empty, this process will repeat.



2 Operations

2.1 Parameters

There is a single parameter in this module, ***NUMBER_OF_SAMPLES_BITS***. This controls the width of the fifo_number_samples and fifo_number_samples_terminal ports. This value is the total number of bits in both of these ports. That makes them both [NUMBER_OF_SAMPLE_BITS-1:0] wide.

This is done because the FIFO that is feeding this module could be any possible depth. This allows the user to adjust these inputs to match the size of the information coming from the FIFO.

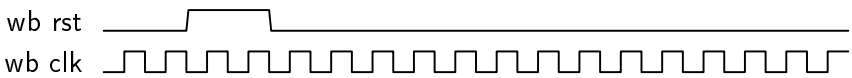
2.2 Signals

Table 1: fifo_to_sram Port Signals

Signal Name	Direction	Size	Behavior
wb_clk	Input	1	Clock for synchronous behavior
wb_rst	Input	1	Synchronous reset
empty	Input	1	Is the FIFO empty?
grant	Input	1	Granted permission to send data to state machine
fifo number samples	Input	NUMBER SAMPLES BITS	how many samples are in the FIFO
fifo number samples terminal	Input	NUMBER SAMPLES BITS	Start processing when we have this many samples
data_done	Input	1	Data done moving to SRAM
fifo_data_in	Input	32	Data from the FIFO to store in SRAM
pop	Output	1	Get the next data from the FIFO
sram_start	Output	1	Handshake to start state machine to write SRAM
sram_data_out	Output	32	Data to write to SRAM

2.3 Bus Cycles

- This shows the basic loop of processing the data



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 fifo_to_sram_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 fifo_to_sram.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 fifo_to_sram.v file into a fifo_to_sram_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 fifo_to_sram_xilinx_synthesis.v and a fifo_to_sram_yosys.log.