INDIVIDUAL COURSEWORK 3 - EXTERNAL RAM INTERFACE FSM

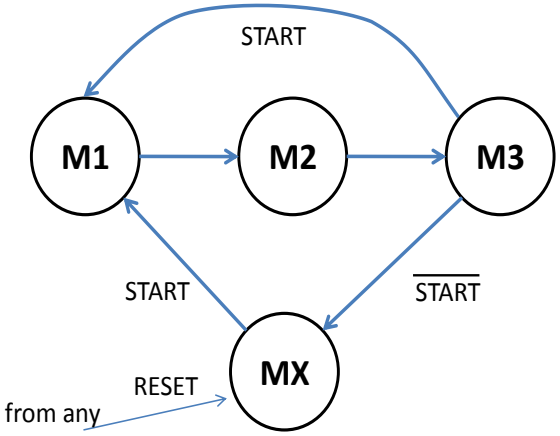

Figure 2

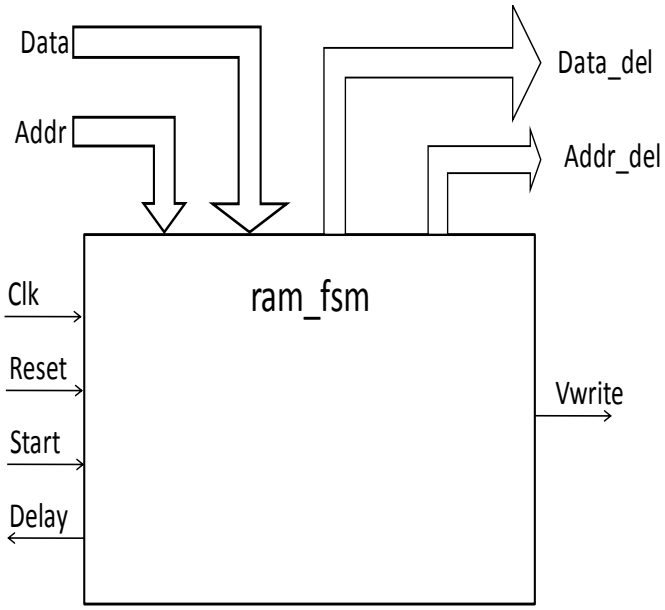

Figure 3

This coursework is to complete the design of & code a state machine that could for example control an external static RAM. Applications information for this is included below, however your deliverable depends only on the specification in this section. You do not have to consider applications to implement this.

The state machine state diagram is shown in Figure 2, it has 4 states. State transitions without Boolean expressions always happen on the next positive clock edge.

Write an entity **ram_fsm as in Figure .** Use transitions as shown in Figure 2 and determine logic to generate vwrite and delay from FSM state and inputs consistent with Figure 3. Note that as shown in Figure 2 the FSM must unconditionally enter state MX on reset.

Add two D registers (flip-flops) clocked by the *negative* edge of **clk** to your entity, with D inputs the *open array vectors* **addr** and **data**, and outputs the *open array vectors* **addr_del** and **data_del** respectively. These inputs and outputs are all ports of ram_fsm as shown in Figure 3. All ports are either std_logic or std_logic_vector as appropriate.

The std_logic outputs vwrite and delay must be implemented as:

Vwrite = M3

Delay = (M1 or M2) and START

Submit this in file **ram_fsm.vhd**.

APPLICATIONS INFORMATION

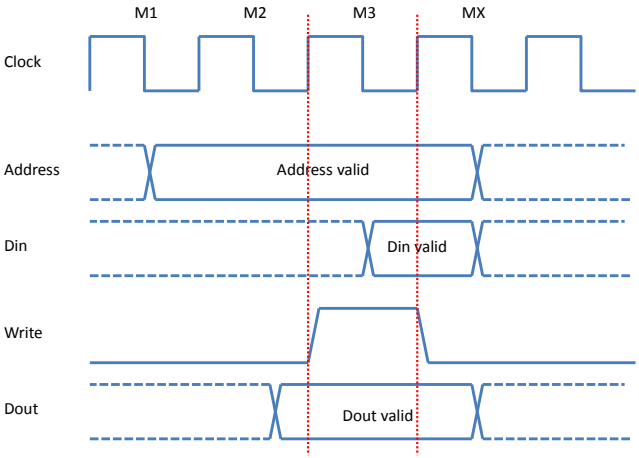

Figure 1

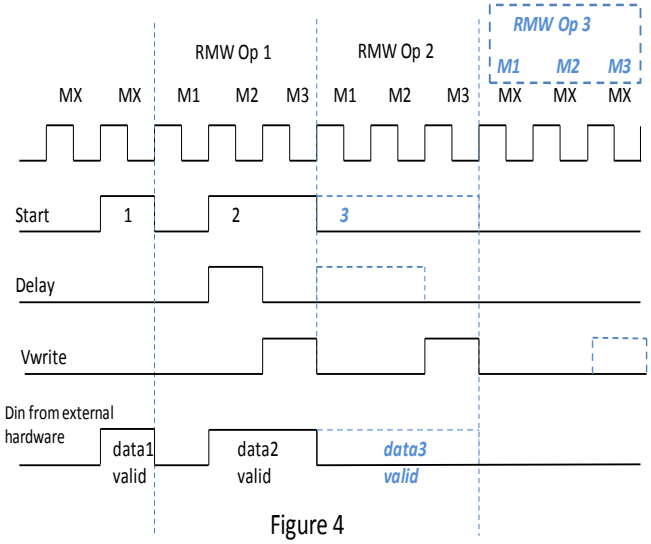

Figure 4

The RAM has inputs Address, Din (data input) Dout (Data output) and Write. Figure 1 shows the waveforms required to perform a read-modify-write (RMW) memory operation. An FSM which sequences RMW memory operations and controls the RAM as required by Figure 1 is shown in Figure 2. States M1,M2,M3 sequence an RMW cycle, MX is entered when the memory is idle. The FSM entity is shown in Figure 3, it has output **vwrite** to the RAM Write signal, and it interfaces with hardware using the RAM through input **start** and output **delay**. Additional required ports: **addr**, **addr_del**, **data**, **data_del** are described below under Assessment.

Figure 4 illustrates the required behaviour of FSM outputs **delay** and **vwrite**. RMW operations 1 & 2 are either followed by idle, or by a further request for operation 3 shown in dotted blue lines. The **start** input is asserted by hardware external to the FSM whenever a memory RMW operation is required. The FSM output signal **delay** is high only if a requested new RMW operation is delayed because a previous operation has not completed. **Delay** must stay high as shown in Figure 4. This is until the clock cycle immediately before the new RMW operation starts. Figure 4 shows that **start** may go high to request a new RMW operation immediately after the previous operation ends. In this case **start** will be continuously high as is shown in operation 2 and 3. If there is no new cycle needed by external hardware **start** will remain low and the memory becomes idle.

The request interface has two lines: start (input to FSM) indicates that a new RAM operation is required, delay (output from the FSM) indicates that the previous operation is not complete and the requested operation must be delayed. Thus the condition for a new operation to begin is START and (not DELAY).

The memory interface has a single control line, vwrite, which instructs the RAM to write the current data (din) into the current address. Note that the RAM has no clock signal.

In order to meet RAM timing spec the data and address in to the RAM are delayed (unusually) by ½ a clock cycle. That is not part of the timing in the FSM, but it is shown in the timing diagram. Obviously a negative clock edge triggered register, as is specified above, will perform the necessary half cycle delay.

If the RAM data access time specification is slower than is given in the timing diagram this can be fixed by a small change to the FSM. State M2 can be made longer: either by replacing M2 by a linear sequence of states: M2a->M2b->M2c...->M2x where the number of states in the sequence determines the time, or by adding a counter which is reset in M1 and counts up to some maximum value in M2. The state transition from M2 is modified so that it only happens when the counter has reached its maximum count. This second method is more flexible because the delay time can be adjusted.

The work you do here will likely be used in a modified form, rather than directly, in the team project.