信息科学与技术学院
School of Information Science and Technology

CS 110
Computer Architecture
Digital Circuits and Systems

Instructors:

Siting Liu & Chundong Wang

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2024/index.html

School of Information Science and Technology (SIST)

ShanghaiTech University

2024/3/28

Administratives

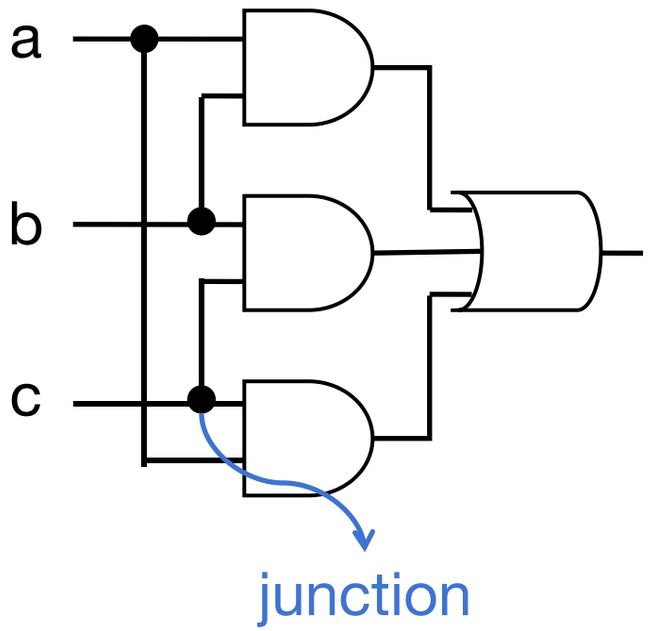
- Lab 5 will be available, please prepare in advance!
- HW 3 will be available soon, ddl April 9th, start early!
- Proj1.1 ddl approaching, April 8th
- Proj1.2 will be released next week
- Future discussion (teaching center 301) schedule:
  - This Friday (Mar. 29th) on CALL/RISC-V, useful for Proj1.1, covered by TA Chen Suting.
  - Next Monday (April 1st) on CALL and digital circuit basics covered by TA Chen Suting and Yang Chao.
  - Next Friday (April 5th) no discussion (QingMing holiday).
  - April 7th (班) discussion on digital circuit basics by TA Yang Chao.
  - April 8th (tentative mid-term I in that week), mid-term review by Yang Chao.
  - After that, the same content for Friday and the next Monday.

Outline

- Digital system
- Combinational logics
- **State elements**
  - **Flip-flops & registers**
  - **Finite state machine (FSM)**
- **Timing constraints**

Warm-up

- What does the following logic circuit do? (Select all that apply)



- A. Output 1 if a=b=1
- B. Output 1 if b=c=1
- C. Output 1 if a=c=1
- D. Output 1 if a=b=c=1
- E. Output 1 if at least two of `a,b,c` is 1
- F. Output the “majority” bit of `a,b,c`
- G. None of the above

Synchronous Digital System (SDS)

- A system that processes digital signals (0s and 1s)
- Synchronous digital systems consist of two basic types of circuits.
  - Combinational logic circuits
    - The outputs sorely depend on the input
    - No way to store information
  - State Elements (**this lecture**)
    - Circuits that store information
    - E.g., **registers** and memory
    - Finite state machine
    - Timing constraints

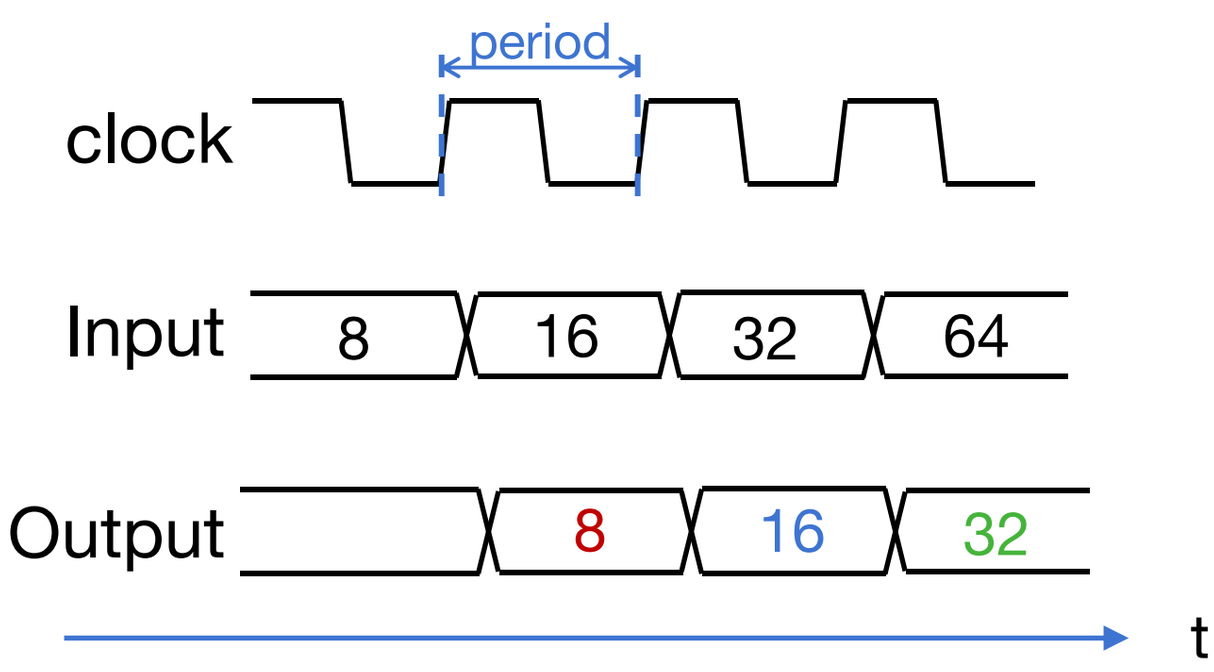
- Our Goal: Implement a RISC-V processor as a synchronous digital system.
- This SDS should have the capabilities to execute RISC-V instructions.

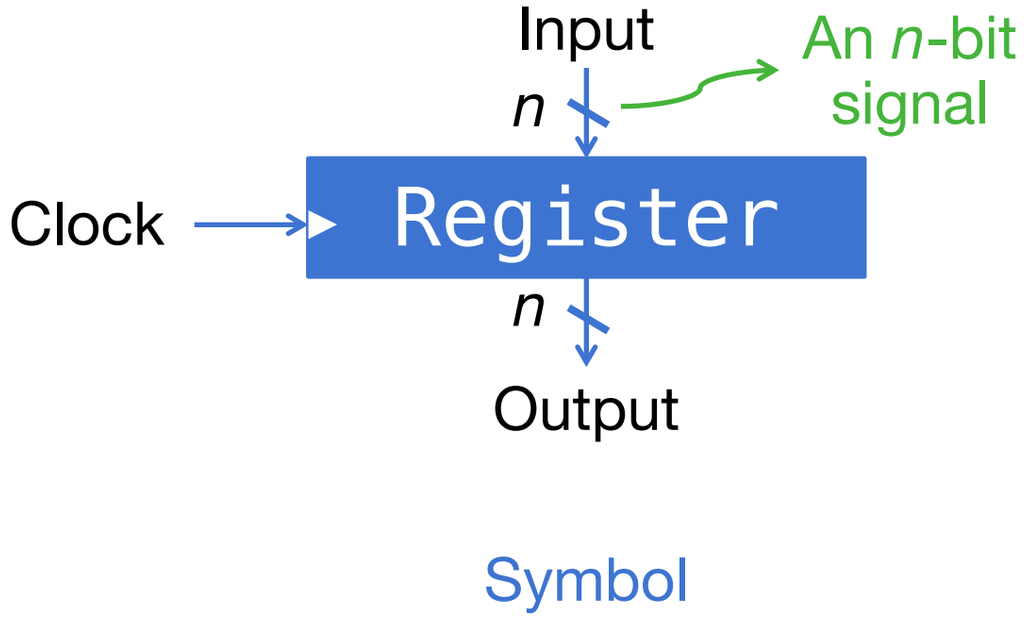
Registers

- A digital element that can store and hold state
- State change governed by a special signal called “clock” (or trigger)
- Sample input on trigger (can be the positive or negative edge of the clock)
- At all other times, ignore the input
- Output follows the value stored

default in this course

- Timing diagram describes its behavior

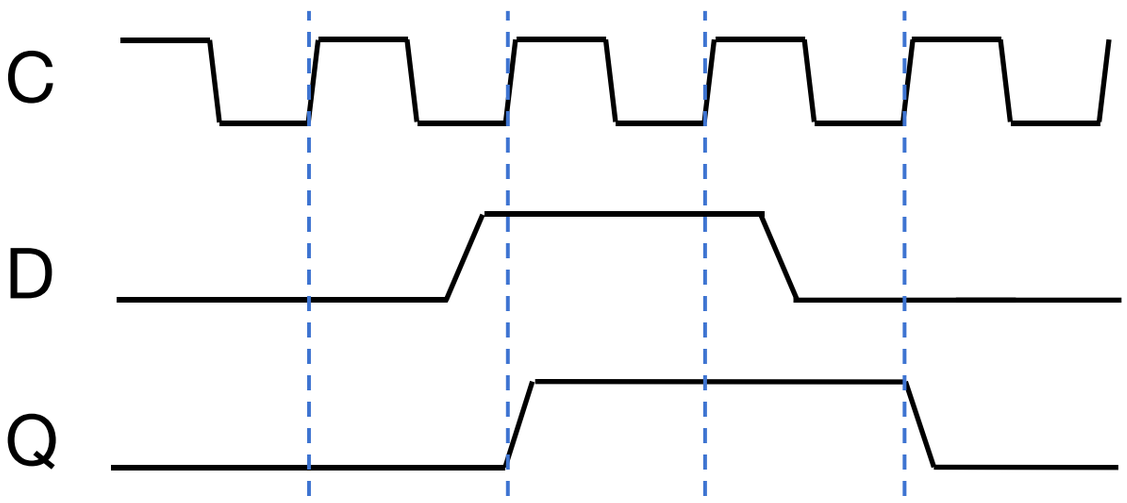


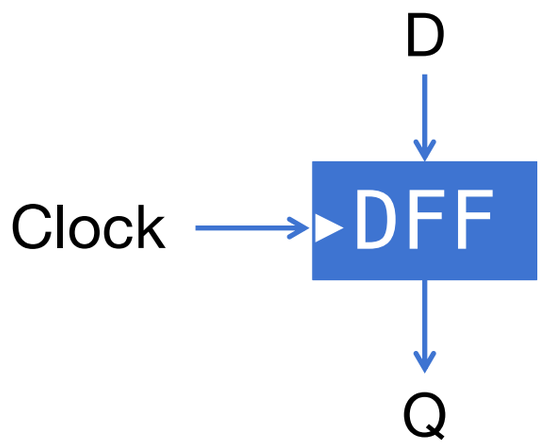


Symbol

Inside a Register

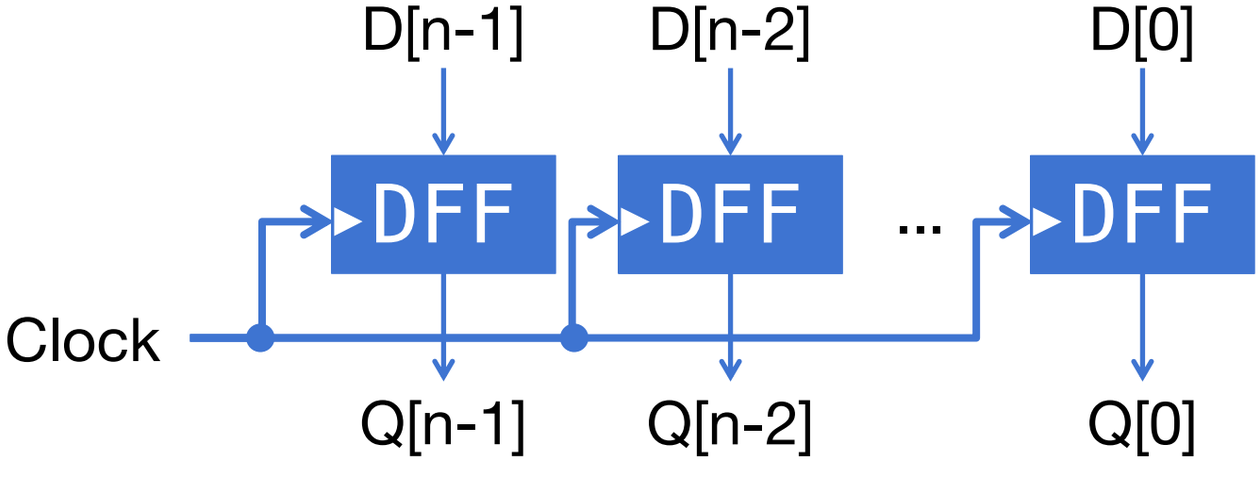
- A register can be implemented by multiple D flip-flops (DFFs)
- Each DFF can store 1 bit (behavior the same as a register)
- Timing diagram

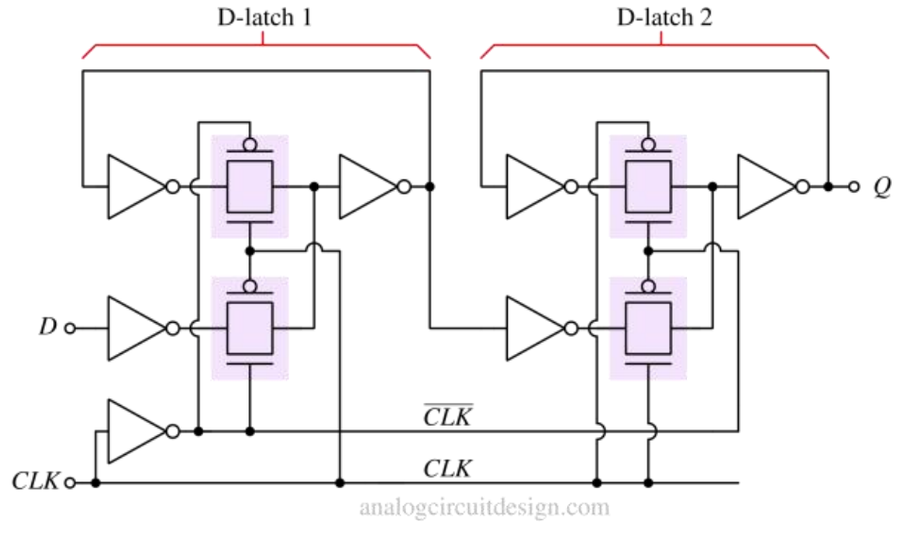




- The DFFs share one clock signal to synchronize

`n-bit register (a possible implementation)`

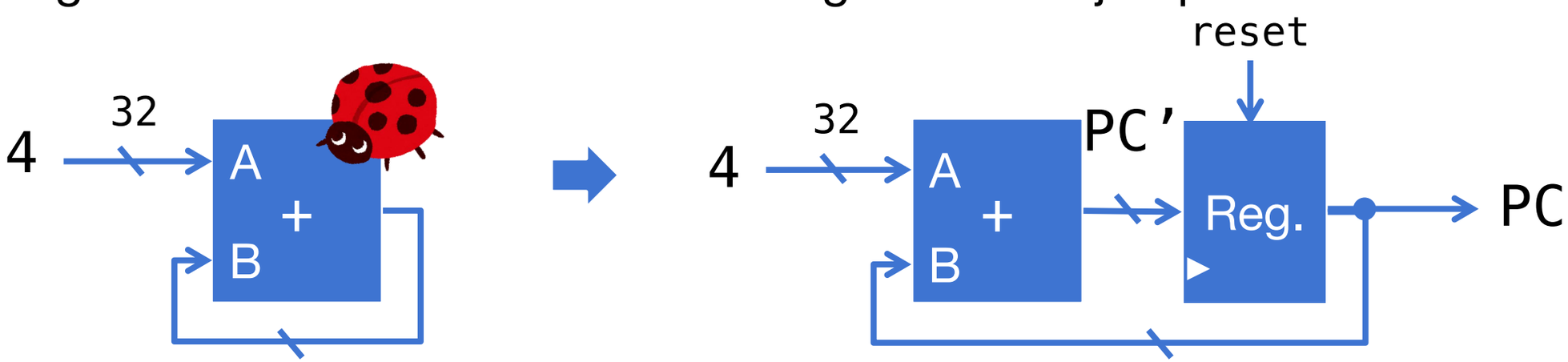




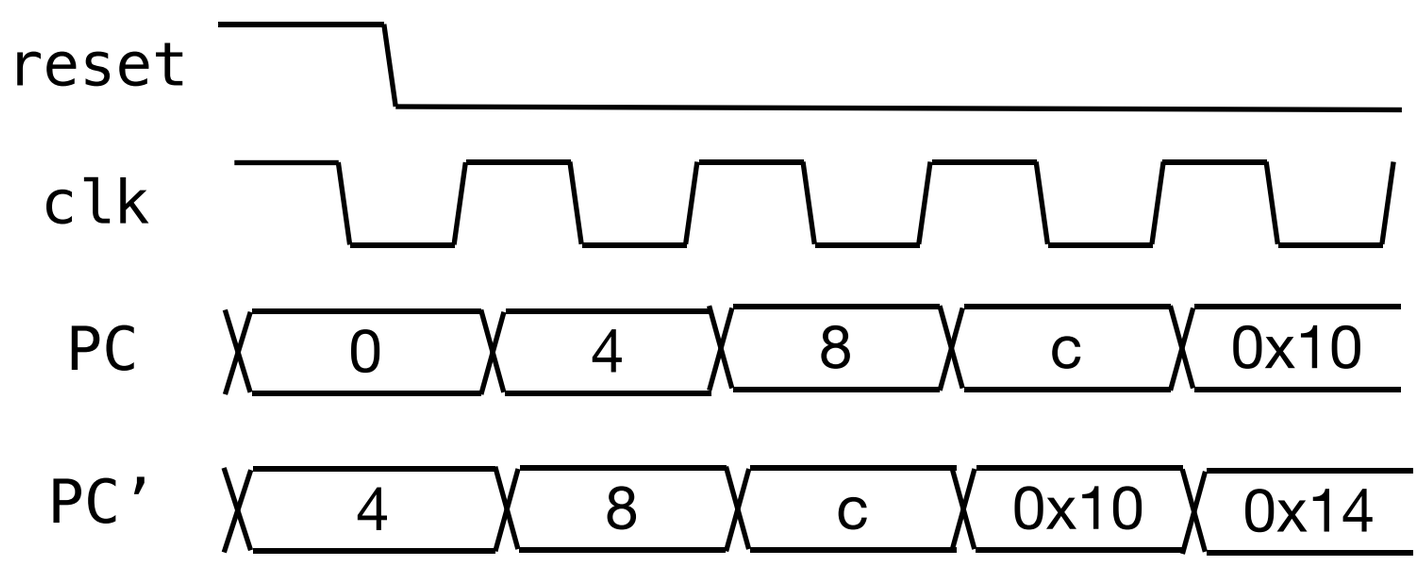
A possible transistor-level implementation of a DFF (optional)

Combine Combinational Circuits and DFFs

- Synchronous digital circuit can have feedback, e.g., iterative accumulator
  - e.g. PC = PC + 4 without considering branch or jump

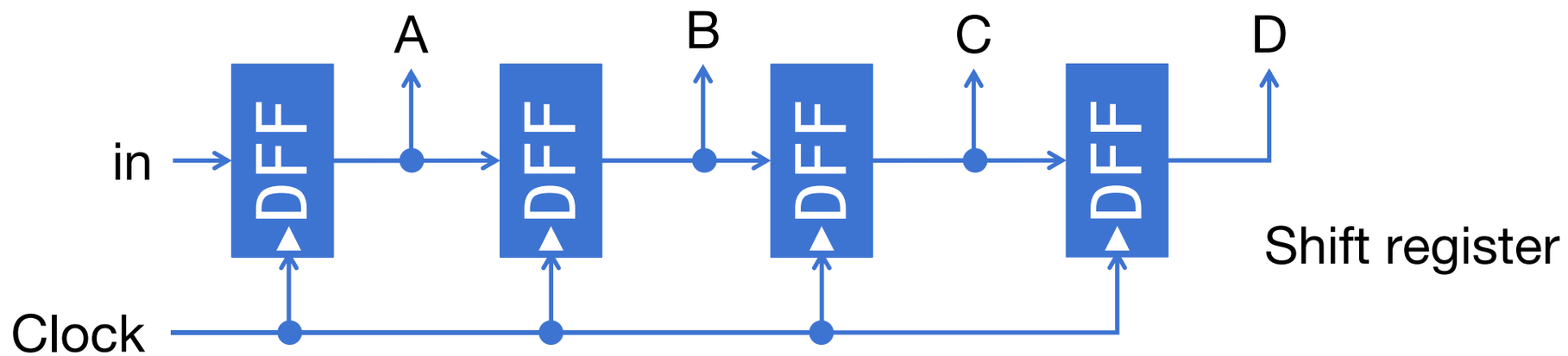


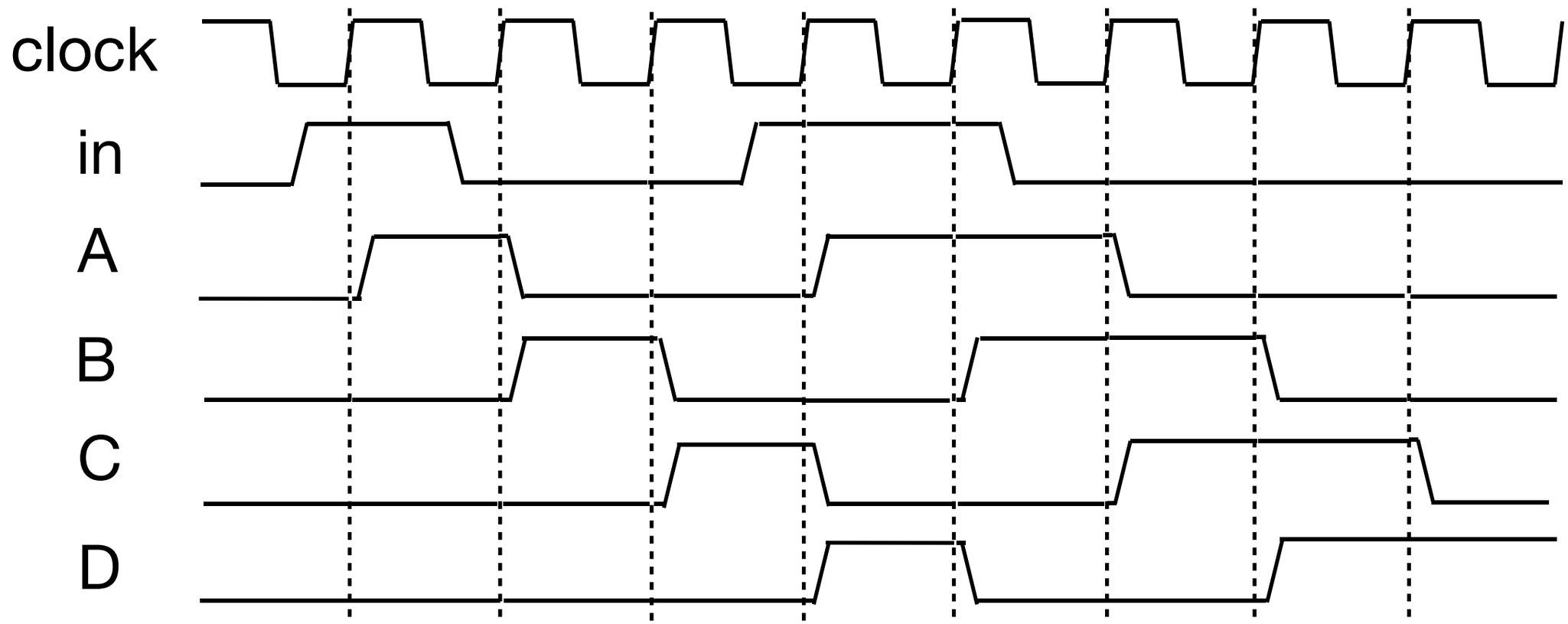
- Timing diagram



Combine Combinational Circuits and DFFs

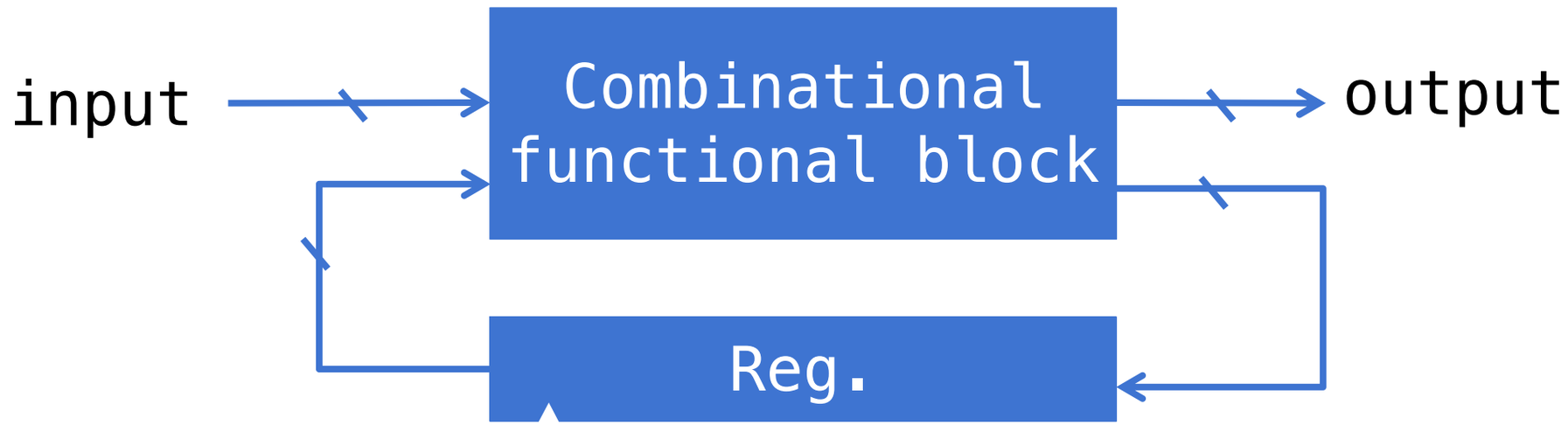
- Synchronous digital circuit that can have multiple stages of DFFs or registers, and all the registers share one clock



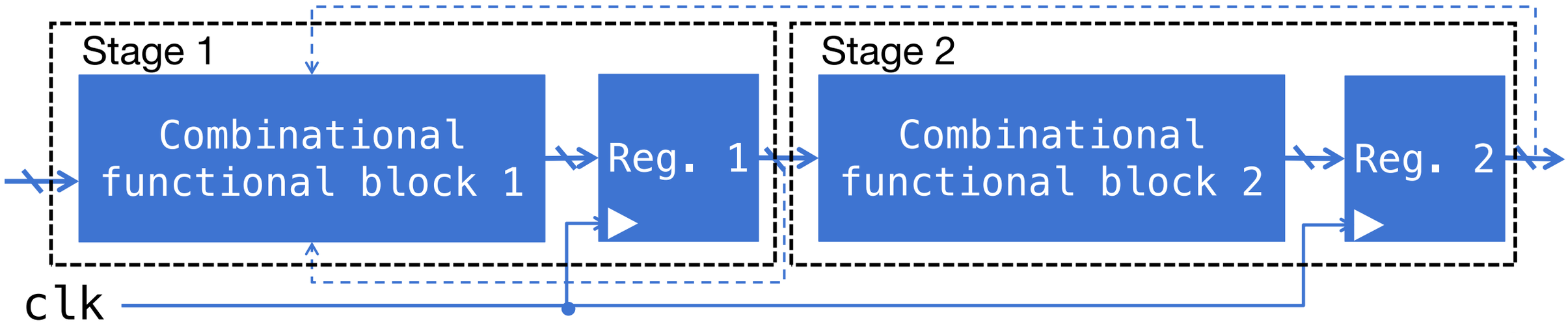


General form of synchronous circuits

- State element can work together with combinational logic (can have feedbacks)

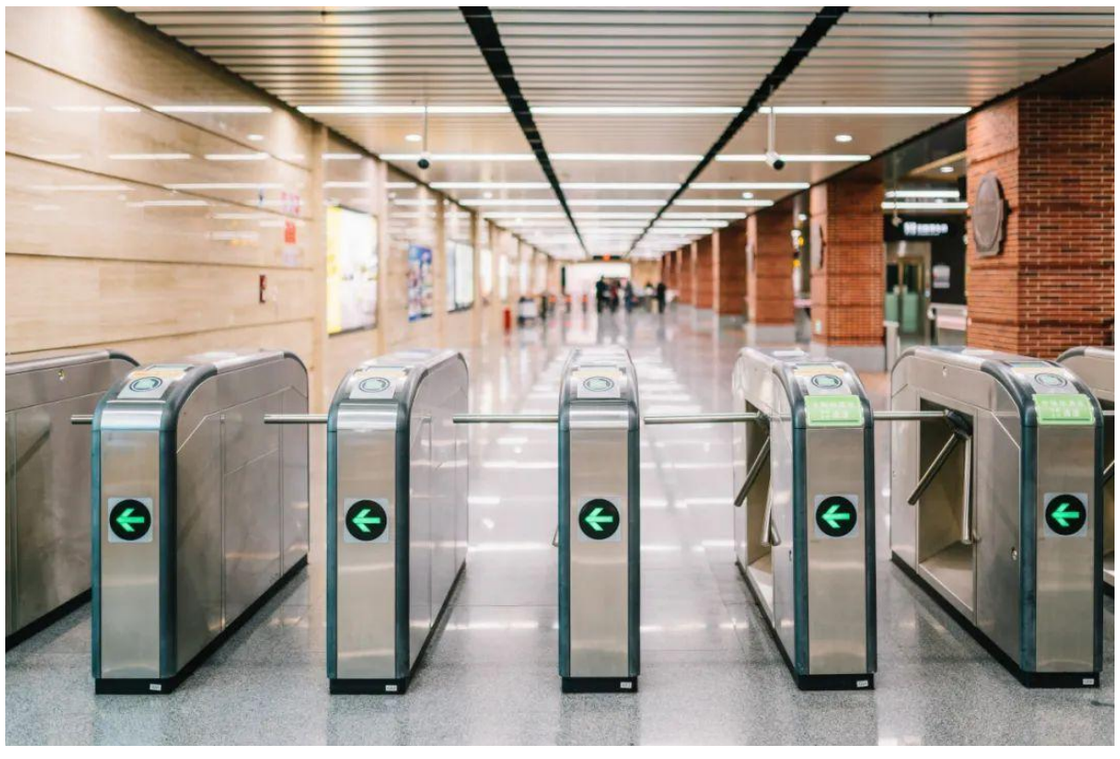


- We can cascade multiple stages of the above synchronous circuit (can have feedbacks across stages)

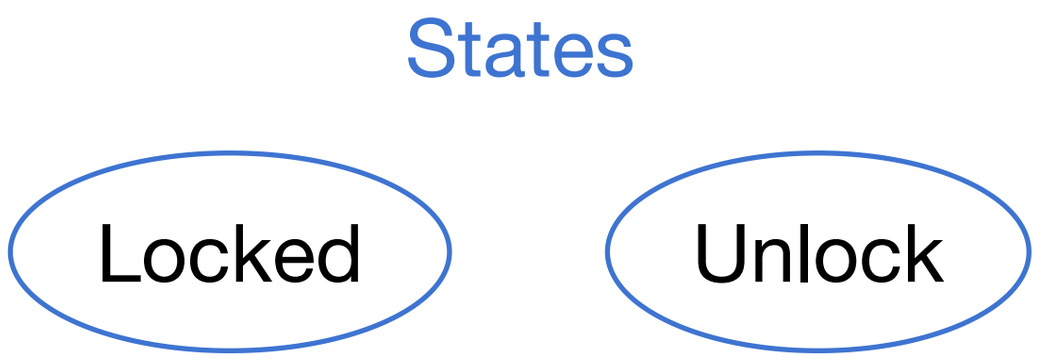


Model of synchronous circuits, FSM

- Finite state machine (FSM): a mathematical model of computation
- It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.
- An FSM is defined by a list of its states, its initial state (entrance), the inputs that trigger each transition, and optional outputs

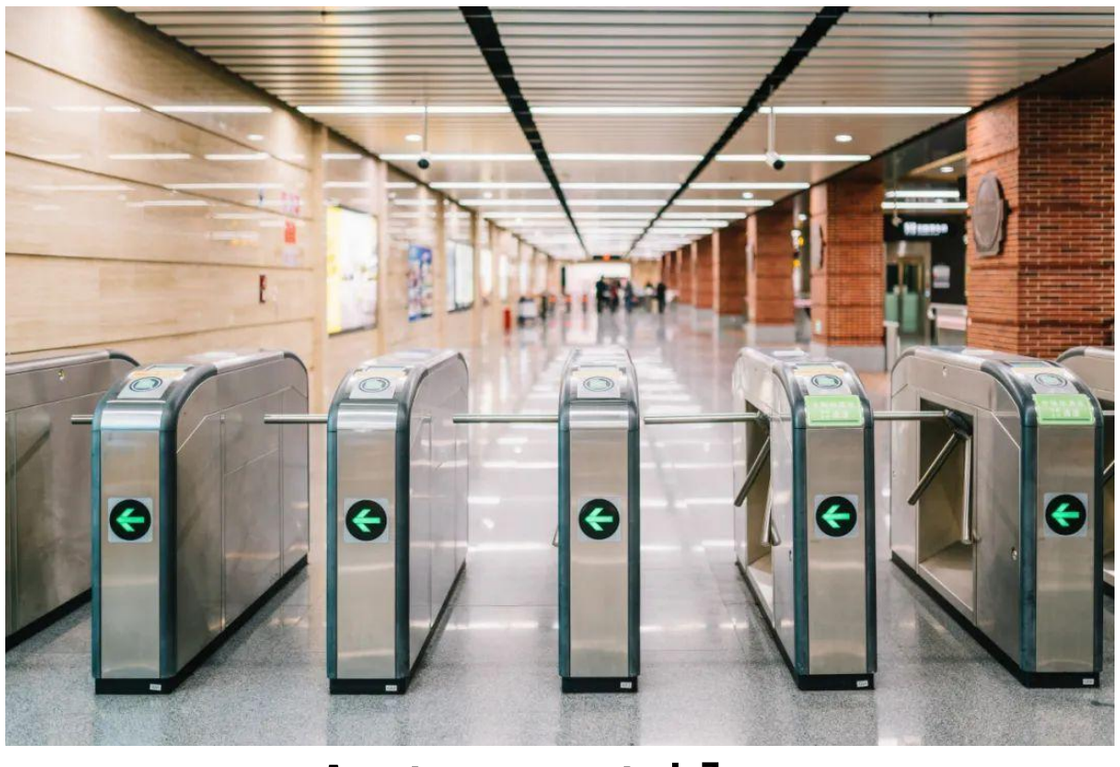


A turnstile

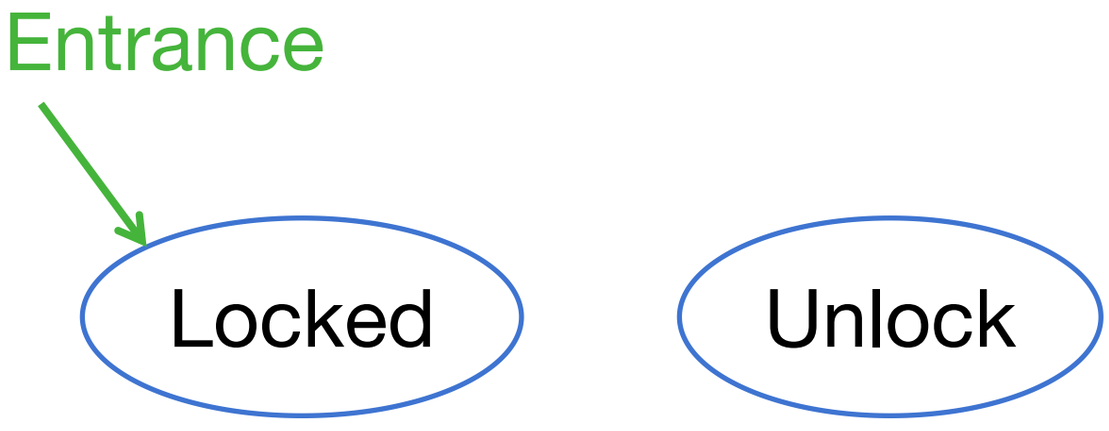


Model of synchronous circuits, FSM

- Finite state machine (FSM): a mathematical model of computation
- It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.
- An FSM is defined by a list of its states, its initial state (entrance), the inputs that trigger each transition, and optional outputs

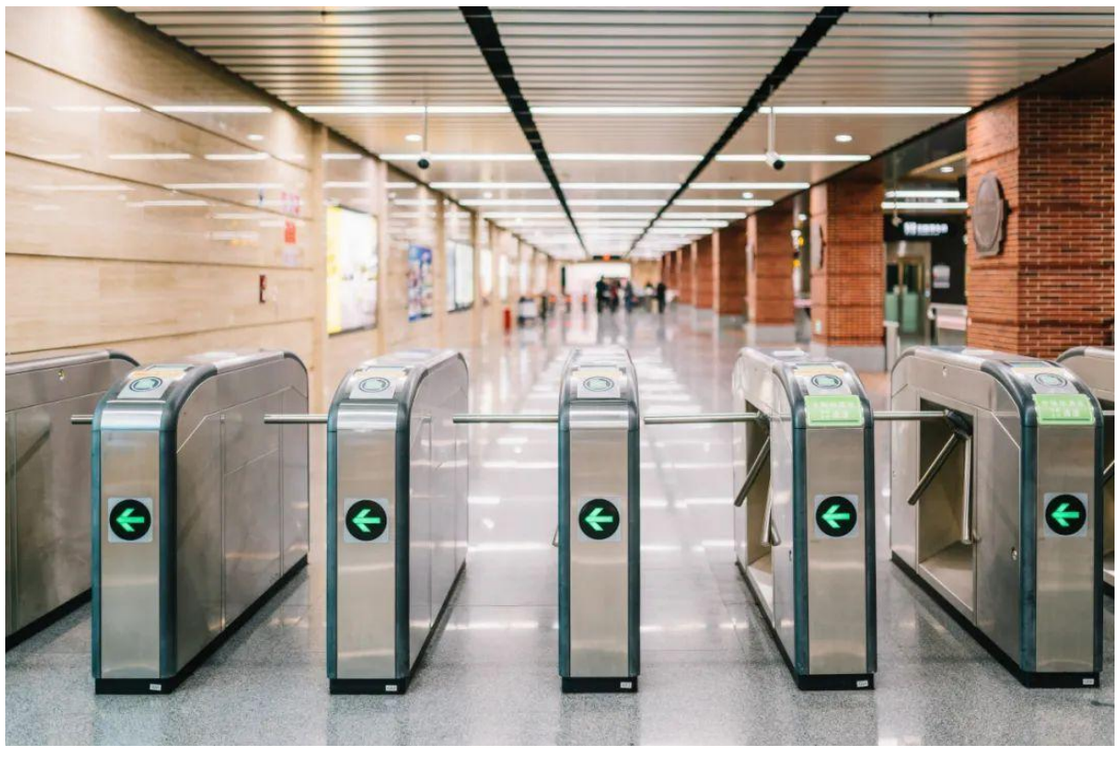


A turnstile

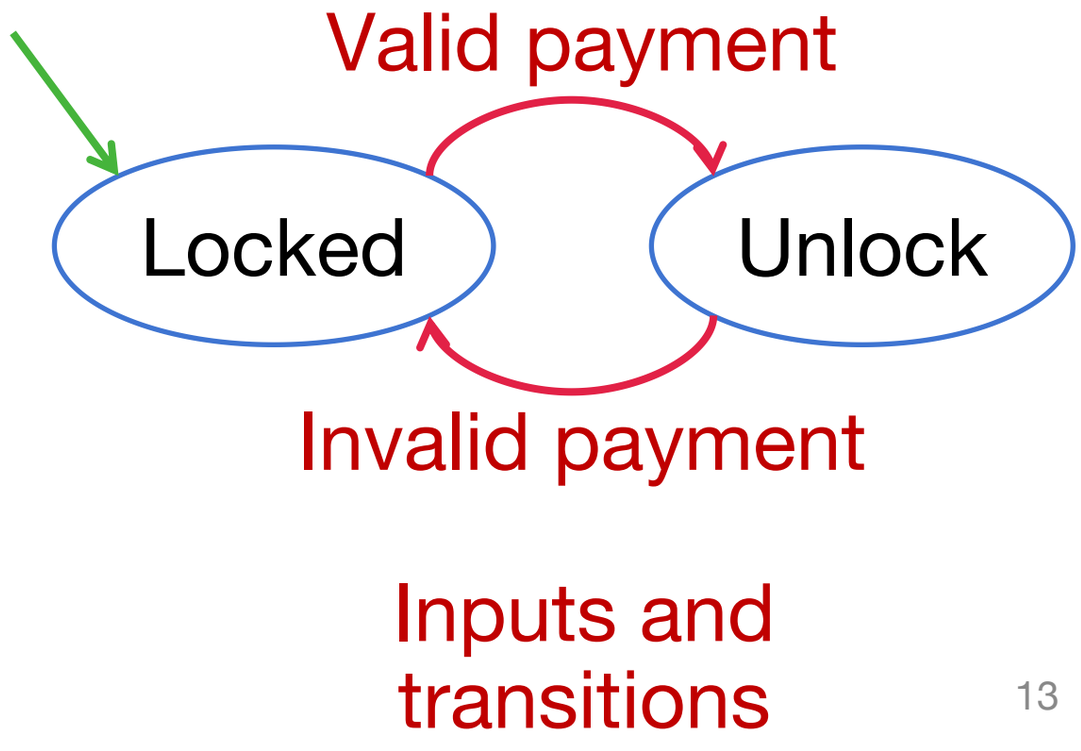


Model of synchronous circuits, FSM

- Finite state machine (FSM): a mathematical model of computation
- It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.
- An FSM is defined by a list of its states, its initial state (entrance), the inputs that trigger each transition, and optional outputs

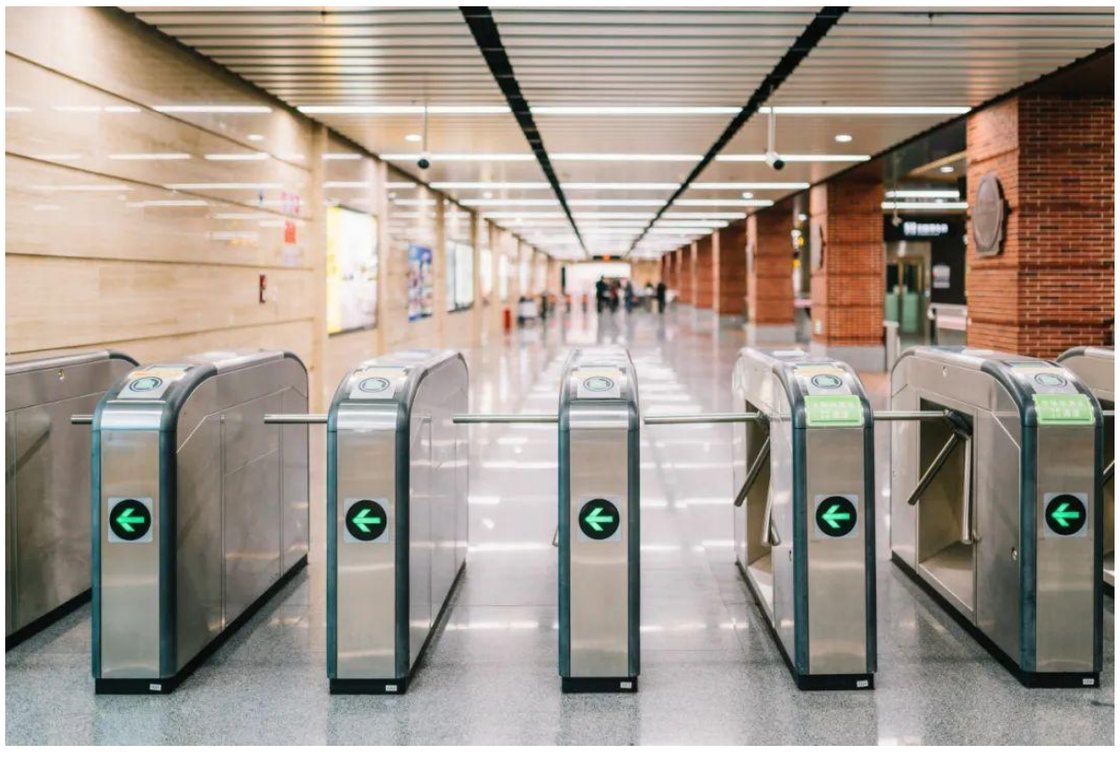


A turnstile

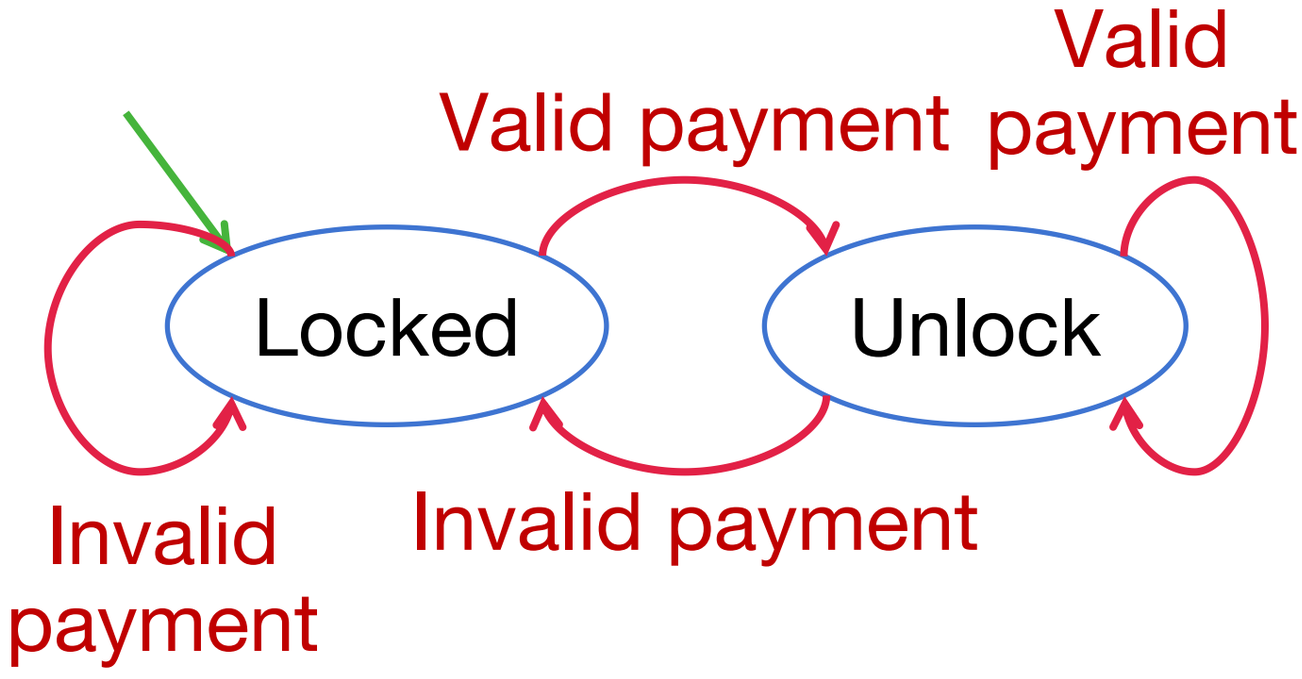


Model of synchronous circuits, FSM

- Finite state machine (FSM): a mathematical model of computation
- It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.
- An FSM is defined by a list of its states, its initial state (entrance), the inputs that trigger each transition, and optional outputs

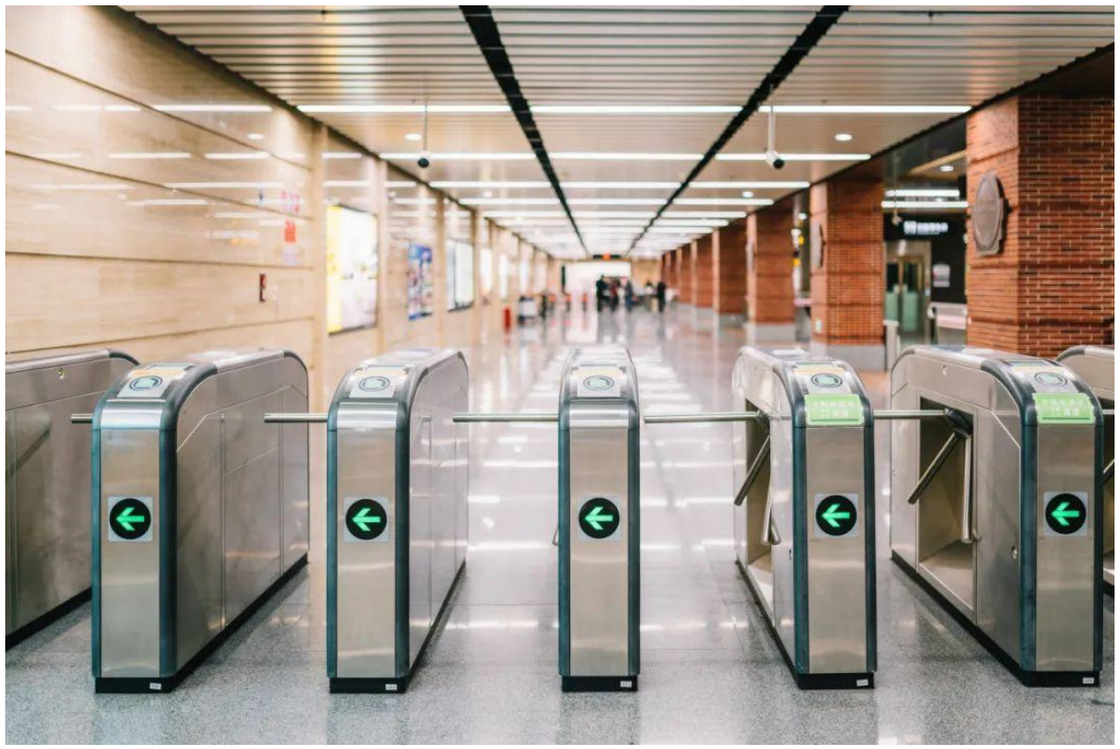


A turnstile

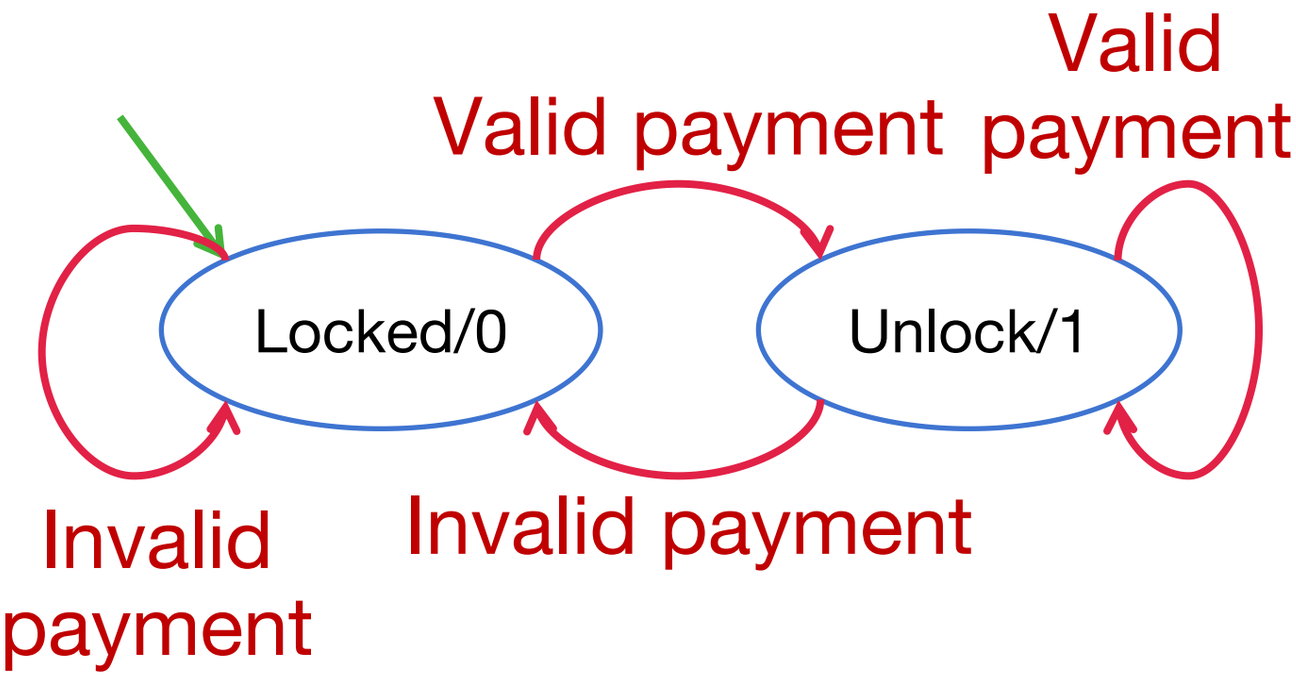


Model of synchronous circuits, FSM

- Define: State “Locked/hold the bar” as ‘0’, “Unlock/release the bar” as ‘1’; use register/DFF to store the state
- Define: “valid payment” as input ‘1’, “invalid payment” as input ‘0’
- Build a synchronous circuit that implements automatic control of the turnstile



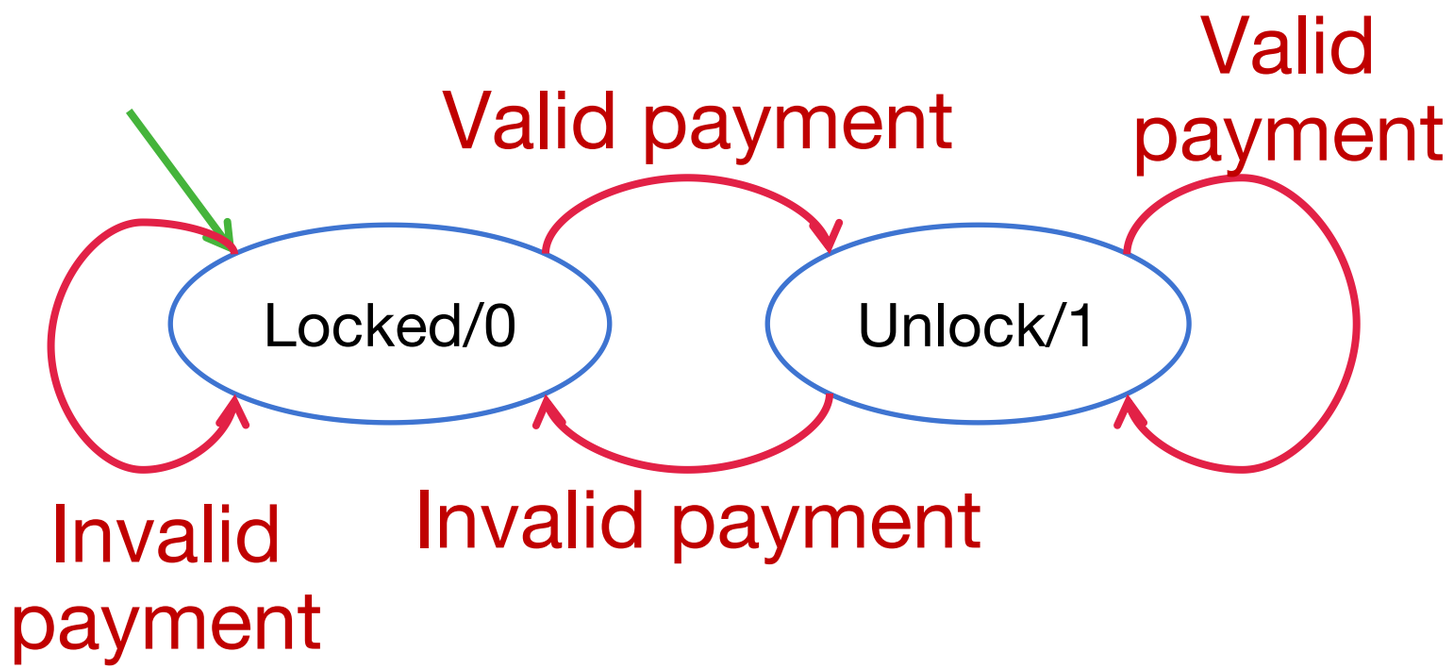
A turnstile



Model of synchronous circuits, FSM

- Define: State “Locked/hold the bar” as ‘0’, “Unlock/release the bar” as ‘1’; use register/DFF to store the state
- Define: “valid payment” as input ‘1’, “invalid payment” as input ‘0’
- Build a synchronous circuit that implements automatic control of the turnstile
- Truth table of this synchronous circuit

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1



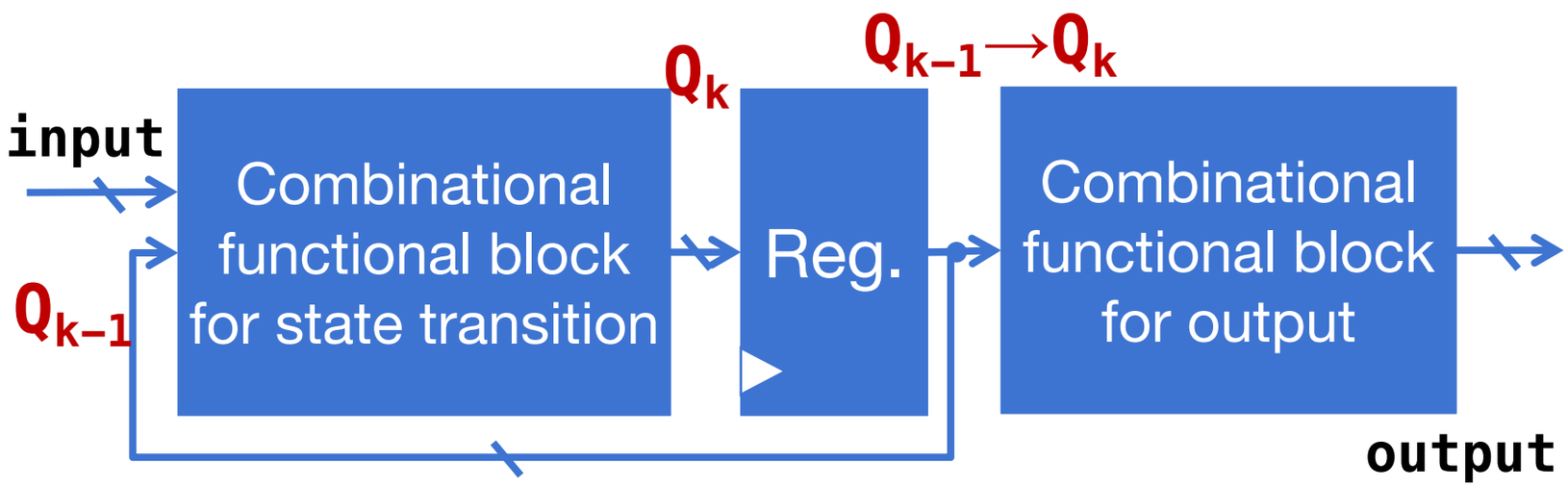
Q_{k-1}: state at the (k-1)th clock cycle
Q_k: state at the (k)th clock cycle

Model of synchronous circuits, FSM

- Define: State “Locked/hold the bar” as ‘0’, “Unlock/release the bar” as ‘1’; use register/DFF to store the state
- Define: “valid payment” as input ‘1’, “invalid payment” as input ‘0’
- Build a synchronous circuit that implements automatic control of the turnstile
- Truth table of this synchronous circuit

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

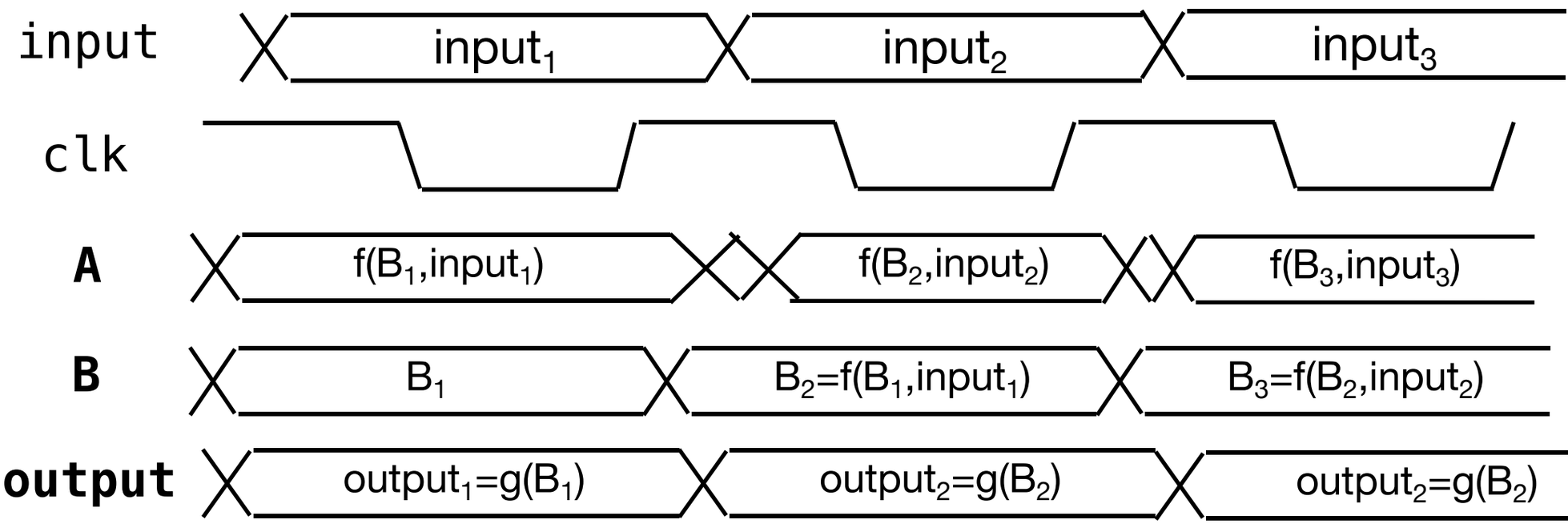
Template of a synchronous circuit that implements the FSM



Q_{k-1}: state at the (k-1)th clock cycle
Q_k: state at the (k)th clock cycle

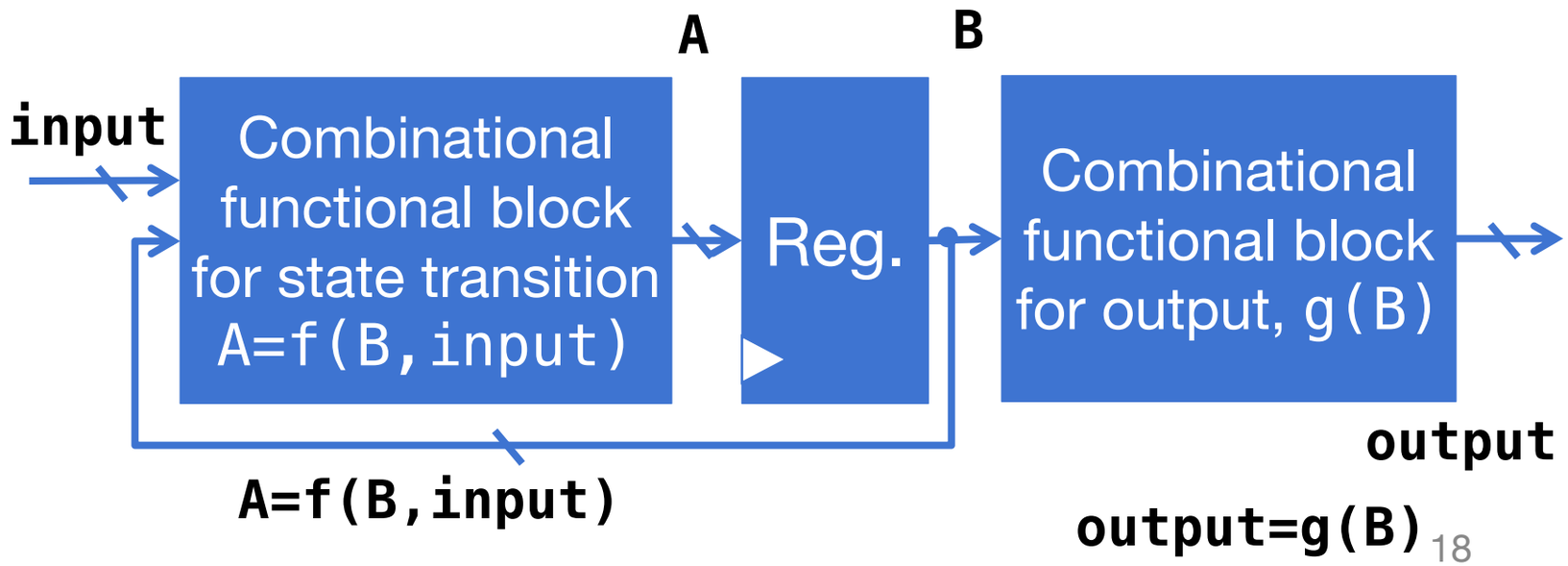
Model of synchronous circuits, FSM

- Timing diagram (general case)



input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

Template of a synchronous circuit that implements the FSM



Model of synchronous circuits, FSM

- Build the circuits

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

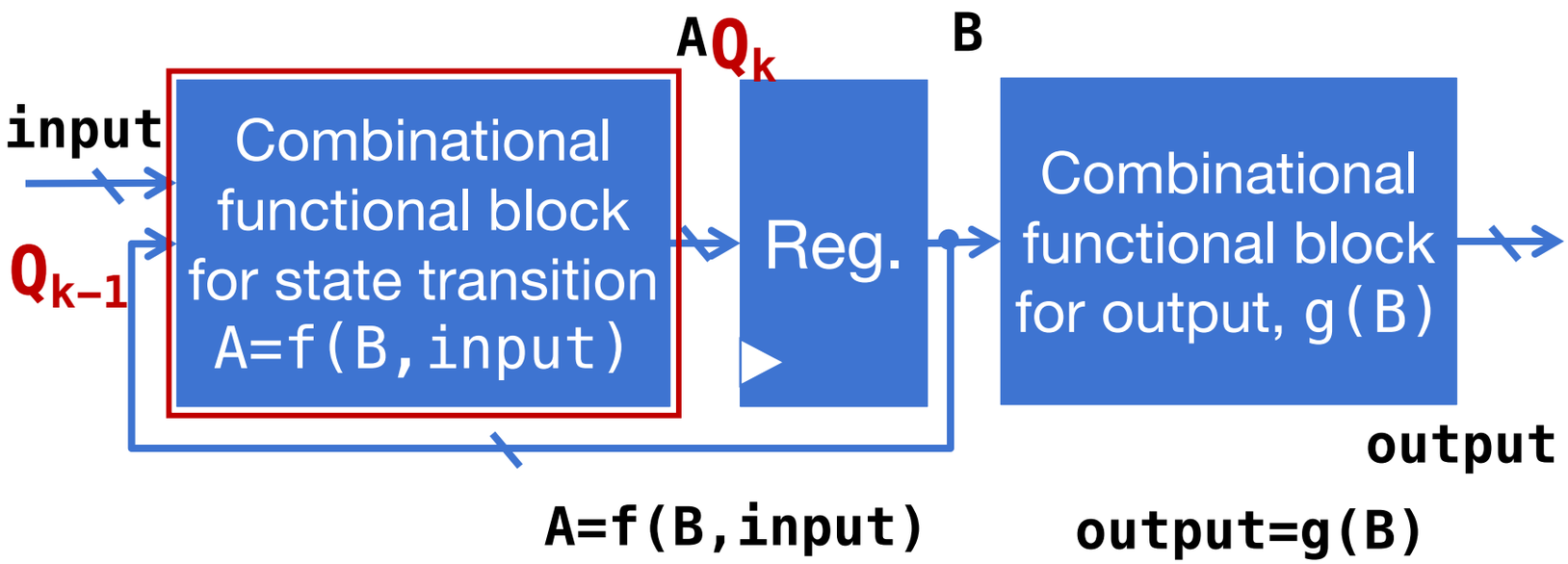
State transition logic

$Q_k=f(input,Q_{k-1})$

⬇

$Q_k=input$

Template of a synchronous circuit that implements the FSM



Model of synchronous circuits, FSM

- Build the circuits

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

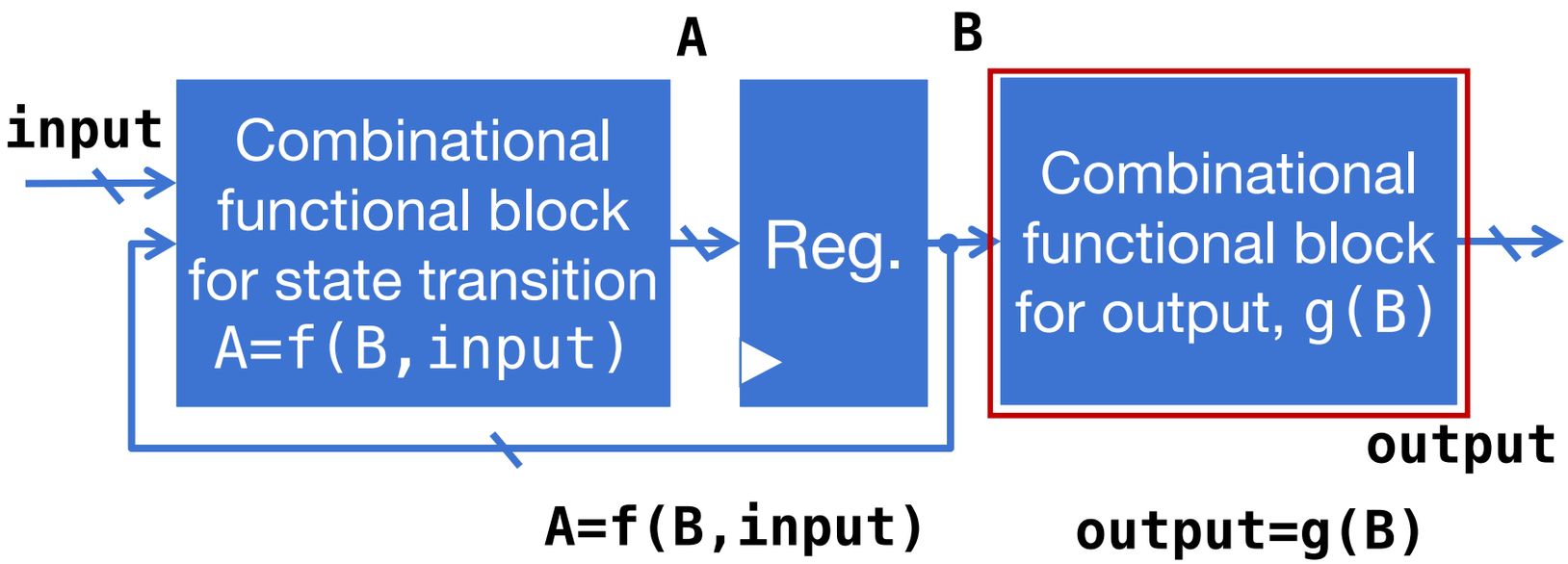
Output logic

$Output=g(Q_k)$

⬇

$Output=Q_k$

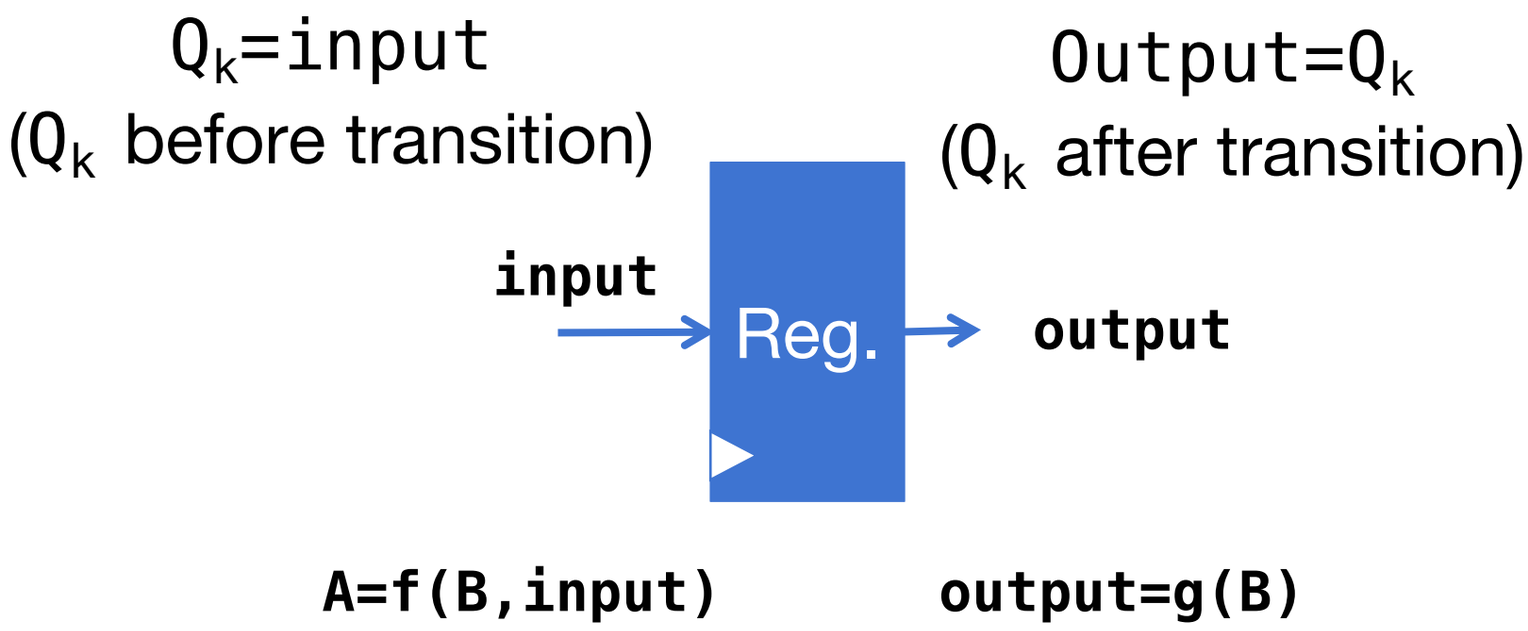
Template of a synchronous circuit that implements the FSM



Model of synchronous circuits, FSM

- Build the circuits

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

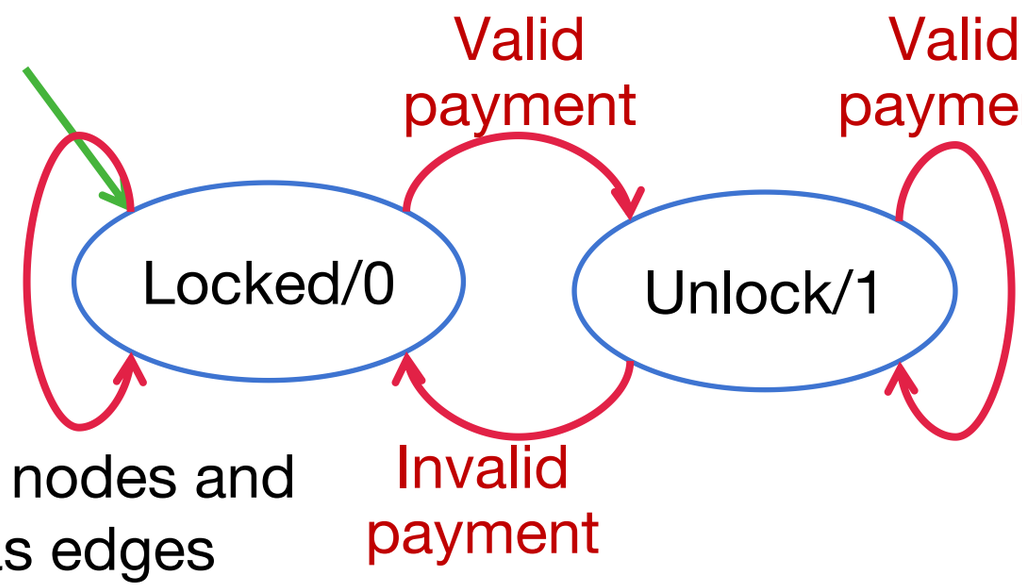


Model of synchronous circuits, FSM

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

Enumerate
Inputs & states

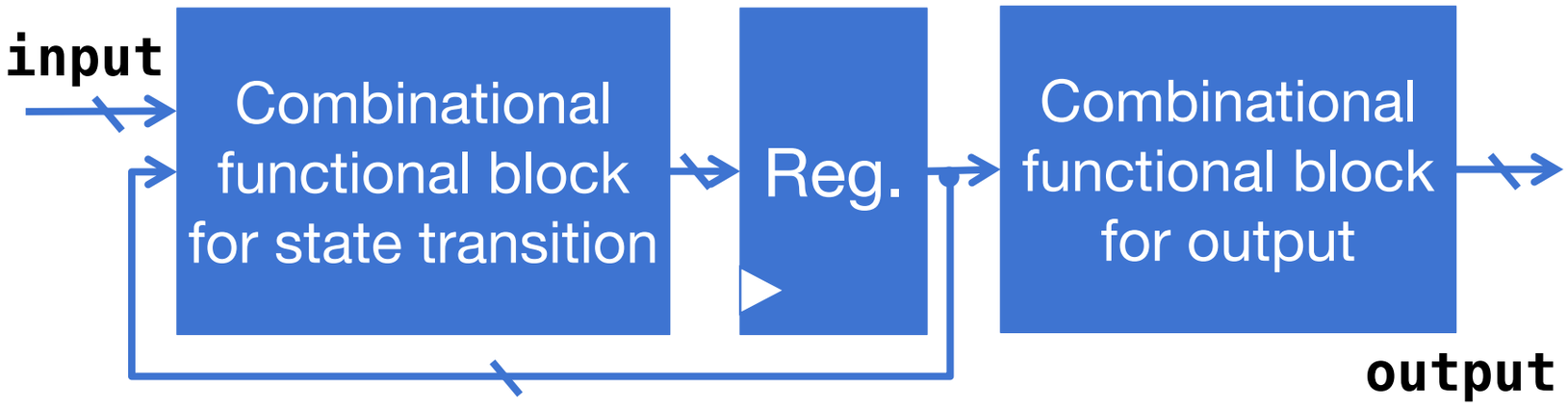
Draw states as nodes and
transitions as edges



Enumerate
Inputs & states

Sum of minterms
or Karnauph map

Enumerate
Inputs & states
& transitions



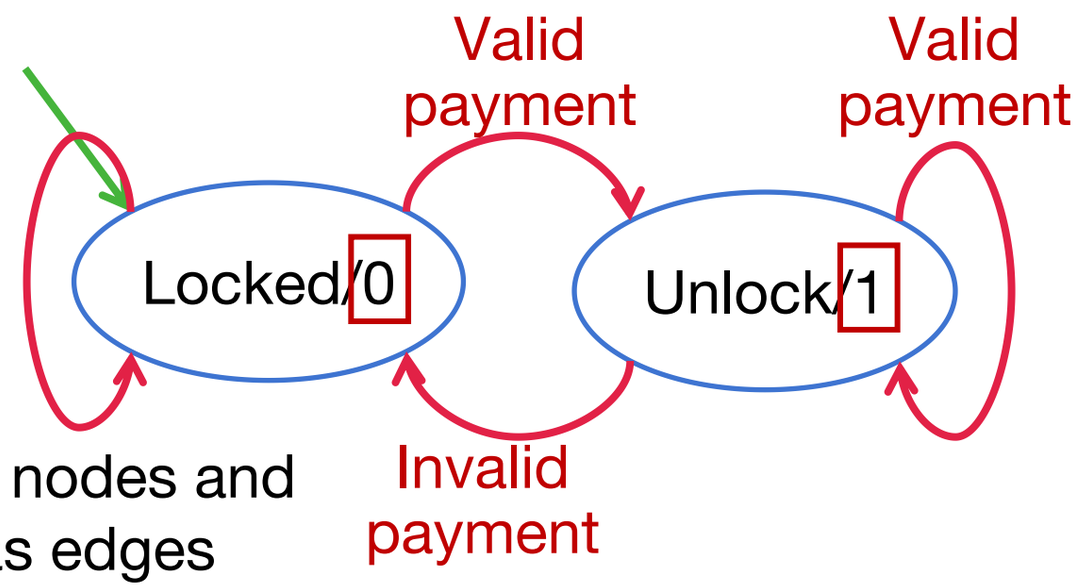
Moore machine vs. Mealy machine

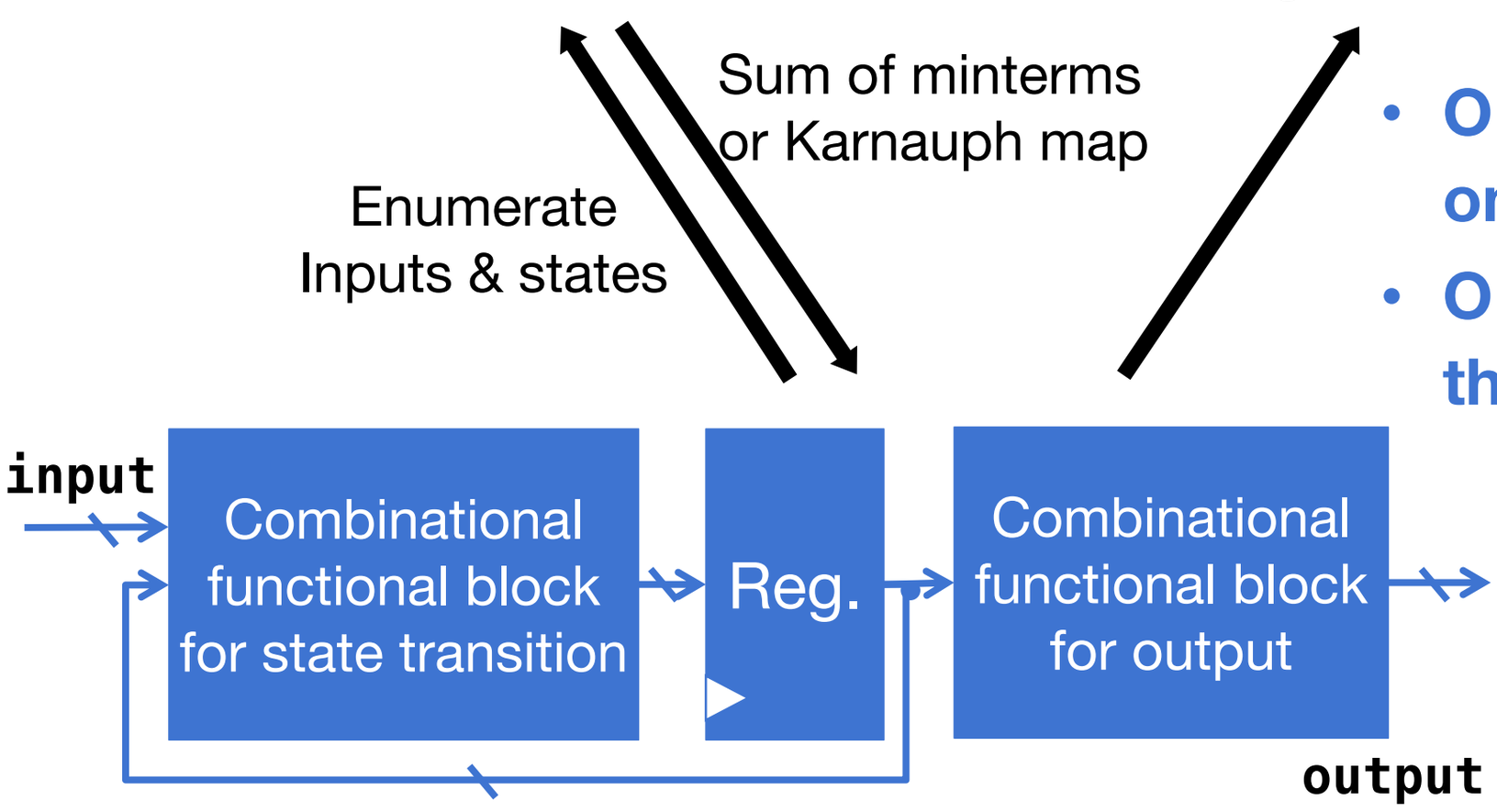
- Moore machine

input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

Enumerate
Inputs & states

Draw states as nodes and transitions as edges





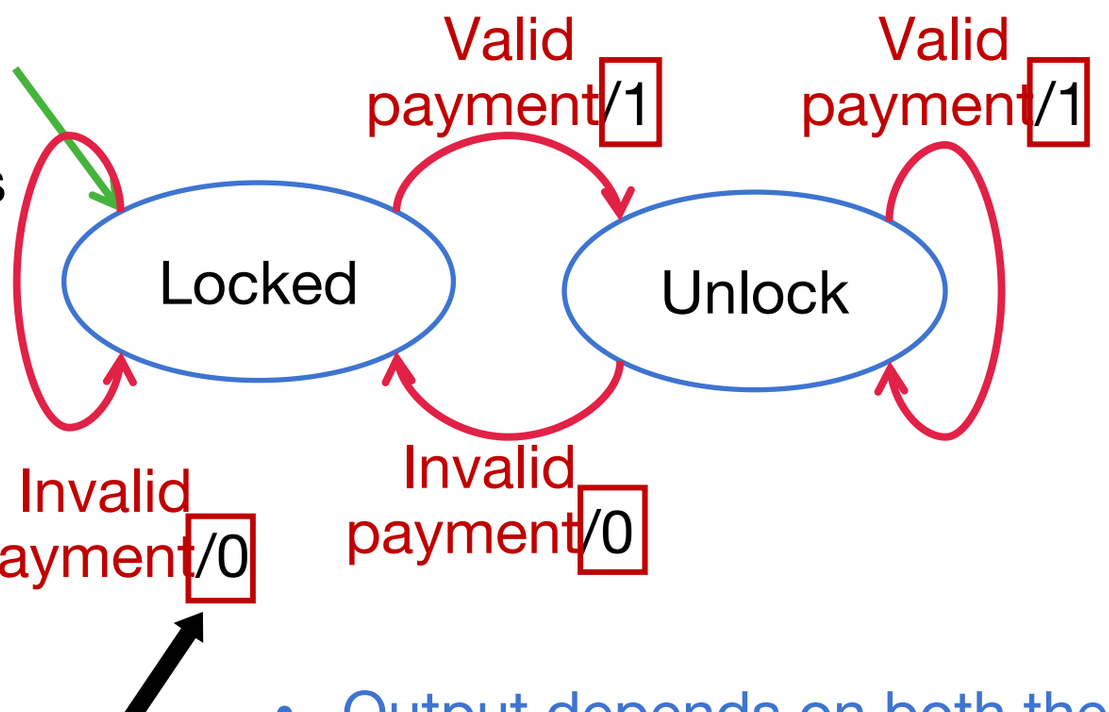
- **Output depends solely on the current state**
- **Output controlled by the clock somewhat**

Moore machine vs. Mealy machine

- Mealy machine

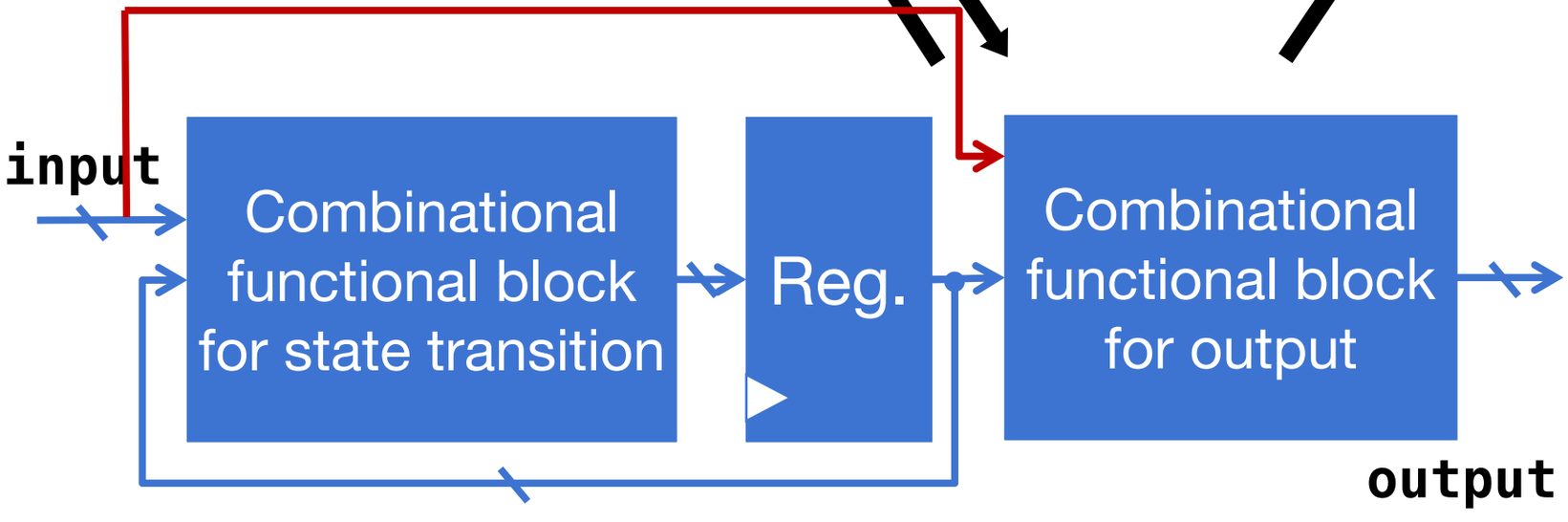
input	Q_{k-1}	Q_k	output
0	0	0	0
0	1	0	0
1	1	1	1
1	0	1	1

Enumerate
Inputs & states



Sum of minterms
or Karnauph map

Enumerate
Inputs & states



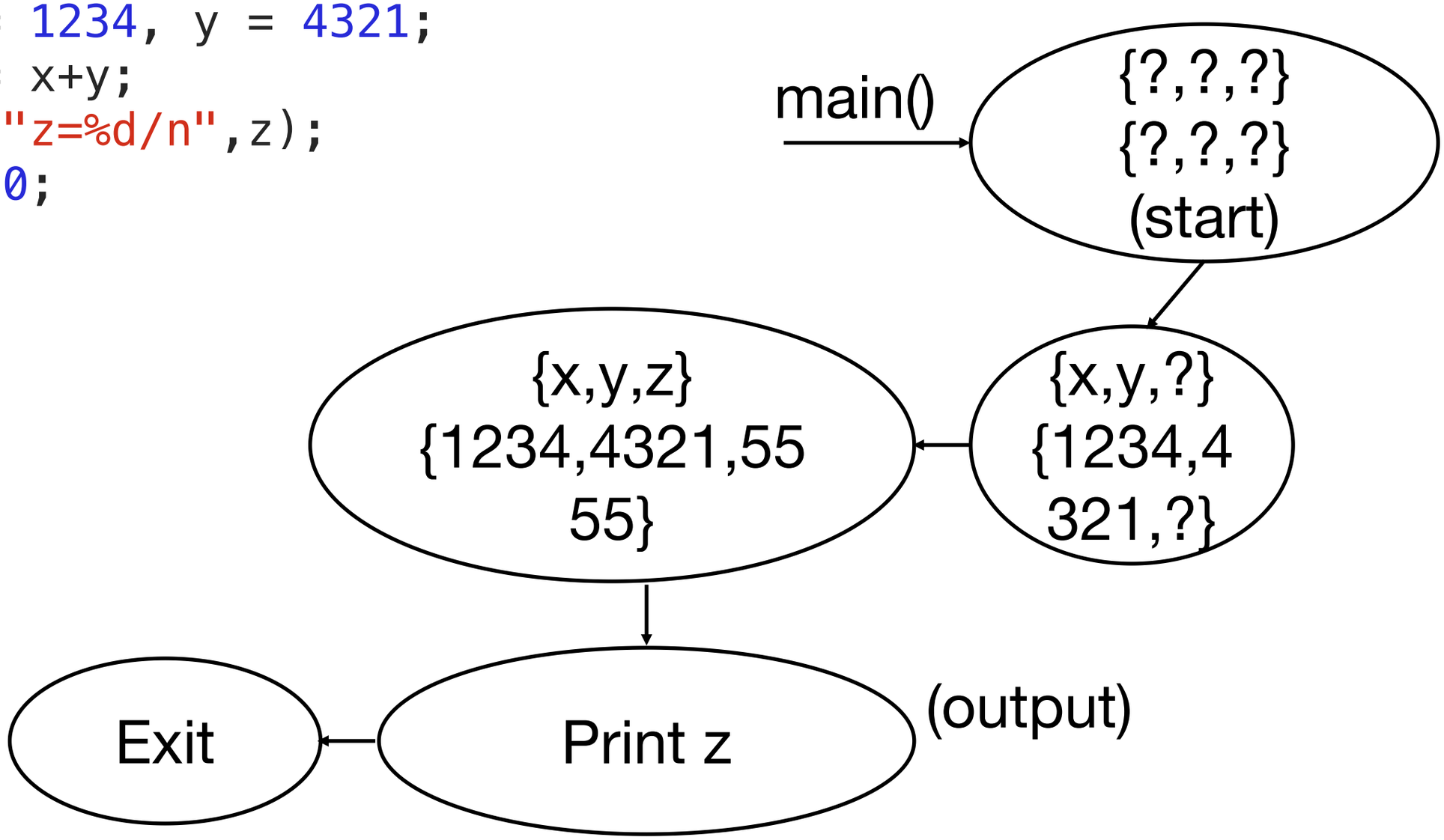
- Output depends on both the current state **and input**
- Output uncontrolled by the clock
- Mealy and Moore machines are interchangable
- Mealy and Moore outputs can co-exist in one synchromous circuit

Summary for building synchronous circuits

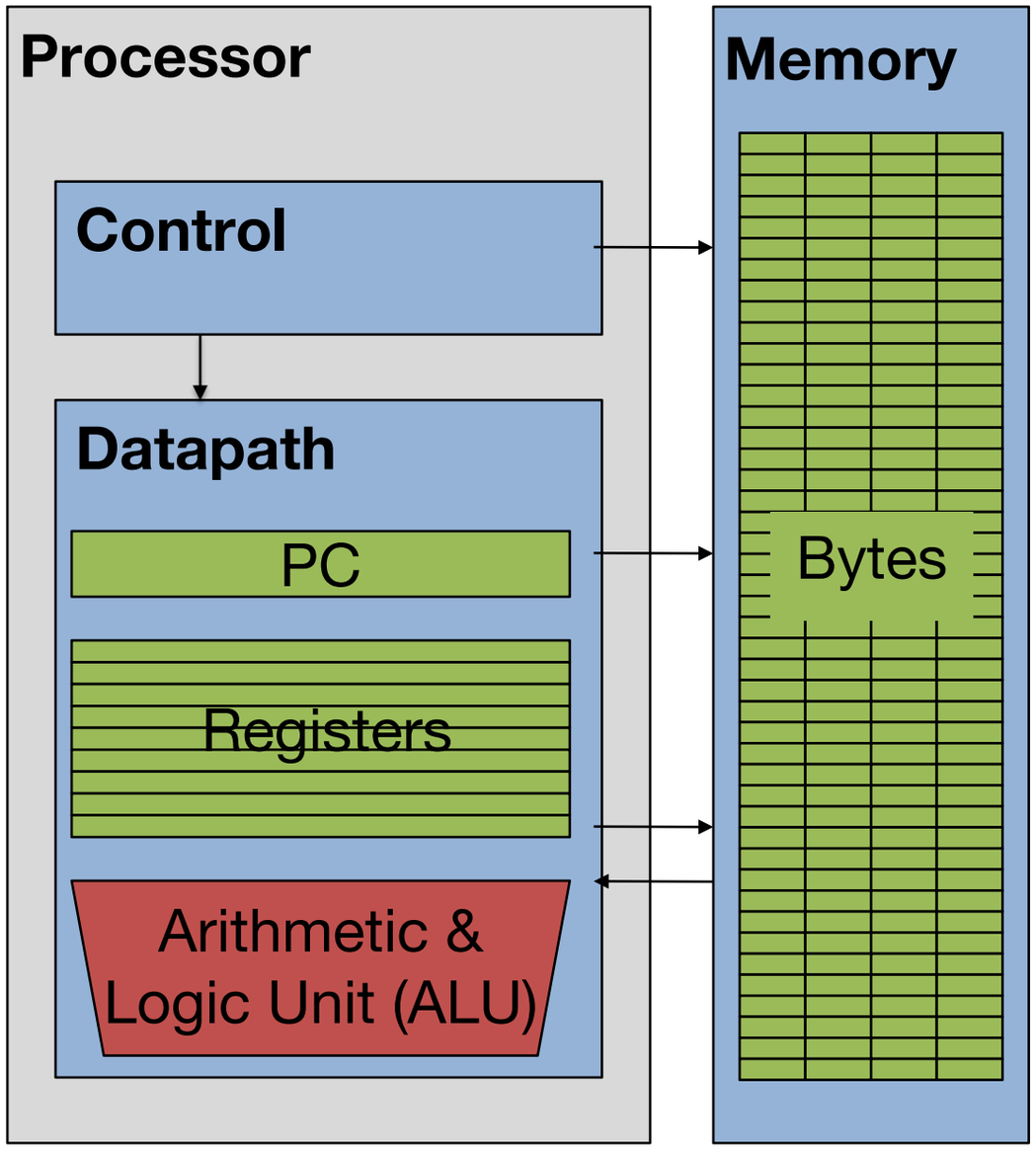
- Step 1: Draw finite state machine of the desired function (we ignore the initialization)
- Step 2: Define/assign binary numbers to represent the states, the inputs and the outputs
- Step 3: Write down the truth table (enumerate input/previous state (and current state) and their corresponding current state (and output))
- Step 4: Use template and decide the combinational block for state transition and output logic
- Otherwise, we can also use simple synchronous circuit blocks to build larger synchronous circuit like for the multi-bit adder

C Program as an FSM

```c
#include <stdio.h>
int main() {//compute 1234 + 4321
   int x = 1234, y = 4321;
   int z = x+y;
   printf("z=%d/n",z);
   return 0;
}
```



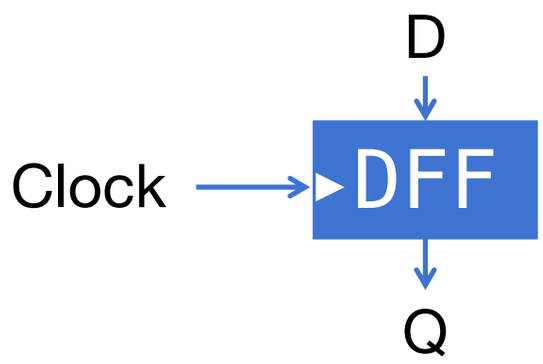
ISA as an FSM



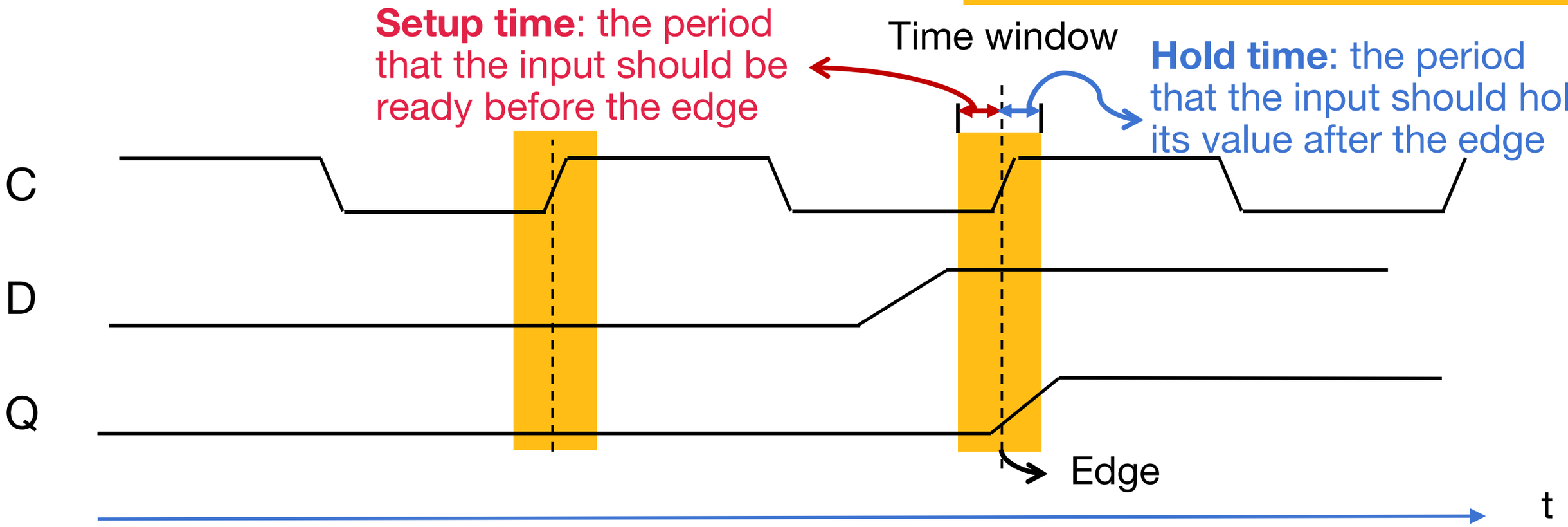
- States: all registers & memory
- Transition: register/memory value change
- Entrance: power on/reset
- Input: instructions, can change registers/memory value

Timing in synchronous circuits

- Can the frequency of the clock be infinity?

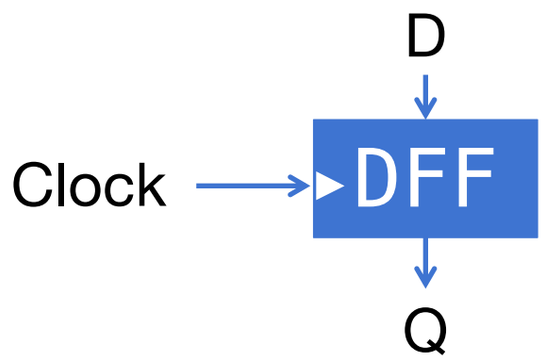


DFFs require a time window to correctly capture a signal, i.e., the signal should be stable during this time window

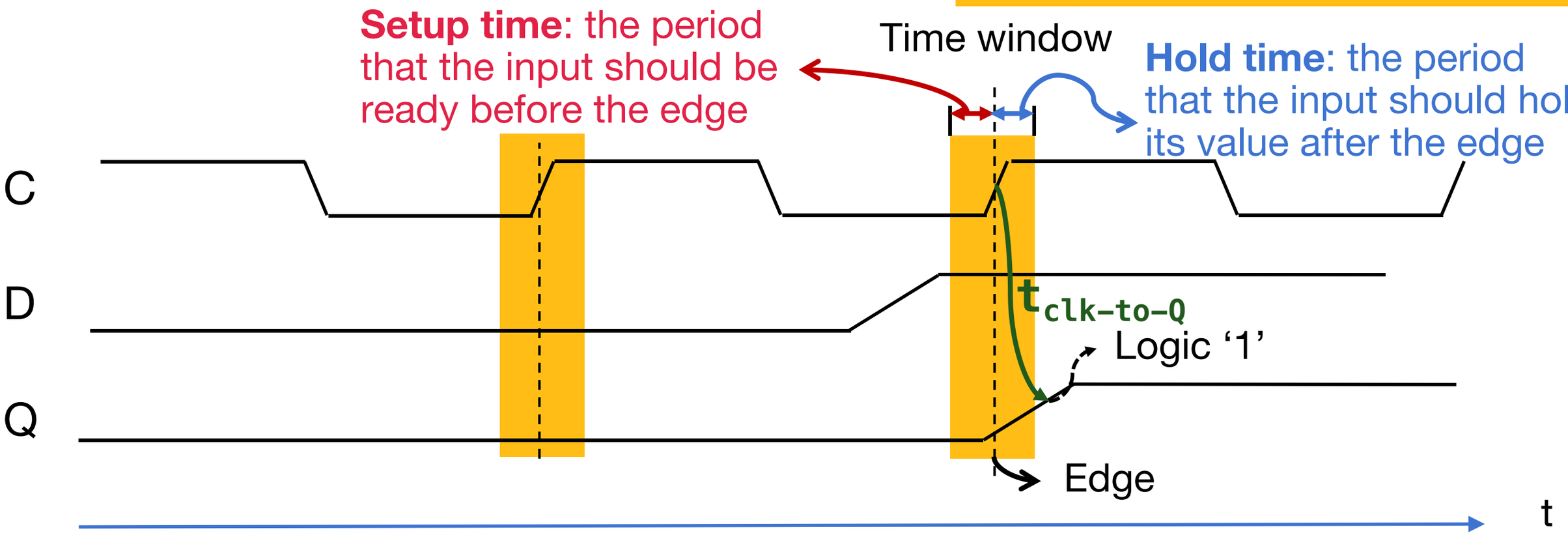


Timing in synchronous circuits

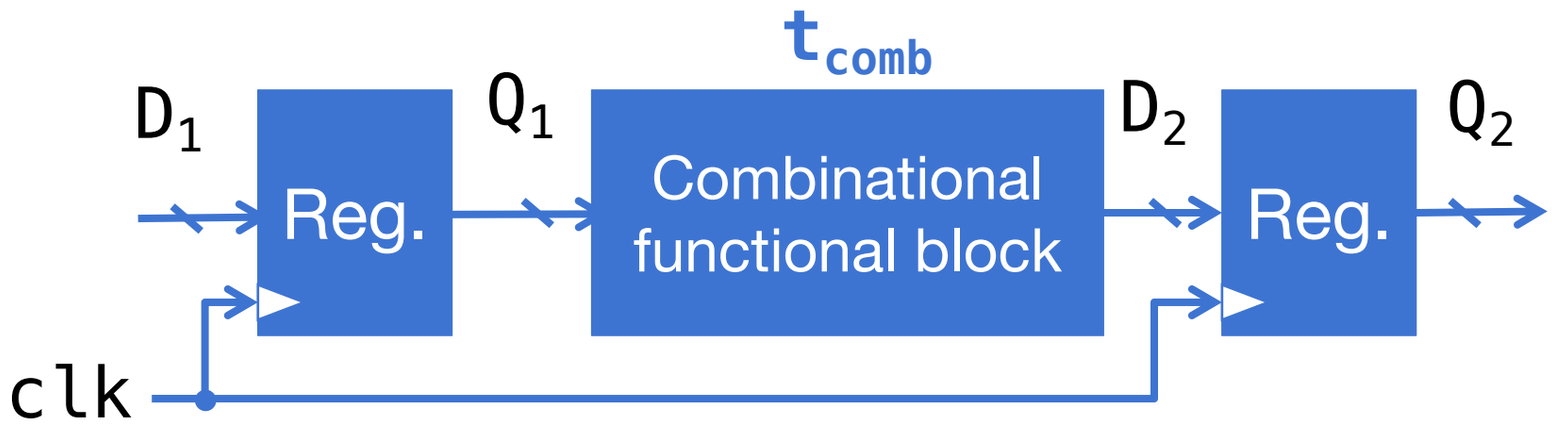
- DFF also has delay: $t_{clk-to-Q}$



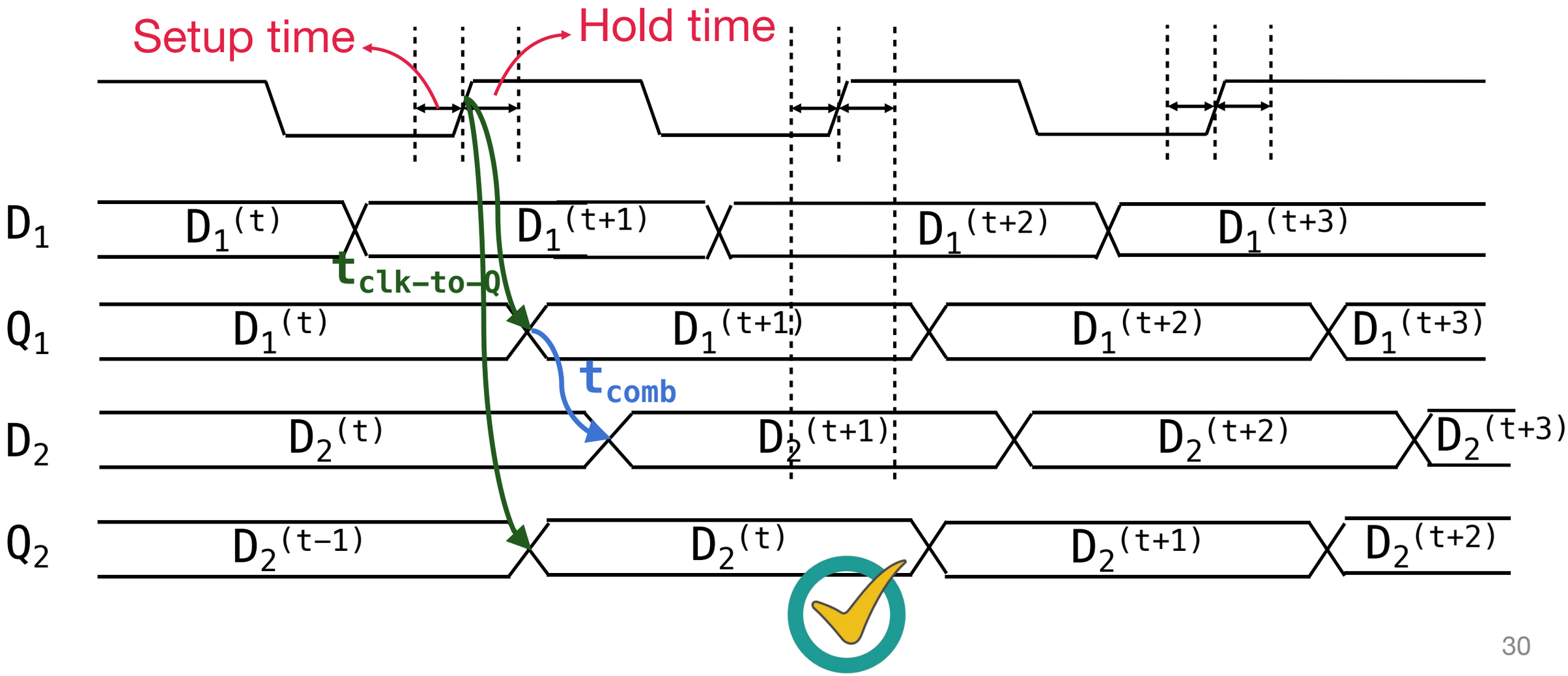
DFFs require a time window to correctly capture a signal, i.e., the signal should be stable during this time window



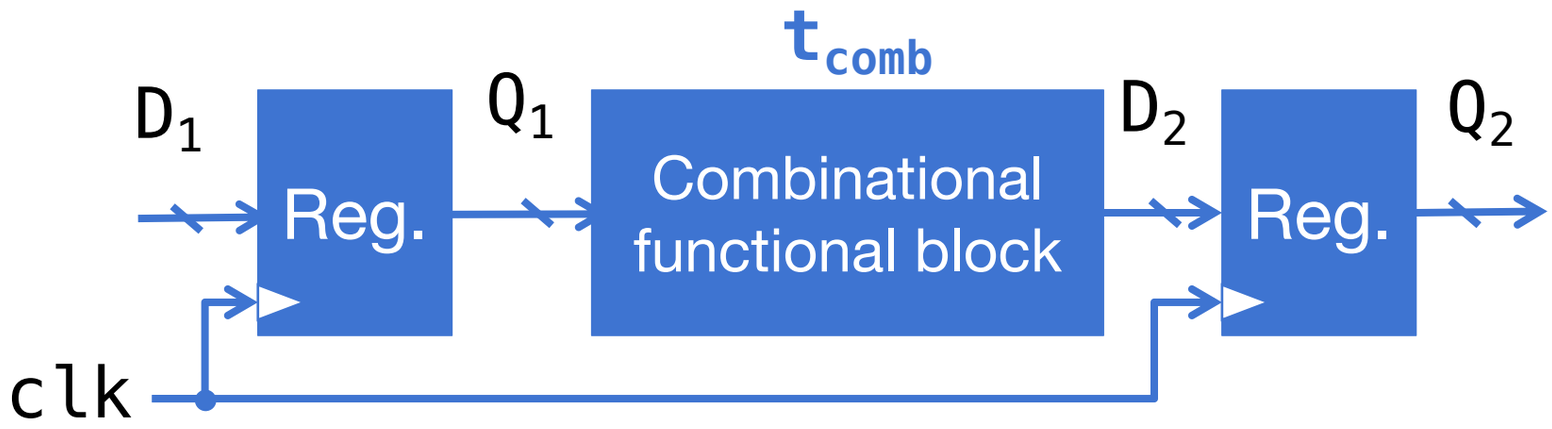
Estimating the max frequency



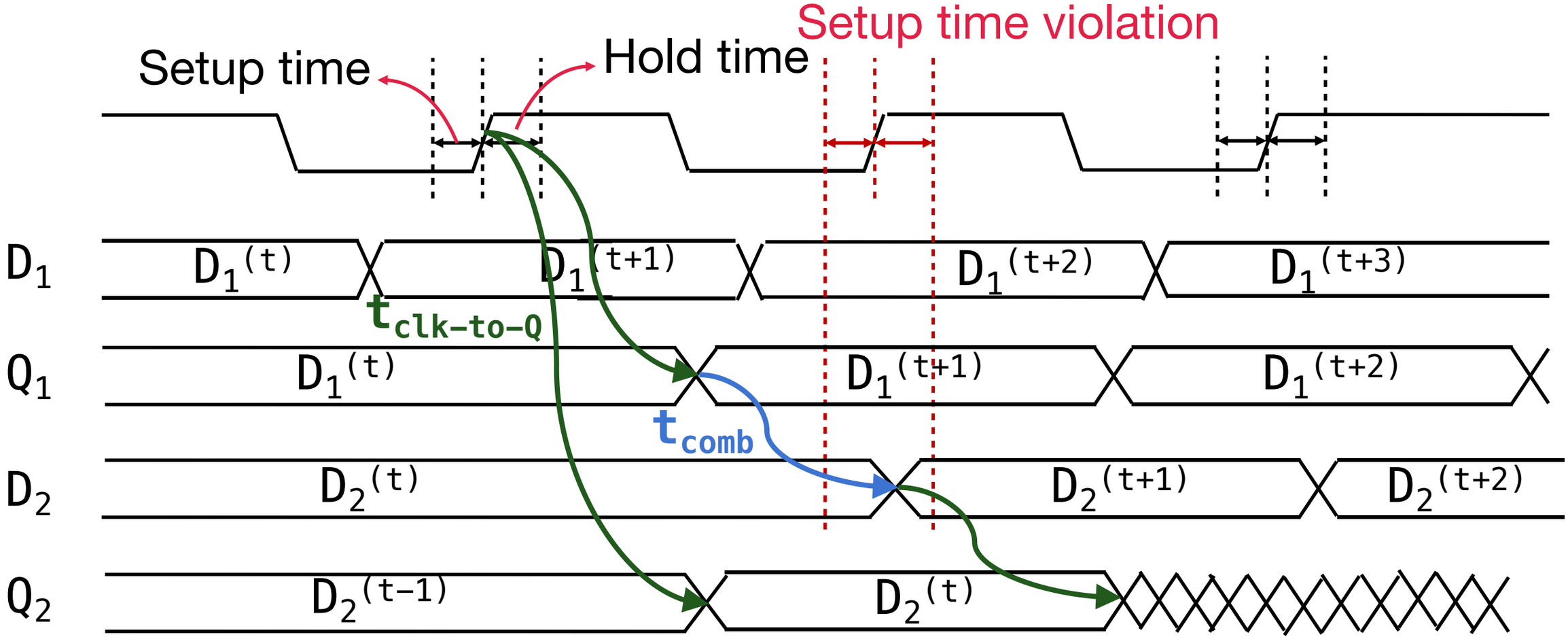
Typical digital system: a mix of combinational circuits & registers



Estimating the max frequency



Typical digital system: a mix of combinational circuits & registers

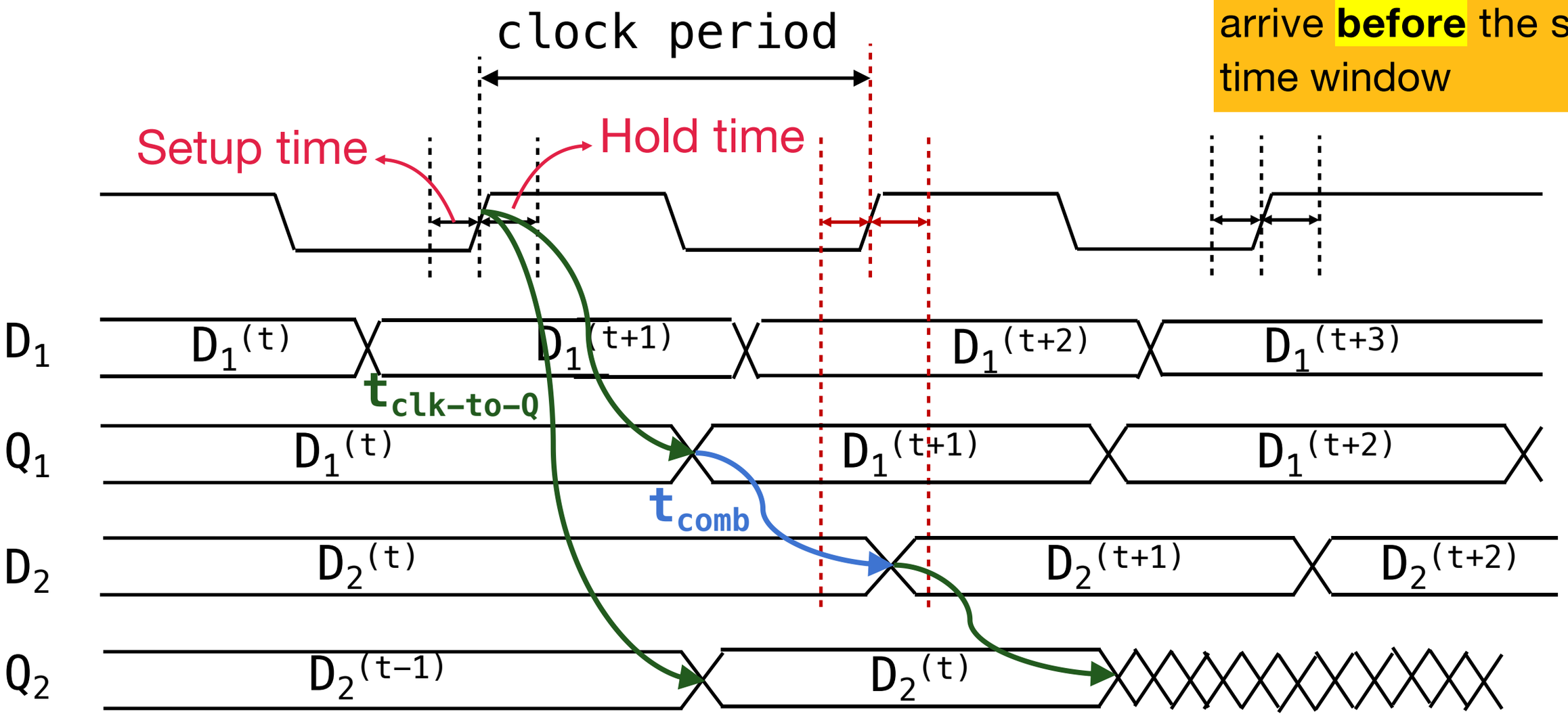


Estimating the max frequency

Max frequency = 1/min clock period

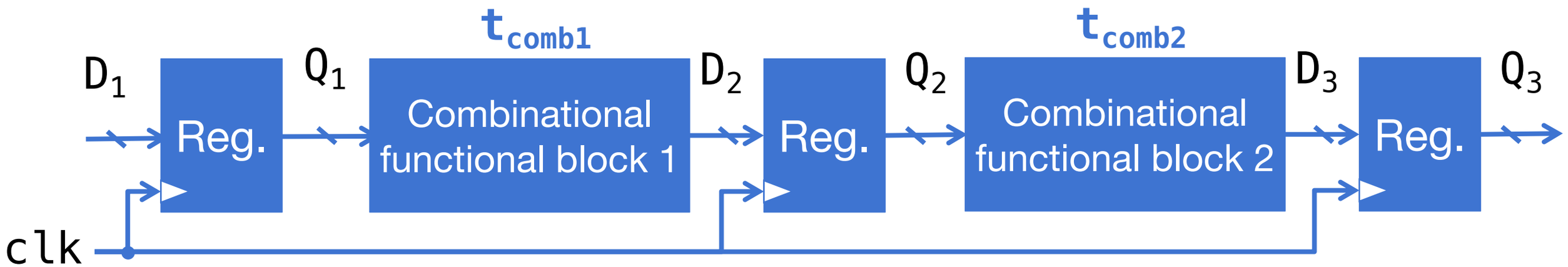
$t_{clk-to-Q} + t_{comb} \leq$ min clock period − setup time

Signal $D_2^{(t+1)}$ should arrive **before** the setup time window



Clarifications

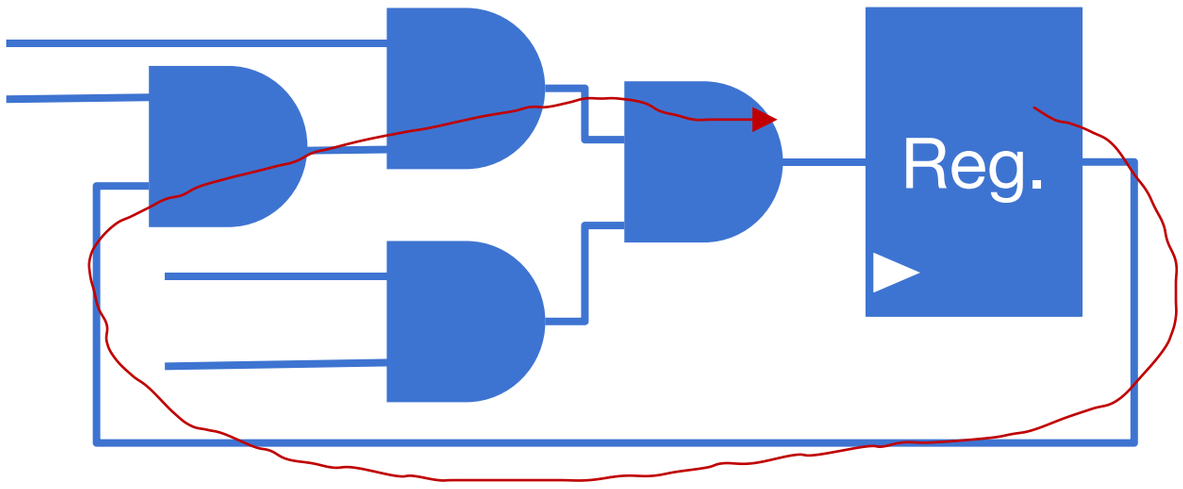
- A synchronous circuit may contain multiple stages of combinational-block-and-register, while the slowest path decide the max frequency.
- The corresponding path is also called critical path.



$$t_{clk\text{-}to\text{-}Q} + t_{comb} \leq \text{min clock period} - \text{setup time}$$

- We simplify the model, so that hold time is not considered. There would also be hold time violation due to some signals changing too fast, but it is not related to the calculation of max frequency of a circuit.

Question (Your turn)



Given	
Clock->Q	1ns
Setup	1ns
Hold	1ns
AND delay	1ns

What is maximum clock frequency?

- A: 5 GHz
- B: 500 MHz
- C: 200 MHz
- D: 250 MHz
- E: 1/6 GHz