Wiki »

EXERCISE: Timing Specification and Debugging on an SRAM interface

Exercise Goals

This exercise will through implementation of an SRAM interface introduce you to:

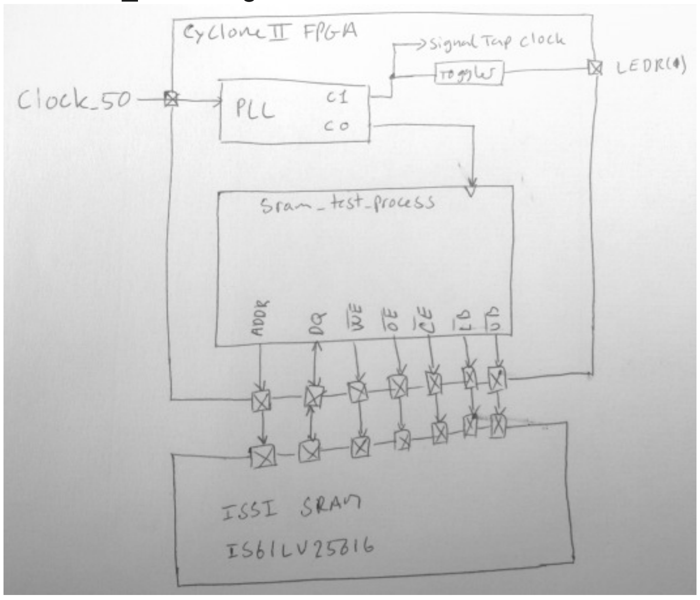
- SRAM Interface
- Timing specification and analysis
- Debugging with Signal Tap Logic Analyzer

Prerequisites

- Quartus II and ModelSim software must be installed and working.
- You may need to install a new license file and use a VPN connection to ASE
- Skeleton code checked out from repository (exercise_sram_test)

Introduction

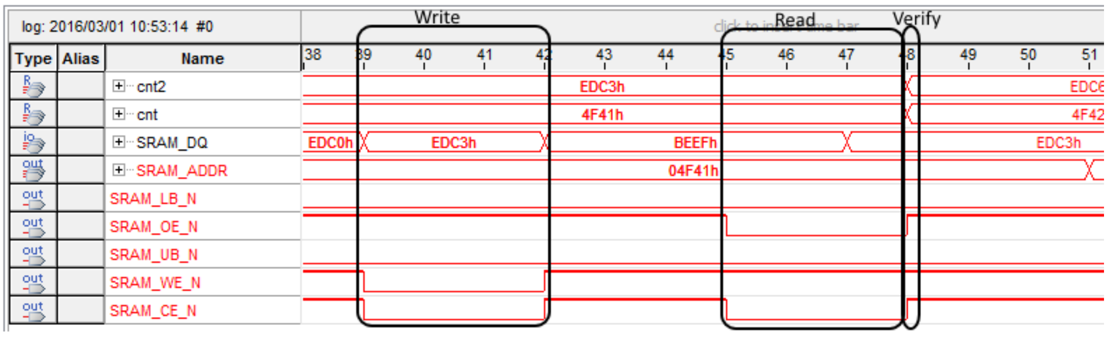
The sram_test design looks like this :



It contains a state machine that basically runs through the SRAM addresses and performs a write-wait-read-verify at each address. If this fails a red led will turn on.

The communication between FPGA and SRAM looks like this:

DE2:



The design will in its basic configuration fail during test. This is because of the SRAM cannot provide data in due time for the validation, due to its internal timing limitations.

It is your job to specify the data interface timing for the SRAM, analyse the timing report results with Time Quest and update the pll output frequency to the maximum permissible.

There should be consistency between your timing analysis and what happens when testing on the board (At least, if your board fails, so must the timing analysis) . If you have no errors during timing analysis, but you do have them during execution on the DE2 board, then your timing specification is incomplete.

Note that Quartus actually uses the timing specification actively during fitting, which means that a well-specified interface will run faster than an unconstrained.

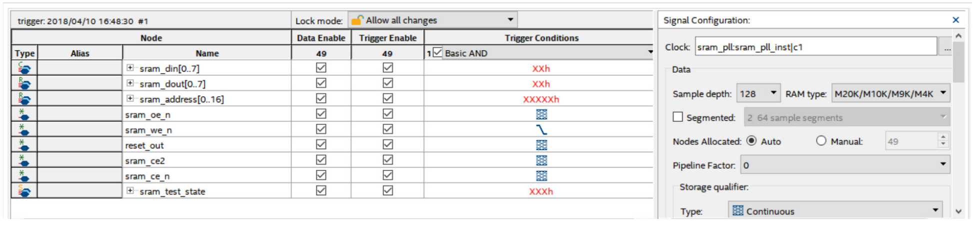
Steps to help you through:

Open the project

In the Project Navigator, double click on the PLL instance and update c0 output from 50MHz to 83MHz, finish, and compile the design.

Create a new Signal Tap II file

From file->new. Its content should look something like this:



trigger: 2018/04/10 16:48:30 #1 — Lock mode: Allow all changes

Node			Data Enable	Trigger Enable	Trigger Conditions
Type	Alias	Name	49	49	1 ✓ Basic AND
		sram_din[0..7]	✓	✓	XXh
		sram_dout[0..7]	✓	✓	XXh
		sram_address[0..16]	✓	✓	XXXXXh
		sram_oe_n	✓	✓	▨
		sram_we_n	✓	✓	⤵
		reset_out	✓	✓	▨
		sram_ce2	✓	✓	▨
		sram_ce_n	✓	✓	▨
		sram_test_state	✓	✓	XXXh

Signal Configuration:

Clock: sram_pll:sram_pll_inst|c1

Data

Sample depth: 128 RAM type: M20K/M10K/M9K/M4K

☐ Segmented: 2 64 sample segments

Nodes Allocated: [✓ Auto] Manual: 49

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Note that we are using the pll's clock, c1 (200 MHz), as sampling clock and this limits the max frequency due to aliasing.

See the Signal Tap Tutorial as reference.

Compile Again

Note if there are any timing errors? Keep Signal tap open, go to the programmer, program the fpga and return to Signal tap. You should see LED lit, as the test fails. Press the acquisition button in Signal Tap to see the data transfer. You should see something similar to the waveform above.

So, No compilation errors, but a design that fails!

Run TimeQuest

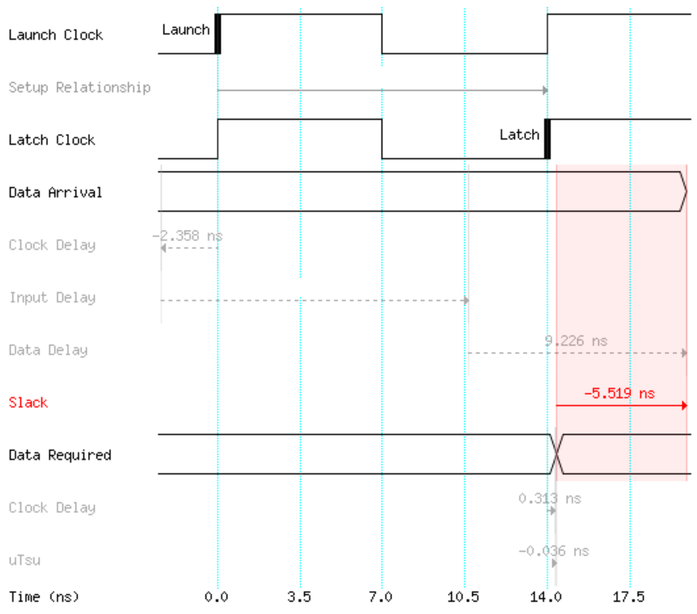
Run the steps "Create Timing Netlist", "Read SDC file" and Update Timing Netlist" from the left window. Note the console windows at the bottom for errors!
Run "Report all I/O Timings", anything interresting? Use the Time Quest Tutorial as reference.

Specify the timing

Open the sram_test.sdc in a text editor and enter the correct timing values (four places!) and uncomment the the four lines. THIS IS THE HARDEST STEP!!! Use SRAM data sheet and timing specification figures to estimate the correct values. SEE NEXT SECTION below.

Analyze timing again

In TimeQuest, reload the sdc file and update the timing netlist.
Run "Report all I/O Timings", you should get some violations now, if you have entered the right timing values.
In the list of timing violations find the line of the violation going from SRAM_DQ[<no>] to LEDR[0]~reg0 (~reg0 because its LEDR[0] registered output and not just a wire). Right-click and choose "report timing", view the waveforms to understand the problem. Note the negative slack time. You should get something like:



Recompile in Quartus

Errors during compilation? Check the slack in TimeQuest (like in previous step), has it changed?
Report Max Frequency, whats the limit?

Update the PLL to the max frequency

In Quartus, in the hierarchy window (upper left) double-click on the pll comopnent, which will open the Mega-Wizard. Adjust the c0 output frequency. Press finish and recompile, analyze and test on DE2. Any arrors? Is it now working? Check result in Signal Tap. Is it visible? We are sampling at 200MHz, what does that imply?

Compile with timing specification

Comment the four lines in your .sdc file, recompile, analyze and test. Do you have a negative slack now? Does it work on the DE2 board? Check result in Signal Tap.

Configure timing

You must specify the timing for the SRAM_DQ interface in the .sdc file.

The sdc file has been prepared for you, and you just need to replace the zeros with your values:

```
 # file: sram_test.sdc

set input_min_delay_IS62WV1288DBLL45 0
set input_max_delay_IS62WV1288DBLL45 0
```

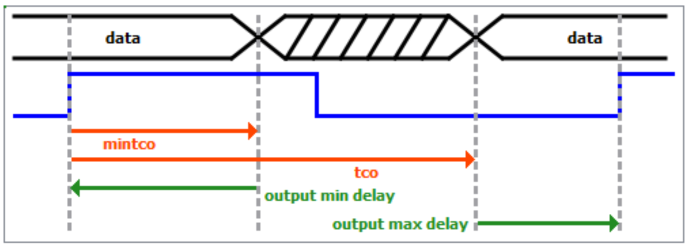
```
set_input_delay -add_delay -max -clock $clk_sys  $input_max_delay_IS62WV1288DBLL45 [
set_input_delay -add_delay -min -clock $clk_sys  $input_min_delay_IS62WV1288DBLL45 [

set output_min_delay_IS62WV1288DBLL45 0
set output_max_delay_IS62WV1288DBLL45 0
set_output_delay -add_delay -max -clock $clk_sys  $output_max_delay_IS62WV1288DBLL45
set_output_delay -add_delay -min -clock $clk_sys  $output_min_delay_IS62WV1288DBLL45
```

You must estimate the timing based on the data sheet for the SRAM attached and from how Time Quest specifies timing.
Note that you should also consider delays from PCB traces (approx 6ps/mm see here for details).

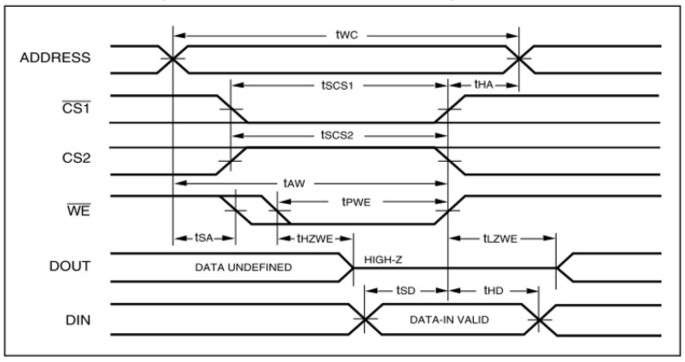
Write-cycle

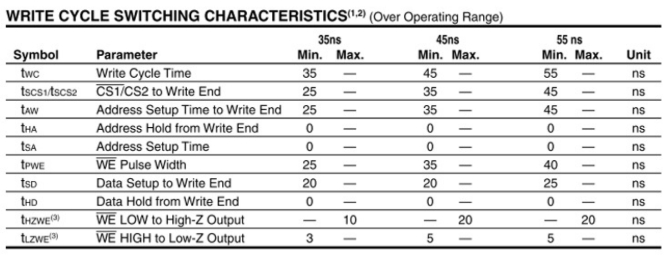
During write, SRAM_DQ is an output from the FPGA to the SRAM device, this means that we must configure the output delay in this case.
Looking at the Time Quest diagram and the data sheet waveforms you should be able to estimate values.



The green times are the ones you must specify, the diagram just shows their relation to setup- and hold times.

WRITE CYCLE NO. 1 ($\overline{\text{CS1}}$/CS2 Controlled, $\overline{\text{OE}}$ = HIGH or LOW)



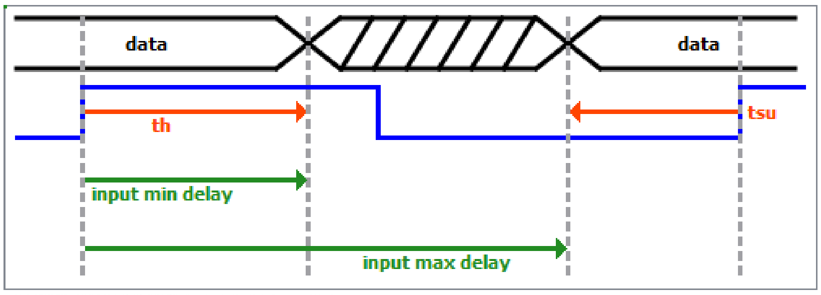


WRITE CYCLE SWITCHING CHARACTERISTICS(1,2) (Over Operating Range)

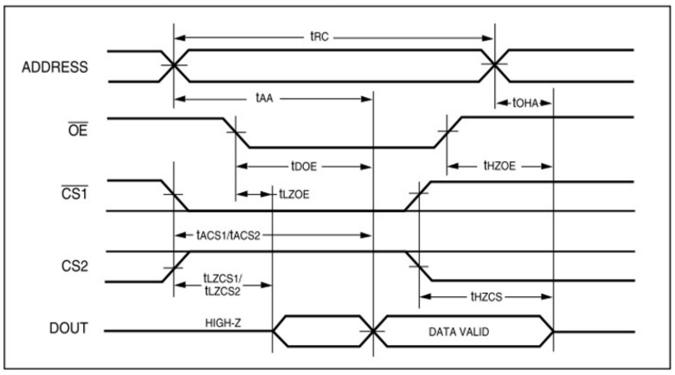
		35ns		45ns		55 ns		
Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Unit
t_{WC}	Write Cycle Time	35	—	45	—	55	—	ns
t_{SCS1}/t_{SCS2}	$\overline{\text{CS1}}$/CS2 to Write End	25	—	35	—	45	—	ns
t_{AW}	Address Setup Time to Write End	25	—	35	—	45	—	ns
t_{HA}	Address Hold from Write End	0	—	0	—	0	—	ns
t_{SA}	Address Setup Time	0	—	0	—	0	—	ns
t_{PWE}	$\overline{\text{WE}}$ Pulse Width	25	—	35	—	40	—	ns
t_{SD}	Data Setup to Write End	20	—	20	—	25	—	ns
t_{HD}	Data Hold from Write End	0	—	0	—	0	—	ns
t_{HZWE}(3)	$\overline{\text{WE}}$ LOW to High-Z Output	—	10	—	20	—	20	ns
t_{LZWE}(3)	$\overline{\text{WE}}$ HIGH to Low-Z Output	3	—	5	—	5	—	ns

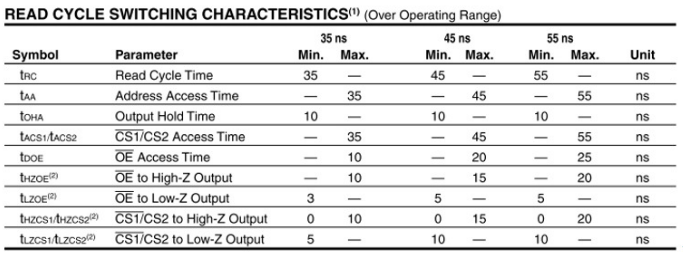
Read-Cycle

This is the same as for the write cycle, just remember that WE/CE a.o. must propagate to the SRAM device, then it looks up a value and returns it after a while and finally it must propagate back to the FPGA!



READ CYCLE NO. 2[1,3] ($\overline{\text{CS1}}$, CS2, $\overline{\text{OE}}$ Controlled)




READ CYCLE SWITCHING CHARACTERISTICS[1] (Over Operating Range)

		35 ns		45 ns		55 ns		
Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Unit
t_{RC}	Read Cycle Time	35	—	45	—	55	—	ns
t_{AA}	Address Access Time	—	35	—	45	—	55	ns
t_{OHA}	Output Hold Time	10	—	10	—	10	—	ns
t_{ACS1}/t_{ACS2}	$\overline{\text{CS1}}$/CS2 Access Time	—	35	—	45	—	55	ns
t_{DOE}	$\overline{\text{OE}}$ Access Time	—	10	—	20	—	25	ns
t_{HZOE}[2]	$\overline{\text{OE}}$ to High-Z Output	—	10	—	15	—	20	ns
t_{LZOE}[2]	$\overline{\text{OE}}$ to Low-Z Output	3	—	5	—	5	—	ns
t_{HZCS1}/t_{HZCS2}[2]	$\overline{\text{CS1}}$/CS2 to High-Z Output	0	10	0	15	0	20	ns
t_{LZCS1}/t_{LZCS2}[2]	$\overline{\text{CS1}}$/CS2 to Low-Z Output	5	—	10	—	10	—	ns

- IMG_1495.JPG (408 KB) Peter Høgh Mikkelsen, 2017-03-01 07:34
- sram_de1soc_wfm.png (22,3 KB) Peter Høgh Mikkelsen, 2017-03-01 07:39
- sram_de1soc_modelsim_wfm.png (225 KB) Peter Høgh Mikkelsen, 2017-03-01 07:42
- signaltap_config_de1soc.PNG (40,6 KB) Peter Høgh Mikkelsen, 2017-03-01 07:46
- read_cycle_timing.png (103 KB) Peter Høgh Mikkelsen, 2018-04-11 09:44
- write_cycle_timing.png (108 KB) Peter Høgh Mikkelsen, 2018-04-11 09:44
- signal_tap_setup.png (136 KB) Peter Høgh Mikkelsen, 2018-04-11 09:50
- license_ase_au_dk.dat (111 Bytes) Peter Høgh Mikkelsen, 2018-08-17 09:38