Introduction to Computer and Programming

Lecture 11

Yue Zhang
Westlake University

August 1, 2023

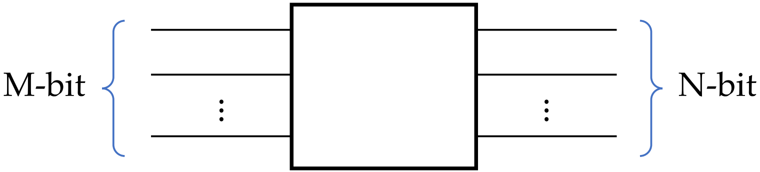
WestlakeNLP

Chapter 11.

Computation Theory

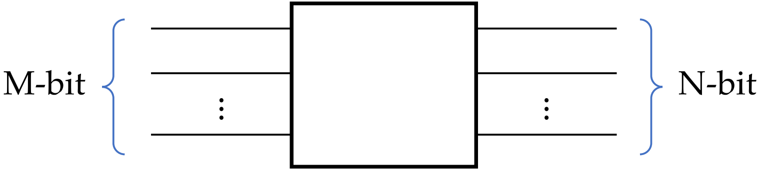
WestlakeNLP

Combinational Logic



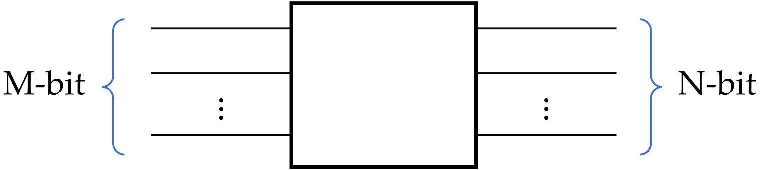
- All M-bit $\rightarrow$ N-bit functions

Combinational Logic



- All M-bit $\rightarrow$ N-bit functions
- But not all functions!
  - what if the input is not bounded? (e.g., large number addition)
  - what if the output is not bounded?

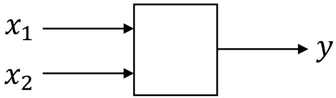
Combinational Logic



- All M-bit $\rightarrow$ N-bit functions
- But not all functions!
    - what if the input is not bounded? (e.g., large number addition)
    - what if the output is not bounded?
- Multi-step Computation

Sequential Logic

- Input – two sequences of bits.
- Output – a sequence of bits.



$$x_1 = [1, 0, 1, 0, 1, 1, 1, \cdots]$$

$$x_2 = [0, 1, 1, 0, 1, 1, 0, \cdots]$$

$$y = [1, 1, 0, 1, 0, 1, 0, \cdots]$$

Adding two numbers from the last bit, remembering carry.

Sequential Logic

- Input – a sequence of M-bit units.
- Output – a sequence of N-bit units.

  e.g., count the number of non-zero values in a sequence.

$$x = [0, 1, 0, 2, 3, \cdots]$$
$$y = [0, 1, 1, 2, 3, \cdots]$$

WestlakeNLP

Sequential Logic

- Input – a sequence of M-bit units.
- Output – a sequence of N-bit units.

  e.g., count the number of non-zero values in a sequence.

  $$x = [0, 1, 0, 2, 3, \cdots]$$
  $$y = [0, 1, 1, 2, 3, \cdots]$$

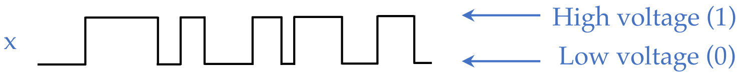
- Indexing notation

  x[t], y[t], e.g., x[1]=0, x[4]=2, y[3]=1

WestlakeNLP

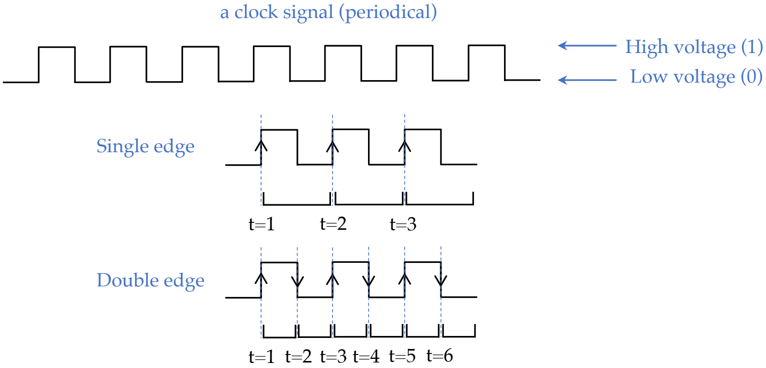
Sequential Logic

- How to model time?
  Can we change x at arbitrary time?



- this 1-bit signal changes whenever it likes.
- but we do not know if there are consecutive 0s!

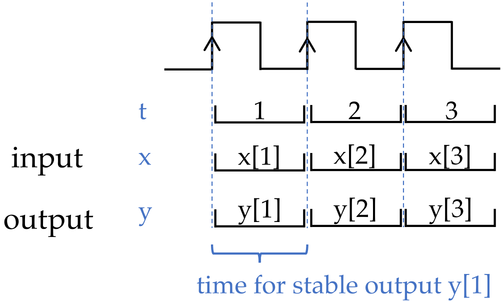
- How to model time?



- There are devices for generating waves.

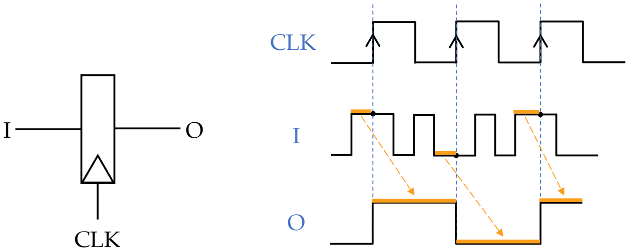
Sequential Logic

- How to integrate clock into computation?



Sequential Logic

- Edge-trigger flipflops (ETFF)

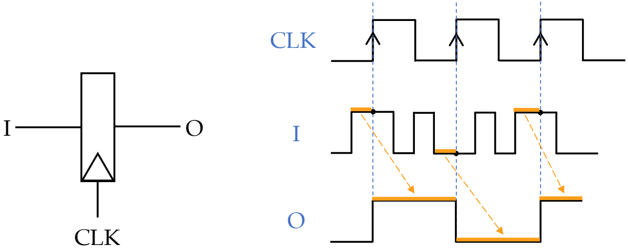


- Set up I before clock tick.
- Hold I after clock tick.

WestlakeNLP

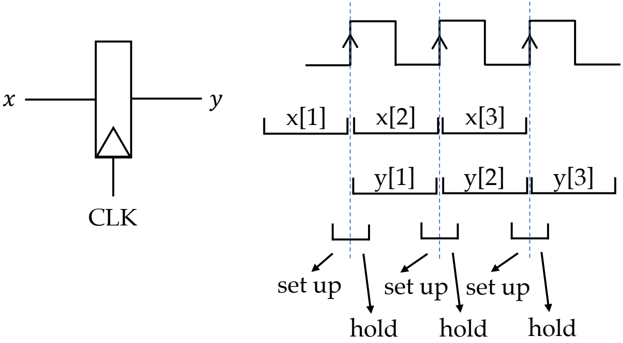
Sequential Logic

- Edge-trigger flipflops (ETFF)



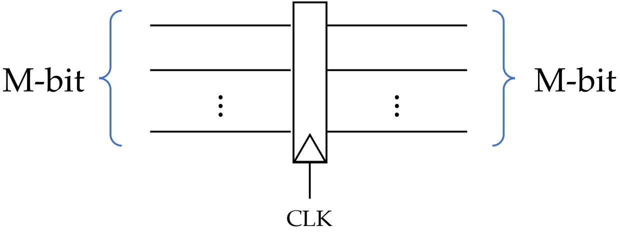
- Set up I before clock tick.
- Hold I after clock tick.
- O samples I at clock tick stage.

- Effective clock



Sequential Logic

- A **register** is a multi-bit ETFF



- It works together with the clock (CLK) to
  1. construct time sequence
  2. ensure output signal stability
- Requires input set up time and hold time.

Sequential Logic

- Build a sequential device.
  count the number of non-zero values in a sequence

$$x = [0, 1, 0, 2, 3, \cdots]$$
$$y = [0, 1, 1, 2, 3, \cdots]$$

- Simplified for illustration.
  - Use only 1-bit output, which is equivalent to indicating whether the count is odd(1) or even(0).
  - Use 2-bit input

$$x = [0, 1, 0, 2, 3, \cdots]$$
$$s = [0, 1, 1, 0, 1, \cdots]$$

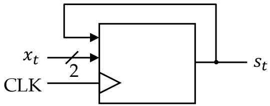
WestlakeNLP

Sequential Logic

- Build a sequential device.

$$\begin{array}{l} \text{current input } x_t \\ \text{current count } s_t \end{array} \longrightarrow \text{next count } s_{t+1} = x_t + s_t$$

x_t	s_t	s_{t+1}
0 0	0	0
	1	1
0 1	0	1
	1	0
1 0	0	1
	1	0
1 1	0	1
	1	0



Sequential Logic

- Build a sequential device.

current input x_t
current count s_t $\longrightarrow$ next count $s_{t+1} = x_t + s_t$

x_t	s_t	s_{t+1}
0 0	0	0
	1	1
0 1	0	1
	1	0
1 0	0	1
	1	0
1 1	0	1
	1	0

$$s_{t+1} = \overline{x_1}\,\overline{x_2}s_t + x_1\overline{x_2}\overline{s_t} + \overline{x_1}x_2\overline{s_t} + x_1x_2\overline{s_t}$$

$$= \overline{x_1}\,\overline{x_2}s_t + (x_1 + x_2)\,\overline{s_t}$$

$$= \overline{(x_1 + x_2)}s_t + (x_1 + x_2)\,\overline{s_t}$$

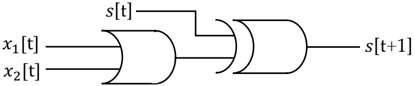
$$= \ (x_1 + x_2)\ XOR\ s_t$$

- Build a sequential device.

current input x_t
current count s_t $\longrightarrow$ next count $s_{t+1} = x_t + s_t$

x_t	s_t	s_{t+1}
0 0	0	0
	1	1
0 1	0	1
	1	0
1 0	0	1
	1	0
1 1	0	1
	1	0

$$
\begin{aligned}
s_{t+1} &= \overline{x_1}\,\overline{x_2}s_t + x_1\overline{x_2}\overline{s_t} + \overline{x_1}x_2\overline{s_t} + x_1x_2\overline{s_t} \\
&= \overline{x_1}\,\overline{x_2}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= \overline{(x_1 + x_2)}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= (x_1 + x_2)\; XOR\; s_t
\end{aligned}
$$



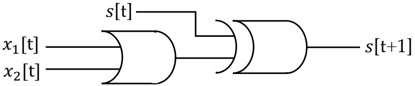
Sequential Logic

- Build a sequential device.

$$\begin{array}{l} \text{current input } x_t \\ \text{current count } s_t \end{array} \longrightarrow \text{next count } s_{t+1} = x_t + s_t$$

x_t	s_t	s_{t+1}
0 0	0	0
	1	1
0 1	0	1
	1	0
1 0	0	1
	1	0
1 1	0	1
	1	0

$$\begin{aligned} s_{t+1} &= \overline{x_1}\,\overline{x_2}s_t + x_1\overline{x_2}\overline{s_t} + \overline{x_1}x_2\overline{s_t} + x_1x_2\overline{s_t} \\ &= \overline{x_1}\,\overline{x_2}s_t + (x_1 + x_2)\,\overline{s_t} \\ &= \overline{(x_1 + x_2)}s_t + (x_1 + x_2)\,\overline{s_t} \\ &= \ (x_1 + x_2)\ XOR\ s_t \end{aligned}$$



How to add clock?

Sequential Logic

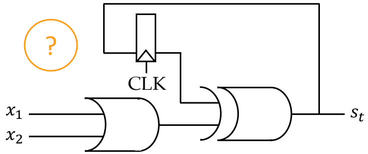
- Build a sequential device.

current input x_t $\longrightarrow$ next count $s_{t+1} = x_t + s_t$

current count s_t

x_t	s_t	s_{t+1}
0 0	0 1	0 1
0 1	0 1	1 0
1 0	0 1	1 0
1 1	0 1	1 0

$$
\begin{aligned}
s_{t+1} &= \overline{x_1}\,\overline{x_2}s_t + x_1\overline{x_2}\,\overline{s_t} + \overline{x_1}x_2\overline{s_t} + x_1x_2\overline{s_t} \\
&= \overline{x_1}\,\overline{x_2}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= \overline{(x_1 + x_2)}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= (x_1 + x_2)\ XOR\ s_t
\end{aligned}
$$



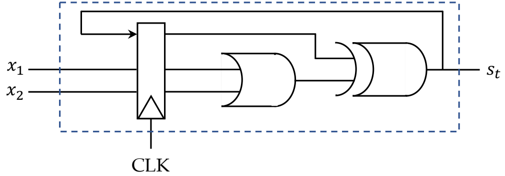
Sequential Logic

- Build a sequential device.

current input x_t
current count s_t $\longrightarrow$ next count $s_{t+1} = x_t + s_t$

x_t	s_t	s_{t+1}
0 0	0	0
	1	1
0 1	0	1
	1	0
1 0	0	1
	1	0
1 1	0	1
	1	0

$$
\begin{aligned}
s_{t+1} &= \overline{x_1}\,\overline{x_2}s_t + x_1\overline{x_2}\overline{s_t} + \overline{x_1}x_2\overline{s_t} + x_1x_2\overline{s_t} \\
&= \overline{x_1}\,\overline{x_2}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= \overline{(x_1 + x_2)}s_t + (x_1 + x_2)\,\overline{s_t} \\
&= (x_1 + x_2)\ XOR\ s_t
\end{aligned}
$$



Sequential Logic

- Build a second sequential device.

Infinite adder

$$1100101 + 1010010$$

$$\Big\Downarrow \quad \text{reverse, bit by bit}$$

$$x_1 = [1, 0, 1, 0, 0, 1, 1]$$

$$x_2 = [0, 1, 0, 0, 1, 0, 1]$$

$$\Big\Downarrow$$

$$y = [1, 1, 1, 0, 1, 1, 0, 1]$$

WestlakeNLP

Sequential Logic

- Build a second sequential device.
  - how to indicate finishing?
  - add a valid signal

$$x_1 = [1, 0, 1, 0, 0, 1, 1, 0, 0, \cdots]$$
$$x_2 = [0, 1, 0, 0, 1, 0, 1, 0, 0, \cdots]$$
$$v_x = [1, 1, 1, 1, 1, 1, 1, 0, 0, \cdots]$$

$$\Downarrow$$

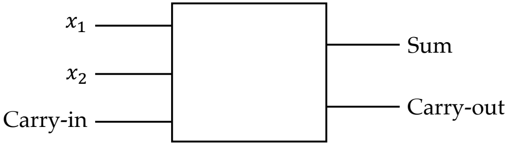
$$y = [1, 1, 1, 0, 1, 1, 0, 1, 0, 0, \cdots]$$
$$v_y = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, \cdots]$$

WestlakeNLP

Sequential Logic

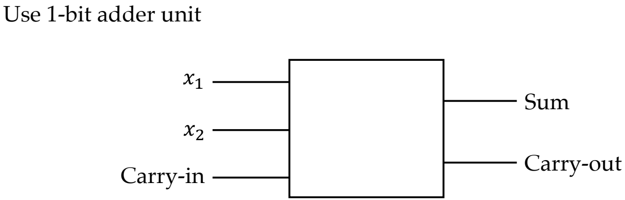
- Build a second sequential device.

Use 1-bit adder unit



Sequential Logic

- Build a second sequential device.



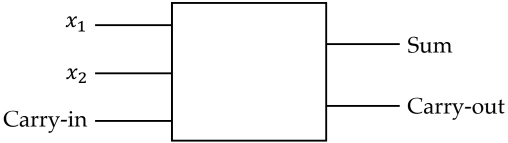
- $y[t+1], CO[t+1] = \text{Adder}(x_1[t], y_1[t], CI[t])$

WestlakeNLP

Sequential Logic

- Build a second sequential device.

Use 1-bit adder unit



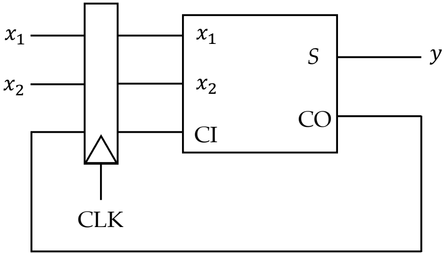
- $y[t+1], CO[t+1] = \text{Adder}(x_1[t], y_1[t], CI[t])$
- How to add clock?

WestlakeNLP

Sequential Logic

- Build a second sequential device.

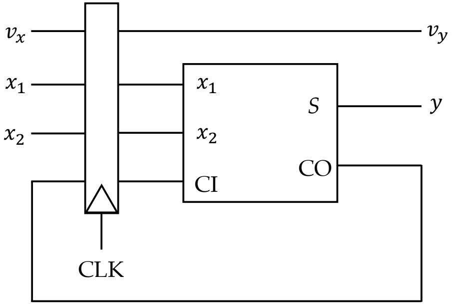
Add clock



WestlakeNLP

- Build a second sequential device.

Add validation



$x_1 = [1, 0, 1, 0, 0, 1, 1]$

$x_2 = [0, 1, 0, 0, 1, 0, 1]$

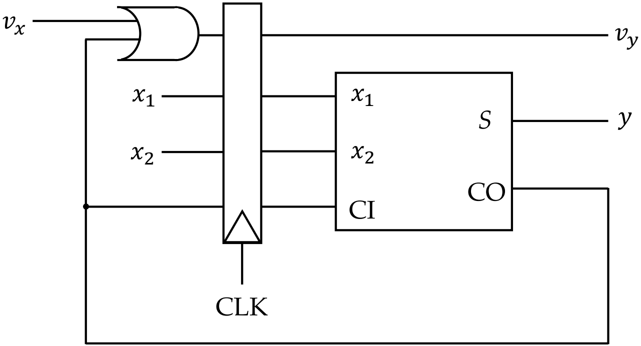
$v_x = [1, 1, 1, 1, 1, 1, 1, 0]$

$y = [1, 1, 1, 0, 1, 1, 0, \boxed{1}]$ ⟶ one more carry!

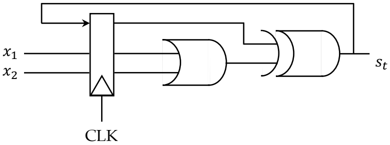
$v_y = [1, 1, 1, 1, 1, 1, 1, 0]$

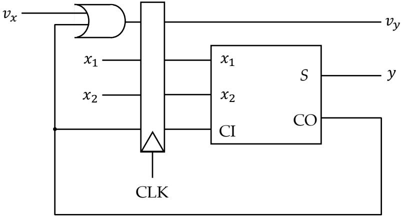
WestlakeNLP

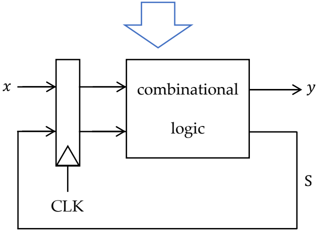
Additional bit



Generalize the two systems







x — input
y — output

$$y[t+1], s[t+1] = f(x[t+1], s[t])$$

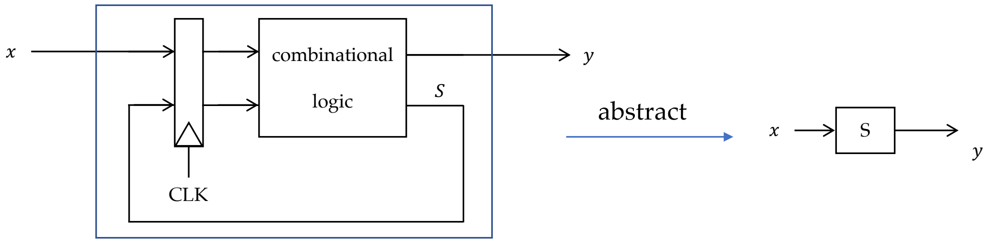
s — state

The feedback info.

WestlakeNLP

Sequential Logic

Finite State Machine



- So initial state S_0

$$y_{t+1}, s_{t+1} = \text{FSM}(x_{t+1}, s_t)$$

WestlakeNLP

Sequential Logic

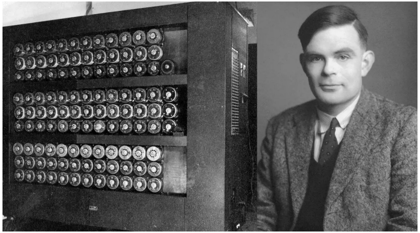
- FSM(sequential logic) can compute more functions than combinational logic.
  - you can turn a sequence of M-bit inputs into one input, by concatenation.
  - thus FSM computes ∞-bit to ∞-bit functions, in theory.
- Does FSM compute all integer functions $y = f(x)$?
- Computation theory investigates this.

WestlakeNLP

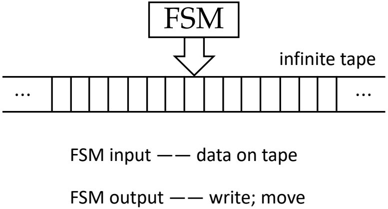
The conclusion

Combinational logic	←	finite input output
FSM	←	finite state
Turing Machines	←	everything a machine can
All integer functions		

Turing Machine



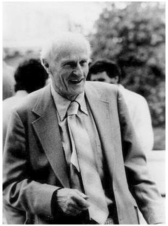
Alan Turing (23 June 1912 – 7 June 1954)



- Three questions:
  1. Turing machine V.S. any machine
  2. Turing machine V.S. all integer functions
  3. Can we make them?

The Church-Turing Thesis

Alonzo Church
(June 14, 1903 – August 11, 1995)



Stephen Cole Kleene
(January 5, 1909 – January 25, 1994)

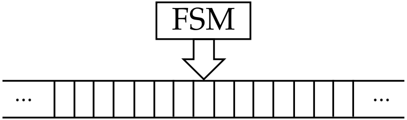
- It has been shown that Turing Machines have the same computation power as recursion(Kleene) and lambda calculus(Church), two other famous computation tools in the 1950s.
- It has been hypothesized that anything computable by a machine is computable by a Turing Machine.

Gödel Incompleteness

Kurt Gödel
April 28, 1906 – January 14, 1978

- It has been **proved** that there are integer functions that Turinng machine cannot compute.

- Turing machines fall back to FSM if the tape is finite!



- Wipes out the advantage of TM over FSM.

Computation Theory

- All computer devices are FSMs!
  - we can increase size
- Turing machines give us important hint.
  - seperate data & code
- Build a programmable FSM.
  - engineers have the stage.

WestlakeNLP