# Digital Logic Recap

ECSE 324: Computer Organization

Fall 2023 - Tutorial 1

Revision History: J. Li (2023), M. Jalaleddine (2022)

# Outline

- Boolean Algebra
- Combinational Circuits
- Sequential Circuits

# Boolean Algebra

- Operations: OR ($+$);

  AND ($\times$) (omit $\times$ for saving space);

  NOT ($'$ or $\bar{\ }$).

- Laws: Commutative ($a+b=b+a, ab=ba$);

  Distributed ($a\times(b+c)=ab+ac, a+bc=(a+b)(a+c)$);

  Associative ($a+(b+c)=(a+b)+c, a(bc)=(ab)c$);

  Identity ($a+0=a, a\times1=a$);

  Complement ($a+a'=1, a\times a'=0$).

# Boolean Algebra

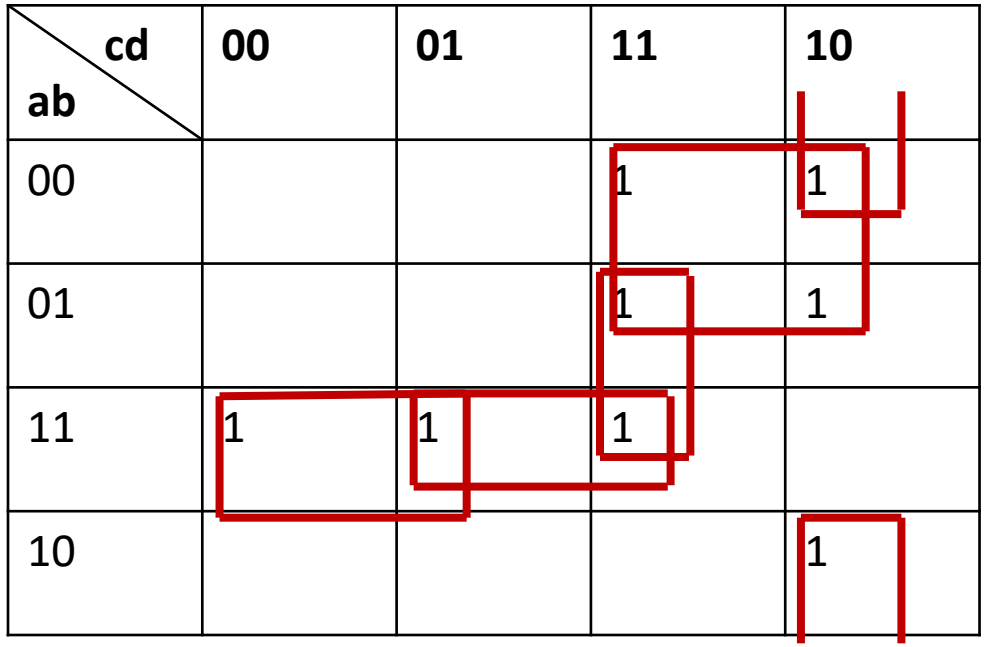
- DeMorgan's Laws: $(a + b + c + \cdots)' = a' \times b' \times c'$ …;

$$(a \times b \times c \times \cdots)' = a' + b' + c' + \cdots.$$

- Sum of Product (SOP): $(a \times b) + (c \times d)$.

- Product of Sum (POS): $(a + b) \times (c + d)$.

# Boolean Algebra



| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | 1 | 1 |
| 01 | | | 1 | 1 |
| 11 | 1 | 1 | 1 | |
| 10 | | | | 1 |

$$f = abc'd' + abc'd + abcd + a'bcd + a'b'cd$$
$$+a'b'cd' + a'bcd' + ab'cd'$$
$$= a'c + b'cd' + bcd + abd + abc'$$

Why SOP? Why POS?

Simplify the logical expression!

# Boolean Algebra

Simplify the following expression:

$$f = (a' + b' + c + d)\times(a' + b' + c + d')\times(a' + b' + c' + d')\times(a + b' + c' + d')\times$$
$$(a + b + c' + d')\times(a + b + c' + d)\times(a + b' + c' + d)\times(a' + b + c' + d)$$

# Boolean Algebra

| cd / ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$$f = (a' + b' + c + d)\times(a' + b' + c + d')\times (a' + b' + c' + d')\times(a + b' + c' + d')\times (a + b + c' + d')\times(a + b + c' + d)\times (a + b' + c' + d)\times(a' + b + c' + d)$$

# Boolean Algebra

| cd / ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$$f = (a' + b' + c + d)\times(a' + b' + c + d')\times (a' + b' + c' + d')\times(a + b' + c' + d')\times (a + b + c' + d')\times(a + b + c' + d)\times (a + b' + c' + d)\times(a' + b + c' + d)$$

1, $f' = abc'd' + abc'd + abcd + a'bcd + a'b'cd + a'b'cd' + a'bcd' + ab'cd'$

# Boolean Algebra

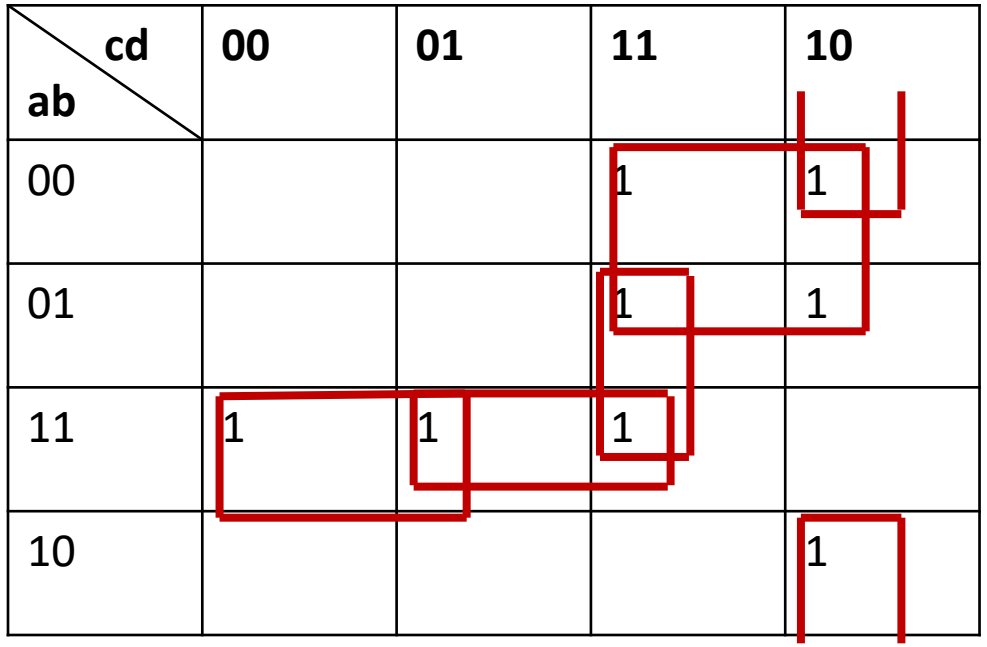
| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | 1 | 1 |
| 01 | | | 1 | 1 |
| 11 | 1 | 1 | 1 | |
| 10 | | | | 1 |

$f = (a' + b' + c + d)\times(a' + b' + c + d')\times$
$(a' + b' + c' + d')\times(a + b' + c' + d')\times$
$(a + b + c' + d')\times(a + b + c' + d)\times$
$(a + b' + c' + d)\times(a' + b + c' + d)$

1, $f' = abc'd' + abc'd + abcd + a'bcd + a'b'cd +$
$a'b'cd' + a'bcd' + ab'cd'$;

2, Label terms in $f'$ in table using 1;

# Boolean Algebra



| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | 1 | 1 |
| 01 | | | 1 | 1 |
| 11 | 1 | 1 | 1 | |
| 10 | | | | 1 |

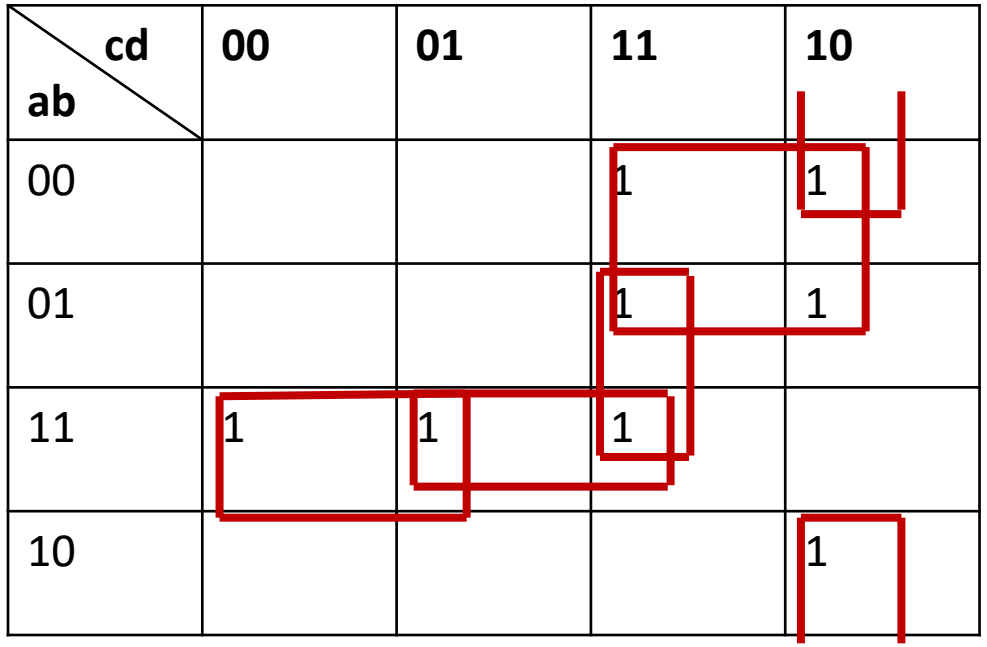
$f = (a' + b' + c + d) \times (a' + b' + c + d') \times (a' + b' + c' + d') \times (a + b' + c' + d') \times (a + b + c' + d') \times (a + b + c' + d) \times (a + b' + c' + d) \times (a' + b + c' + d)$

1, $f' = abc'd' + abc'd + abcd + a'bcd + a'b'cd + a'b'cd' + a'bcd' + ab'cd'$;

2, Label terms in $f'$ in table using 1;

3, Simplify $f'$ based on 1: $f' = a'c + b'cd' + bcd + abd + abc'$;

# Boolean Algebra



| ab \ cd | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | 1 | 1 |
| 01 | | | 1 | 1 |
| 11 | 1 | 1 | 1 | |
| 10 | | | | 1 |

$f = (a' + b' + c + d) \times (a' + b' + c + d') \times (a' + b' + c' + d') \times (a + b' + c' + d') \times (a + b + c' + d') \times (a + b + c' + d) \times (a + b' + c' + d) \times (a' + b + c' + d)$

1, $f' = abc'd' + abc'd + abcd + a'bcd + a'b'cd + a'b'cd' + a'bcd' + ab'cd'$;

2, Label terms in $f'$ in table using 1;

3, Simplify $f'$ based on 1: $f' = a'c + b'cd' + bcd + abd + abc'$;

4, Apply DeMorgan's Law to recover $f$: $f = (a + c') \times (b + c' + d) \times (b' + c' + d') \times (a' + b' + d') \times (a' + b' + c)$.

# Boolean Algebra

Given two binary 8-bit vectors $\boldsymbol{x} = \{x_8 \ldots x_1\}$ , $\mathbf{y} = \{y_8 \ldots y_1\}$

Use AND, OR, and NOT operations to:

1. Get the vector made up of the most significant 4 bits of $\boldsymbol{x}$ and the least significant 4 bits of $\boldsymbol{y}$

2. Set the even bits of $\boldsymbol{x}$ to zero

3. Set the even bits of $\boldsymbol{y}$ to one

# Boolean Algebra

Given two binary 8-bit vectors $\boldsymbol{x} = \{x_8 \dots x_1\}$ , $\mathbf{y} = \{y_8 \dots y_1\}$

Use AND, OR, and NOT operations to:

1. Get the vector made up of the most significant 4 bits of $\boldsymbol{x}$ and the least significant 4 bits of $\boldsymbol{y}$

   $1, \boldsymbol{f_1} = \mathbf{AND}(\boldsymbol{x}, 11110000)$ ; $2, \boldsymbol{f_2}, = \mathbf{AND}(y, 00001111)$; $3, \boldsymbol{f} = \mathbf{OR}(\boldsymbol{f_1}, \boldsymbol{f_2})$.

2. Set the even bits of $\boldsymbol{x}$ to zero

   $\boldsymbol{f} = \mathbf{AND}(\boldsymbol{x}, 01010101)$

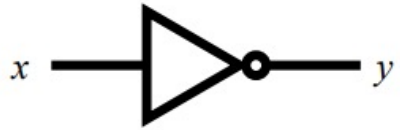
3. Set the even bits of $\boldsymbol{y}$ to one

   $\boldsymbol{f} = \mathbf{OR}(\boldsymbol{y}, 10101010)$

# Combinational Circuits

Logic gates:

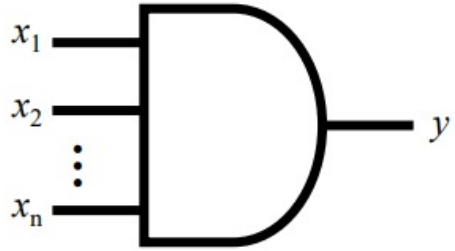
NOT gate: $y = x'$



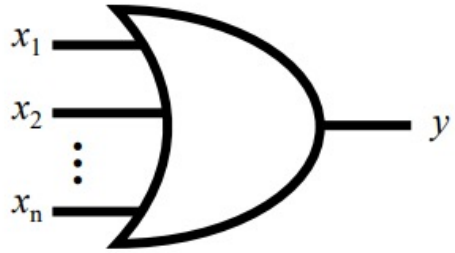

AND gate: $y = x_1 \times x_2$


OR gate: $y = x_1 + x_2$


$n$ inputs AND gates: $y = x_1 \times x_2 \times \cdots$

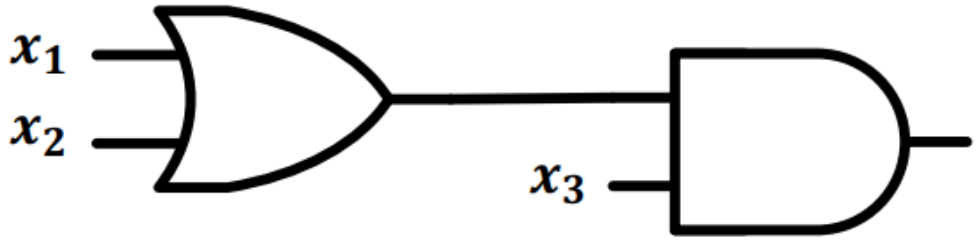



$n$ inputs OR gates: $y = x_1 + x_2 + \cdots$

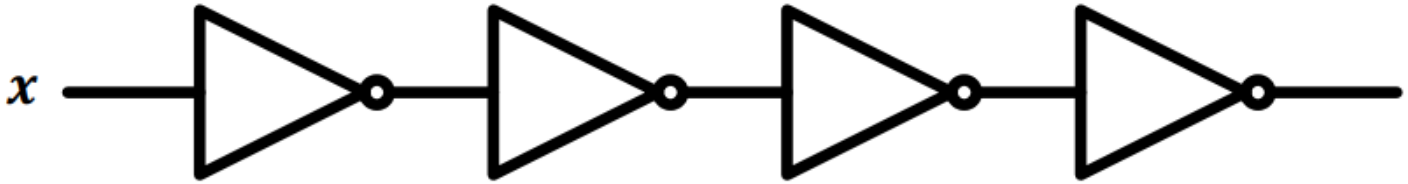


# Combinational Circuits

$(x1 + x2) \times x3$:




$(((x')')')'$:

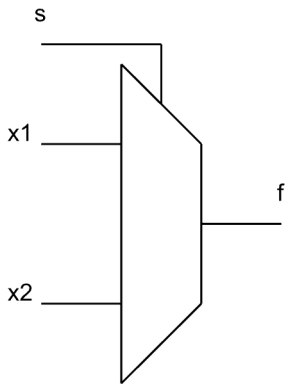


# Combinational Circuits

What is the difference between a demultiplexer and a decoder?

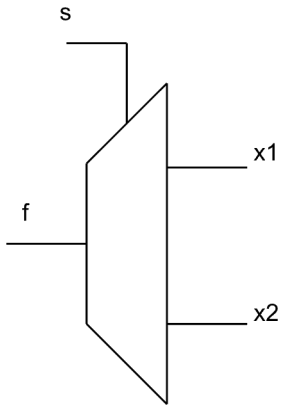
# Combinational Circuits

Multiplexer:




| s | f |
|---|---|
| 0 | x1 |
| 1 | x2 |

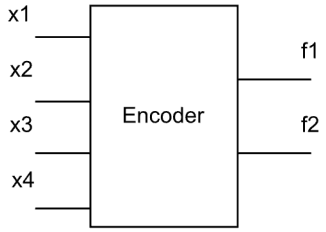
Demultiplexer:




| s | x1 | x2 |
|---|---|---|
| 0 | f | |
| 1 | | f |

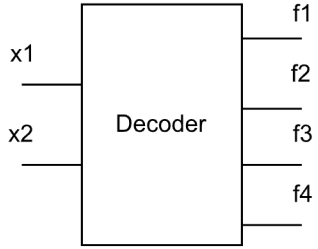
# Combinational Circuits

Encoder:




| x1 | x2 | x3 | x4 | f1 | f2 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

Decoder:




| x1 | x2 | f1 | f2 | f3 | f4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

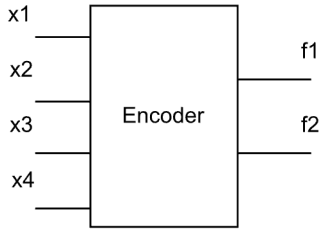
# Combinational Circuits

What is the difference between a demultiplexer and a decoder?

Answer: A demultiplexer steers the input to the desired output port;
A decoder interprets patterns of input bits and generate a unique output.

# Combinational Circuits

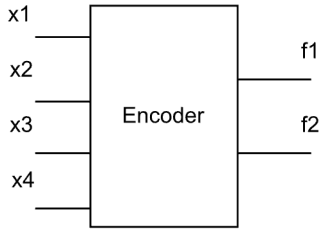
Design a combinational circuit for the following one-hot encoder:




| x1 | x2 | x3 | x4 | f1 | f2 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

# Combinational Circuits

Design a combinational circuit for the following one-hot encoder:




| x1 | x2 | x3 | x4 | f1 | f2 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

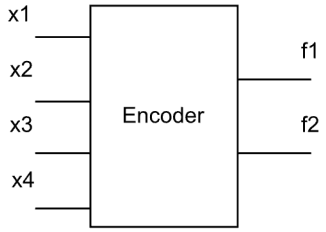
| x1x2 \ x3x4 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | 1 | x | 1 |
| 01 | 0 | x | x | x |
| 11 | x | x | x | x |
| 10 | 0 | x | x | x |

x: don’t care.

$f1 = x1' \times x2'$

# Combinational Circuits

Design a combinational circuit for the following one-hot encoder:




| x1 | x2 | x3 | x4 | f1 | f2 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

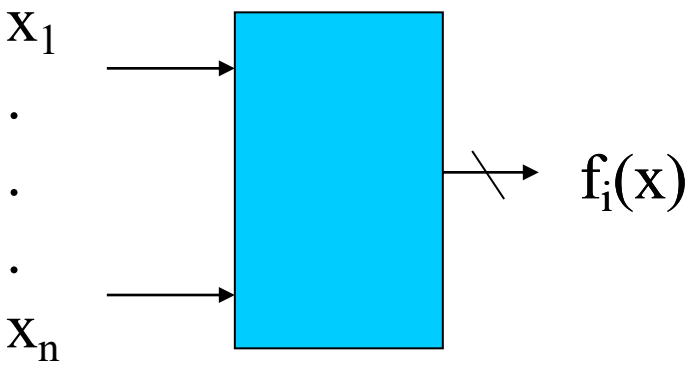
| x1x2 \ x3x4 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | 1 | x | 0 |
| 01 | 1 | x | x | x |
| 11 | x | x | x | x |
| 10 | 0 | x | x | x |

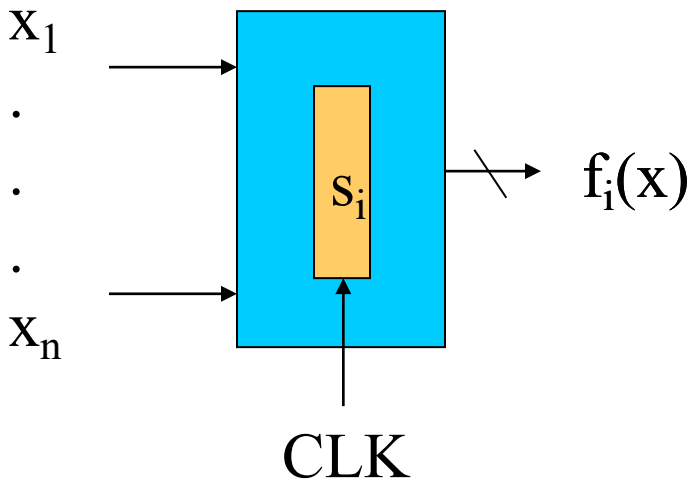
x: don’t care.

$$f2 = x1' \times x3'$$

# Sequential Circuits

Combinatorial vs Sequential:




Combinational: $y_i = f_i(x_1,..,x_n)$

Sequential: 1) Memory 2) Time Steps (Clock)

$y_i^t = f_i(x_1^t,\ldots,x_n^t, s_1^t, \ldots,s_m^t)$

$S_i^{t+1} = g_i(x_1^t,\ldots,x_n^t, s_1^t,\ldots,s_m^t)$

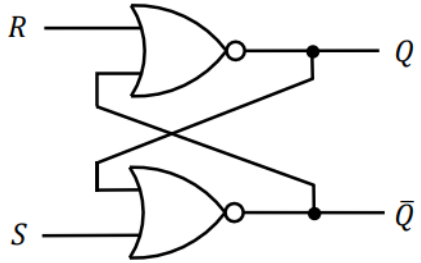
# Sequential Circuits

What is the difference between a latch and a flip-flop?

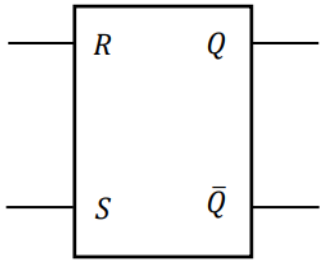
# Sequential Circuits

SR Latch:

Circuit




Graphical symbol

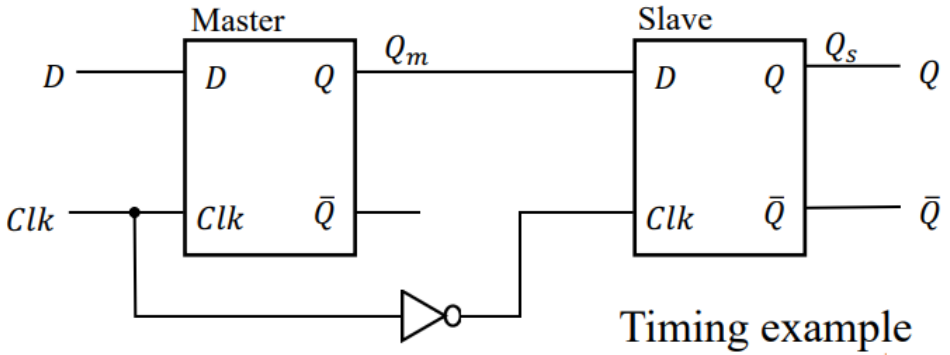



Characteristic table

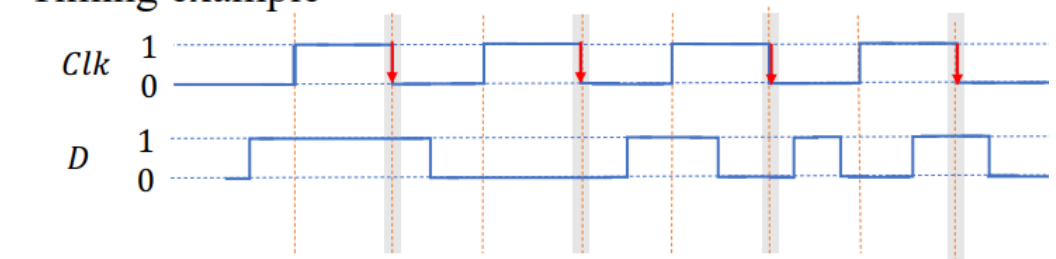
| $S$ | $R$ | $Q(t+1)$ |
|---|---|---|
| 0 | 0 | $Q(t)$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $d$ |

# Sequential Circuits

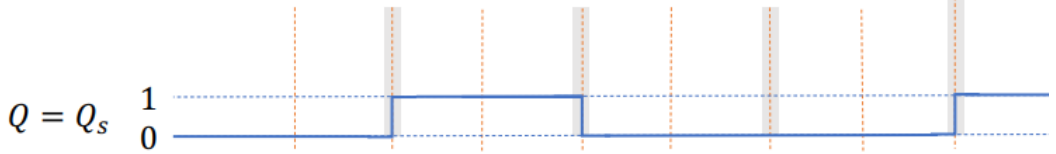
Flip-flop:




Timing example




$Q$ latches on the input $D$ precisely at the negative edge of $Clk$




**Registers are just many FFs!**

# Sequential Circuits

What is the difference between a latch and a flip-flop?

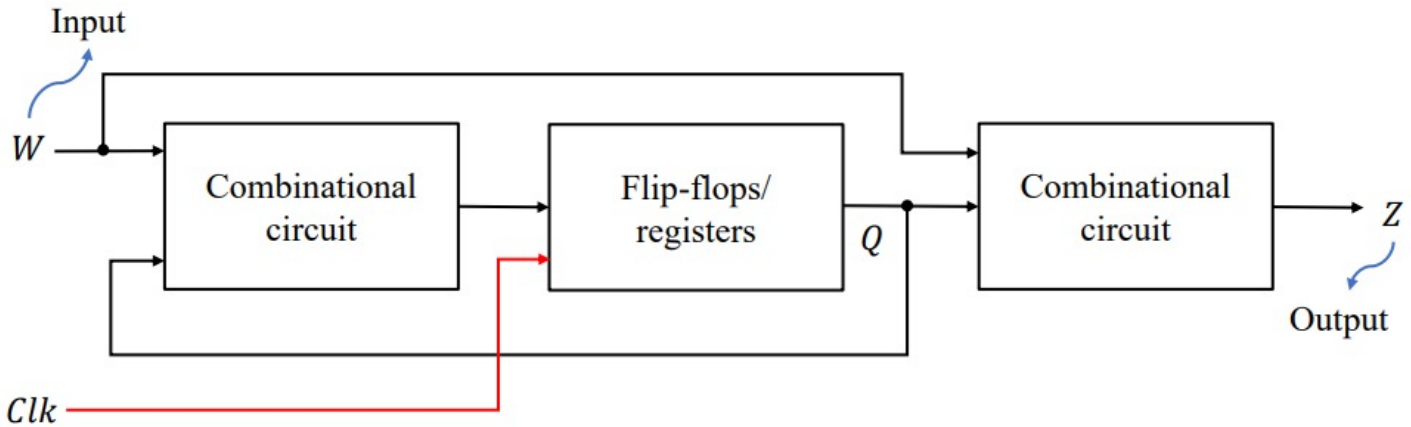
Answer: The output of the latch is level sensitive;
The output of the flip-flop is edge sensitive.

# Sequential Circuits

What is the difference between different designs (Mealy and Moore Machine) of a finite state machine (FSM)?

# Sequential Circuits

FSM (Mealy):

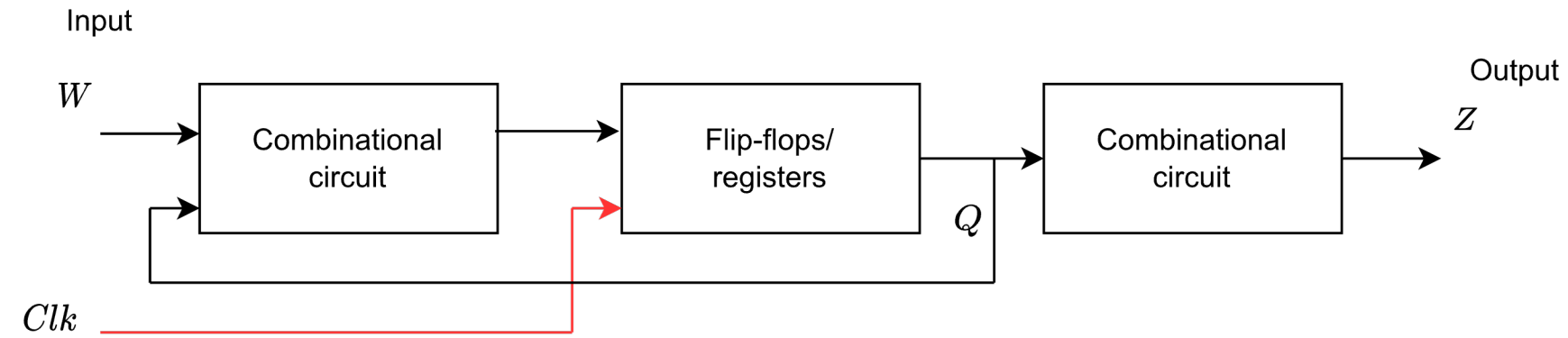



FSM is a sequential circuit

Synchronous:
Clock controls/times the operation of the circuit

# Sequential Circuits

FSM (Moore):




Synchronous:
clock controls/times the operation of the circuit

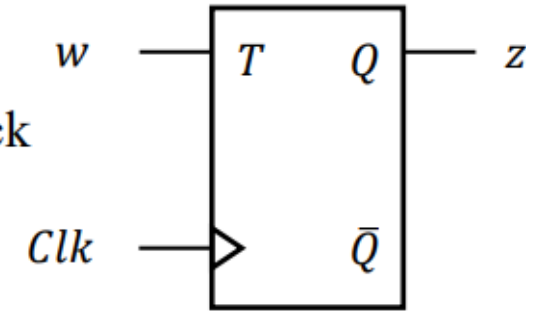
# Sequential Circuits

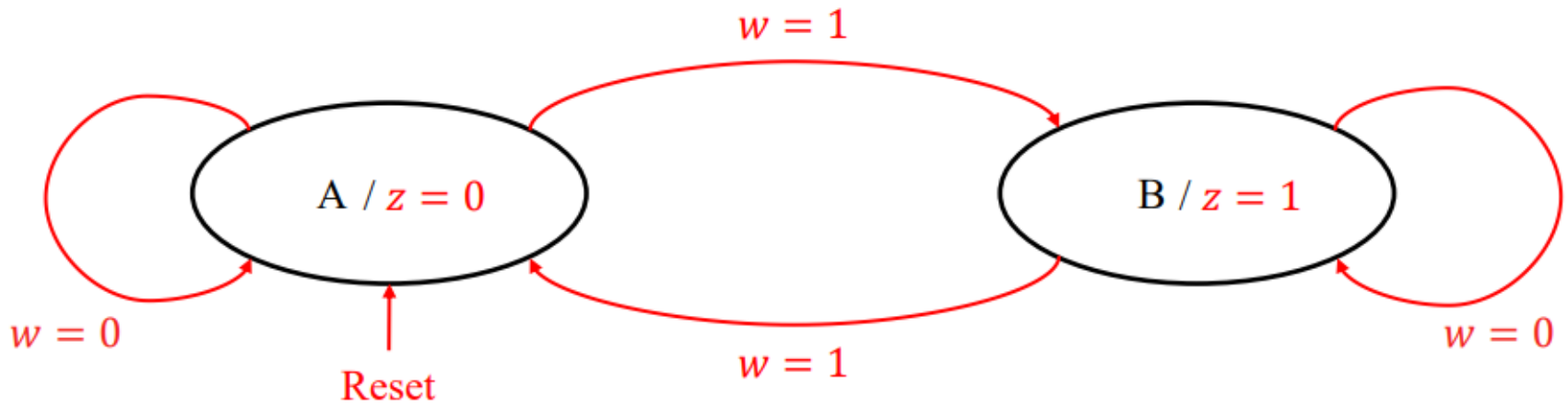
What is the difference between different designs (Mealy and Moore Machine) of a finite state machine (FSM)?

Answer: Mealy's next state and output depend on the present state and the input;
Moore's next state depends on the present state and the input, while the output only depends on the present state.

# Sequential Circuits

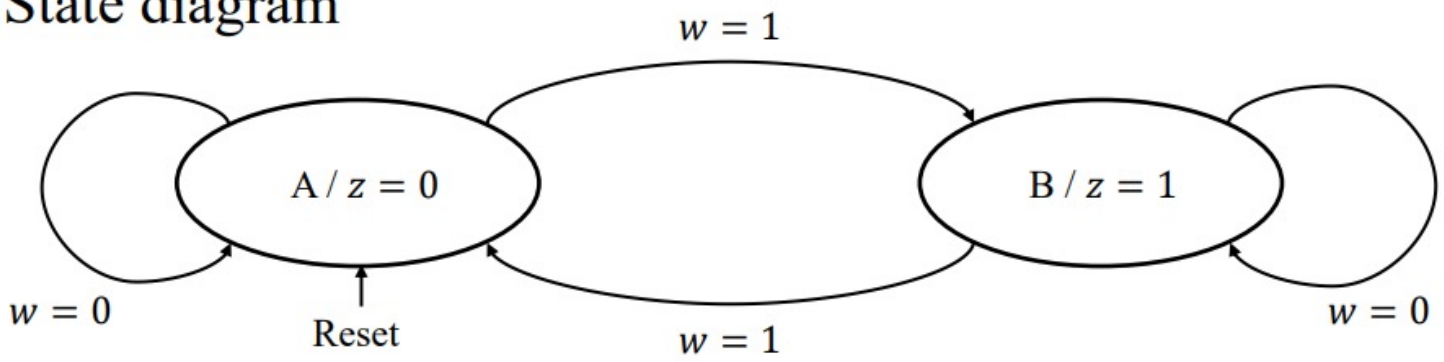
- The FSM (TFF) can be in one of two states
  - State A: $Q = 0$
  - State B: $Q = 1$
- Depending on the input, the FSM may change states at each positive edge of the clock
  - When $w = 0$ the state does not change
  - When $w = 1$ the state toggles
- The output $z$ depends on the state only → Moore FSM
- Initial state: the state at which the FSM "starts"



# State Diagram and Table

State diagram




State table

| Present state | Next state | | Output $z$ |
|---|---|---|---|
| | $w = 0$ | $w = 1$ | |
| A | A | B | 0 |
| B | B | A | 1 |

# Sequential Circuits

Write down a state transition diagram for a sequence detector that detects two consecutive bit 1s in the input data stream. 1 bit arrives at the input per clock cycle. For example, a bit 0 arrives at the clock period 1, a bit 1 arrives at the clock period 2, a bit 1 arrives at the clock period 3, and the detector outputs the signal for successfully detecting two consecutive bit 1s. If the next incoming bit is 1 once two consecutive bit 1s are detected, the detector still outputs the signal for successfully detecting two consecutive bit 1s.

# Sequential Circuits

Write down a state transition diagram for a sequence detector that detects two consecutive bit 1s in the input data stream. 1 bit arrives at the input per clock cycle. For example, a bit 0 arrives at the clock period 1, a bit 1 arrives at the clock period 2, a bit 1 arrives at the clock period 3, and the detector outputs the signal for successfully detecting two consecutive bit 1s. If the next incoming bit is 1 once two consecutive bit 1s are detected, the detector still outputs the signal for successfully detecting two consecutive bit 1s.

Answer: 1, Types of FSM -> Moore;
2, # of state -> 4;
3, State transition diagram ->