香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

Lecture 04: Building Blocks of a Processor

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

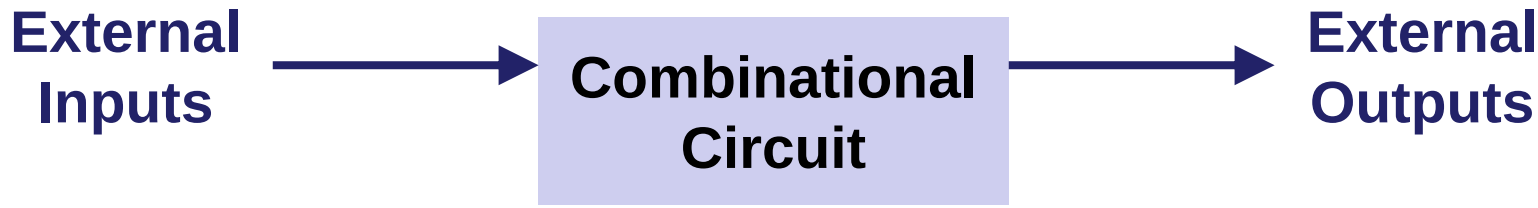
Outline

- 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

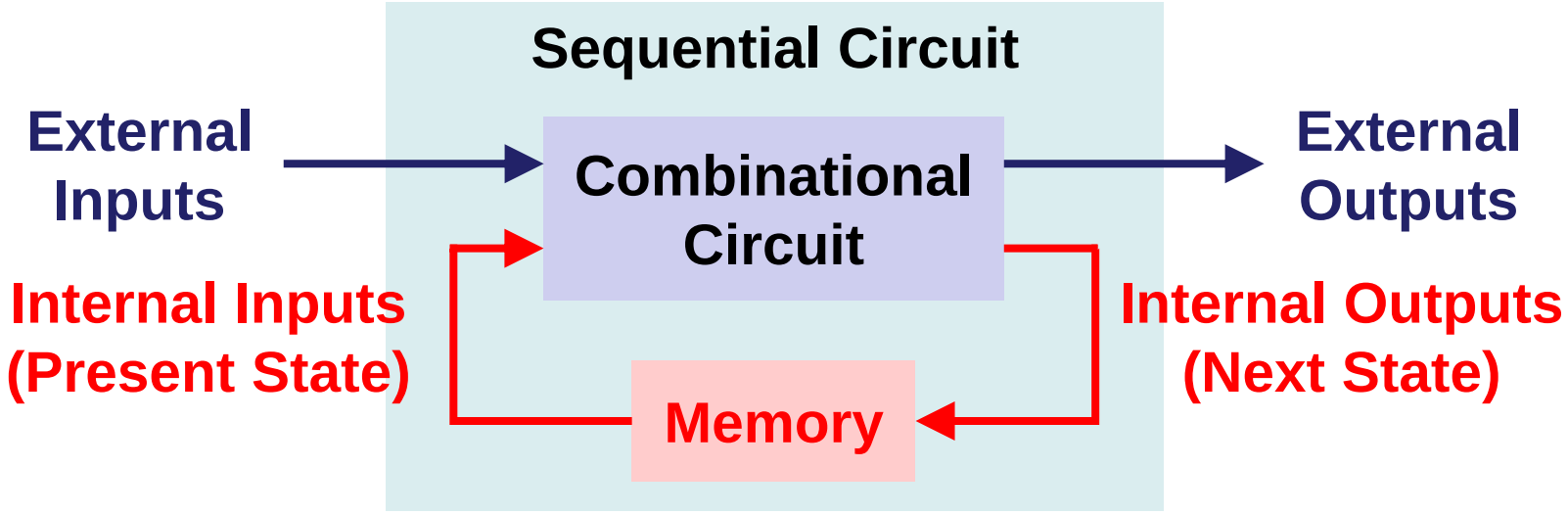
Combinational Circuit

- **Combinational Circuit**: **no memory**
  - Outputs are a function of the present inputs only.
    - As soon as inputs change, the values of previous inputs are lost.
    - That is, combinational logic circuits have no memory.
  - Example: inverter, tri-state buffer, encoder/decoder, multiplexer, bi-directional bus, etc.



Sequential Circuit

- **Sequential Circuit**: **has memory**
  - The outputs may depend upon the present inputs, the past inputs, and the previous outputs (i.e., state).
    - That is, the output of a sequential circuit may depend upon its previous outputs and so in effect has some form of memory.
    - It changes states and outputs based on some conditions, such as inputs or clock signal.
  - Example: latch, flip-flops (FFs), etc.

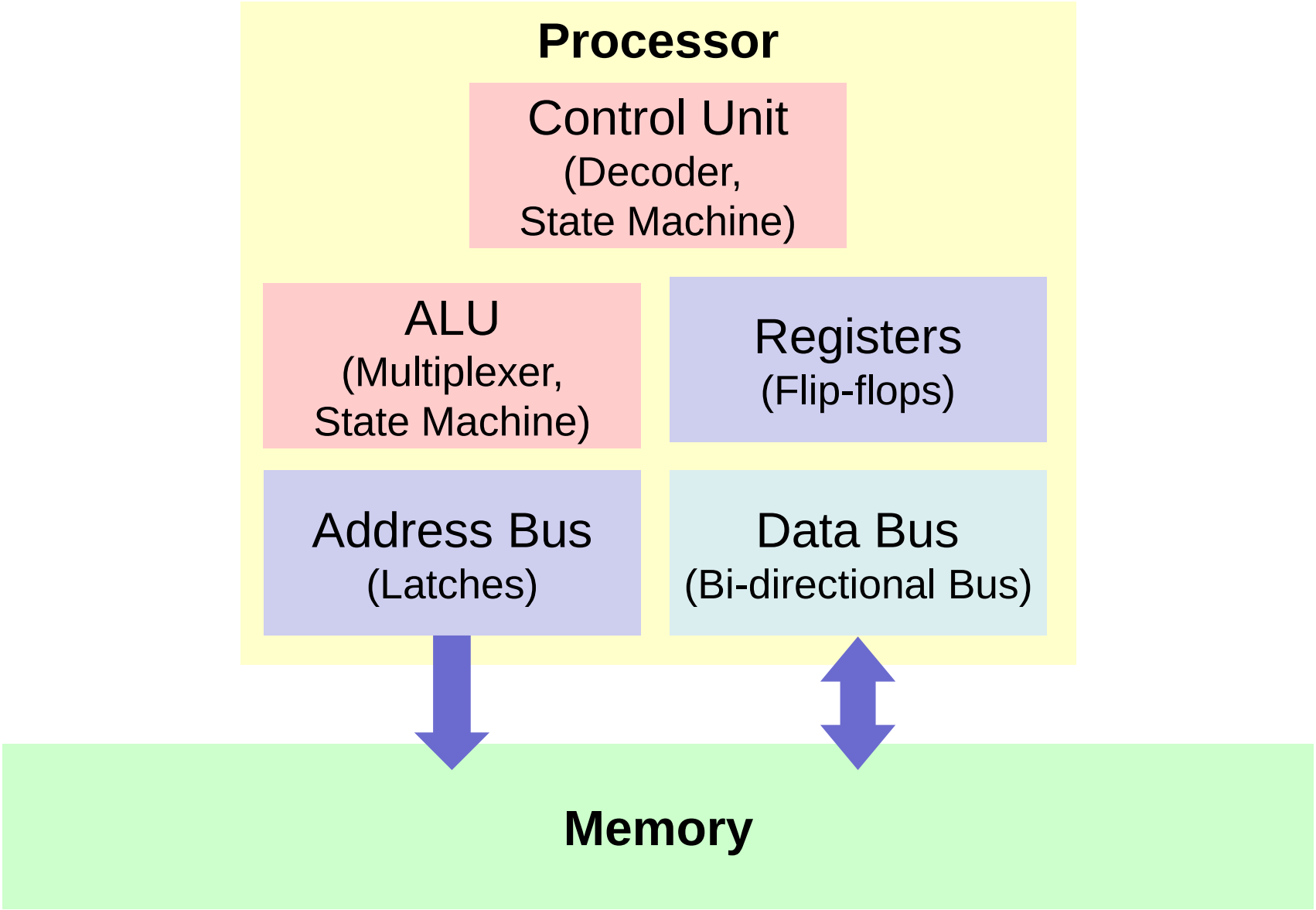


Outline

- 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

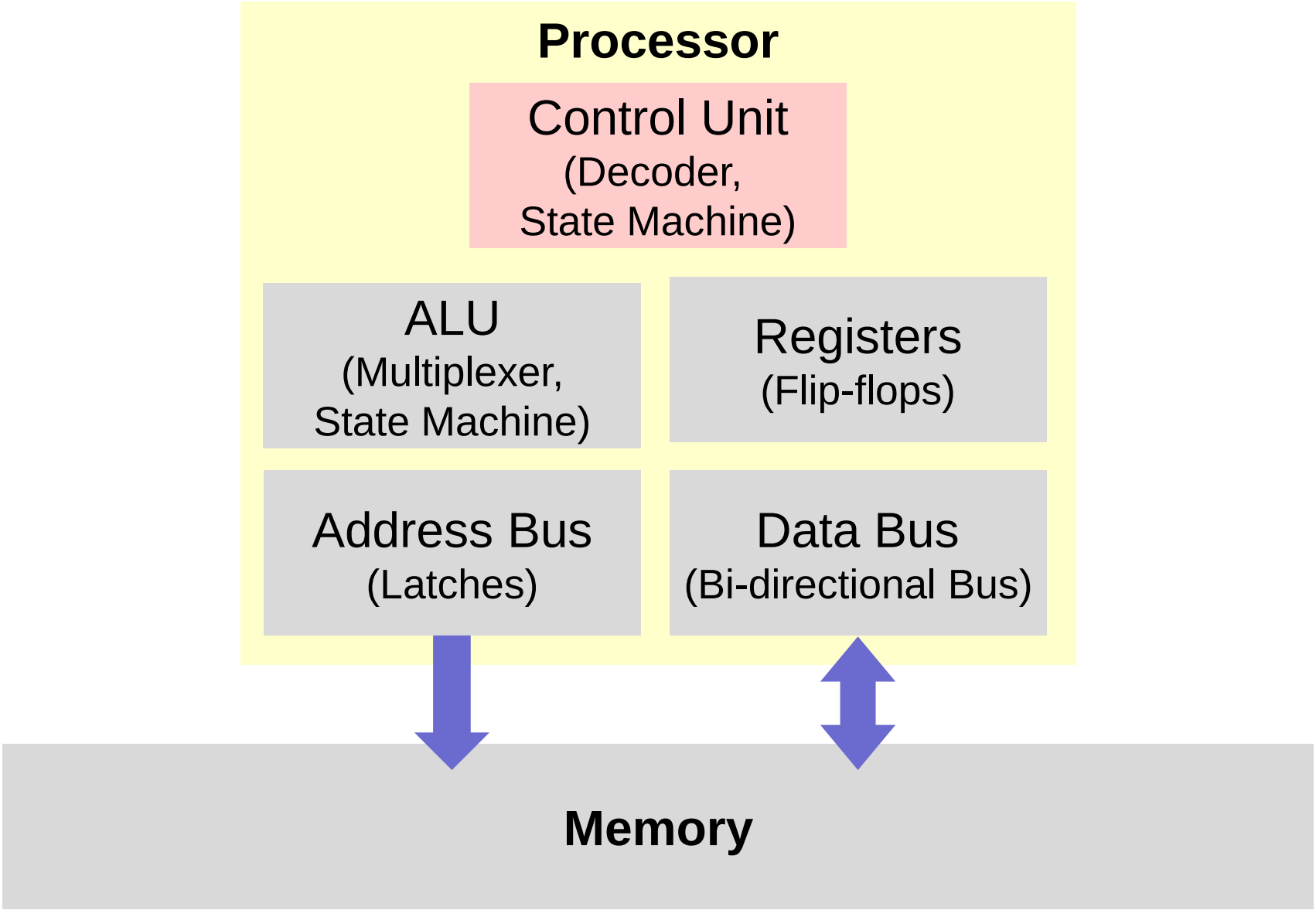


Outline

- 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

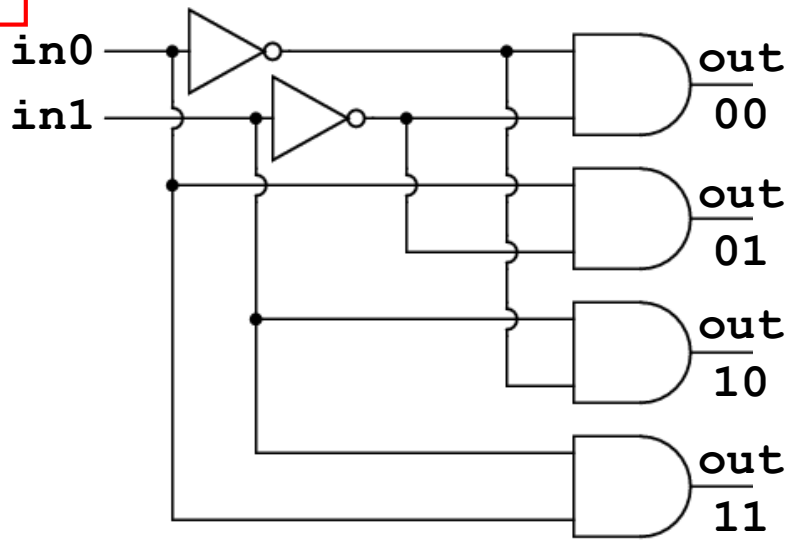
out01

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 (in1, in2)
  begin




  end process;
end encoder_ex_arch;
```

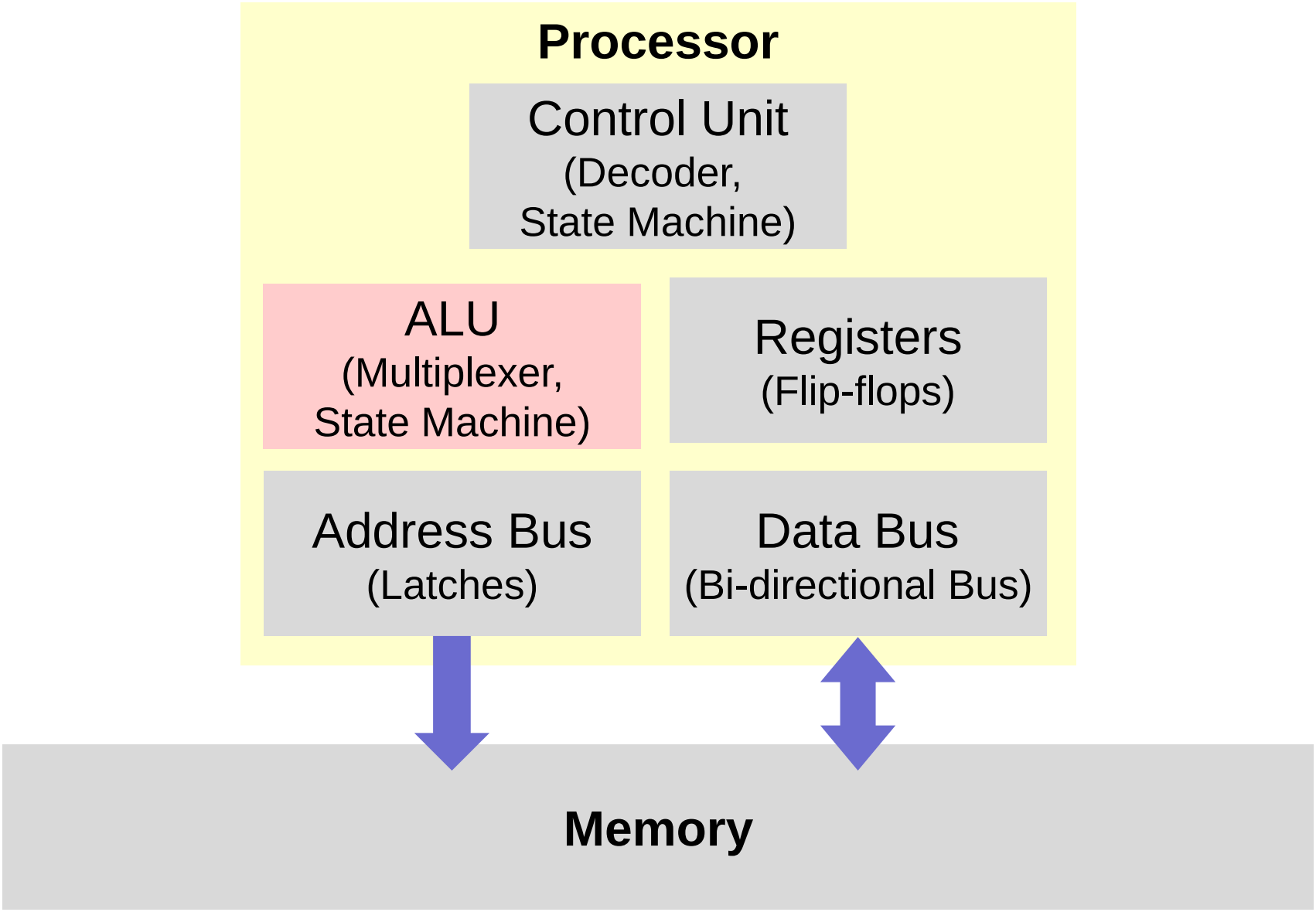
in 00	in 01	in 10	in 11	out 0	out 1
1	0	0	0	**0**	**0**
0	**1**	0	0	**0**	**1**
0	0	**1**	0	**1**	**0**
0	0	0	**1**	**1**	**1**

Outline

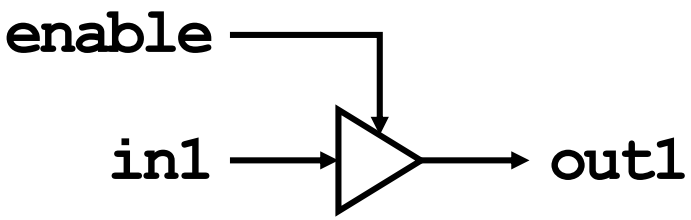
- Combinational Circuit and Sequential Circuit

- 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



Recall: Tri-state Buffer

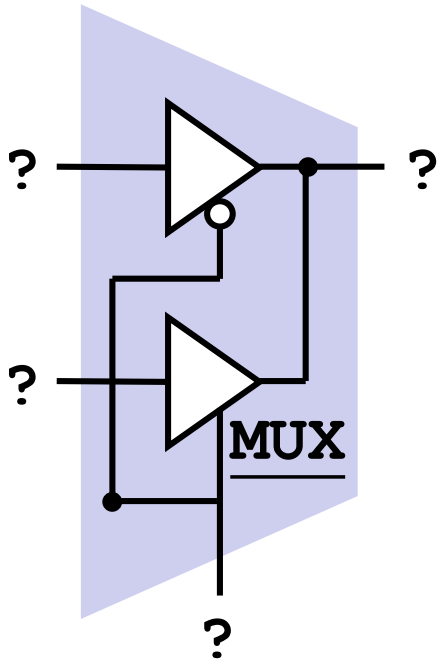


in1	enable	out1
0	0	Z
1	0	Z
0	1	0
1	1	1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tri_ex is
port (in1, enable: in std_logic;
                ut1: 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;
```

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;
```

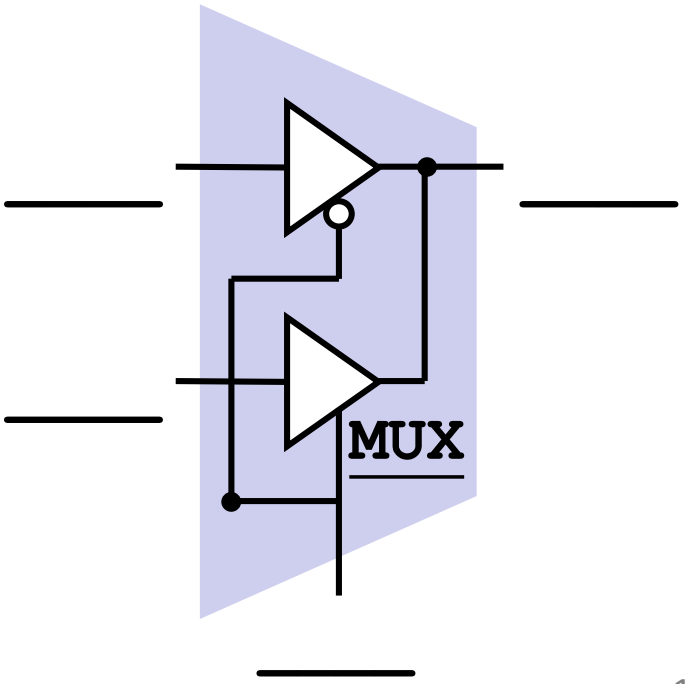


Class Exercise 4.2

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;
```

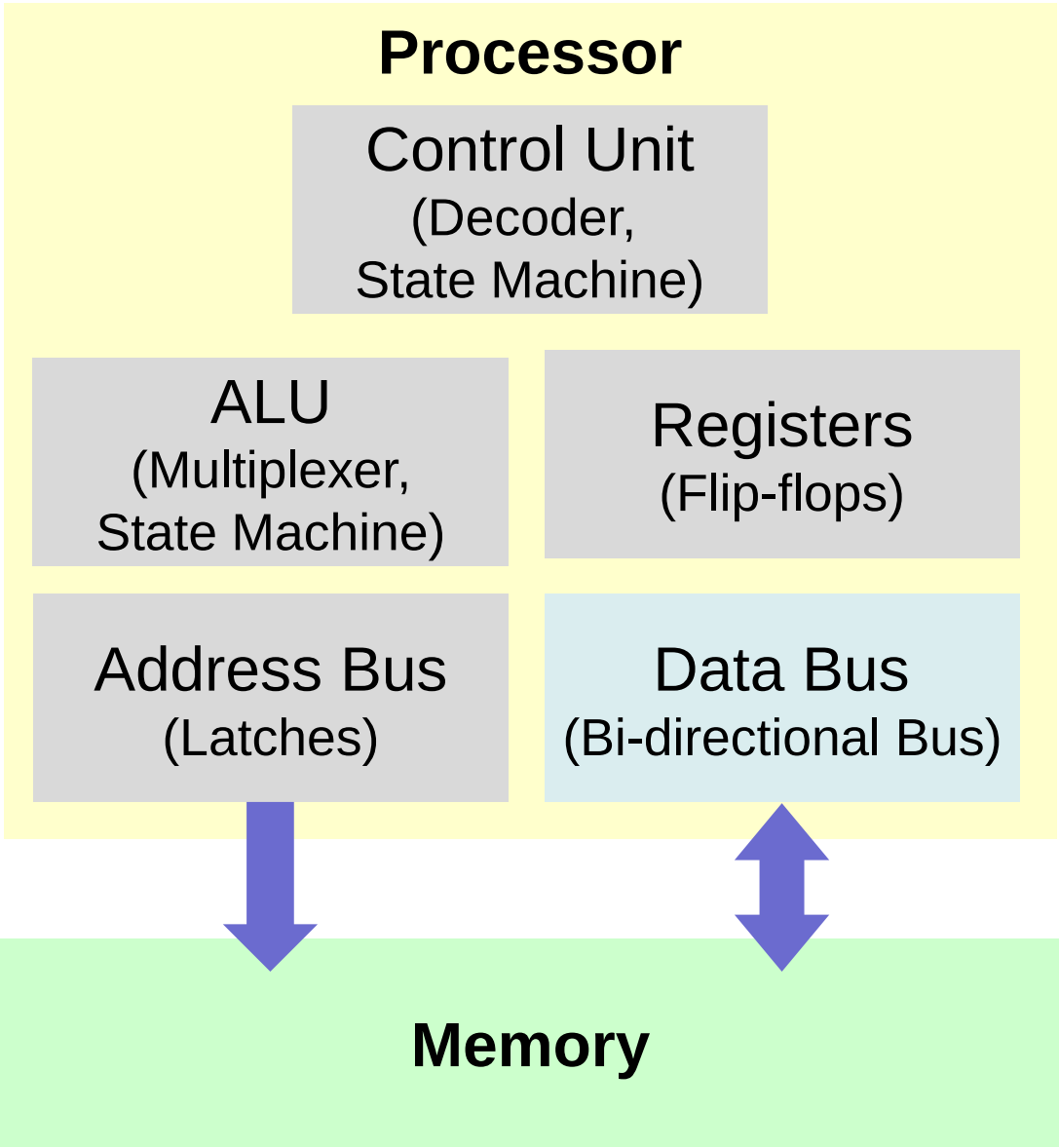


Outline

- Combinational Circuit and Sequential Circuit

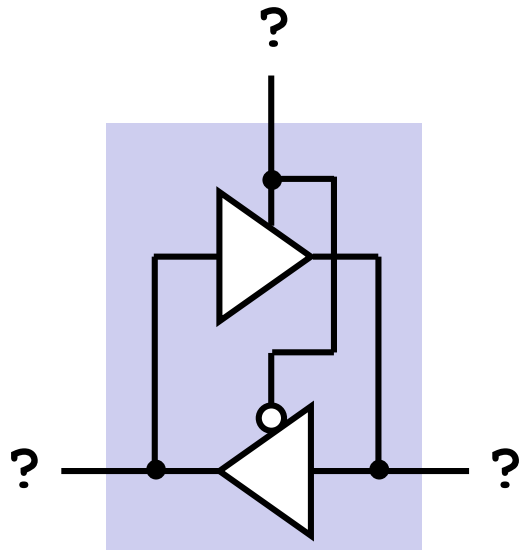
- Building Blocks of a Processor
  - Combinational Circuit: No Memory
    - Decoder
    - Multiplexer
    - Bi-directional Bus
  - Sequential Circuit: Has Memory
    - Latch
    - Flip-flop with Asynchronous Reset
    - Flip-flop with Synchronous Reset

Building Blocks: Bi-directional Bus



Combinational Circuit: Bi-directional Bus

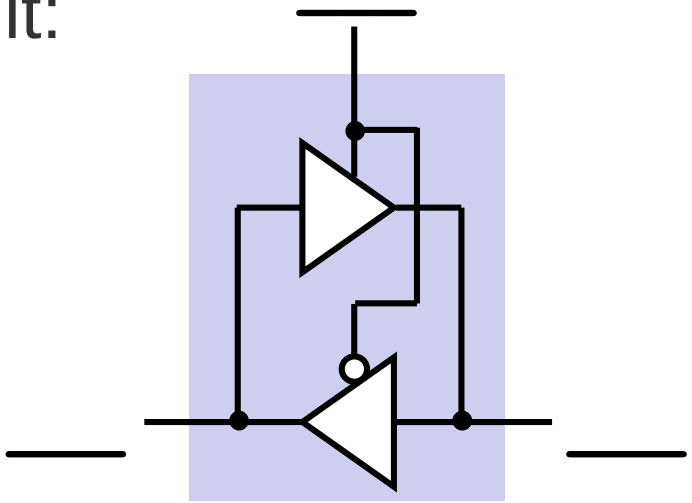
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity inout_ex is
port (io1, io2: inout std_logic;
          ctrl: in std_logic);
end inout_ex;
architecture inout_ex_arch of inout_ex is
begin
    io1 <= io2 when ctrl = '1' else 'Z';
    -- io1 follows "io2.in"
    io2 <= io1 when ctrl = '0' else 'Z';
    -- io2 follows "io1.in"
end inout_ex_arch;
```



Class Exercise 4.3

Student ID: _____ Date:
Name: _____ _____

- Specify the I/O signals in the circuit:



```
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
    io1 <= io2 when ctrl = '1' else 'Z';
    -- io1 follows “io2.in”
    io2 <= io1 when ctrl = '0' else 'Z';
    -- io2 follows “io1.in”
end inout_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

Latches and Flip Flops

- Latches and Flip-flops (FF) are the basic elements used to store information.
  - Each latch and flip flop can store one bit of data.
  - The output not only depends on the current inputs, but also depends on the previous input and outputs (has memory!).

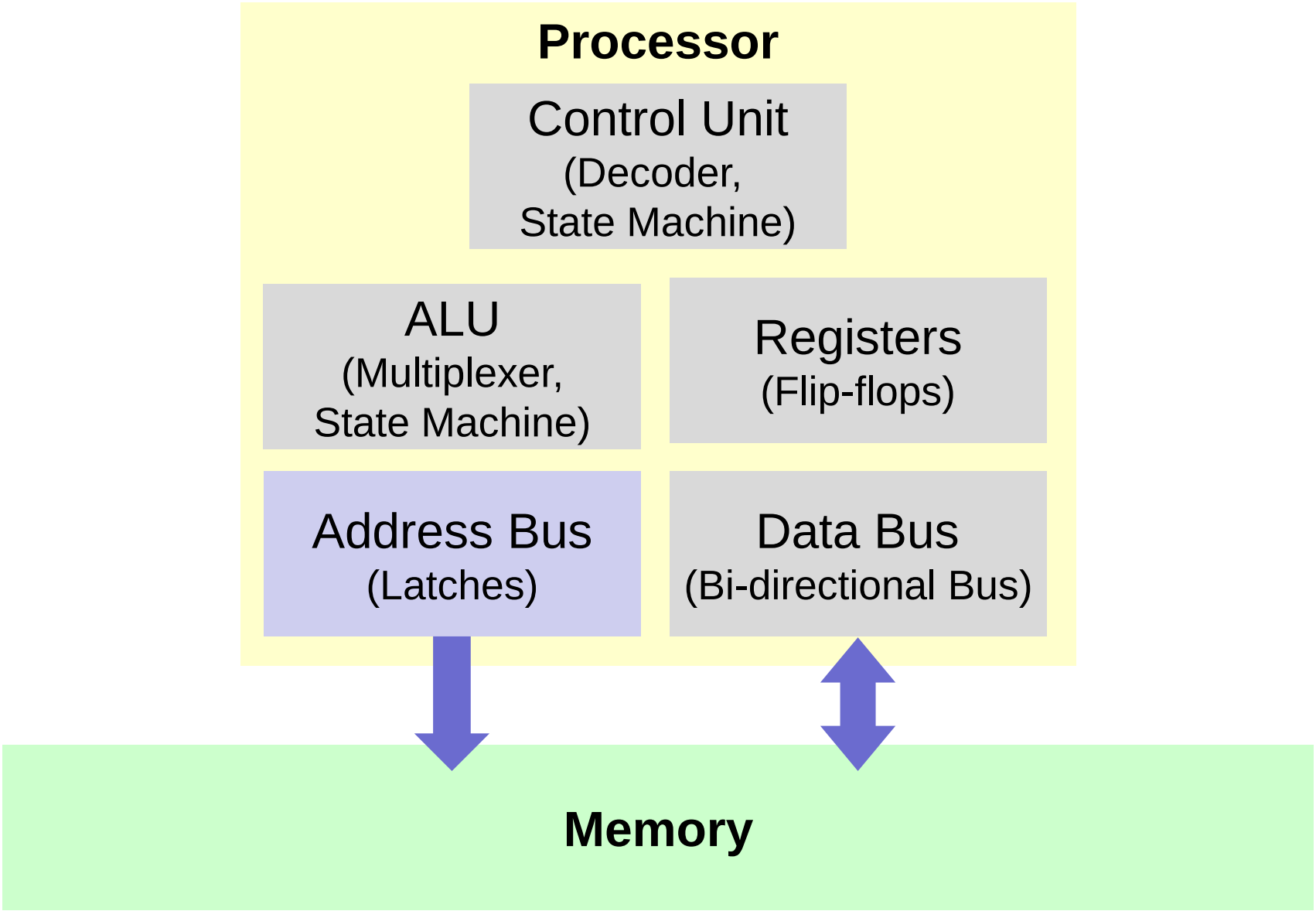
- The main difference between latch and flip-flop:
  - A latch continuously checks input and changes the output whenever there is a change in input.
  - A flip-flop continuously checks input and changes the output only at times determined by the clock signal.
    - That is, a flip flop has a clock signal.

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

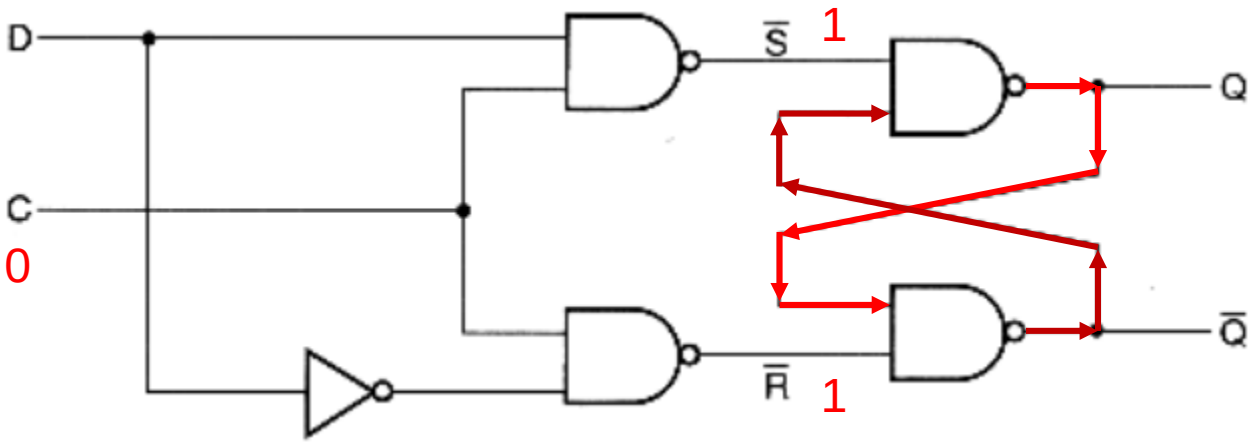


Sequential Circuit: Latch (1/2)

- Latches are asynchronous.
  - The output of the latch only depends on its 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!



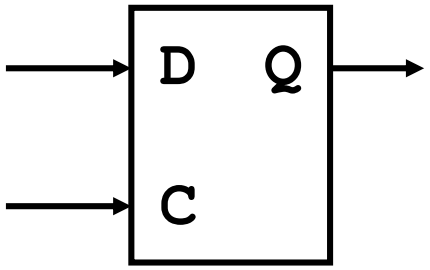
Data need to be held.

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;
```



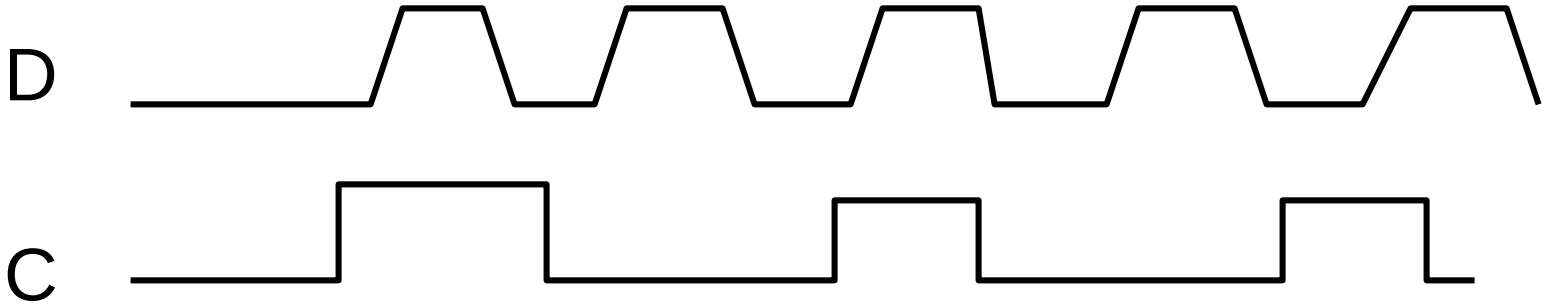
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

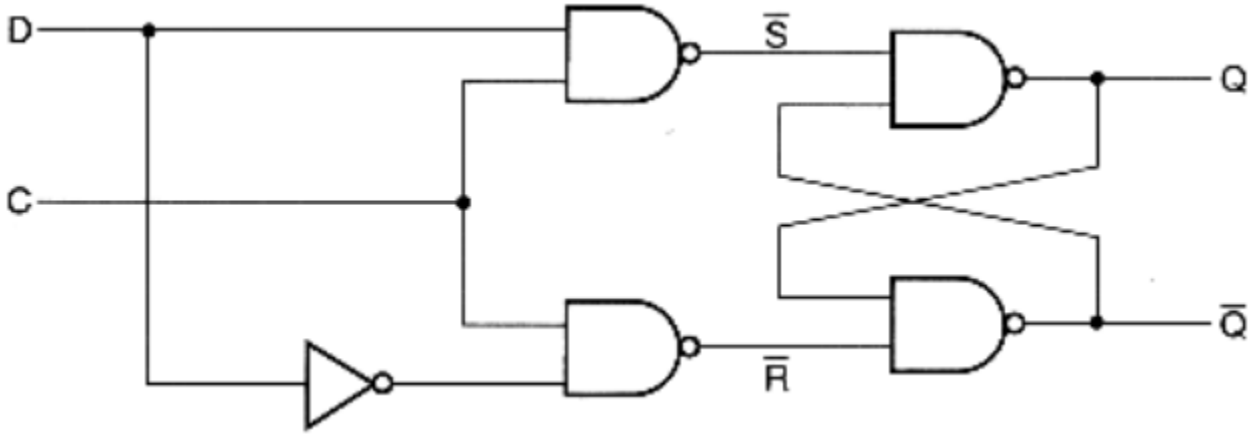
https://www.edgefx.in/digital-electronics-latches-and-flip-flops/

Class Exercise 4.4

Student ID: _____ Date:
Name: _____ _____

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





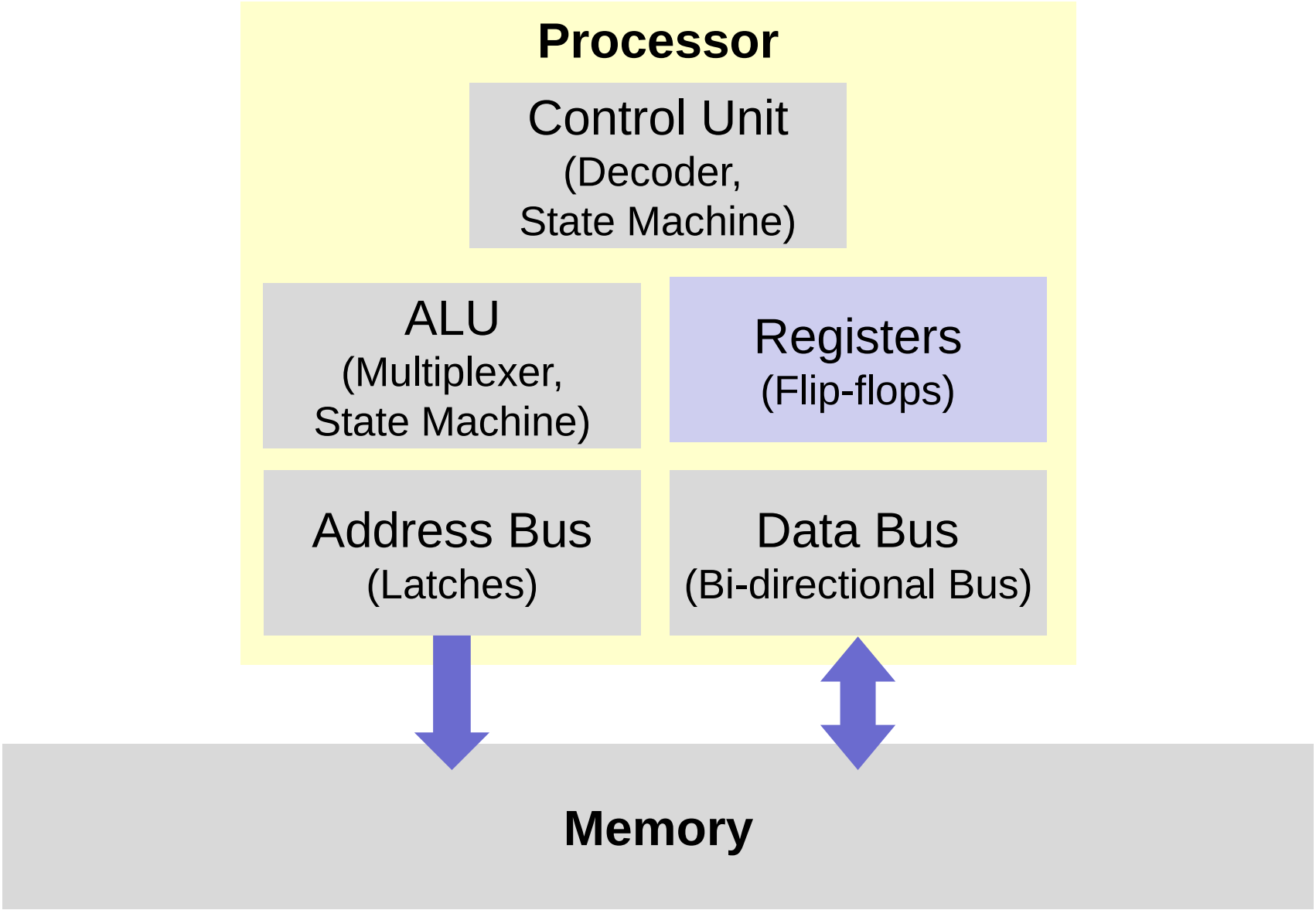
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

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

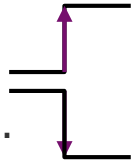
Building Blocks: Flip-flops



Sequential Circuit: Flip-flop

- A Latch is a memory device to store one bit of data.
  - It has no CLOCK signal.
  - It changes output only in response to data input.
  - The value is set asynchronously.

- A Flip-flop (FF) is a clock-controlled memory device for storing one bit of data.
  - Different from a Latch, it has a CLOCK control signal input.
  - It stores the input value (i.e., low or high) and outputs the stored value only in response to the CLOCK signal.
    - The output Q can follow the input D in two ways:
      - **Positive-edge-triggered**: At every **L to H** transition of CLOCK.
      - **Negative-edge-triggered**: At every **H to L** transition of CLOCK.

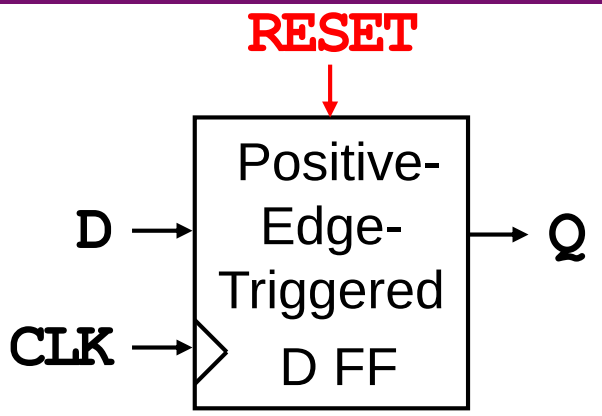


  - The value can be reset asynchronously or synchronously.

http://macao.communications.museum/eng/exhibition/secondfloor/moreinfo/FlipFlop.html

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:

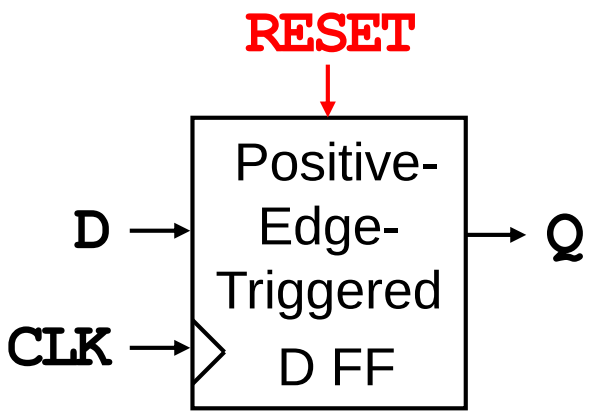
```
...
 9   process(CLK, RESET) -- sensitivity list
10   begin
11      if (RESET = '1') then
12         Q <= '0'; -- Reset Q
13      elsif CLK = '1' and CLK'event then
14         Q <= D; -- Q follows input D
15      end if;
        -- no change (so has memory)
16   end process;
...
```

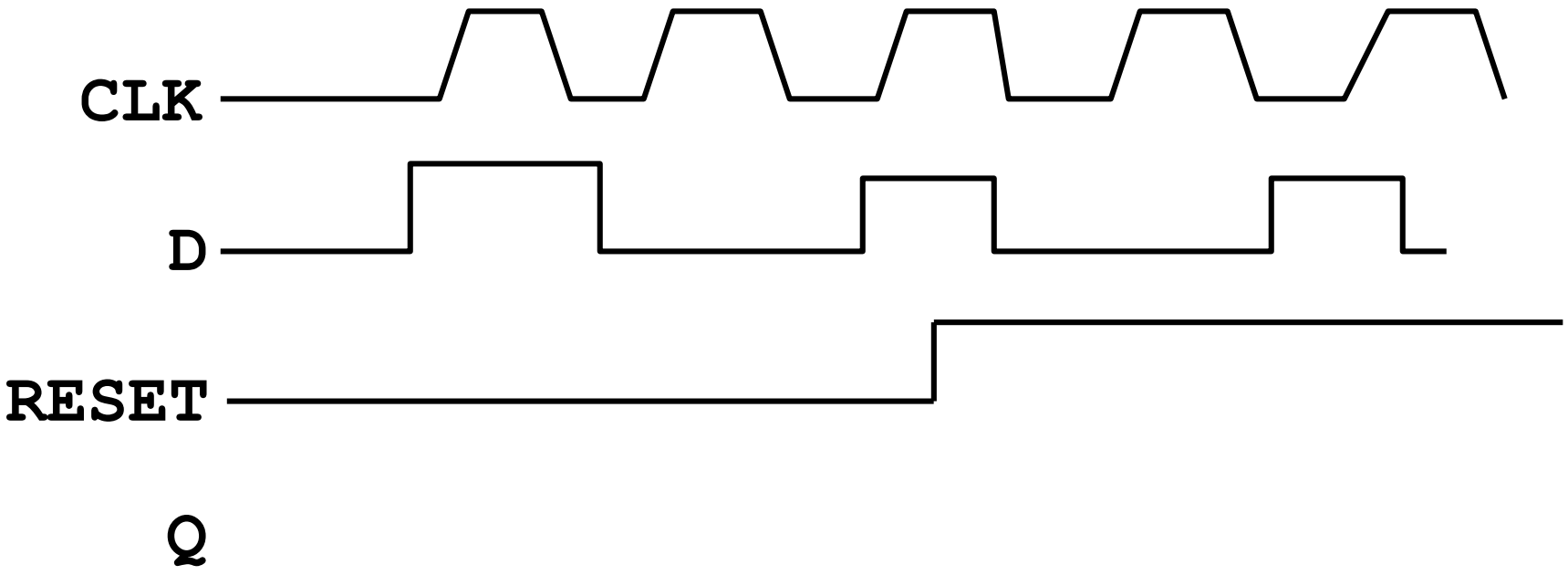
  - When will line 9 be executed?
    Answer: _____
  - Which signal is more "powerful"? **CLK** or **RESET**?
    Answer: _____

Class Exercise 4.6

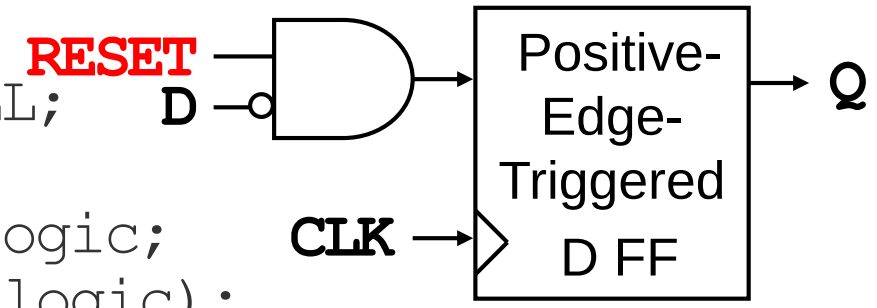
Student ID: _____ Date:
Name: _____ _____

- 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;
```

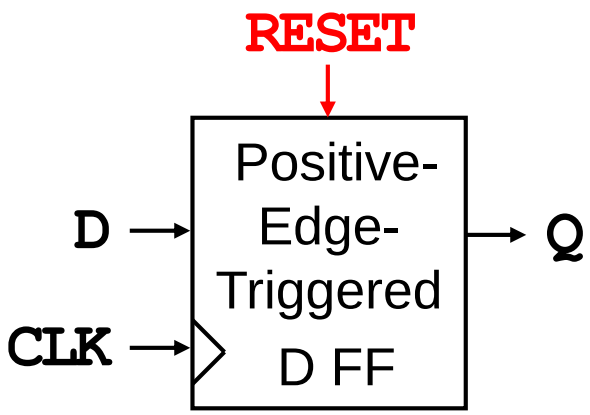
← **RESET** can be removed (why?)

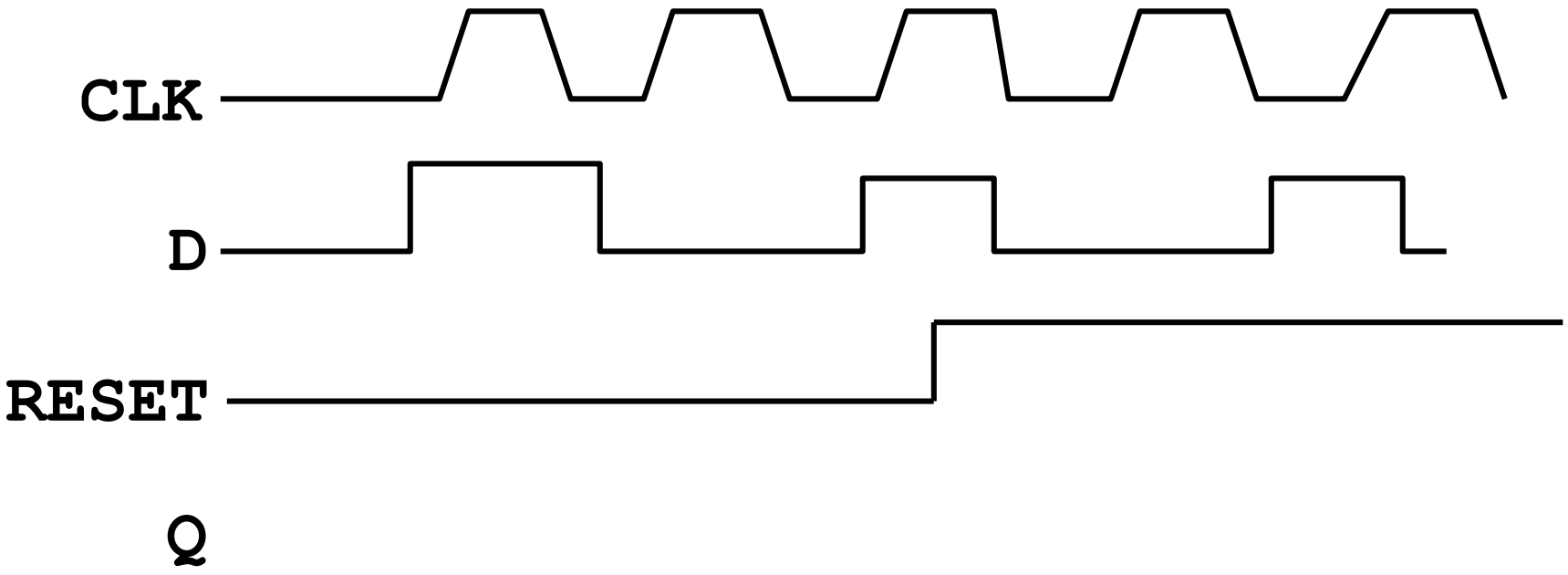
← Positive-edge-triggered

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.





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

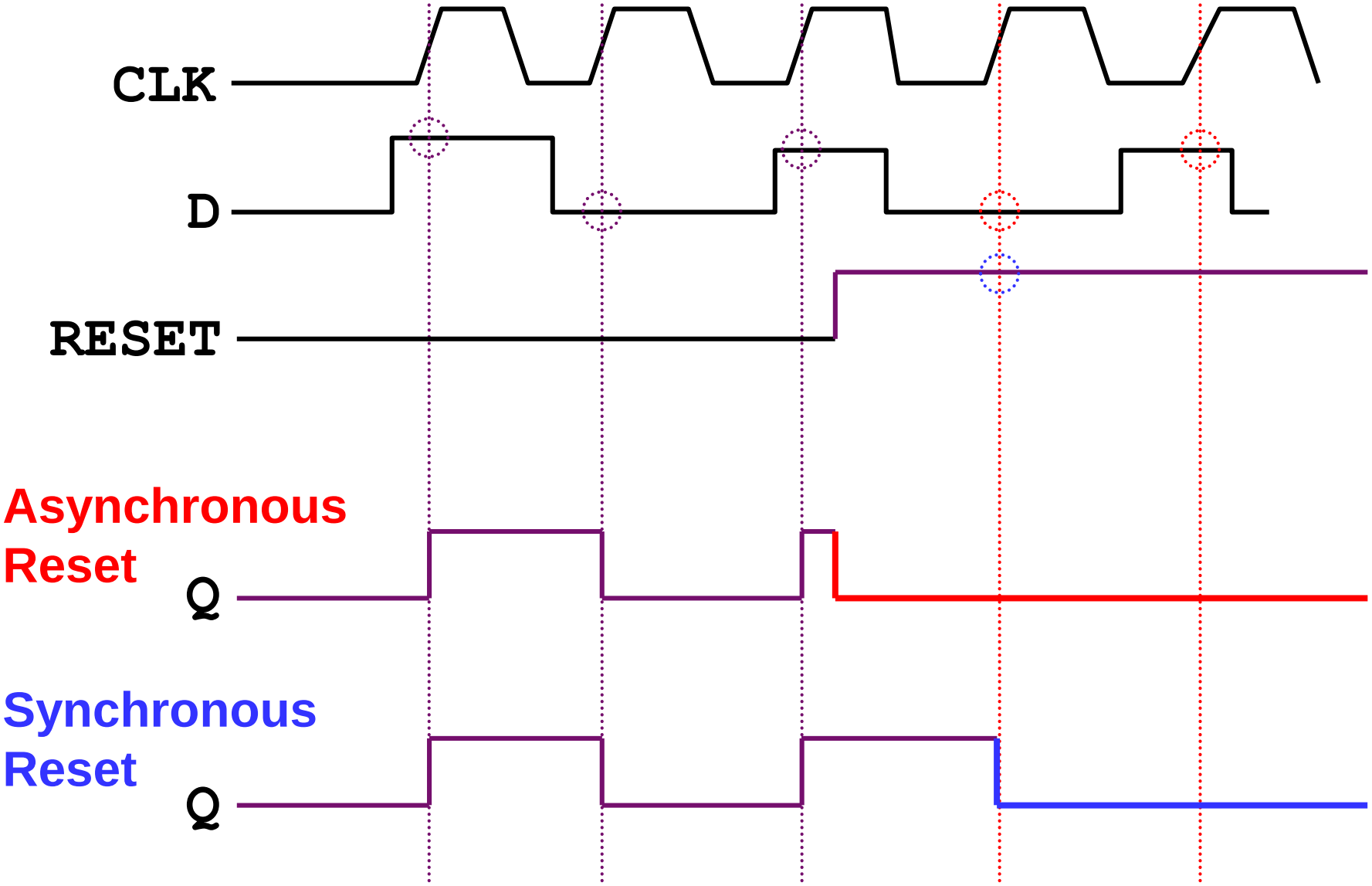
- The order of the statements inside the process determines asynchronous reset or synchronous reset
  - **Asynchronous Reset** (check **RESET** first!)

```
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!