Solutions to Assignment 7

Question 1

Design

Define Inputs and Outputs

The first step in designing a state machine is to define the inputs and outputs. These have already been defined in the assignment.

Define States

The next step is to define the required states. When designing a Moore state machine (this is usually the best choice) there will be at least one state for each unique combination of outputs. In addition, changes in the outputs will happen at the same time as changes of states.

From the problem description and the resulting simplified timing diagrams it appears that the output transitions need only happen on the rising edge of a clock. This design will therefore change state on the rising edge of the clock. This means that the DRAM controller states will lag (be delayed by) half a clock period relative to the CPU bus state.

From the timing diagram we can identify the following combinations of outputs:

Output				State
hold	rama	ras	cas	Name
0	row	0	1	M1
0	col	0	0	M2
0	X	0	0	M3
1	X	1	0	R1
1	X	0	0	R2
1	X	1	1	RN
0	X	1	1	MN

where 'X' means the value driven on the DRAM's address inputs does not matter and 'RN' and 'MN' mean "refresh cycle next" and "memory cycle next" (see the section "Transition Conditions"). Since for a Moore state machine the outputs are a function of the state there must be at least this many states.

Since there must be some way to differentiate the second and third clock periods of the read cycle, there must be another state:

Output				State
hold	rama	ras	cas	Name
1	X	0	0	R3

Figure 1 shows the state of the control signal lines over three bus cycles and the corresponding controller states.

Define Transition Conditions

The next step in the design process is to define the state transition as a function of the current state and the inputs. From the description of the problem, the states always sequence through the M1 to M3 and the R1 to R3 states in order.

Since the HOLD output must be asserted in the cycle *before* the start of a refresh bus cycle, it's necessary to assert HOLD in the last state of either a memory or refresh bus cycle if the *next* cycle will be a refresh cycle. Thus we can design the state machine to transition from either M3 or R3 to MN or RN depending on the value of RFRSH.

There are alternative ways of choosing states and transition conditions, any of which will be correct as long as the requirements are met.

The transition conditions are thus:

current state	rfrsh	next state
M1	X	M2
M2	X	M3
R1	X	R2
R2	X	R3
M3	0	MN
R3	0	MN
M3	1	RN
R3	1	RN
MN	X	M1
RN	X	R1

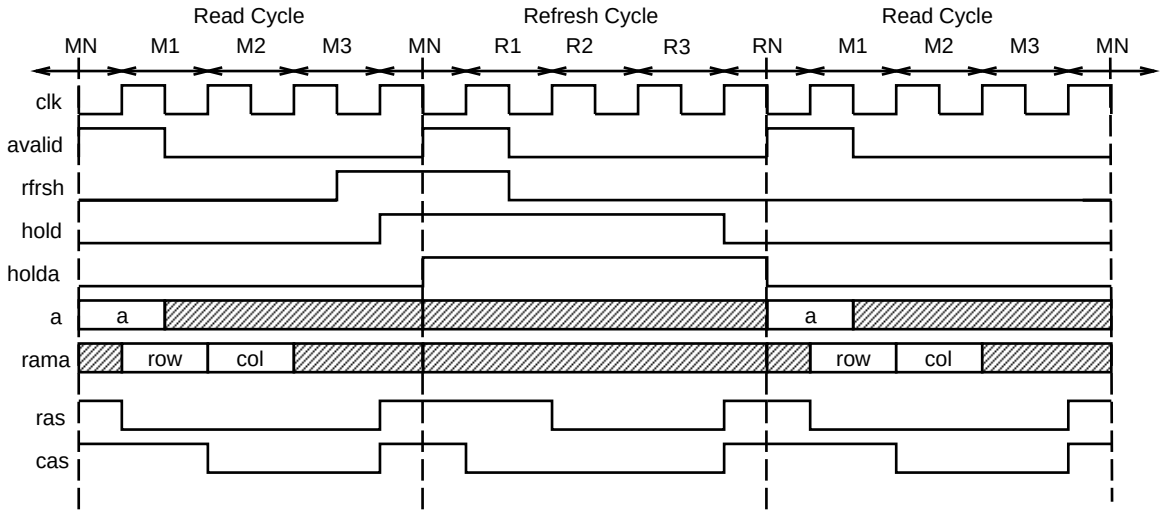


Figure 1: Controller States.

Define Outputs

The final step is to define the values of the outputs for each of the states. In this case the outputs corresponding to each state have already been defined.

Data Registers

A requirement of this design is that the 20-bit CPU address, A, must be latched at the start of state M1 so that the MS and LS halves can be output to the DRAM in states M1 and M2. This can be done with a 20-bit latch which is loaded at the start of M1 and a multiplexer to select either the MS or LS 10 bits of the latched address.

Initialization/Synchronization

A practical circuit must have a way of initializing its state. In this design the AVALID input allows us to detect the start of a bus cycle. If AVALID is detected at the start of any state the state must be M1 or R1 depending on the value of RFRSH.

VHDL Description

As usual, we split up the design into a combinational process for computing the state transitions and a sequential process to synchronize state transitions with the clock edge. We also use two processes to latch the CPU address (A) at the start of M1 and to output the correct half of the address on RAMA during states M1 and M2.

In the following architecture we use an enumeration type for the state variables to make the code easier to read. The synthesizer will assign values to the states.

```
-- Solution to ELEC 464 Assignment 7
-- Simple DRAM controller
-- Ed Casas, 96/11/6

entity dramc is
   port ( clk, rfrsh, avalid, holda : in bit ;
      a : in bit_vector (19 downto 0) ;
      hold, ras, cas : out bit ;
      rama : out bit_vector (9 downto 0) ) ;
end ;


architecture rtl of dramc is
   -- states
   type statetype is (M1,M2,M3,R1,R2,R3,MN,RN) ;
   -- latched CPU address
   signal tmpa : bit_vector (19 downto 0) ;
   -- next state
   signal state, nexts : statetype ;
begin

   -- compute next state
   process(state,avalid,rfrsh)
   begin

      if avalid = '1' then
        -- synchronize to start of CPU cycle
        if rfrsh = '1' then
           nexts <= R1 ;
        else
           nexts <= M1 ;
        end if ;
```

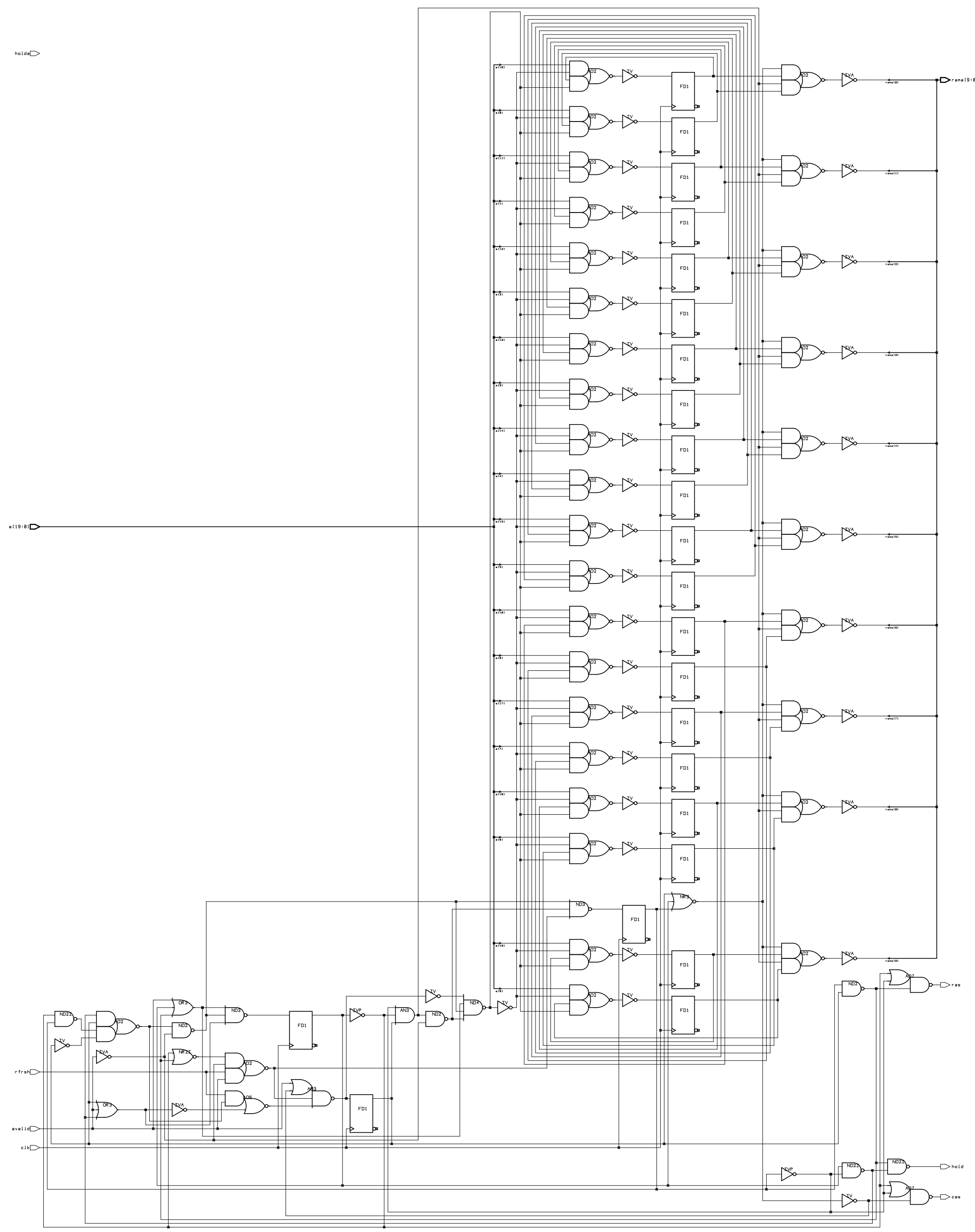
```
   else
      if state = M1 then
         nexts <= M2 ;
      elsif state = M2 then
         nexts <= M3 ;
      elsif state = M3 then
         if rfrsh = '1' then
            nexts <= RN ;
         else
            nexts <= MN ;
         end if ;
      elsif state = R1 then
         nexts <= R2 ;
      elsif state = R2 then
         nexts <= R3 ;
      elsif state = R3 then
         if rfrsh = '1' then
            nexts <= RN ;
         else
            nexts <= MN ;
         end if ;

      -- the following conditions should not
      -- happen since avalid should be true at
      -- start of R1 or M1

      elsif state = RN then
         nexts <= R1 ;
      else -- state = MN then
         nexts <= M1 ;
      end if ;
   end if ;
end process ;

--  outputs
process(state)
begin
   case state is
      when M1 => hold <= '0' ;
                 ras  <= '0' ; cas <= '1' ;
      when M2 => hold <= '0' ;
                 ras  <= '0' ; cas <= '0' ;
      when M3 => hold <= '0' ;
                 ras  <= '0' ; cas <= '0' ;

      when R1 => hold <= '1' ;
                 ras  <= '1' ; cas <= '0' ;
      when R2 => hold <= '1' ;
                 ras  <= '0' ; cas <= '0' ;
      when R3 => hold <= '1' ;
                 ras  <= '0' ; cas <= '0' ;
      when RN => hold <= '1' ;
                 ras  <= '1' ; cas <= '1' ;
      when MN => hold <= '0' ;
                 ras  <= '1' ; cas <= '1' ;
   end case ;
end process ;


-- state transitions
process(clk,nexts)
begin
   if clk'event and clk='1' then
      state <= nexts ;
   end if ;
end process ;

-- output muxed address to DRAM in M1/M2
   process(state,tmpa)
   begin
     case state is
        when M1 => rama <= tmpa (19 downto 10) ;
        when M2 => rama <= tmpa ( 9 downto  0) ;
         -- otherwise "don't care"
        when others => rama <= "0000000000" ;
     end case ;
   end process ;

   -- latch CPU address at start of M1
   process(clk,nexts,a)
   begin
      if clk'event and clk='1' then
         if nexts=M1 then
            tmpa <= a ;
         end if ;
      end if ;
   end process ;

end rtl ;
```

Simulation

The VHDL description was analyzed and run with the test input shown in Figure 1. The results are not given here to conserve space.

Synthesis

The synthesized schematic is given on the next page.


holda
a[19:0]
rfrsh
avalid
clk
rana[9:0]
ras
hold
cas
FD1