香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

Lecture 04: Combinational Circuit and Sequential Circuit

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Recall: Concurrent vs. Sequential

- **Concurrent Statement**
  - ① Statements inside the architecture body can be executed concurrently, except statements enclosed by a **`process`**.
  - ② Every statement will be executed once whenever any signal in the statement changes.
- **Sequential Statement**
  - ① Statements within a **`process`** are executed sequentially, and the result is obtained when the process is complete.
  - ② **`process(sensitivity list)`**: When one or more signals in the sensitivity list change state, the process executes once.
  - ③ A **`process`** can be treated as one concurrent statement in the architecture body.

Recall: Concurrent vs. Sequential

Concurrent Statement

when-else

```
b <= "1000" when a = "00" else
     "0100" when a = "01" else
     "0010" when a = "10" else
     "0001" when a = "11";
```

with-select-when

```
with a select
b <= "1000" when "00",
     "0100" when "01",
     "0010" when "10",
     "0001" when "11";
```

Sequential Statement

if-then-else

```
if    a = "00" then b <= "1000"
elsif a = "01" then b <= "1000"
elsif a = "10" then b <= "1000"
else                  b <= "1000"
end if;
```

case-when

```
case a is
    when "00" => b <= "1000";
    when "01" => b <= "0100";
    when "10" => b <= "0010";
    when others => b <= "0001";
end case;
```

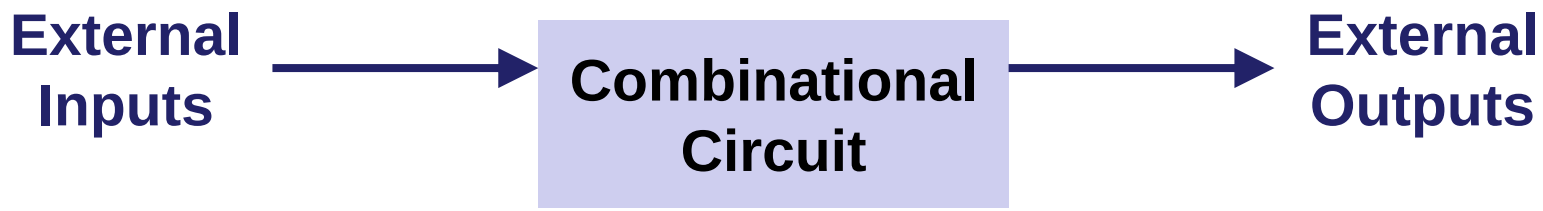
Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Combinational Circuit

- **Combinational Circuit**: **no memory**
  - Outputs are a function of the *present* inputs only.
    - As soon as inputs change, the values of previous outputs are lost.
    - It has no internal state (i.e., has no memory).
    - Common Examples: Full/Half Adders (*Lab01*), Encoders/Decoders (*Lab02*), Multiplexers (*Lab03*), Bi-directional Bus (*Lec04*), etc.
  - **Rule**: You can build a combinational circuit using **either** concurrent **or** sequential (i.e., `process`) statements.



Combinational Logic as a Process

- Consider a simple combinational logic:

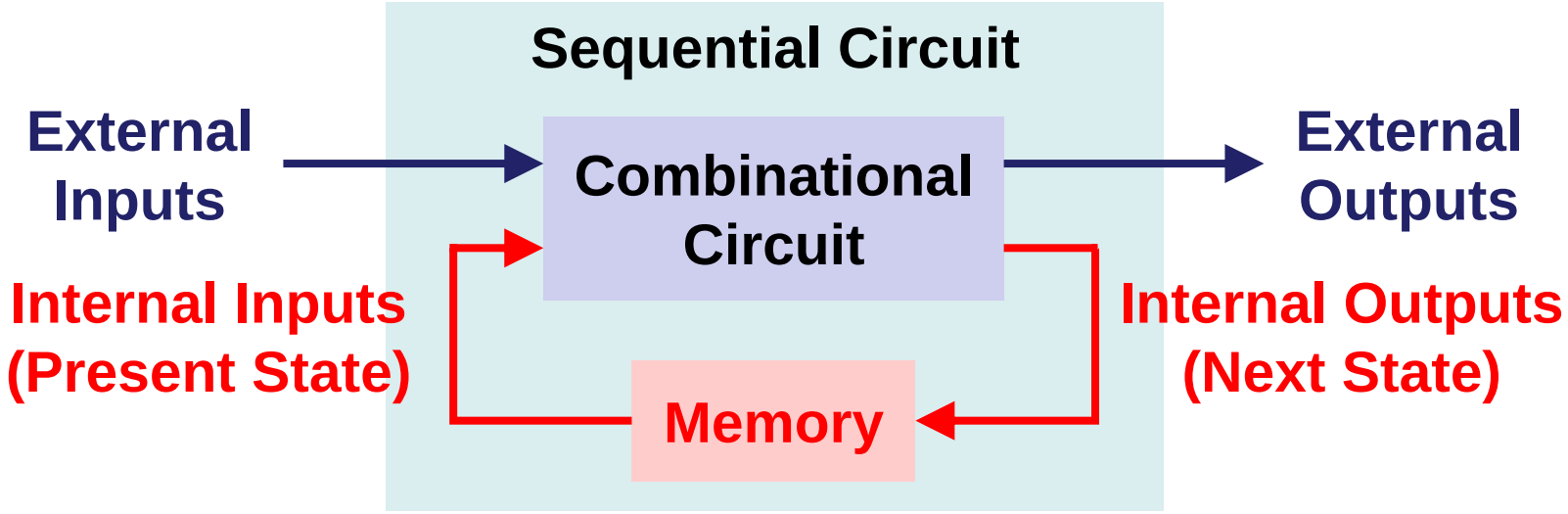
```
c <= a and b;
```

- This logic can be also modeled as a process:
  - All signals referenced in process must be in sensitivity list.

```
entity And_Good is
  port (a, b: in std_logic; c: out std_logic);
end And_Good;
architecture Synthesis_Good of And_Good is
begin
  process (a, b) -- sensitive to signals a and/or b
  begin
    c <= a and b; -- c updated
  end process;
end;
```

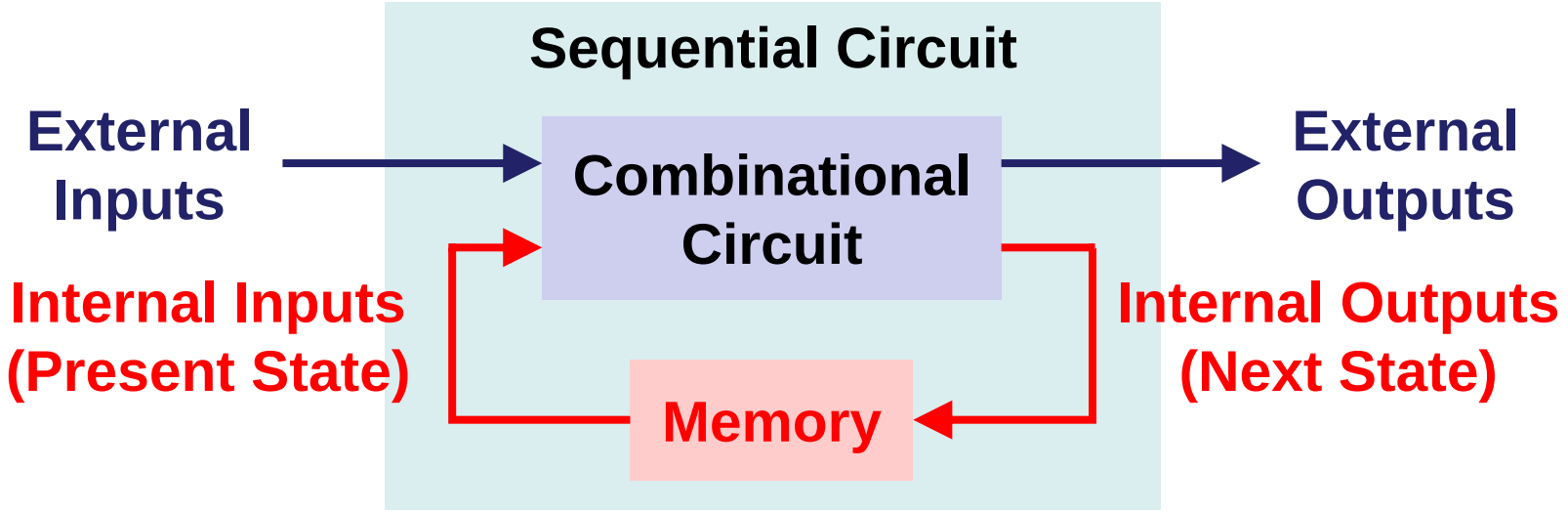
Sequential Circuit

- **Sequential Circuit**: **has memory**
  - Outputs are a function of the *present* inputs and the *previous* outputs (i.e., the **internal state**).
    - It changes outputs based on inputs; but the outputs also depend upon previous outputs (i.e., the **internal state**) (i.e., has memory).
    - Example: Latch (*Lec04*), Flip-Flops (*Lec04*), Counters (*Lec05*), etc.
  - **Rule**: You **must** build a sequential circuit with **only** sequential (i.e., `process`) statements.



Combinational vs. Sequential Circuit

- **Combinational Circuit**: **no memory**
  - ① Outputs are a function of the *present* inputs only.
  - ② **Rule**: Use **either** concurrent **or** sequential statements.
- **Sequential Circuit**: **has memory**
  - ① Outputs are a function of the *present* inputs and the *previous* outputs (i.e., the **internal state**).
  - ② **Rule**: **Must** use sequential (i.e., `process`) statements.

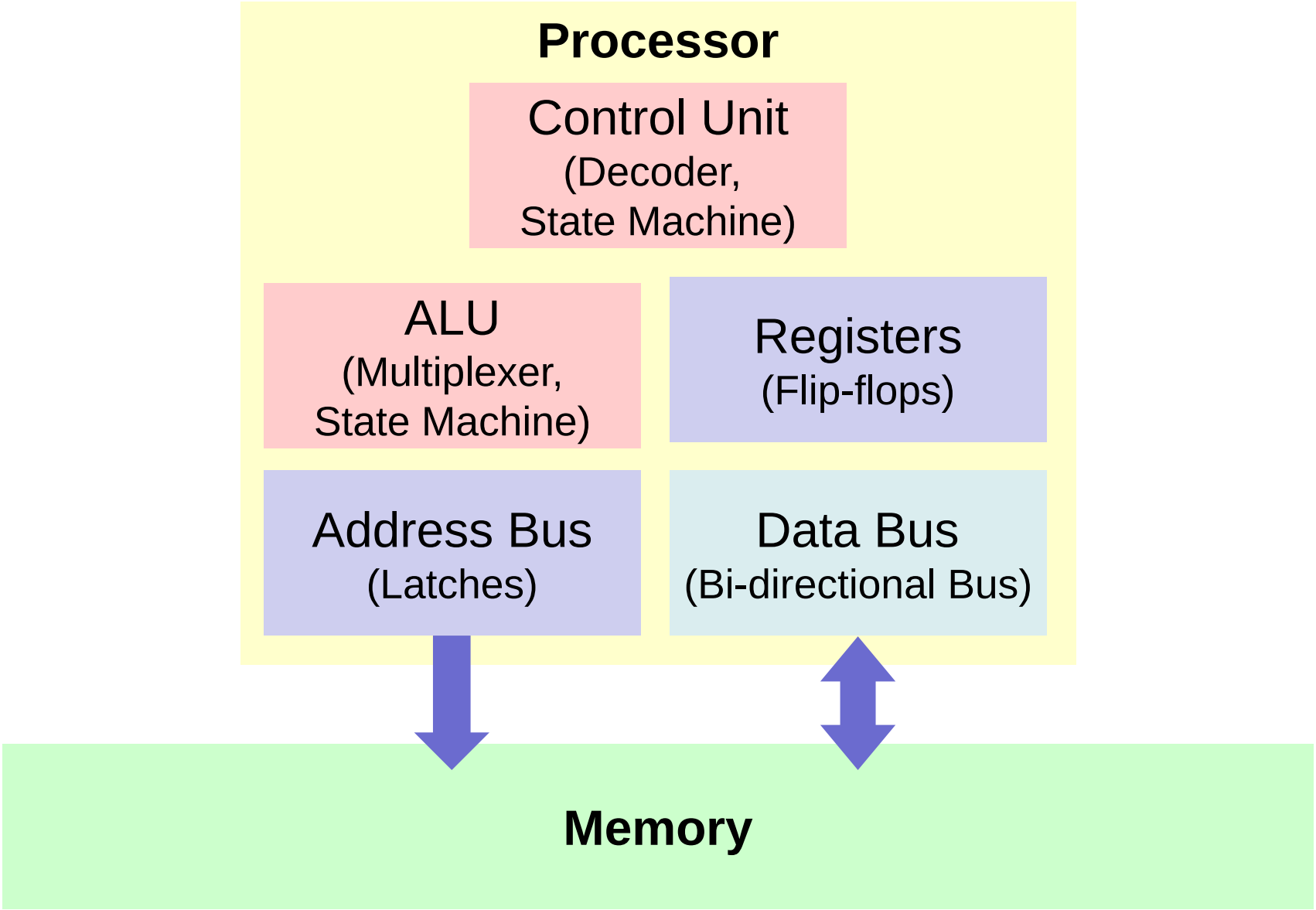


Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Typical Processor Organization

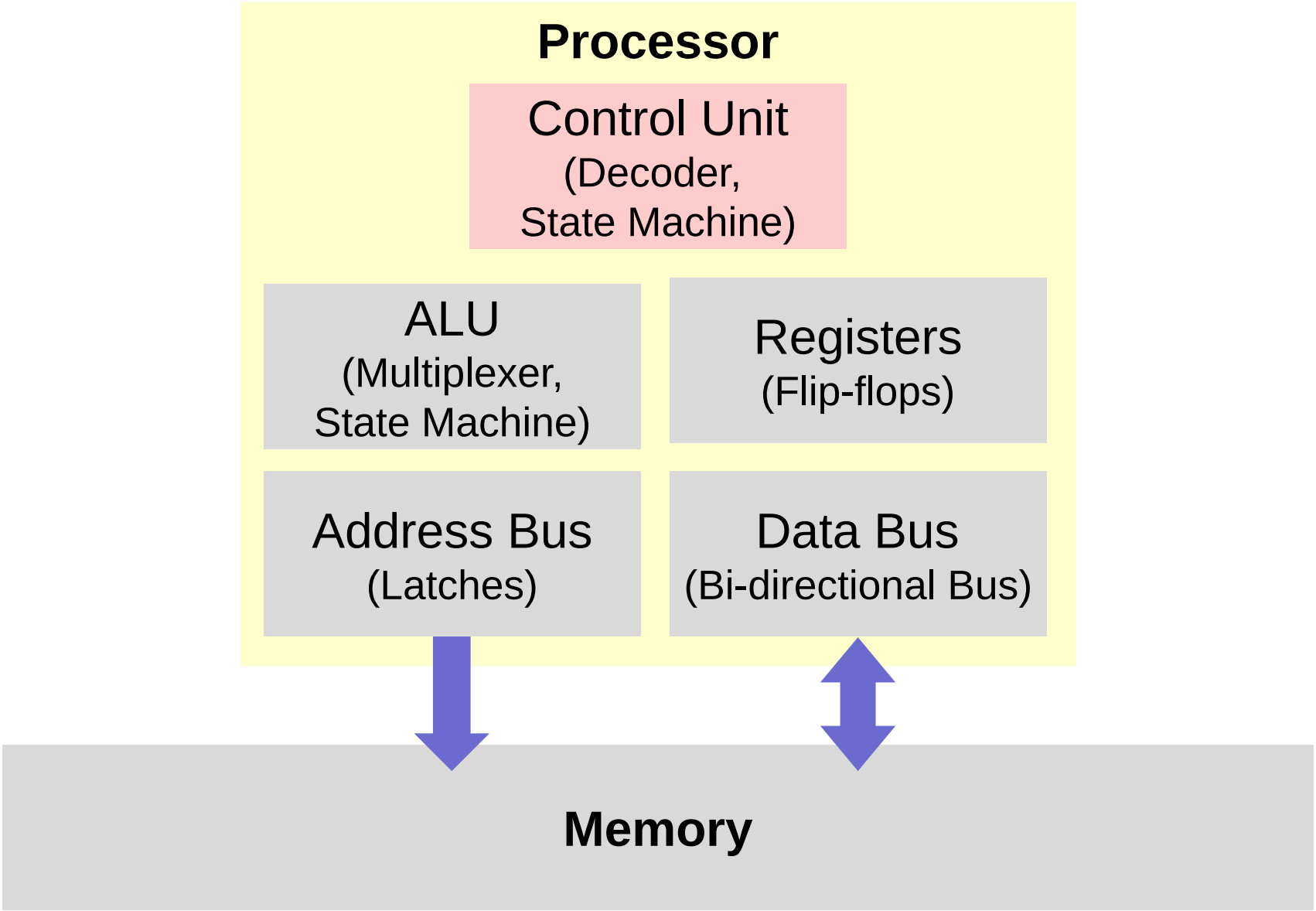


Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Building Blocks: Decoder



Combinational Circuit: Decoder (1/2)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decoder_ex is
port (in0,in1: in std_logic;
      out00,out01,out10,out11: out std_logic);
end decoder_ex;
architecture decoder_ex_arch of decoder_ex is
begin
  process (in0, in1)
  begin
    if in0 = '0' and in1 = '0' then
      out00 <= '1';
    else
      out00 <= '0';
    end if;
    if in0 = '0' and in1 = '1' then
      out01 <= '1';
    else
      out01 <= '0';
    end if;
```

out00

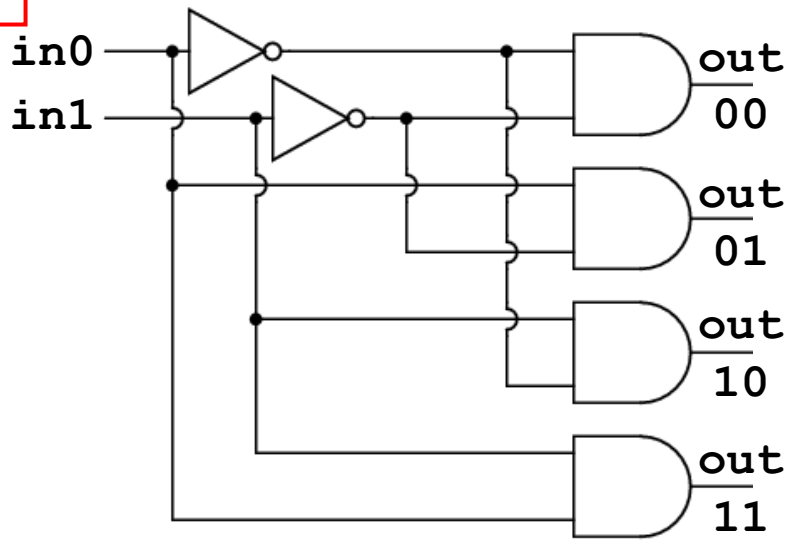
out01

in 0	in 1	out 00	out 01	out 10	out 11
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Combinational Circuit: Decoder (2/2)

```
    ...
    if in0 = '1' and in1 = '0' then
      out10 <= '1';
    else                                  out10
      out10 <= '0';
    end if;
    if in0 = '1' and in1 = '1' then
      out11 <= '1';
    else                                  out11
      out11 <= '0';
    end if;
  end process;
end decoder_ex_arch;
```

in 0	in 1	out 00	out 01	out 10	out 11
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



https://www.allaboutcircuits.com/textbook/digital/chpt-9/decoder/

Class Exercise 4.1

Student ID: _____ Date:
Name: _____ _____

- Implement the Encoder based on the given table:

```
port(
                                                              );
…
architecture encoder_ex_arch of encoder_ex is
begin
  process (                          )
  begin




  end process;
end encoder_ex_arch;
```

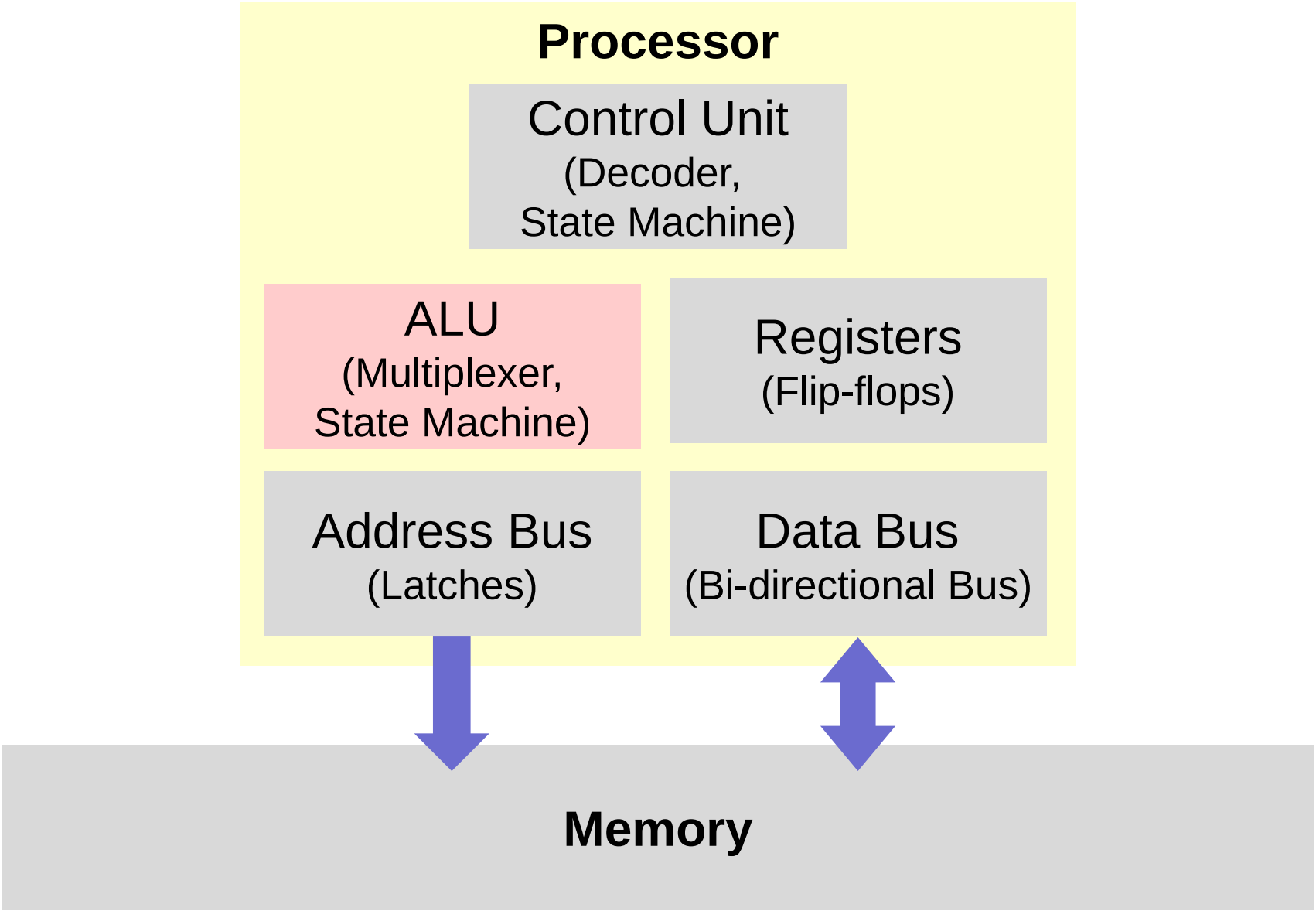
in 00	in 01	in 10	in 11	out 0	out 1
1	0	0	0	**0**	**0**
0	**1**	0	0	**0**	**1**
0	0	**1**	0	**1**	**0**
0	0	0	**1**	**1**	**1**

Outline

- Combinational Circuit and Sequential Circuit

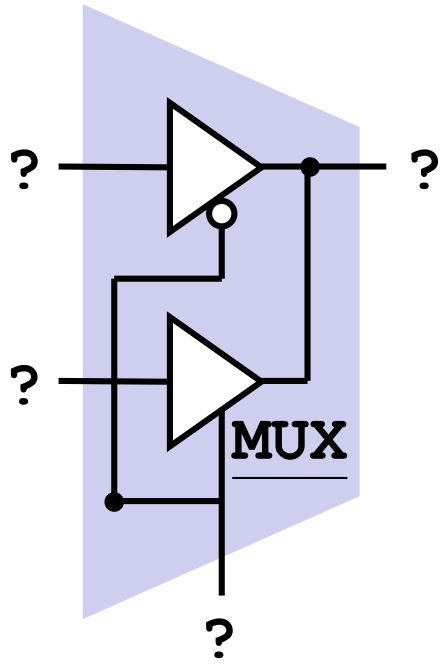
- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Building Blocks: Multiplexer

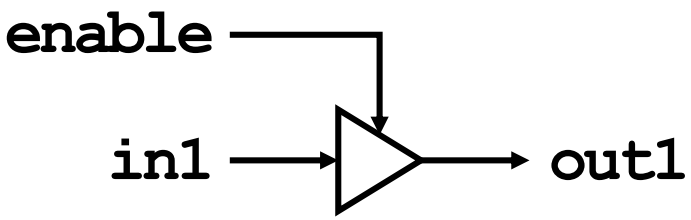


Combinational Circuit: Multiplexer

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux_ex is
port (in1,in2,sel: in std_logic;
                out1: out std_logic);
end mux_ex;
architecture mux_ex_arch of mux_ex is
begin
  process (in1, in2, sel)
  begin
    if sel = '0' then
      out1 <= in1; -- select in1
    else
      out1 <= in2; -- select in2
    end if;
  end process;
end mux_ex_arch;
```



Recall: Tri-state Buffer



in1	enable	out1
0	0	Z
1	0	Z
0	1	0
1	1	1

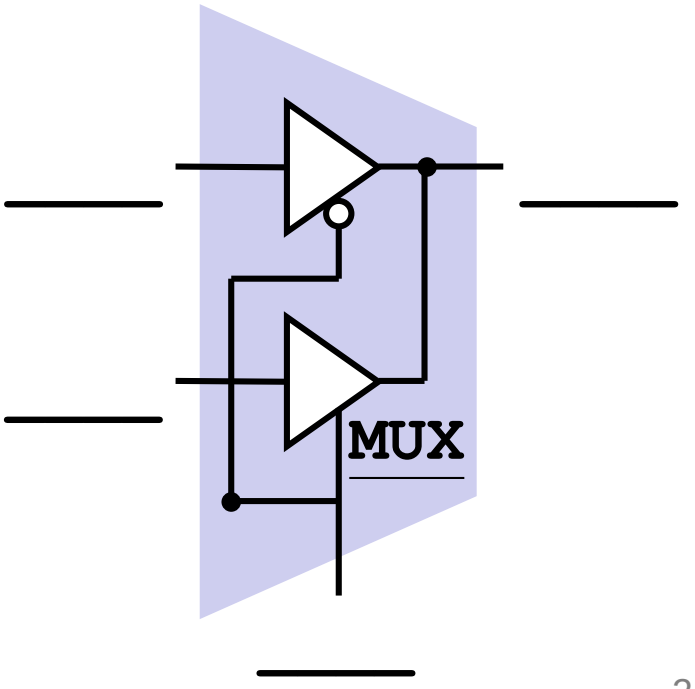
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tri_ex is
port (in1, enable: in std_logic;
                out1: out std_logic);
end tri_ex;
architecture tri_ex_arch of tri_ex is
begin
    out1 <= in1 when enable = '1' else 'Z';
end tri_ex_arch;
```

Class Exercise 4.2

Student ID: _____ Date:
Name: _____ _____

- Specify the I/O signals in the circuit:

```
entity mux_ex is
port (in1,in2,sel: in std_logic;
                out1: out std_logic);
end mux_ex;
architecture mux_ex_arch of mux_ex is
begin
  process (in1, in2, sel)
  begin
    if sel = '0' then
      out1 <= in1;
    else
      out1 <= in2;
    end if;
  end process;
end mux_ex_arch;
```

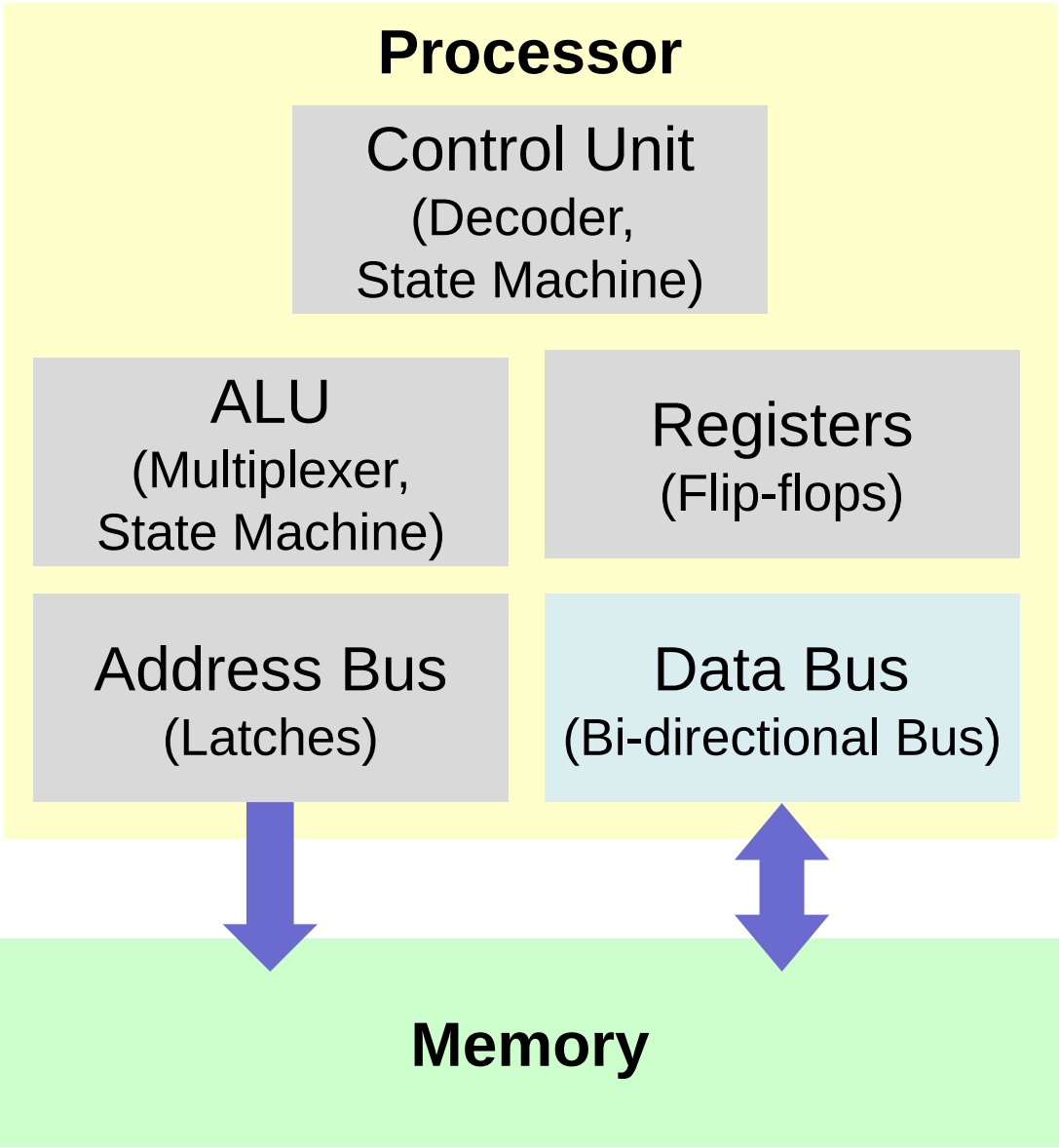


Outline

- Combinational Circuit and Sequential Circuit

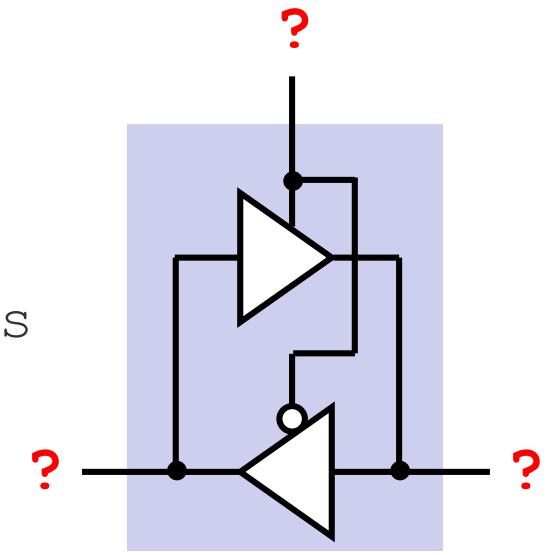
- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Building Blocks: Bi-directional Bus



Combinational Circuit: Bi-directional Bus

```
entity inout_ex is
port (io1, io2: inout std_logic;
           ctrl: in std_logic);
end inout_ex;
architecture inout_ex_arch of inout_ex is
begin
  process (io1, io2, ctrl) is begin
    if (ctrl = '1') then io1 <= io2;
    else io1 <= 'Z';
    end if;
  end process;
  process (io1, io2, ctrl) is begin
    if (ctrl = '0') then io2 <= io1;
    else io2 <= 'Z';
    end if;
  end process;
end inout_ex_arch;
```



io1 follows "**io2.in**"

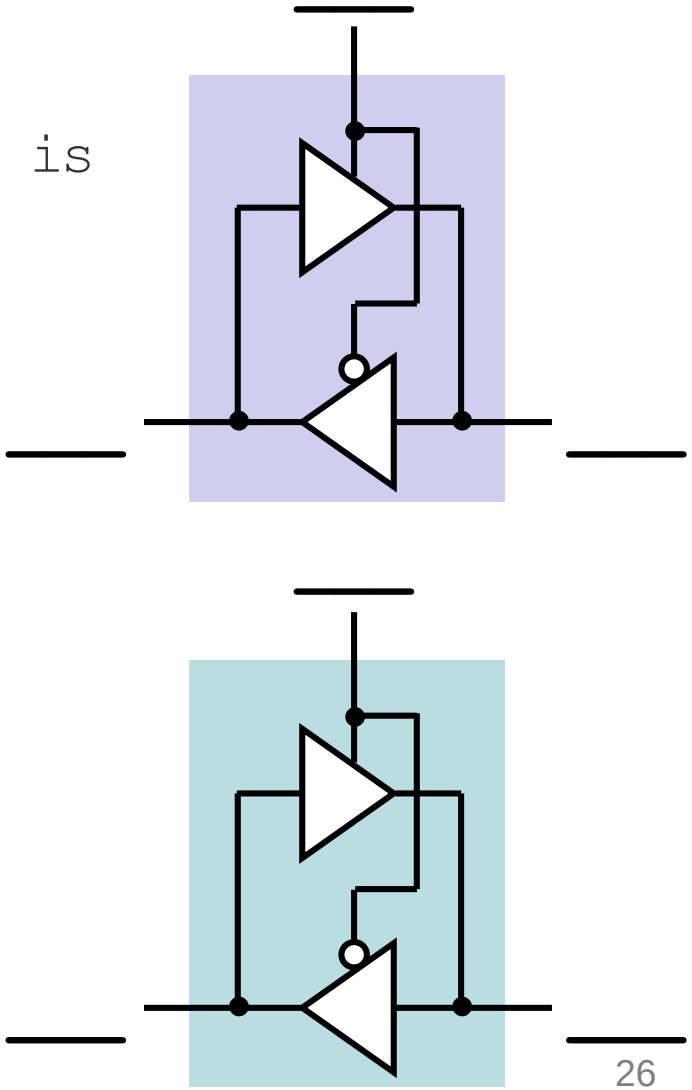
io2 follows "**io1.in**"

Class Exercise 4.3

Student ID: _____ Date:
Name: _____ _____

```
entity inout_ex is
port (io1, io2: inout std_logic;
           ctrl: in std_logic);
end inout_ex;
architecture inout_ex_arch of inout_ex is
begin
  process (io1, io2, ctrl) is begin
    if (ctrl = '1') then io1 <= io2;
    else io1 <= 'Z';
    end if;
  end process;
  process (io1, io2, ctrl) is begin
    if (ctrl = '0') then io2 <= io1;
    else io2 <= 'Z';
    end if;
  end process;
end inout_ex_arch;
```

- Specify I/O signals:



Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Latches and Flip Flops

- Latches and Flip-flops (FF) are the basic elements used to store information.
  - Each latch and flip flop can keep one bit of data.

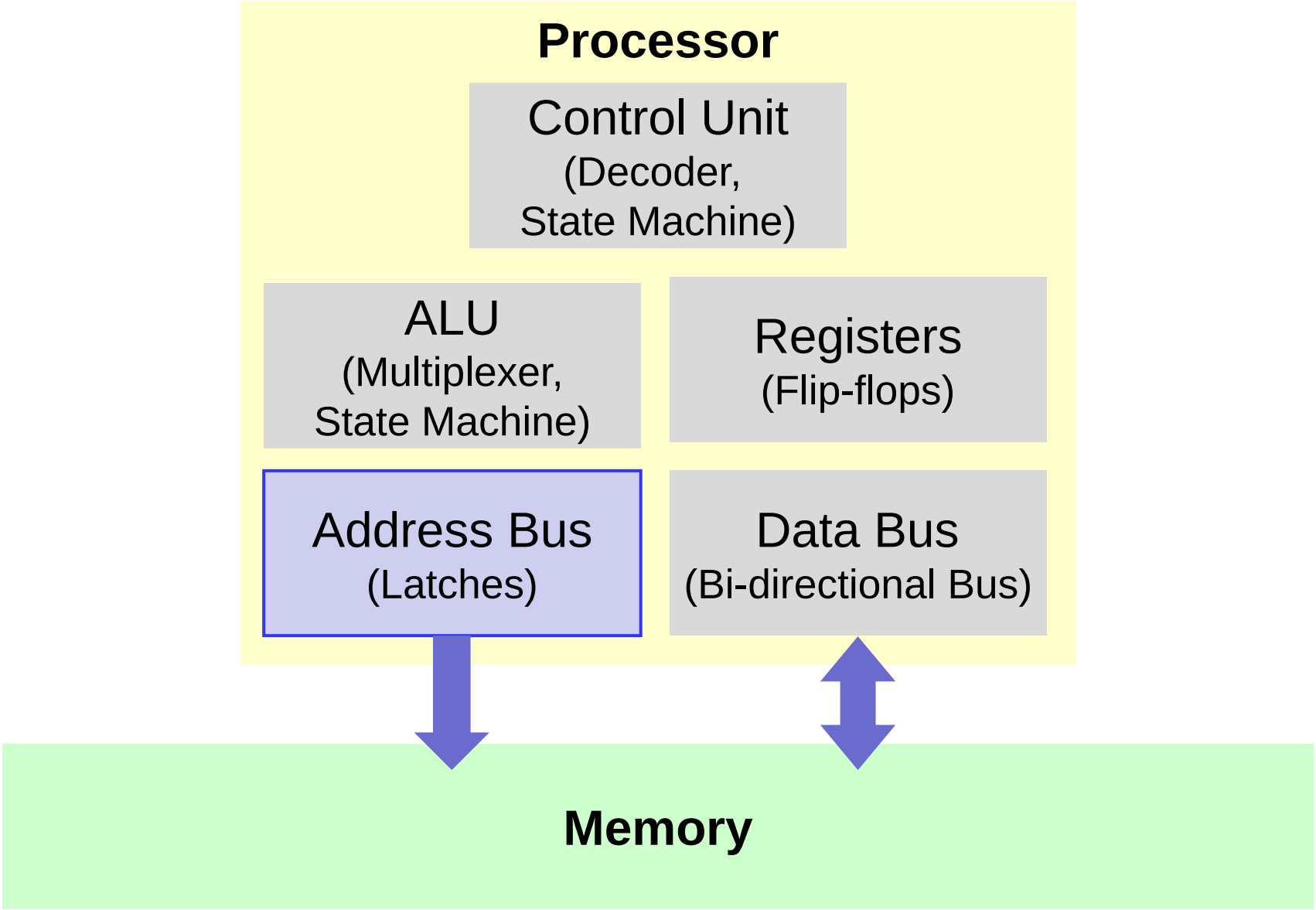
- The main difference between latch and flip-flop:
  - A latch continuously checks input and changes the output **whenever** there is a change in input.
    - A latch has **no** clock signal.
  - A flip-flop continuously checks input and changes the output **only at times** determined by the clock signal.
    - A flip flop has a clock signal.

Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Building Blocks: Latch

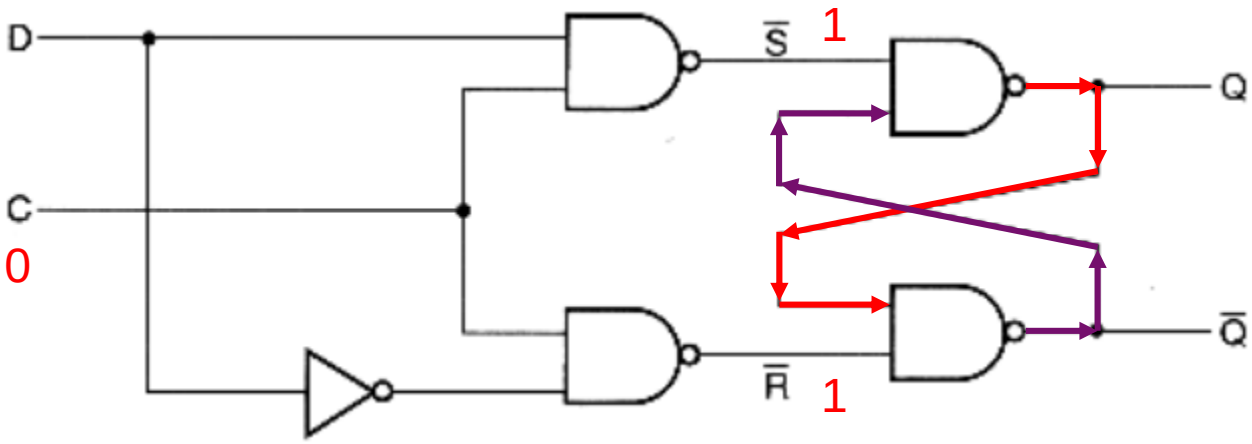


Sequential Circuit: Latch (1/2)

- Latches are asynchronous (no CLOCK signal).
  - It changes output **only in response** to input.
- **Case Study**: **D Latch**
  - When enable line C is high, the output Q follows input D.

  → That is why D latch is also called as transparent latch.
    - When enable line C is asserted, the latch is said to be transparent.
  - When C falls, the last state of D input is trapped and held.

  → That is why the latch has memory!



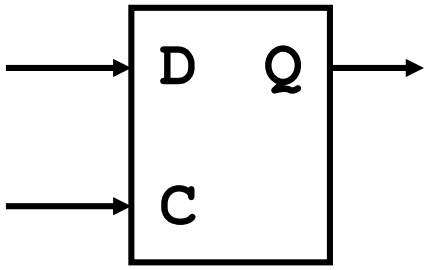
Data need to be held.

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

https://www.edgefx.in/digital-electronics-latches-and-flip-flops/

Sequential Circuit: Latch (2/2)

```
 1 library IEEE;--(ok vivado 2014.4)
 2 use IEEE.STD_LOGIC_1164.ALL;
 3 entity latch_ex is
 4 port (C, D: in std_logic;
 5           Q: out std_logic);
 6 end latch_ex;
 7 architecture latch_ex_arch of latch_ex is
 8 begin
 9   process(C, D) -- sensitivity list
10   begin
11     if (C = '1') then
12       Q <= D;
13     end if;
       -- no change (memory)
14   end process;
15 end latch_ex_arch;
```



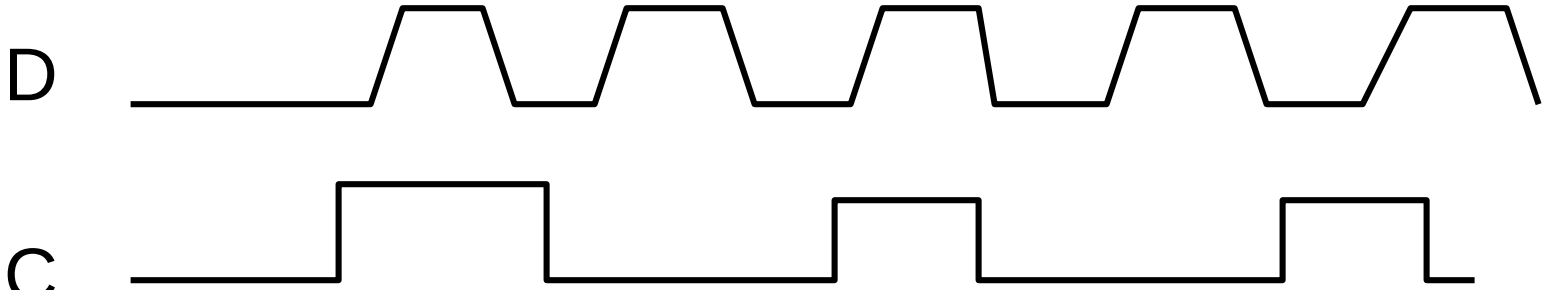
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

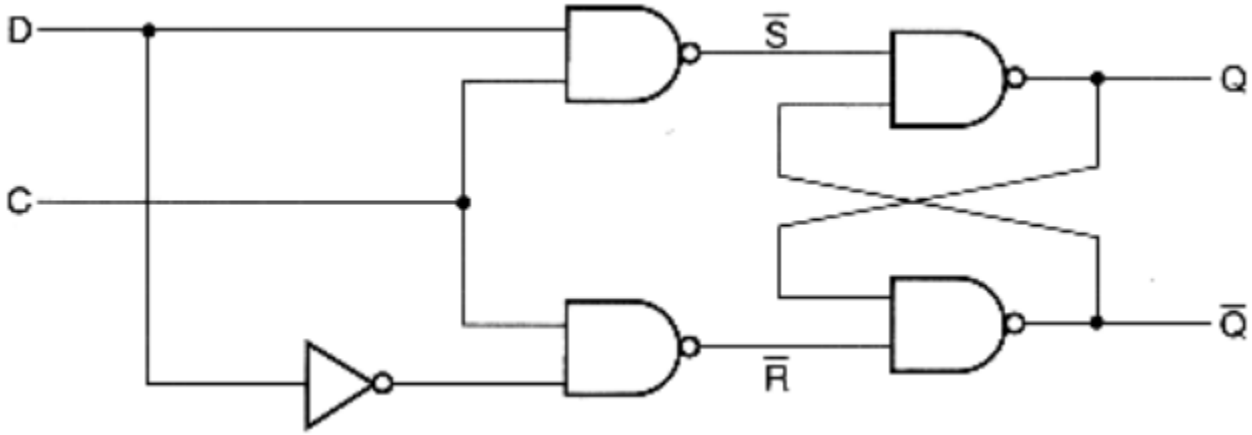
https://www.edgefx.in/digital-electronics-latches-and-flip-flops/

Class Exercise 4.4

Student ID: _____ Date:
Name: _____ _____

- Given a D latch, draw Q in the following figure:





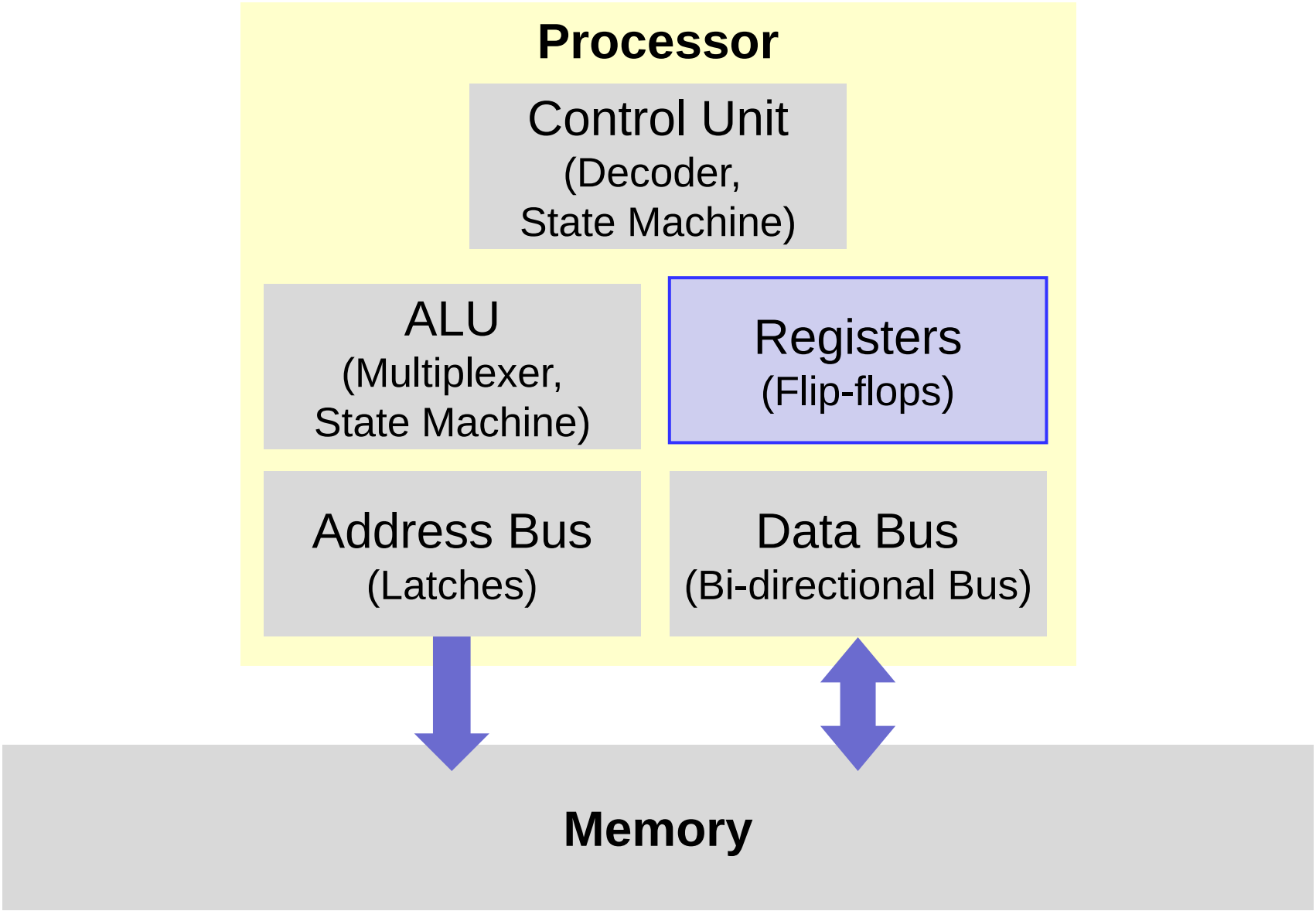
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

Outline

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

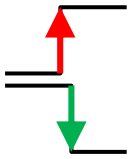
Building Blocks: Flip-flops



Sequential Circuit: Flip-flop

- A **Latch** is a non-clock-controlled memory device.
  ① It has **no** CLOCK signal.
  ② It changes output only in response to data input. (i.e., the value is set asynchronously).

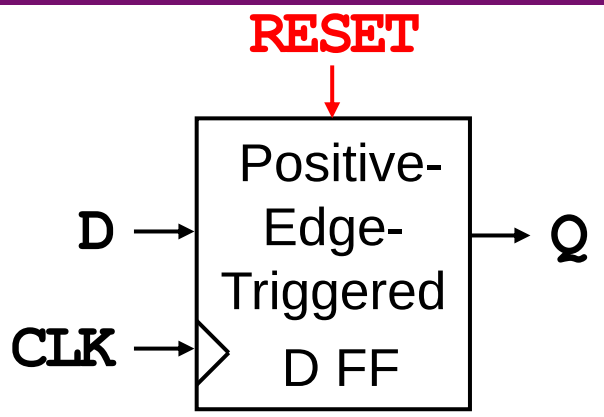
- A **Flip-flop (FF)** is a clock-controlled memory device.
  ① Different from a Latch, it **has** a CLOCK signal as input.
  ② It stores the input value (i.e., low or high) and **outputs** the stored value **only in response to** the CLOCK signal.
    - **Positive-Edge-Triggered**: At every **low to high** of CLOCK.
    - **Negative-Edge-Triggered**: At every **high to low** of CLOCK.



  ③ The value can be **reset** asynchronously or synchronously.
    - **Async. Reset**: Reset the value **anytime**.
    - **Sync. Reset**: Reset the value on **positive** or **negative** **clock edges**.

Positive-Edge-Triggered FF with Async. Reset

```
 1 library IEEE;
 2 use IEEE.STD_LOGIC_1164.ALL;
 3 entity dff_async is
 4 port(D,CLK,RESET: in std_logic;
 5                   Q: out std_logic);
 6 end dff_async;
 7 architecture dff_async_arch of dff_async is
 8 begin
 9   process(CLK, RESET) -- sensitivity list
10   begin
11      if (RESET = '1') then
12         Q <= '0'; -- Reset Q immediately
13      elsif CLK = '1' and CLK'event then
14         Q <= D; -- Q follows input D
15      end if;
        -- no change (so has memory)
16   end process;
17 end dff_async_arch;
```



← Positive-edge-triggered

Recall: Attributes (Lec01)

- Another important signal attribute is the **`'event`**.
  - This attribute yields a Boolean value of TRUE if an event has just occurred on the signal.
  - It is used primarily to determine if a **clock** has transitioned.
- Example (*more in Lec04*):

```
…
port(my_in, clock: in std_logic;
             my_out: out std_logic);
…
if clock = '1' and clock'event then
     my_out <= my_in;
```

Class Exercise 4.5

Student ID: _____ Date:
Name: _____ _____

- Consider the following VHDL implementation of a positive-edge-triggered FF with asynchronous reset:

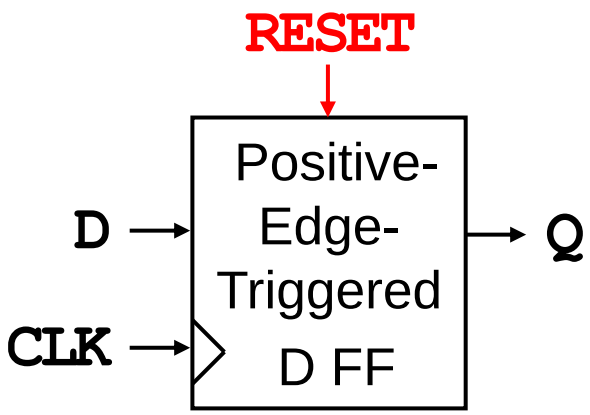
```
...
 9   process(CLK, RESET) -- sensitivity list
10   begin
11      if (RESET = '1') then
12         Q <= '0'; -- Reset Q
13      elsif CLK = '1' and CLK'event then
14         Q <= D; -- Q follows input D
15      end if;
        -- no change (so has memory)
16   end process;
...
```

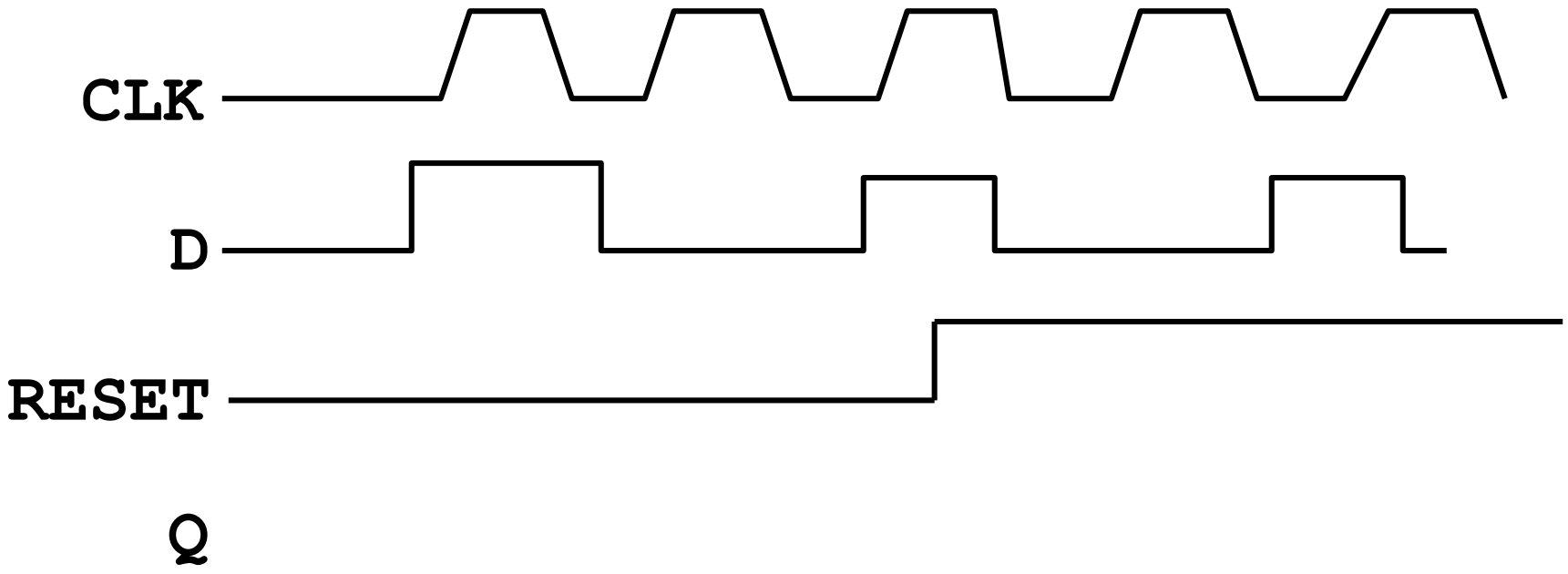
  - When will line 9 be executed?
    Answer: _____
  - Which signal is more "powerful"? **CLK** or **RESET**?
    Answer: _____

Class Exercise 4.6

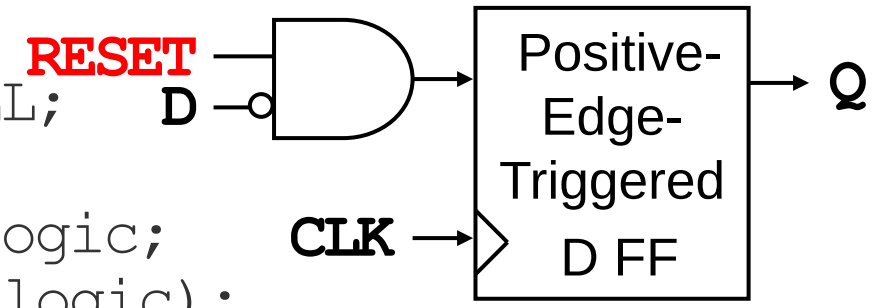
Student ID: _____ Date:
Name: _____ _____

- Given a “50%” Positive-edge-triggered D Flip-flop with **async.** reset, draw the output Q.
  - “50%” means it changes state when clock is 50% between high and low.





Positive-Edge-Triggered FF with Sync. Reset




```
 1 library IEEE;
 2 use IEEE.STD_LOGIC_1164.ALL;
 3 entity dff_sync is
 4 port(D,CLK,RESET: in std_logic;
 5                   Q: out std_logic);
 6 end dff_sync;
 7 architecture dff_sync_arch of dff_sync is begin
 8 process(CLK)
 9   begin
10     if CLK = '1' and CLK'event then
11       if (RESET = '1') then
12         Q <= '0'; -- Reset Q
13       else
14         Q <= D; -- Q follows input D
15       end if;
16     end if;
       -- no change (so has memory)
17   end process;
18 end dff_syn_arch;
```

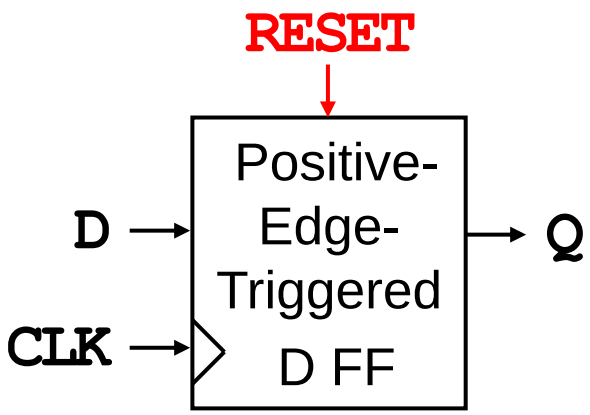
← **RESET** can be removed (why?)

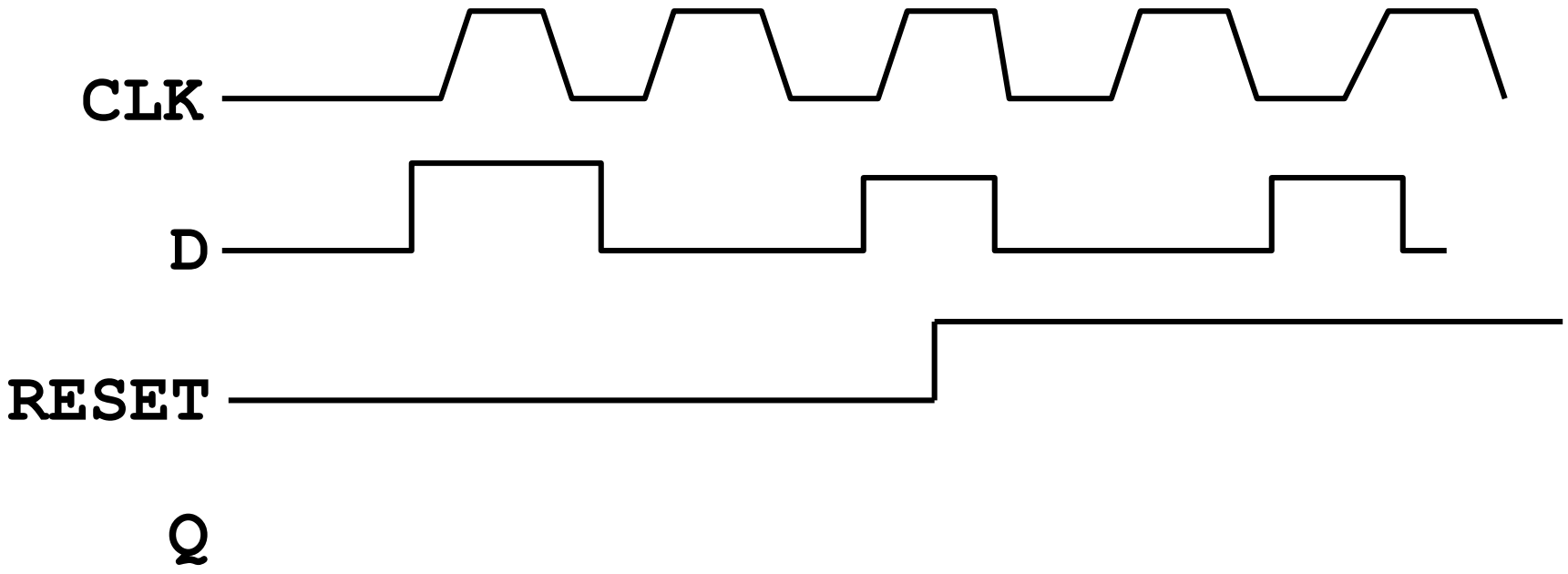
← Positive-edge-triggered

Class Exercise 4.7

Student ID: _____ Date:
Name: _____ _____

- Given a “50%” Positive-edge-triggered D Flip-flop with **sync.** reset, draw the output Q.
  - “50%” means it changes state when clock is 50% between high and low.





Async. Reset vs. Sync. Reset (1/2)

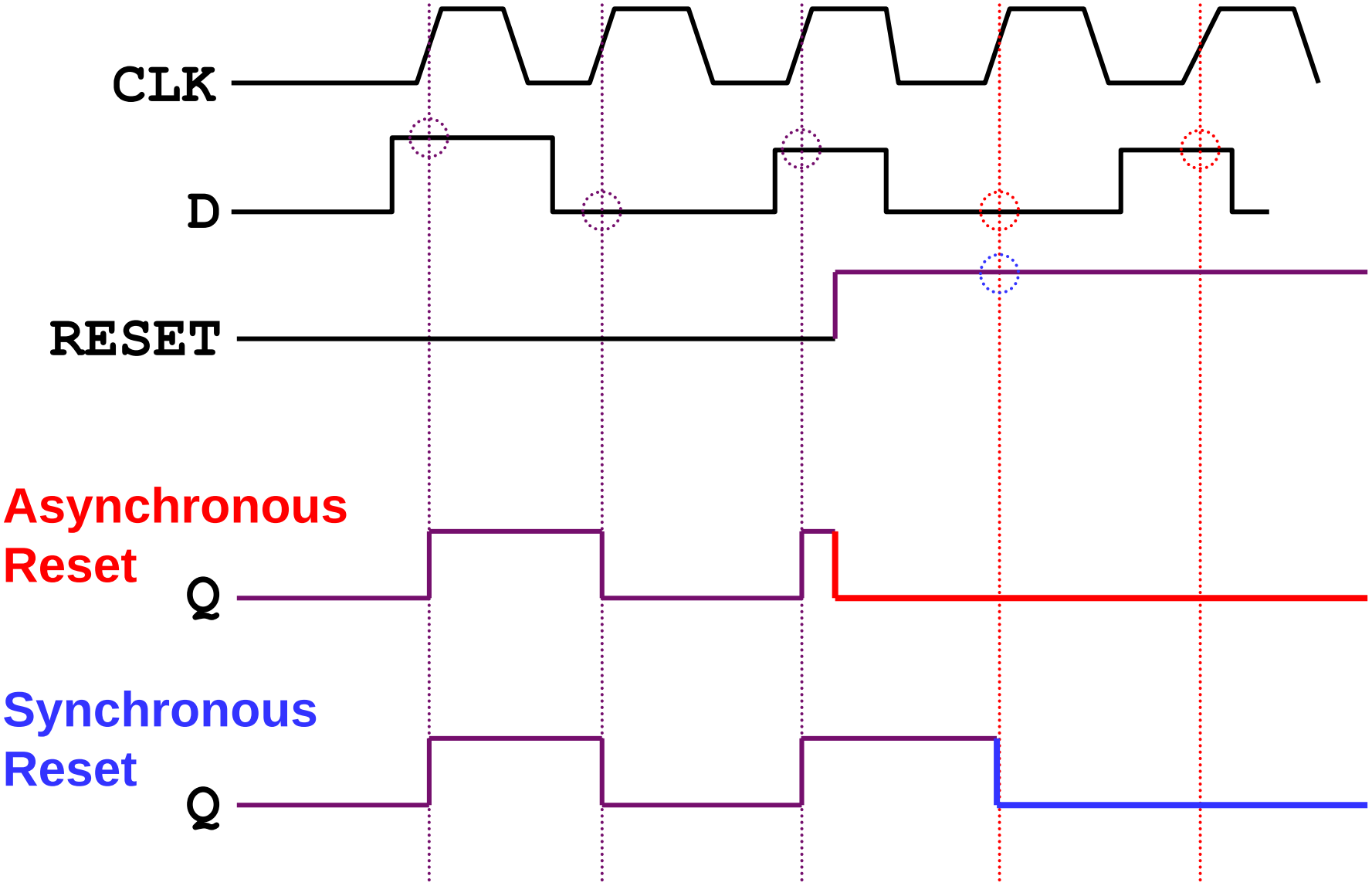
- The order of the statements inside the process determines asynchronous reset or synchronous reset.
  - **Asynchronous Reset** (check **RESET** first!)

```
11      if (RESET = '1') then
12         Q <= '0'; -- Reset Q
13      elsif CLK = '1' and CLK'event then
14         Q <= D; -- Q follows input D
15      end if;
```

  - **Synchronous Reset** (check **CLK** first!)

```
10     if CLK = '1' and CLK'event then
11       if (RESET = '1') then
12         Q <= '0'; -- Reset Q
13       else
14         Q <= D; -- Q follows input D
15       end if;
16     end if;
```

Aysnc. Reset vs. Sync. Reset (2/2)



Summary

- Combinational Circuit and Sequential Circuit

- Building Blocks of a Processor
  - Combinational Circuit: **No Memory**
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: **Has Memory**
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

What's the next? Finite State Machine!