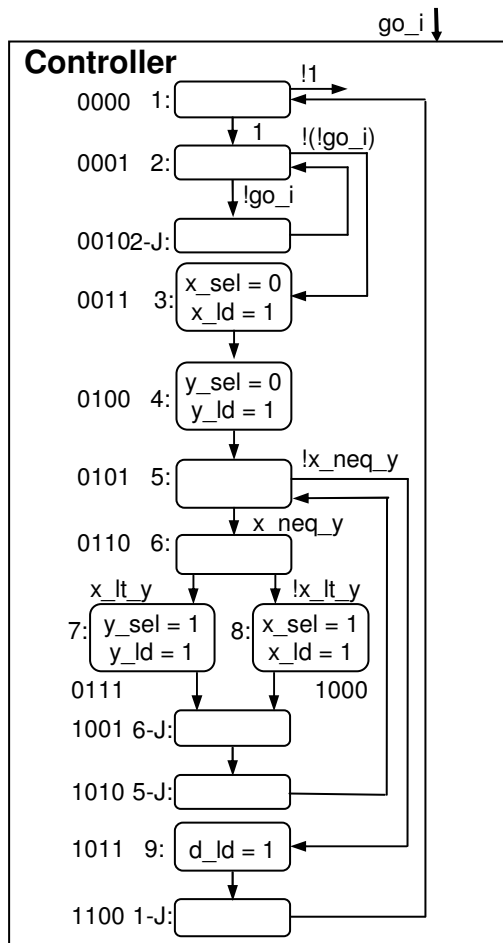


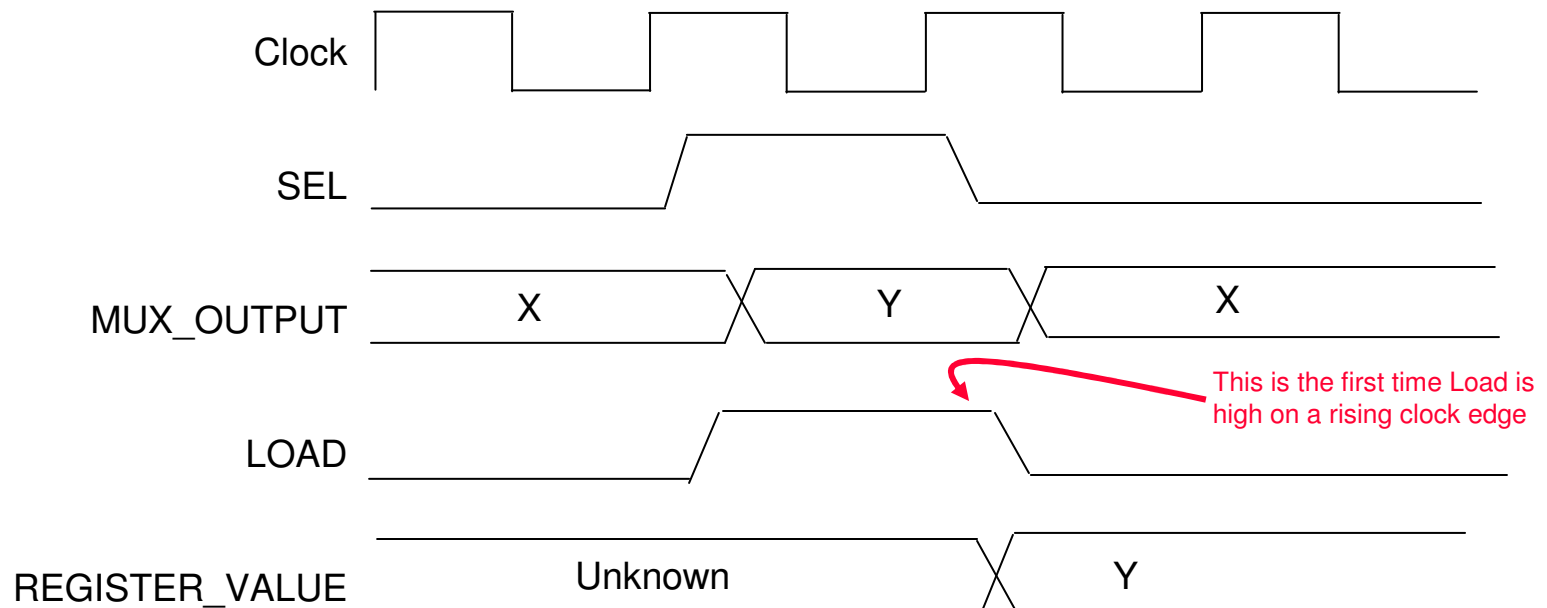
# A Note Timing...



If we are selecting an output, and loading in the same state, how do we know the correct data is getting loaded?

Answer: Although we set the load signal high in one state, the register does not actually read until the next clock cycle.

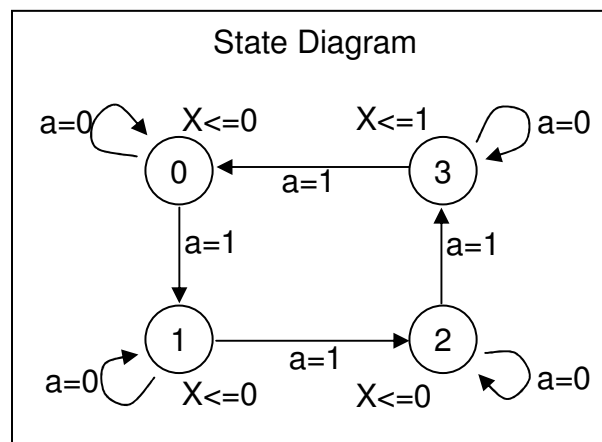
# A Note Timing...



# State Machines in VHDL...

---

Remember this state machine? You are going to see it in lab too.



# State Machines in VHDL...

---

First, your entity:

```
entity simp_statemachine is
```

```
port (  
  CLK : in std_logic;  
  A   : in std_logic;  
  RESET : in std_logic;  
  X   : out std_logic);
```

```
end simp_statemachine;
```

# State Machines in VHDL...

---

Then your architecture:

*architecture RTL of simp\_statemachine is*

*-- This is where we define all the different values our state signal can take*

*type STATE\_TYPE is (HOME, LOW\_STATE1, LOW\_STATE2,  
HIGH\_STATE1);*

*-- This is what is stored in the state register*

*signal CURRENT\_STATE : STATE\_TYPE;*

*-- This value is determined by the combinational logic*

*signal NEXT\_STATE : STATE\_TYPE;*

*begin -- RTL*

# State Machines in VHDL...

---

```
COMBINATIONAL_STATE_LOGIC : process (CURRENT_STATE, A)  
begin -- process COMBINATIONAL_STATE_LOGIC  
  case CURRENT_STATE is  
    when HOME    =>  
      if A = '1' then  
        NEXT_STATE <= LOW_STATE1;  
      else  
        NEXT_STATE <= HOME;  
      end if;  
    when LOW_STATE1 =>  
      if A = '1' then  
        NEXT_STATE <= LOW_STATE2;  
      else  
        NEXT_STATE <= LOW_STATE1;  
      end if;
```

# State Machines in VHDL...

---

```
when LOW_STATE2 =>  
  if A = '1' then  
    NEXT_STATE <= HIGH_STATE1;  
  else  
    NEXT_STATE <= LOW_STATE2;  
  end if;
```

```
when HIGH_STATE1 =>  
  if A = '1' then  
    NEXT_STATE <= HOME;  
  else  
    NEXT_STATE <= HIGH_STATE1;  
  end if;
```

```
when others => null;      -- You always have to have a "when  
                          -- others" statement
```

```
end case;  
end process COMBINATIONAL_STATE_LOGIC;
```

# State Machines in VHDL...

---

Make your state register:

```
STATE_REGISTER : process (CLK, RESET, NEXT_STATE)  
begin -- process STATE_REGISTER  
  if CLK'event and CLK = '1' then -- rising clock edge  
    if RESET = '1' then  
      CURRENT_STATE <= HOME;  
    else  
      CURRENT_STATE <= NEXT_STATE;  
    end if;  
  end if;  
end process STATE_REGISTER;
```

# State Machines in VHDL...

---

**Create your logic for your output and close the process:**

```
X <= '1' when CURRENT_STATE = HIGH_STATE1 else '0';  
end RTL;
```

# Massive Team ICE!

---

```
while (1){
  a = inputA
  b = inputB
  z = 1

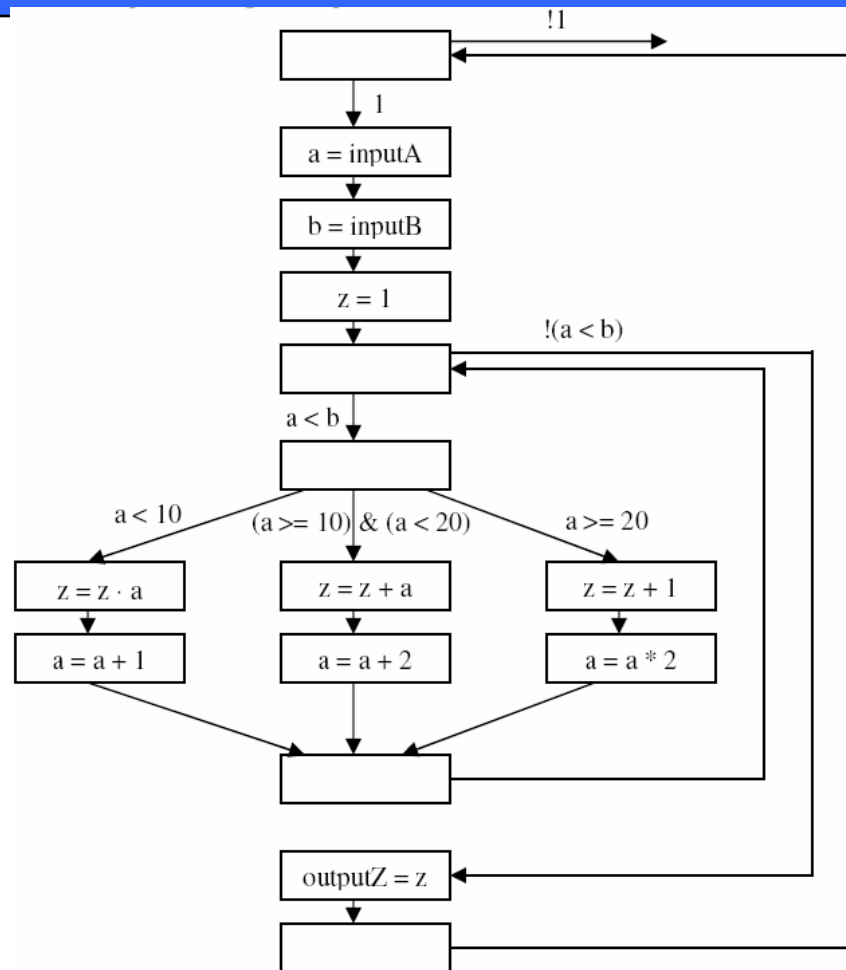
  while (a < b){

    if (a < 10){
      z = z * a
      a = a + 1
    }
    else if ( ( a >= 10) and (a < 20) )
      z = z + a
      a = a + 2
    else
      z = z + 1
      a = a * 2
    }
  }
  outputZ = z
}
```

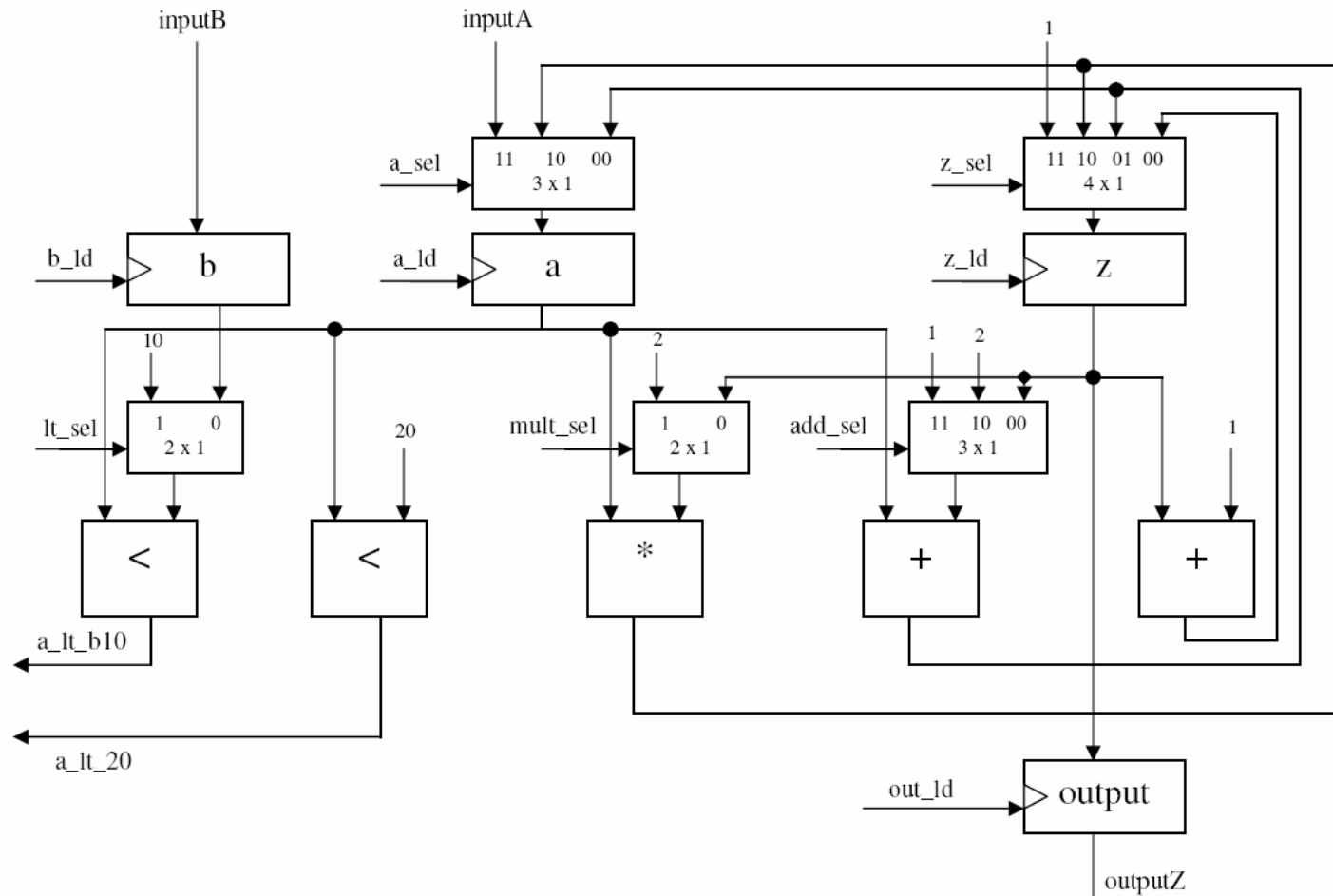
Get in teams of four.

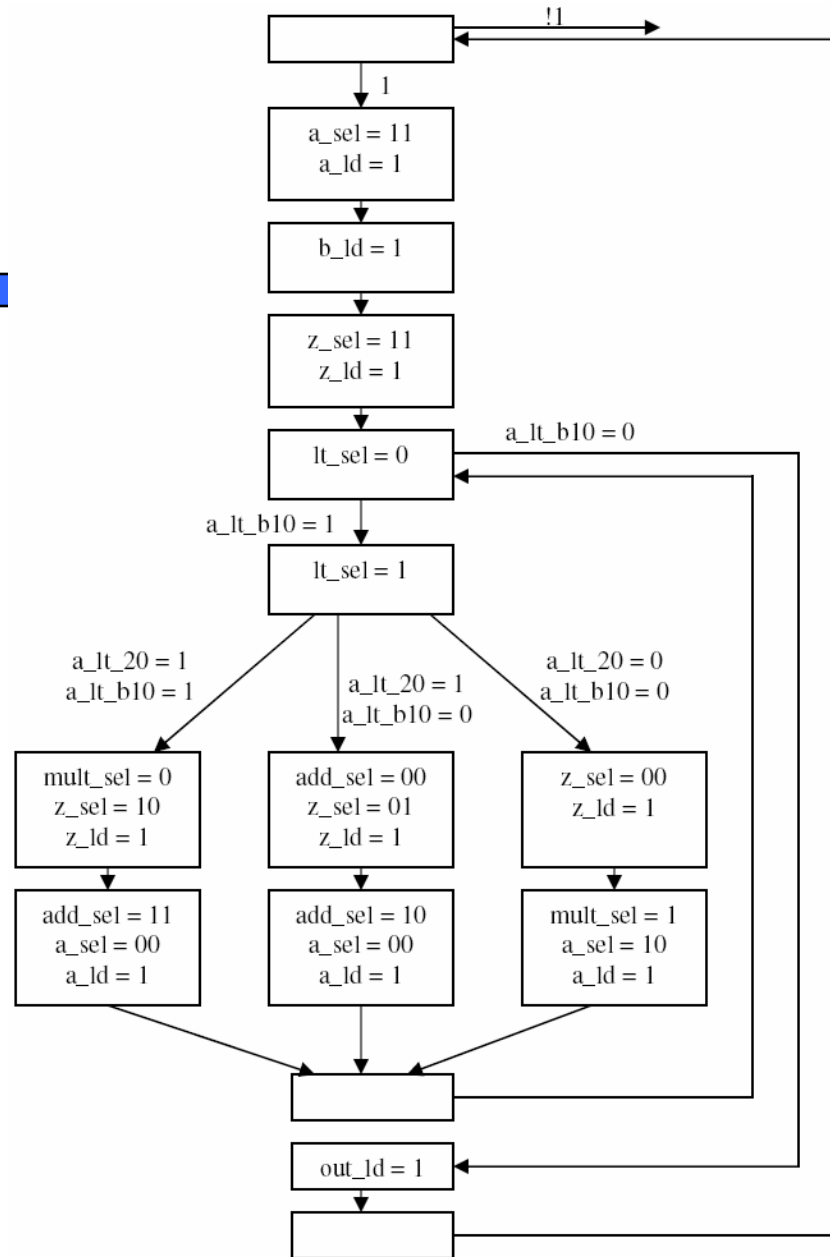
- First: Create an FSMD for this algorithm
- Second: Make a datapath
- Third: Make a matching FSM

# Massive Team ICE!



# Massive Team ICE!





# To Do before next Tuesday

---

- Finish Homework 2
- Keep an eye out for your scanned homework and quiz in your CS e-mail account.