Exercise 10:

a) RAM Controller

The communication between the FPGA and the external PSRAM is done by the module *ram_controller*.

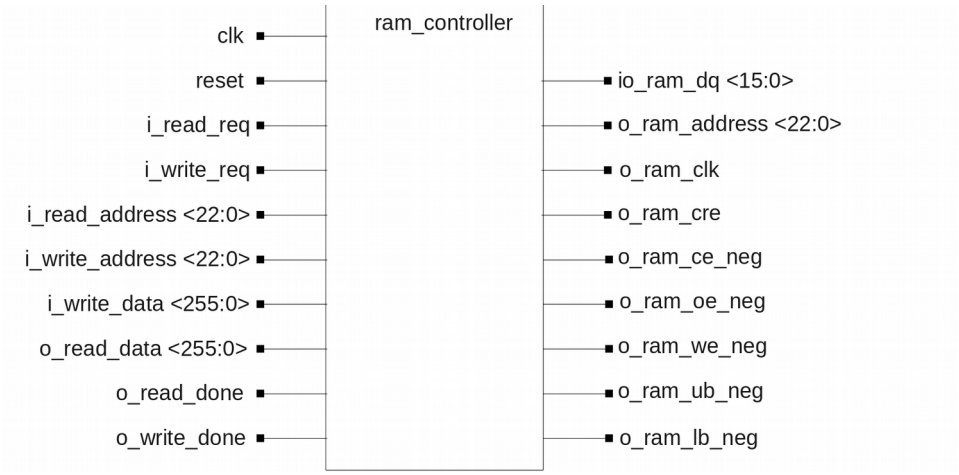


Figure 1: ram_controller

Port declaration:

- *clk* : in std_logic
- *reset* : in std_logic
- *i_read_req* : in std_logic
- *i_write_req* : in std_logic
- *i_read_address* : in std_logic_vector (22 downto 0)
- *i_write_address* : in std_logic_vector (22 downto 0)
- *i_write_data* : in std_logic_vector (255 downto 0)
- *o_read_done* : out std_logic
- *o_write_done* : out std_logic
- *o_read_data* : out std_logic_vector (255 downto 0)

- *io_ram_dq* : inout std_logic_vector (15 downto 0)
- *o_ram_address* : out std_logic_vector (22 downto 0)
- *o_ram_ce_neg* : out std_logic
- *o_oe_neg* : out std_logic
- *o_we_neg* : out std_logic
- *o_ram_ub_neg* : out std_logic
- *o_ram_lb_neg* : out std_logic

The PSRAM used on the board uses a DRAM architecture. This has some important drawbacks from a programmer's point of view: Since Data is addressed via rows internally, accessing or writing data is delayed by the time needed to open the corresponding row. Additionally, the capacitors used to store the information in DRAM lose their charge over time. This mandates periodic refreshes of all rows by opening them and writing back the result. If this refresh cycle collides with a read or write request from the FPGA, the access is delayed even further.

All together, this causes extremely long access times when the PSRAM is used as an asynchronous SRAM, since the on-chip controller assumes the worst case for every access (possible refresh collision and initial row opening latency). To overcome this problem, the PSRAM has a synchronous interface which masks the initial access latency by transferring larger bursts of data from the same row that was initially opened. This synchronous interface is more complex to design, so this part will be provided to you. However, the PSRAM has be be configured initially to use the synchronous interface via asynchronous signals.

Your task is to configure the PSRAM for synchronous operation.

The complete parameters are:

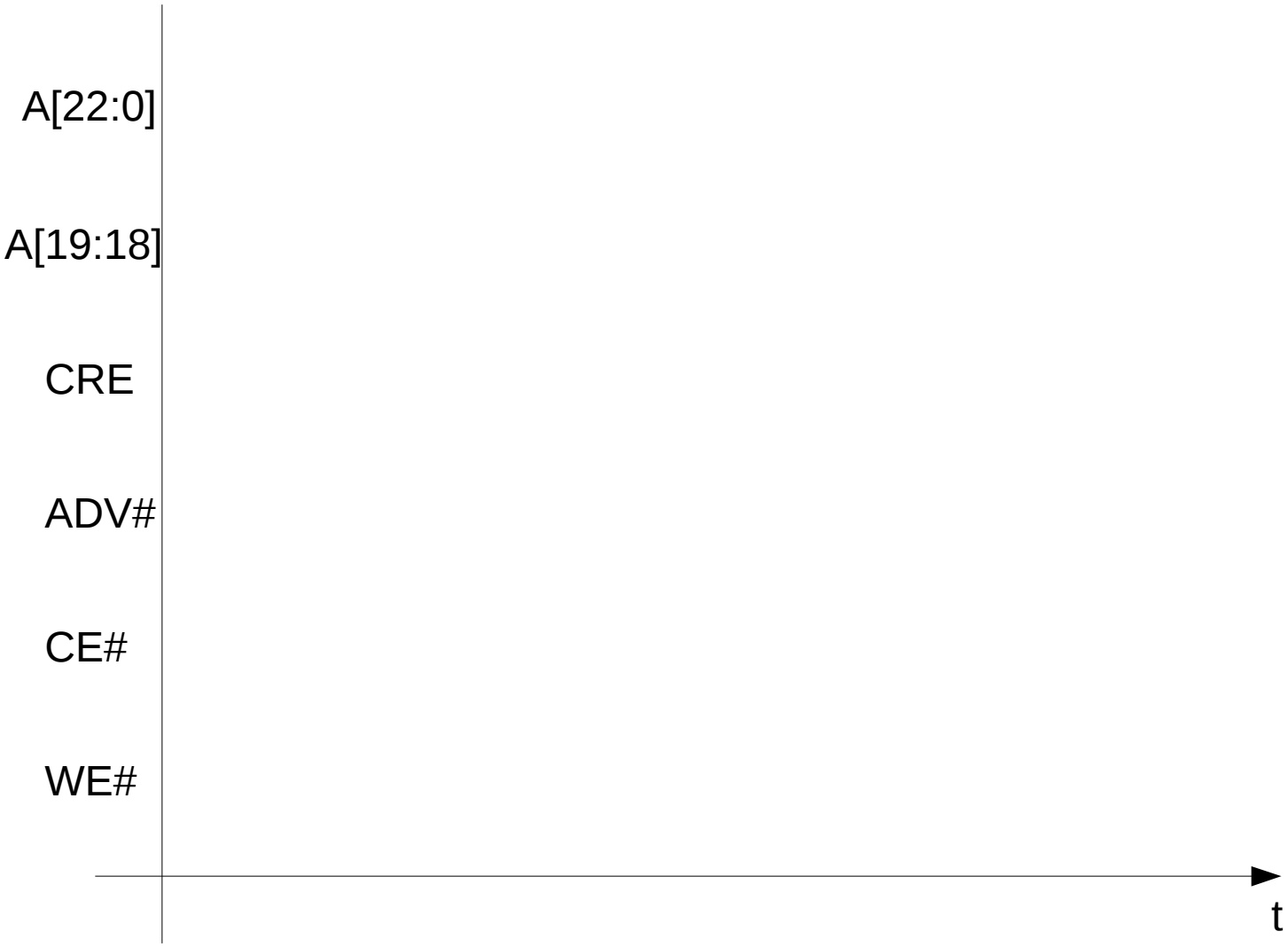
- synchronous operation
- fixed initial latency
- ... of 4 clock cycles
- wait active high
- ... asserted during delay cycles
- 1/2 drive strengh
- no burst wrap
- continous bursts until the end of row

The procedure for writing to control registers using CRE, the correct register and the bit positions as well as the timing can be found in the PSRAM's Datasheet in your documentation folder.

The controller should finish the configuration in as few clock cycles as possible. Note: Your “clk” input runs at 100Mhz, so one clock cycle equals 10ns.

To simulate the interaction between the controller and the SRAM, a simulation model of the SRAM is provided. The necessary Modelsim directives for compiling this Verilog model are already included in the .do-file.

Draw your chosen timing in the diagram on the following page.


A[22:0]
A[19:18]
CRE
ADV#
CE#
WE#
t