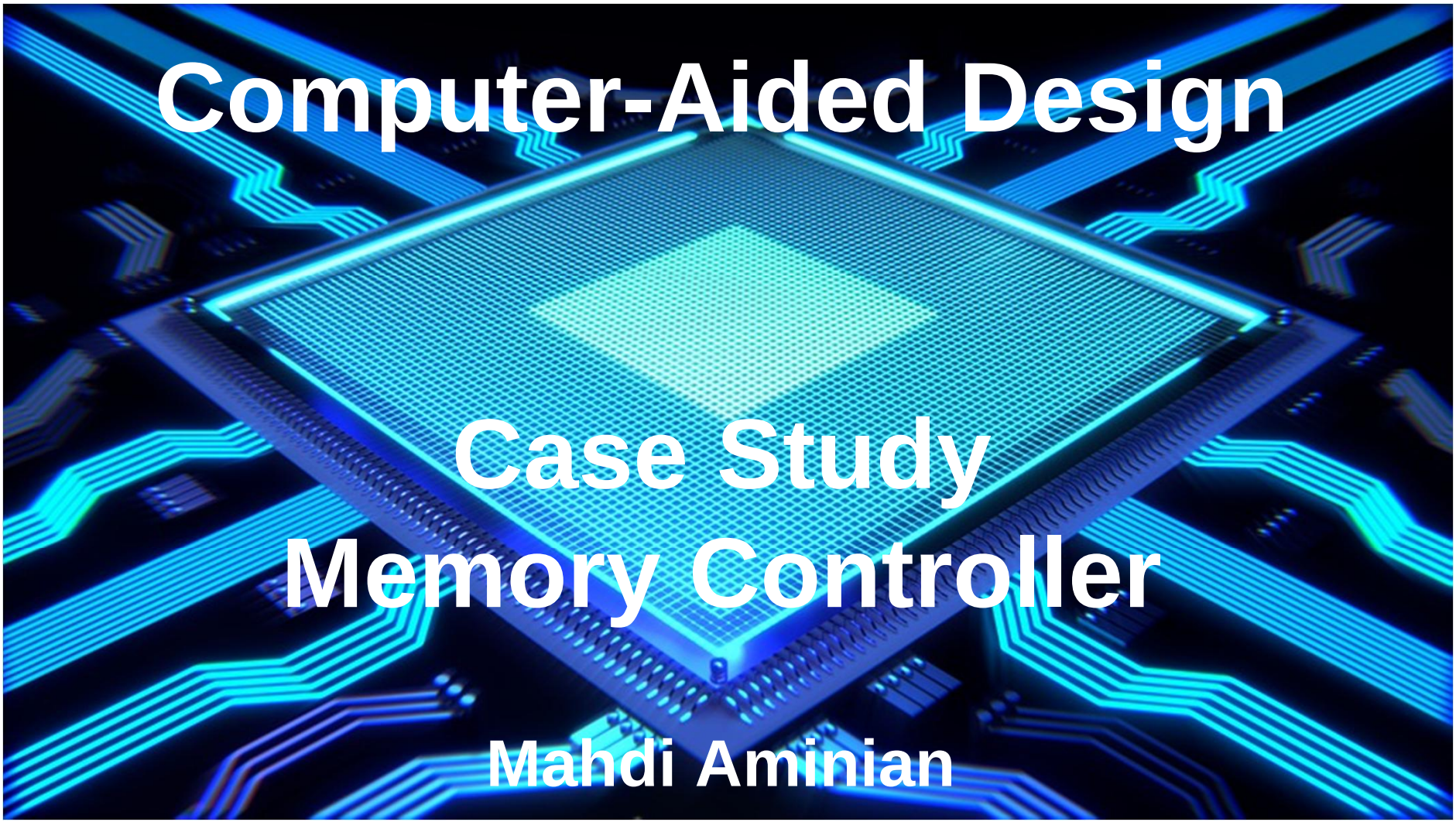


Computer-Aided Design

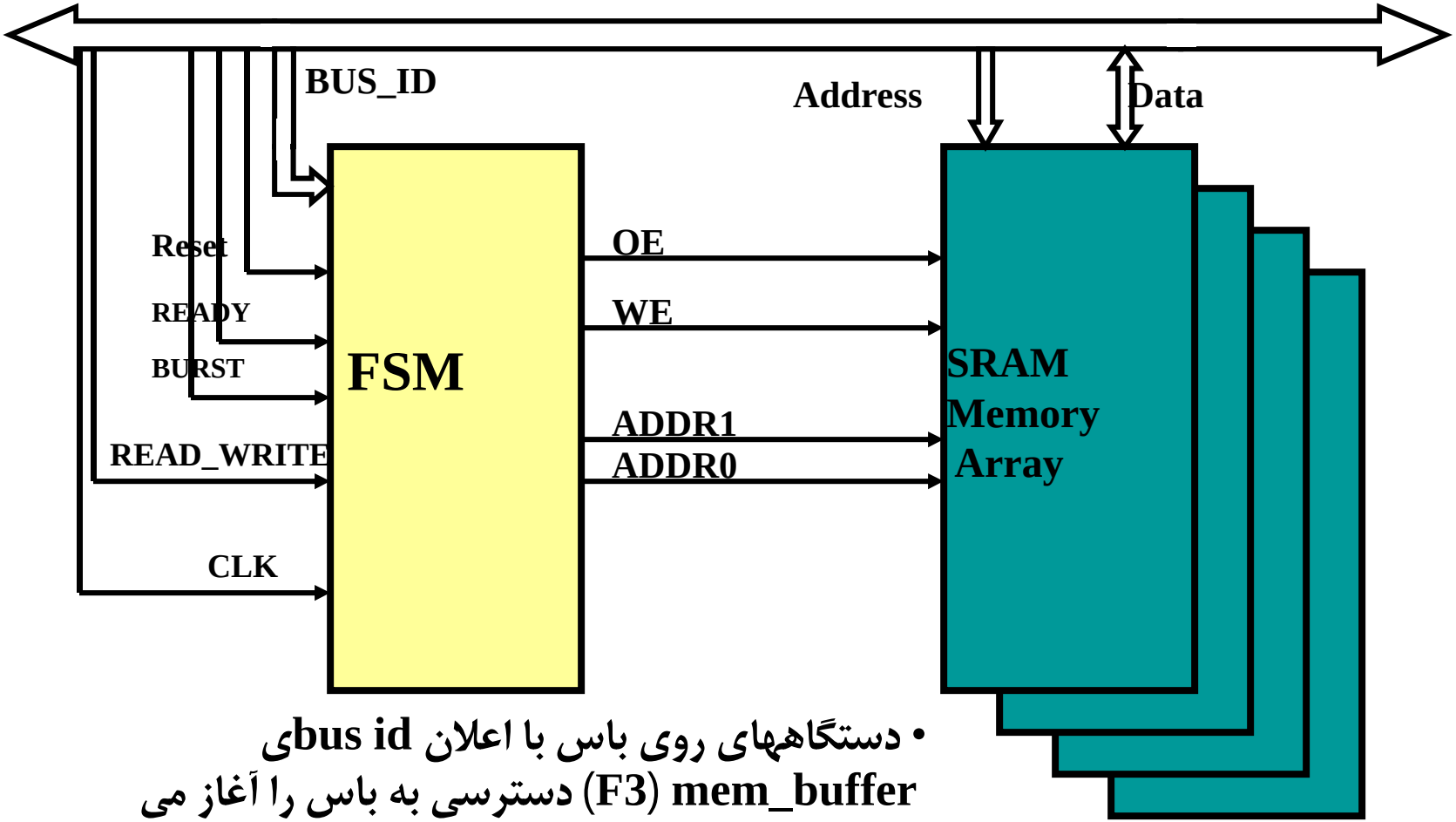
Case Study
Memory Controller

Mahdi Aminian

The University Of Guilan
Rasht - Iran

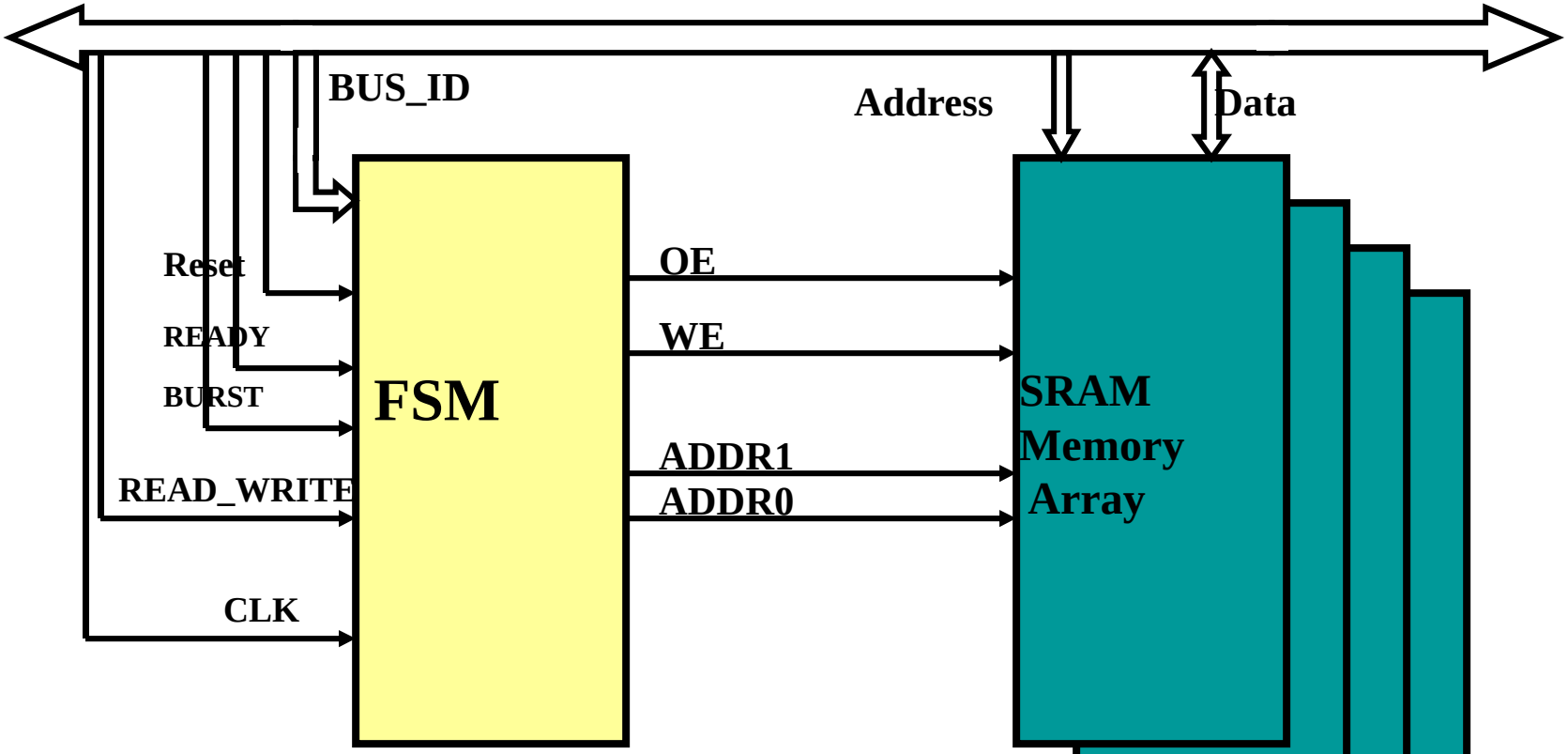
Some slides courtesy of:
- “Hardware Systems Modeling”, A. Vachoux, EPFL
- CAD slides from Dr. Saheb Zamani

Memory Controller



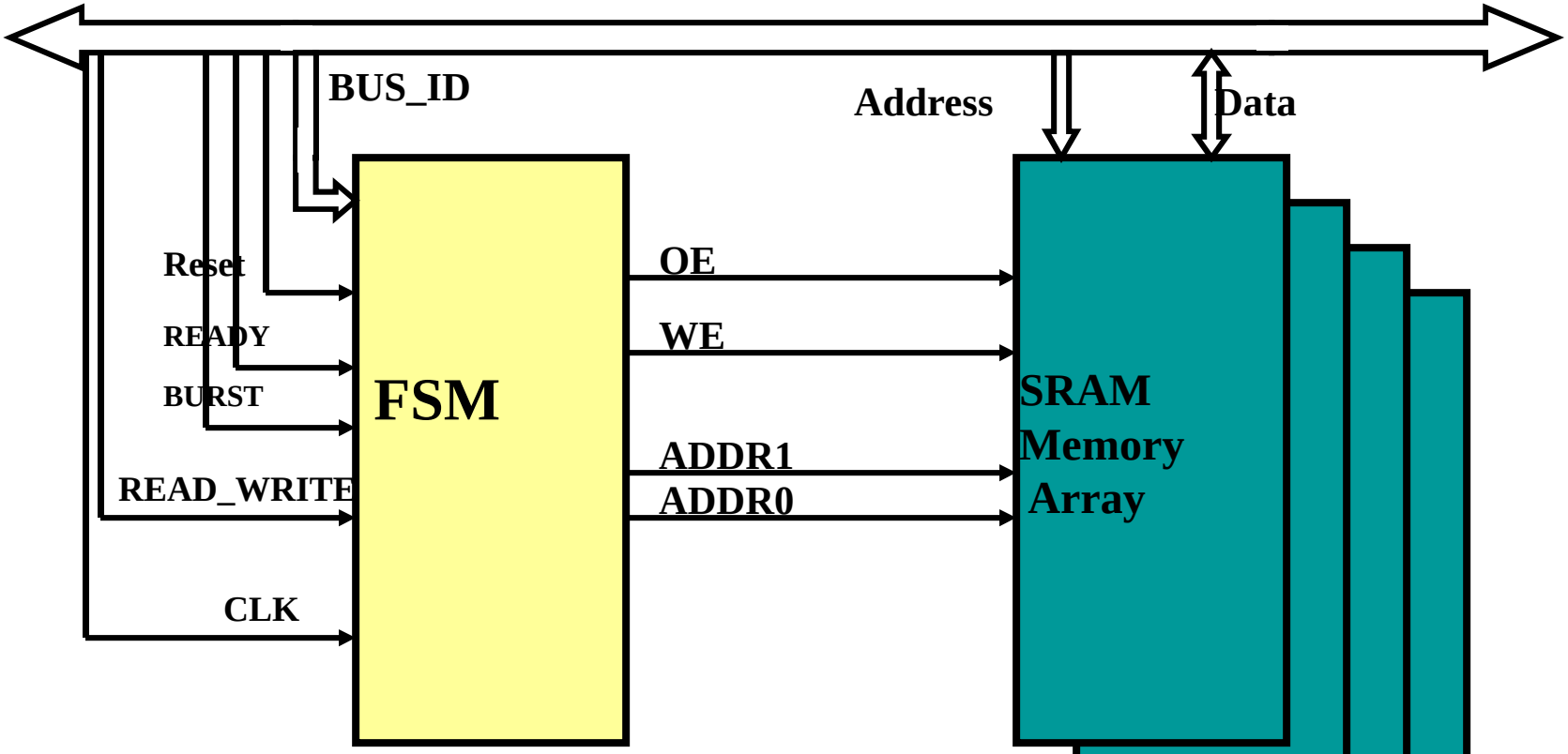
- دستگاههای روی باس با اعلان bus id ی mem_buffer (F3) دسترسی به باس را آغاز می کنند.

Memory Controller



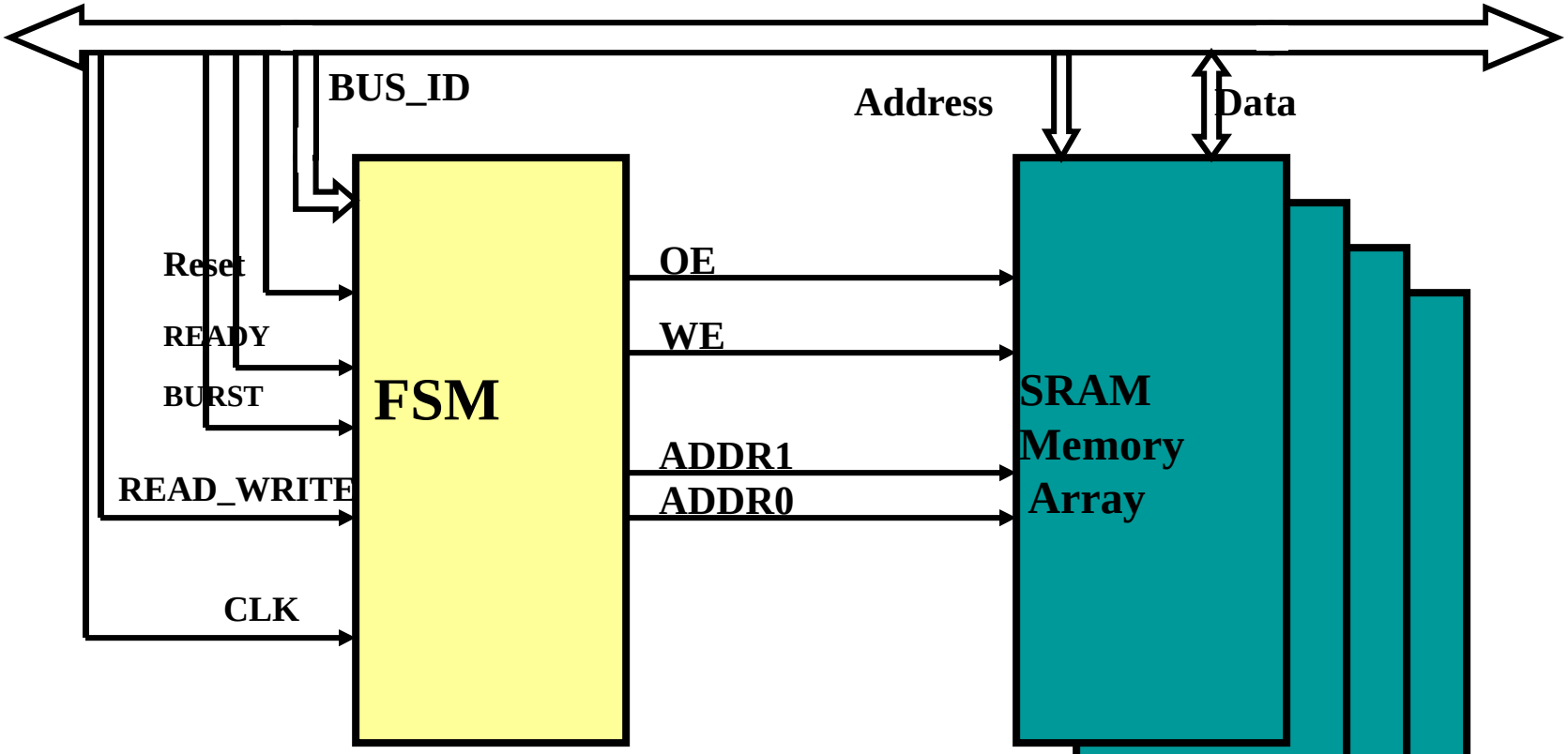
- یک سیکل بعد, READ_WRITE = ‘1’ می شود تا بگوید که یک خواندن می خواهد انجام شود (یا ‘0’ برای نوشتن).

Memory Controller



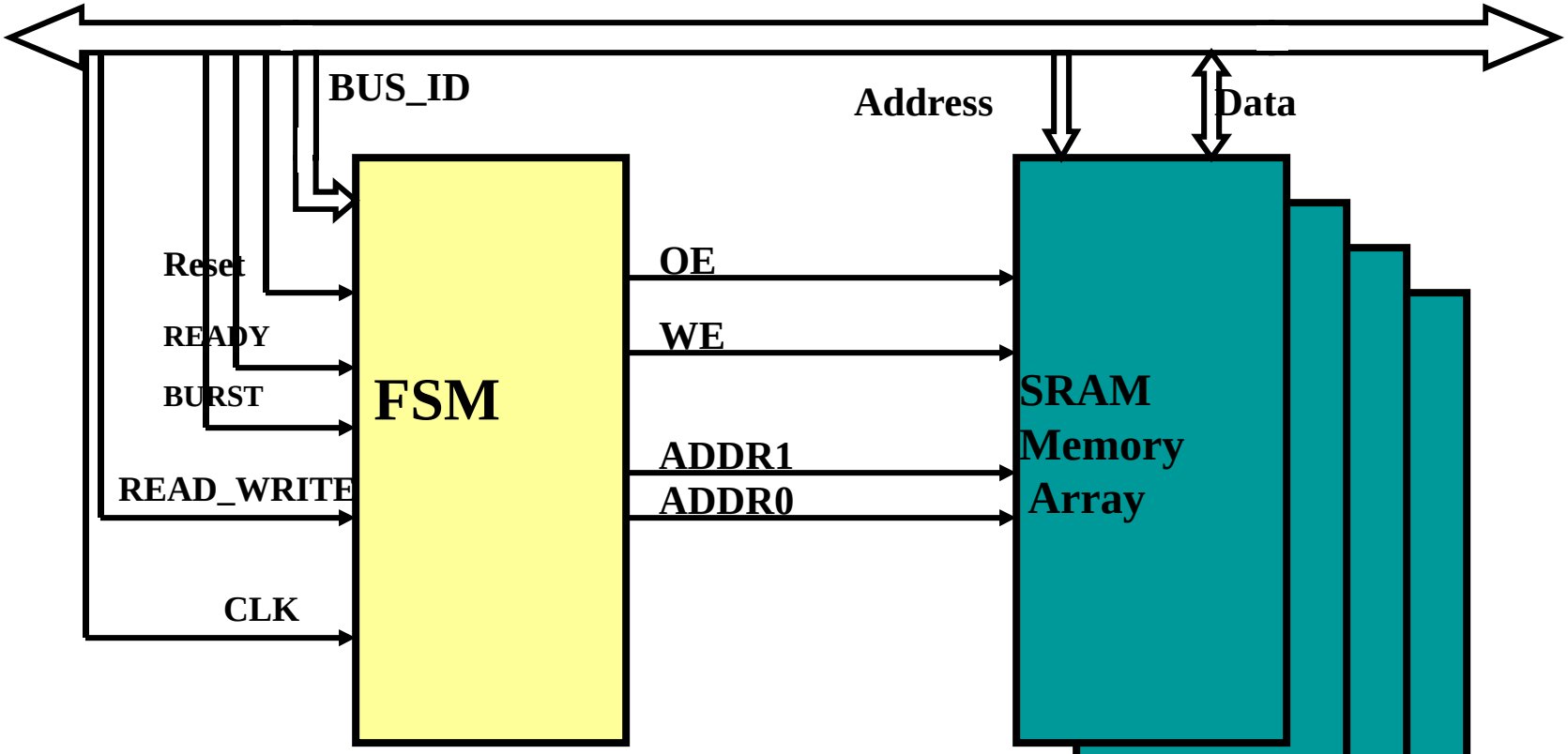
•برای خواندن ممکن است 4کلمه ای (burst read)باشد: باید در مدت اولین سیکل, burst فعال باشد.

Memory Controller



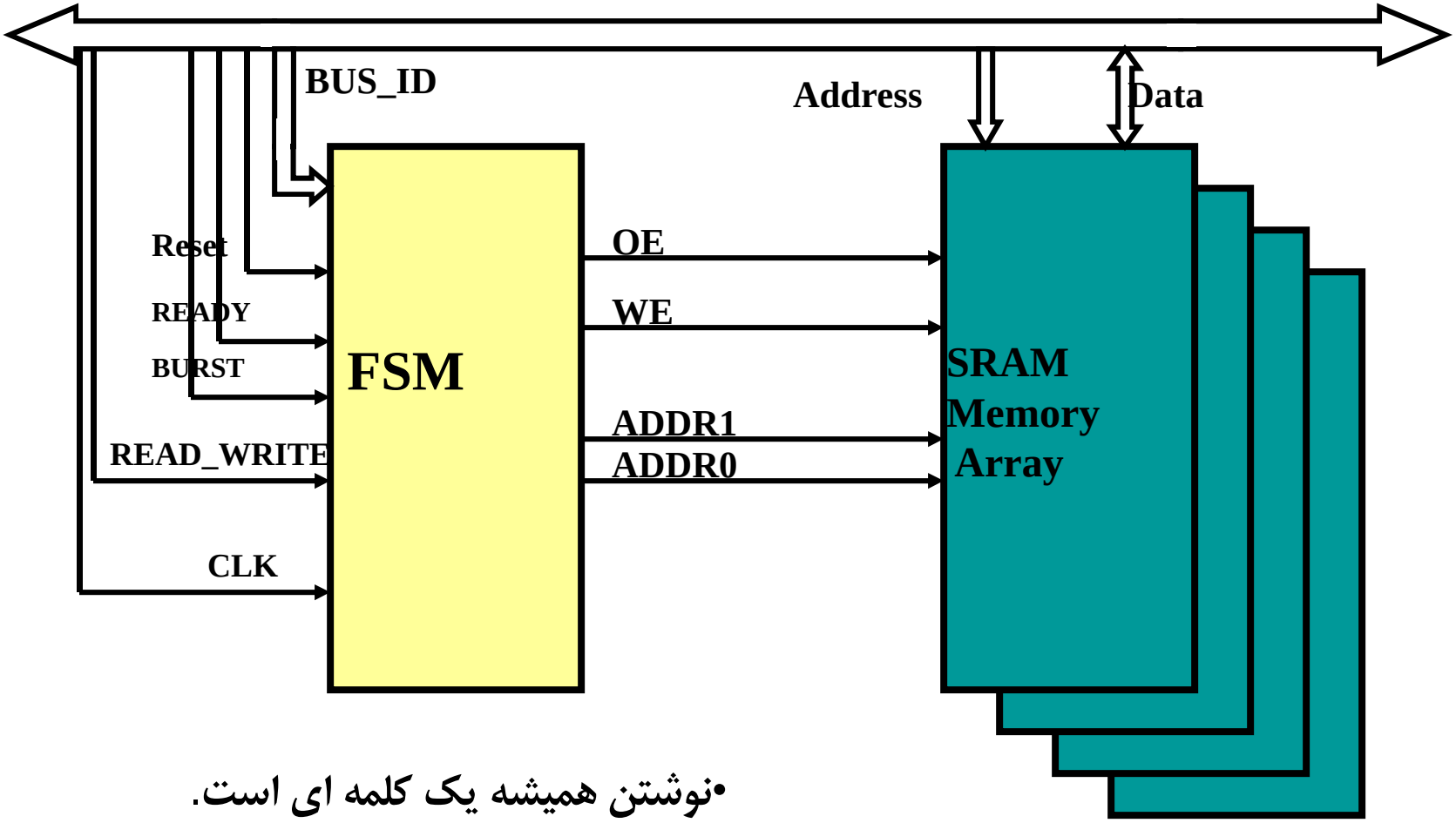
•کنترلر به 4 محل از بافر دسترسی می یابد (به محلهای بعدی بعد از فعال کردنهای متوالی ready دسترسی می یابد).

Memory Controller



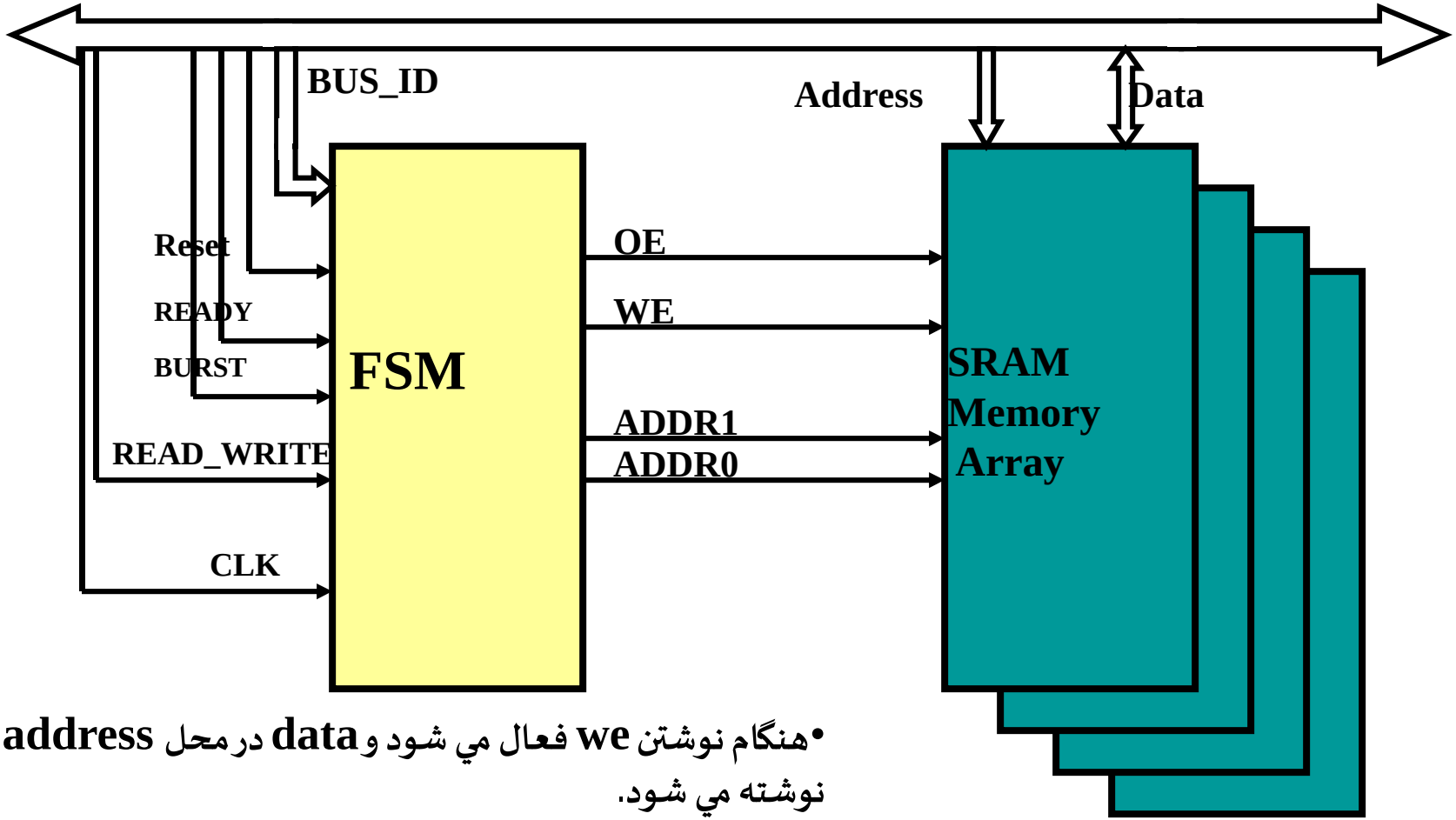
• کنترلر oe را برای mem_buffer در طول خواندن فعال می کند و دو بایت پایین آدرس را در حالت burst افزایش می دهد.

Memory Controller



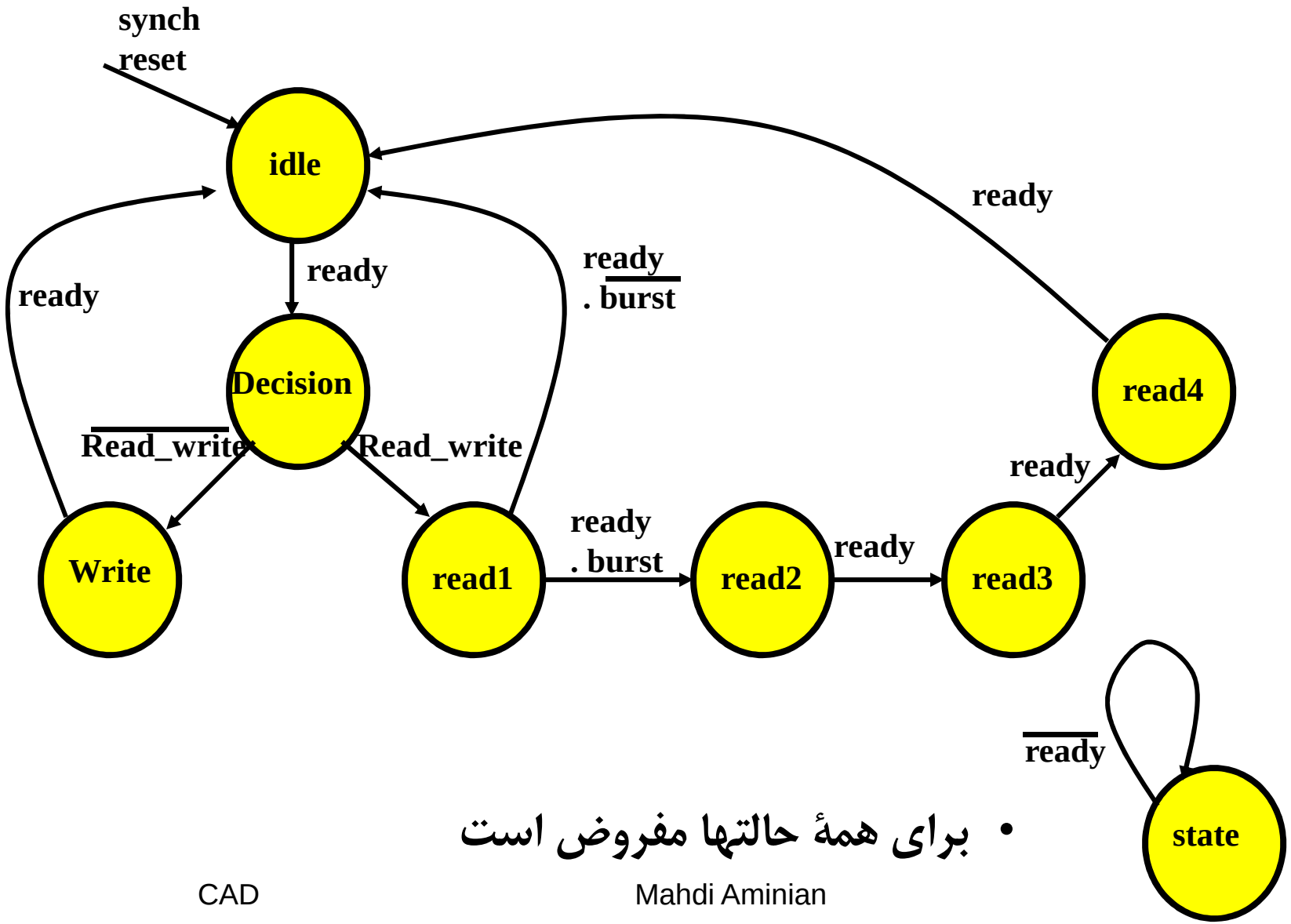
•نوشتن همیشه یک کلمه ای است.

Memory Controller



- هنگام نوشتن **we** فعال مي شود و **data** در محل **address** نوشته مي شود.
- خواندن و نوشتن با اعلان **ready** خاتمه مي يابد.

Memory Controller



- برای همهٔ حالتها مفروض است

VHDL Code (2-process)

```
library ieee;
use ieee.std_logic_1164.all;
entity memory_controller is
   port (
      reset, read_write, ready, burst, clk    : in std_logic;
      bus_id                                  : in std_logic_vector(7 downto 0);
      oe, we                                  : out std_logic;
      addr                                    : out std_logic_vector(1 downto 0));
end memory_controller;

architecture state_machine of memory_controller is
            type StateType is (idle, decision, read1, read2, read3, read4, write);
            signal present_state, next_state : StateType;
```

VHDL Code

```
begin
state_comb:process(reset, bus_id, present_state, burst, read_write, ready)
begin
  if (reset = '1') then
    oe <= ‘0'; we <= '-'; addr <= "--";
    next_state <= idle;
  else
    case present_state is
      when idle    =>    oe <= '0'; we <= '0'; addr <= "00";
        if (bus_id = "11110011“ and ready = ‘1’) then
          next_state <= decision;
        else
          next_state <= idle;
        end if;
      when decision=>      oe <= '0'; we <= '0'; addr <= "00";
        if (read_write = '1') then
          next_state <= read1;
        else                    --read_write='0'
          next_state <= write;
        end if;
```

- **Don’t cares assigned to outputs à optimized**

In every case, a signal must be assigned to the outputs; otherwise, unwanted latches.

VHDL Code

```
when read1   =>      oe <= '1'; we <= '0'; addr <= "00";
  if (ready = '0') then
    next_state <= read1;
  elsif (burst = '0') then
    next_state <= idle;
  else
    next_state <= read2;
  end if;
when read2   =>      oe <= '1'; we <= '0'; addr <= "01";
  if (ready = '1') then
    next_state <= read3;
  else
    next_state <= read2;
  end if;
when read3   =>      oe <= '1'; we <= '0'; addr <= "10";
  if (ready = '1') then
    next_state <= read4;
 else
    next_state <= read3;
 end if;
```

VHDL Code

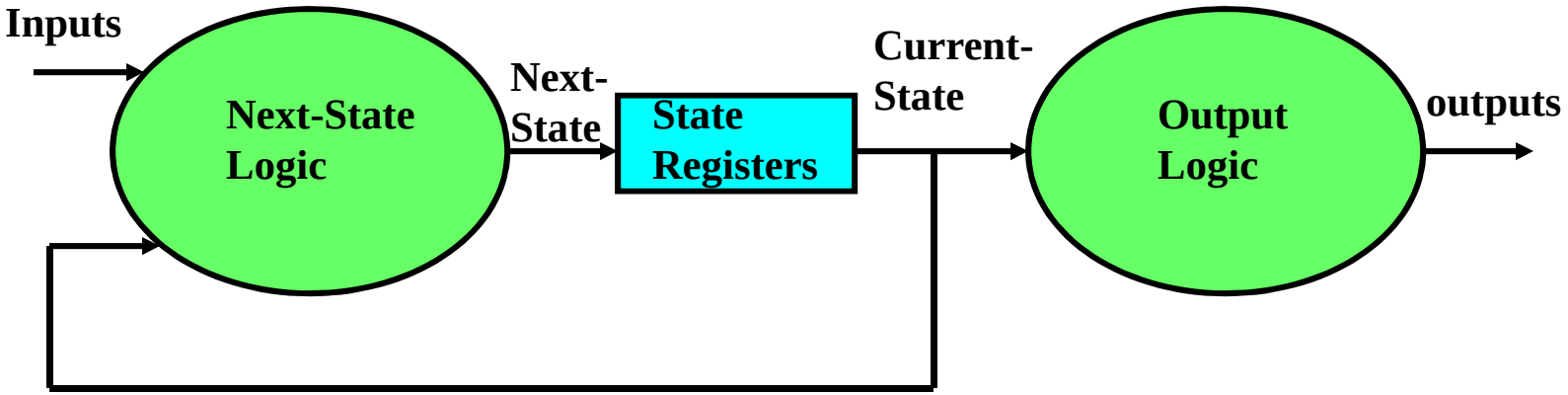
```
    when read4   =>      oe <= '1'; we <= '0'; addr <= "11";
      if (ready = '1') then
        next_state <= idle;
      else
        next_state <= read4;
      end if;
    when write   =>      oe <= '0'; we <= '1'; addr <= "00";
      if (ready = '1') then
        next_state <= idle;
      else
        next_state <= write;
      end if;
   end case;
  end if;
end process state_comb;
```

VHDL Code

```
state_clocked:process(clk) begin
    if rising_edge(clk) then
        present_state <= next_state;
    end if;
end process state_clocked;

end;
```

تولید خروجیها در ماشینهای Moore

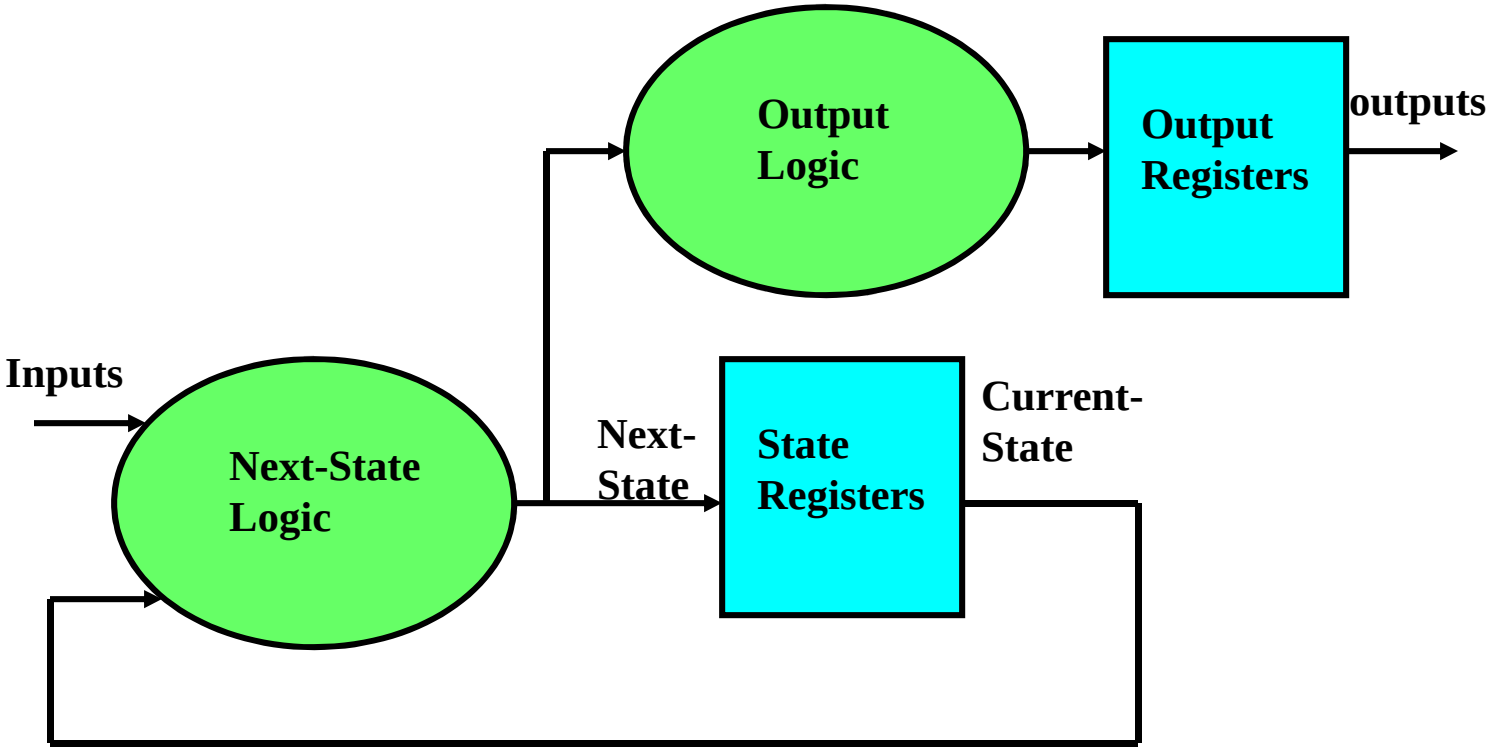
1) خروجیهایی که از بیتهای حالت به طور ترکیبی دیکد شده اند: (کد قبل)



- مزایا:
  - گویایی کد
  - نگهداری آسانتر
- اشکال:
  - کند

تولید خروجیها در ماشینهای Moore

2) خروجیهایی که از رجیسترهای خروجی به طور موازی دیکد می شوند:



- انتساب به خروجیها باید در خارج ازپروسسی که انتقال حالات در آن تعریف می شود انجام گیرد.

```
architecture state_machine of memory_controller is
   type StateType is (idle, decision, read1, read2, read3, read4, write);
   signal present_state, next_state : StateType;
   signal addr_d: std_logic_vector(1 downto 0);          -- D-input to addr
f-flops
begin
state_comb:process(bus_id, present_state, burst, read_write, ready)
begin
   case present_state is                          -- addr outputs not defined
      when idle    =>       oe <= '0'; we <= '0';       -- addr is absent.
         if (bus_id = "11110011“ and ready = ‘1’) then
            next_state <= decision;
         else
            next_state <= idle;
         end if;
      when decision=>        oe <= '0'; we <= '0';
         if (read_write = '1') then
            next_state <= read1;
         else                  --read_write='0'
            next_state <= write;
         end if;
```

- فرض: فقط برای addr این کار را انجام می دهیم
  (برای we و oe مشکل زمانی نداریم)

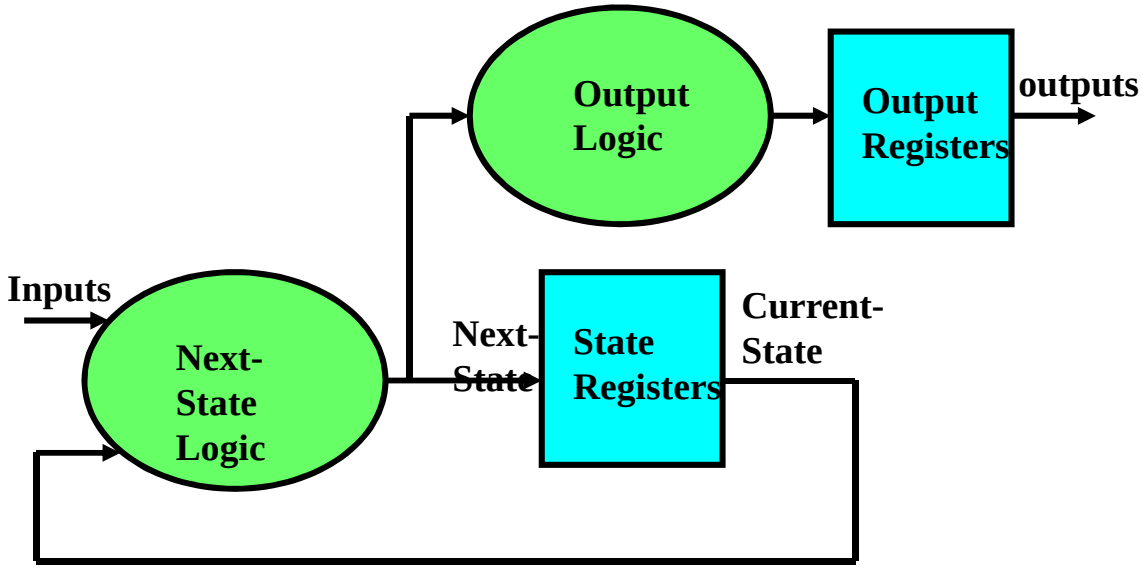
```
when read1   =>          oe <= '1'; we <= '0';
   if (ready = '0') then
      next_state <= read1;
   elsif (burst = '0') then
      next_state <= idle;
   else
      next_state <= read2;
   end if;
when read2   =>          oe <= '1'; we <= '0';
   if (ready = '1') then
      next_state <= read3;
   else
      next_state <= read2;
   end if;
when read3   =>          oe <= '1'; we <= '0';
   if (ready = '1') then
      next_state <= read4;
   else
      next_state <= read3;
   end if;
```

```
    when read4   =>          oe <= '1'; we <= '0';
        if (ready = '1') then
            next_state <= idle;
        else
            next_state <= read4;
        end if;
     when write   =>          oe <= '0'; we <= '1';
        if (ready = '1') then
            next_state <= idle;
        else
            next_state <= write;
        end if;
  end case;
end process state_comb;

 with next_state select              -- D-input to addr flip-flops
     addr_d <= "01" when read2,          -- defined here.
           "10" when read3,
           "11" when read4,
           "00" when others;
```

```
state_clocked:process(clk, reset) begin
    if reset = '1' then
        present_state <= idle;
        addr <= "00";           -- asynchronous reset for addr flops
    elsif rising_edge(clk) then
        present_state <= next_state;
        addr <= addr_d;         -- value of addr_d stored in addr
    end if;
end process state_clocked;

end state_machine;
```

تولید خروجیها در ماشینهای Moore

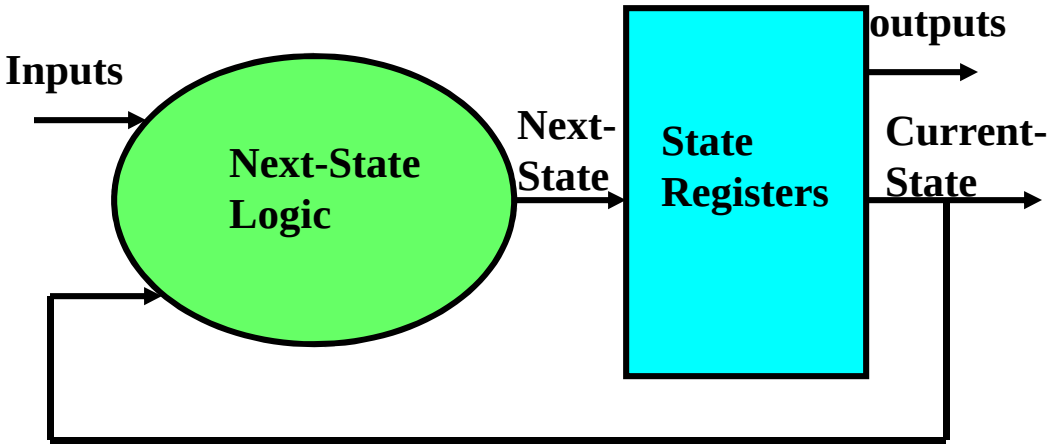
- مشکلات:
  - 2 FF اضافه.
- برای انتشار بیتهای حالت به FFهای addr, از دو مدار ترکیبی رد می شود (اگر در PLD از 2 سلول استفاده کند می تواند فرکانس ماکزیمم را محدود کند)



تولید خروجیها در ماشینهای Moore

3) خروجیهایی که مستقیماً در بیتهای حالت انکد شده اند (**Medvedev**)

(مانند شمارنده ها):



- **State encoding باید به دقت انجام شود.**
- **FFهای بیشتری لازم دارد.**
- **برای خروجی به مدار ترکیبی نیاز ندارد (سرعت بیشتر).**

State Encoding

- فرض: فقط برای addr این کار را انجام می دهیم (برای we و oe مشکل زمانی نداریم)

	Addr(1)	Addr(0)	s1	s2
Idle	0	0	0	0
decision	0	0	0	1
Read1	0	0	1	0
Read2	0	1	0	0
Read3	1	0	0	0
Read4	1	1	0	0
Write	0	0	1	1

State Encoding

- اگر برای we و oe هم بخواهیم به همین صورت encode کنیم:

	Addr(1)	Addr(0)	oe	we	s0
Idle	0	0	0	0	0
decision	0	0	0	0	1
Read1	0	0	1	0	0
Read2	0	1	1	0	0
Read3	1	0	1	0	0
Read4	1	1	1	0	0
Write	0	0	0	1	0

VHDL Code

```
architecture state_machine of memory_controller is
-- state signal is a std_logic_vector rather than an enumeration type
   signal state : std_logic_vector(4 downto 0);
   constant idle    : std_logic_vector(4 downto 0) := "00000";
   constant decision: std_logic_vector(4 downto 0) := "00001";
   constant read1   : std_logic_vector(4 downto 0) := "00100";
   constant read2   : std_logic_vector(4 downto 0) := "01100";
   constant read3   : std_logic_vector(4 downto 0) := "10100";
   constant read4   : std_logic_vector(4 downto 0) := "11100";
   constant write   : std_logic_vector(4 downto 0) := "00010";
begin
state_tr:process(reset, clk)
begin      -- One-process FSM
   if reset = '1' then
      state <= idle;
   elsif rising_edge(clk) then
      case state is                -- outputs not defined here
        when idle    =>
          if (bus_id = "11110011") then
             state <= decision;
          end if;                  -- no else; implicit memory
```

VHDL Code

```
when decision=>
  if (read_write = '1') then
    state <= read1;
  else                  --read_write='0'
    state <= write;
  end if;
when read1   =>
  if (ready = '0') then
    state <= read1;
  elsif (burst = '0') then
    state <= idle;
  else
    state <= read2;
  end if;
when read2   =>
  if (ready = '1') then
    state <= read3;
  end if;               -- no else; implicit memory
```

```
        when read3   =>
          if (ready = '1') then
            state <= read4;
          end if;              -- no else; implicit memory
        when read4   =>
          if (ready = '1') then
            state <= idle;
          end if;              -- no else; implicit memory
        when write   =>
          if (ready = '1') then
            state <= idle;
          end if;              -- no else; implicit memory
        when others =>
            state <= "-----";        -- don't care if undefined state
             end case;
             end if;
end process state_tr;

-- outputs associated with register values
  we <= state(1);
  oe <= state(2);
  addr <= state(4 downto 3);
end state_machine;
```