



CS110A

FSM and SDS

Chundong Wang



An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input:	<u>0 1 0</u> <u>0 1 0</u> 1 0 1 1 0
Output:	0 0 0 1 0 0 1 0 0 0 0

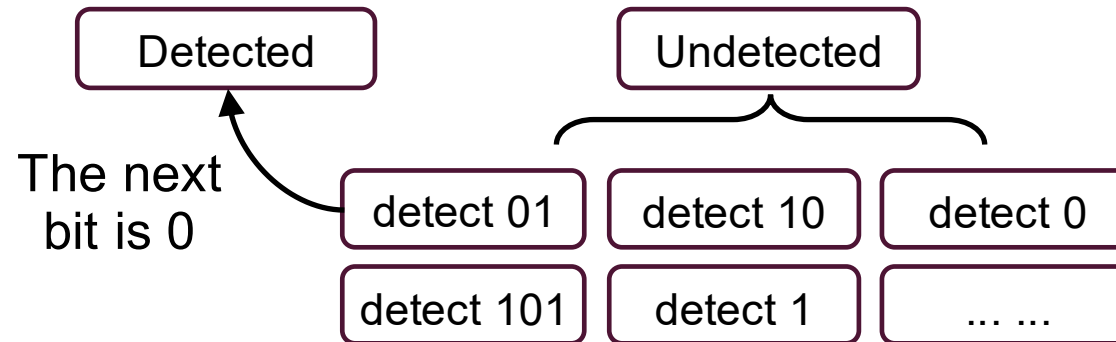
- Step 1: Draw finite state machine of the desired function (we ignore the initialization)
- Step 2: Define/assign binary numbers to represent the states, the inputs and the outputs
- Step 3: Write down the truth table (enumerate input/previous state (and current state) and their corresponding current state (and output))
- Step 4: Use template and decide the combinational block for state transition and output logic

An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 1: Draw finite state machine of the desired function

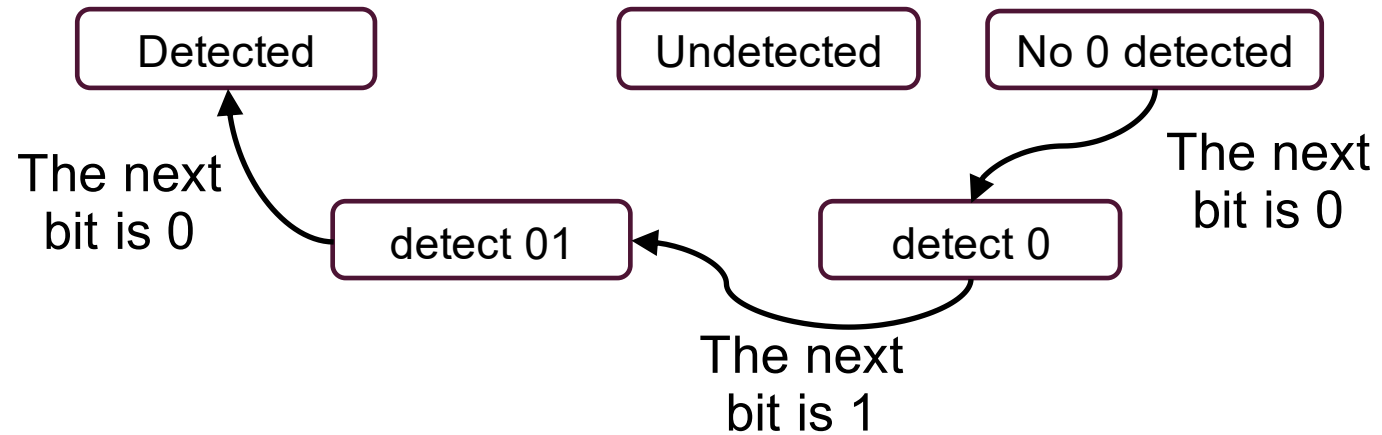


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 1 0
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 1: Draw finite state machine of the desired function

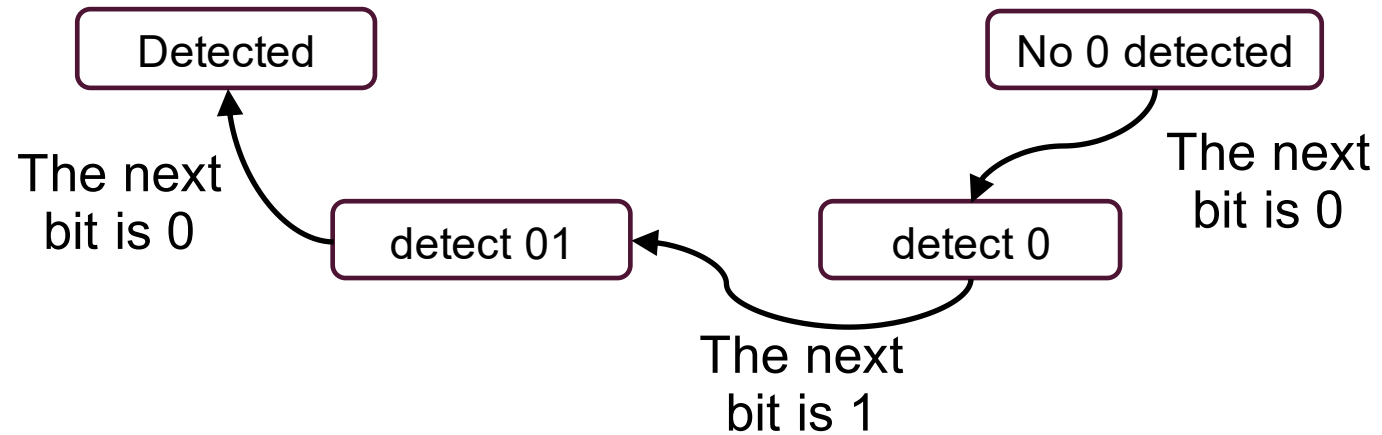


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 1: Draw finite state machine of the desired function

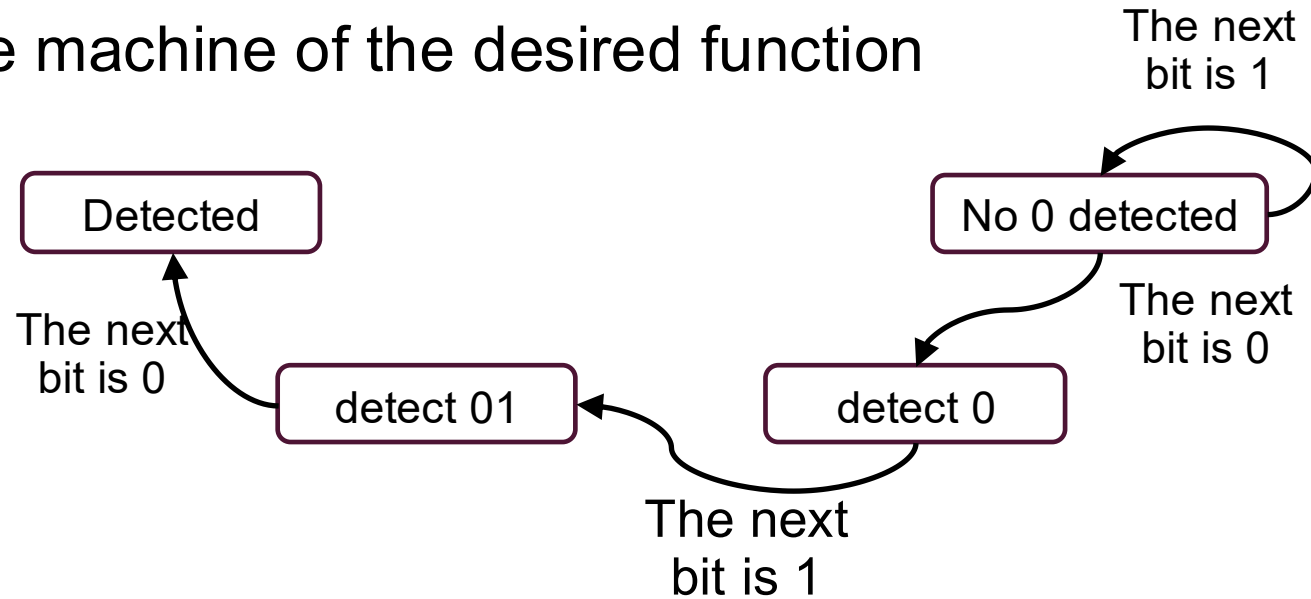


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 1 0
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 1: Draw finite state machine of the desired function

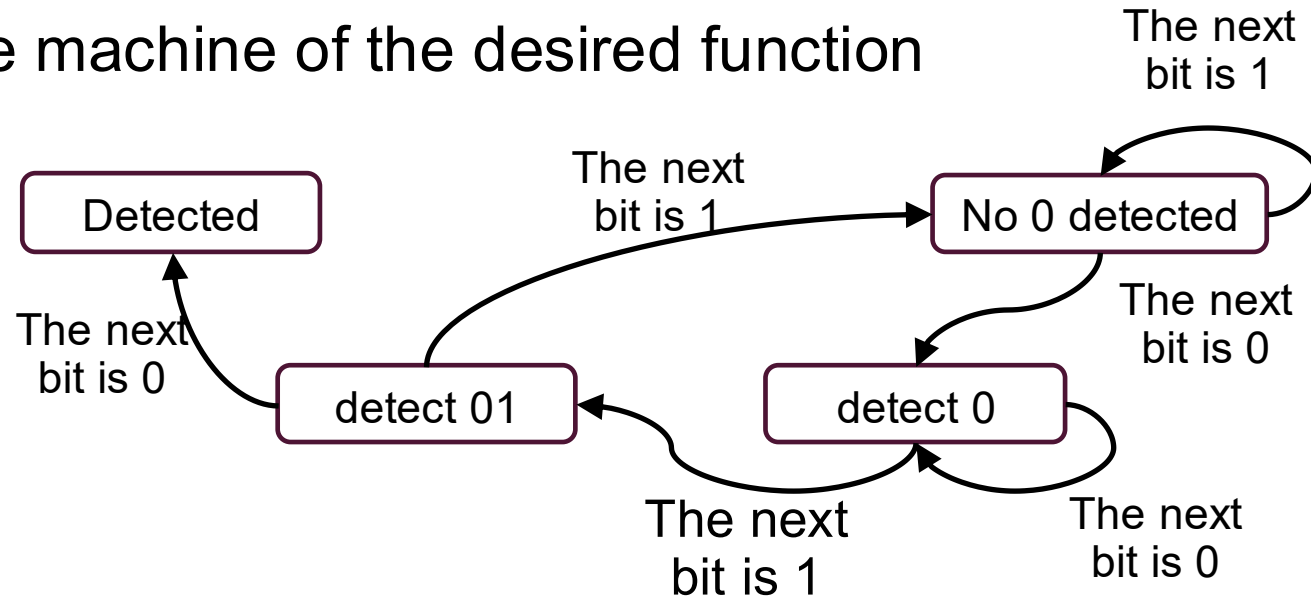


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 1: Draw finite state machine of the desired function

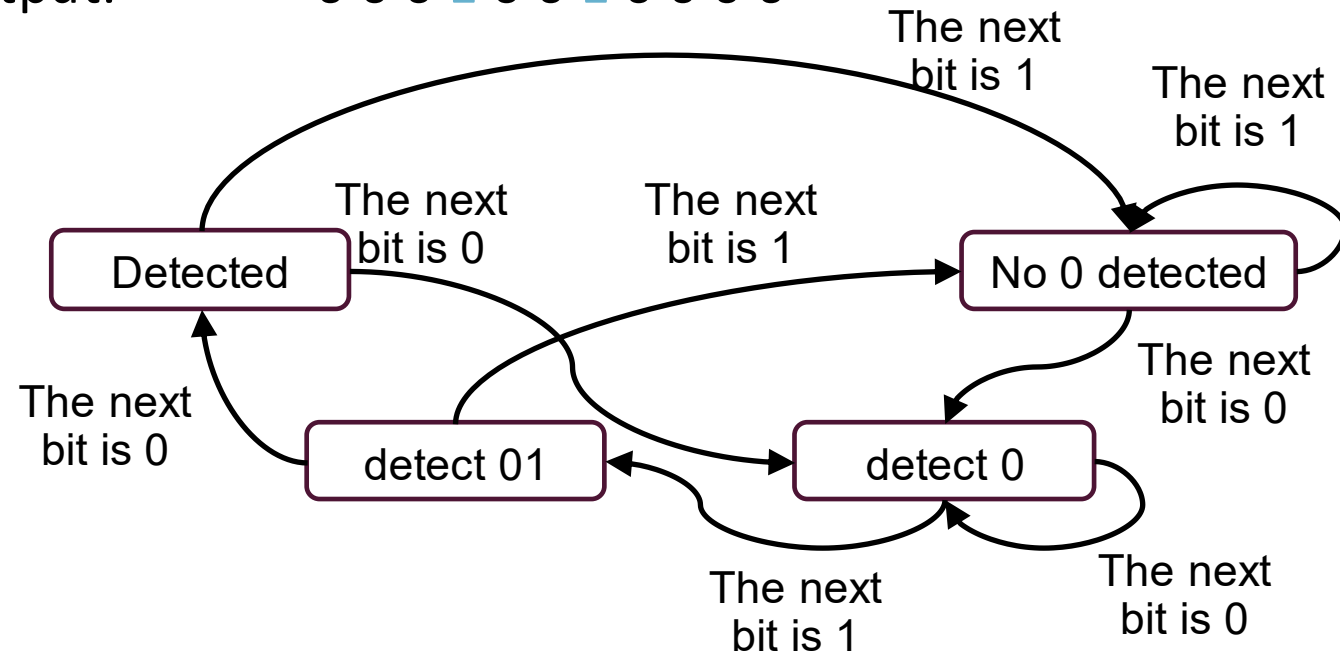


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

Output: 0 0 0 1 0 0 1 0 0 0 0



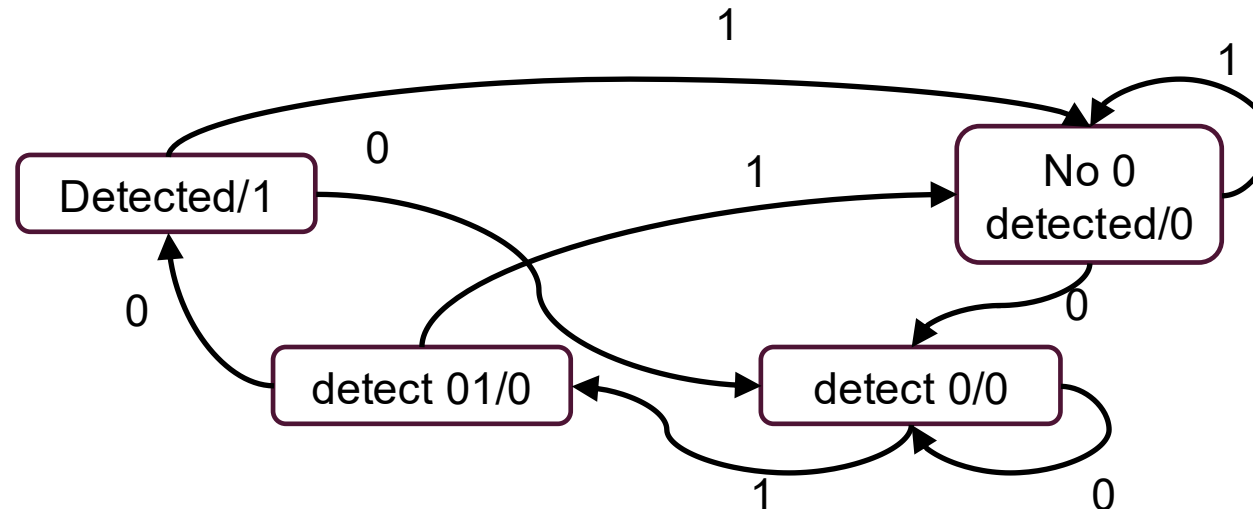
- Step 1: Draw finite state machine of the desired function

An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 1: Draw finite state machine of the desired function

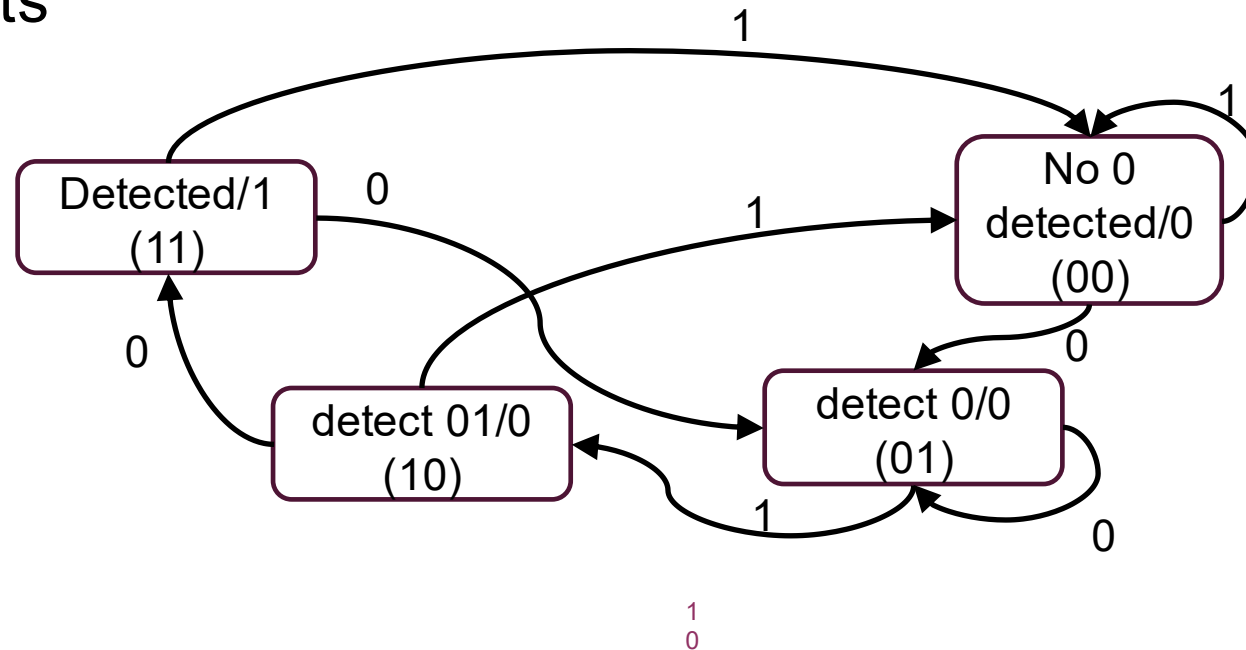


An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0
Output: 0 0 0 **1** 0 0 **1** 0 0 0 0

- Step 2: Define/assign binary numbers to represent the states, the inputs and the outputs



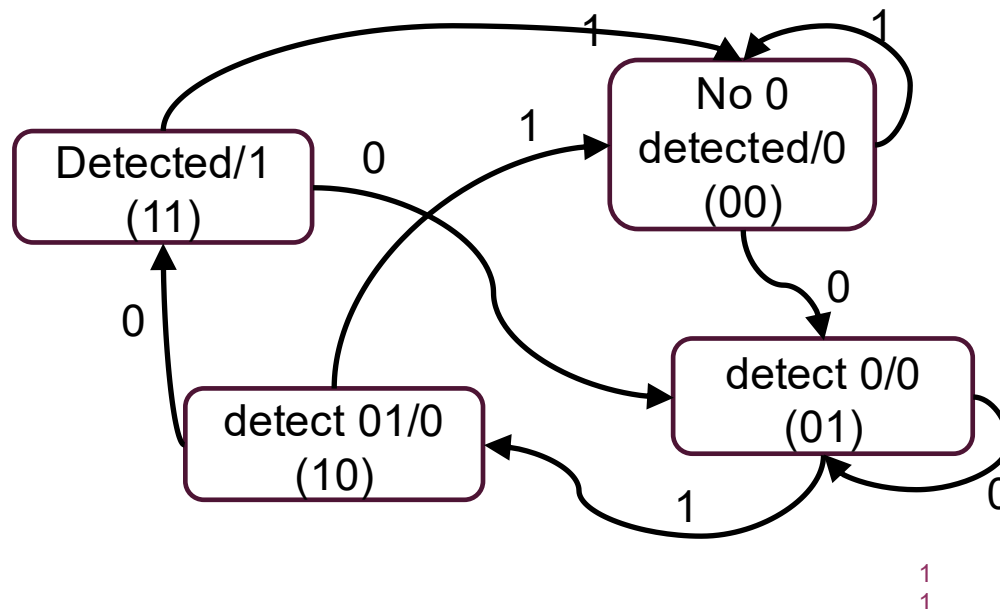
An example of FSM

- A classic problem: sequence detection for “010” (non-overlapping)

Input: 0 1 0 0 1 0 1 0 1 1 0

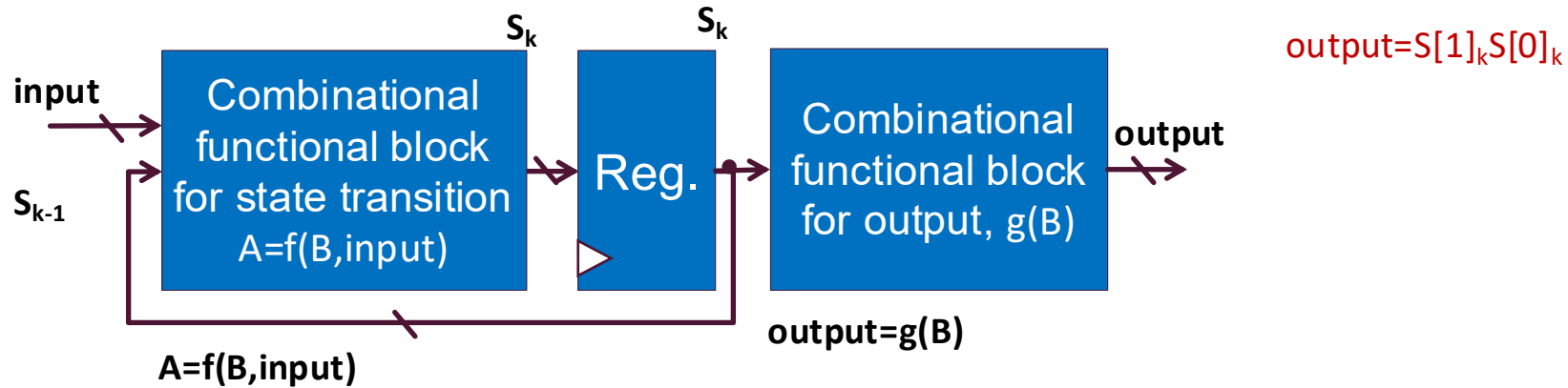
Output: 0 0 0 1 0 0 1 0 0 0 0

- Step 3: Write down the truth table (enumerate input/previous state (and current state) and their corresponding current state (and output))

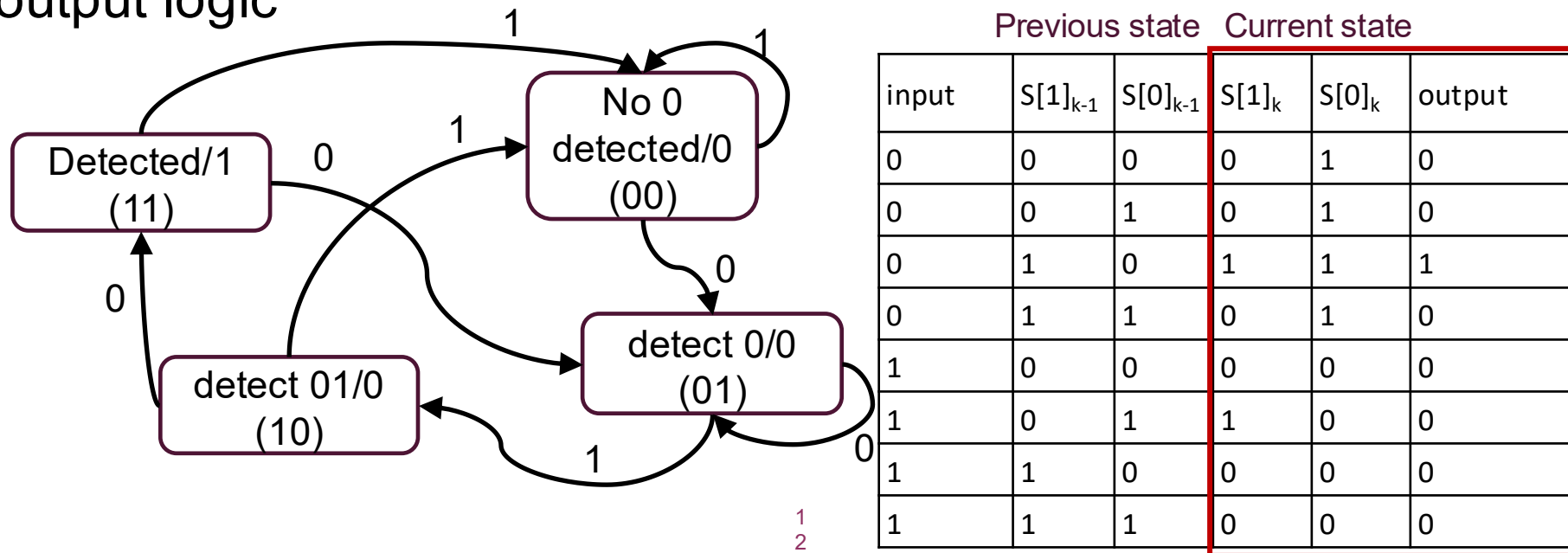


		Previous state		Current state	
input	$S[1]_{k-1}$	$S[0]_{k-1}$	$S[1]_k$	$S[0]_k$	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

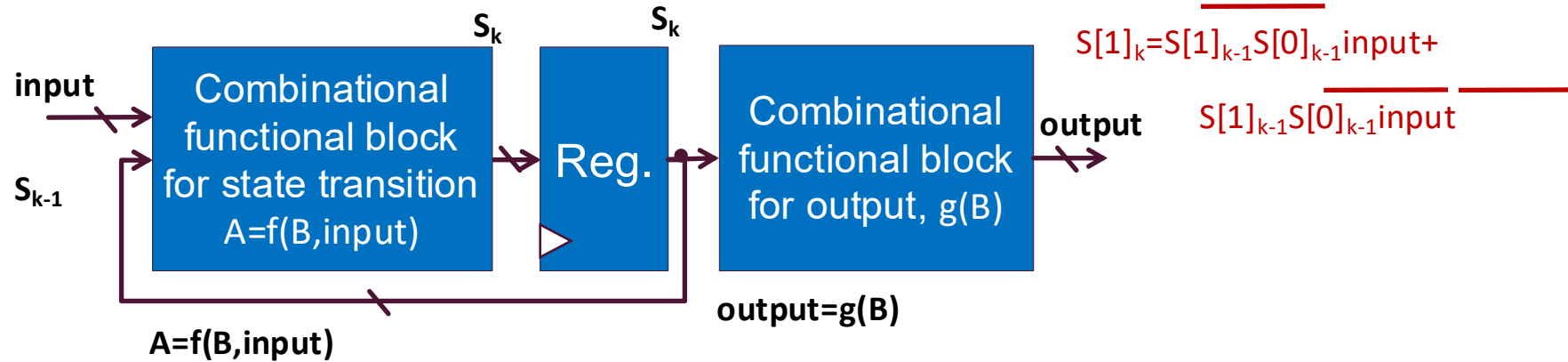
An example of FSM



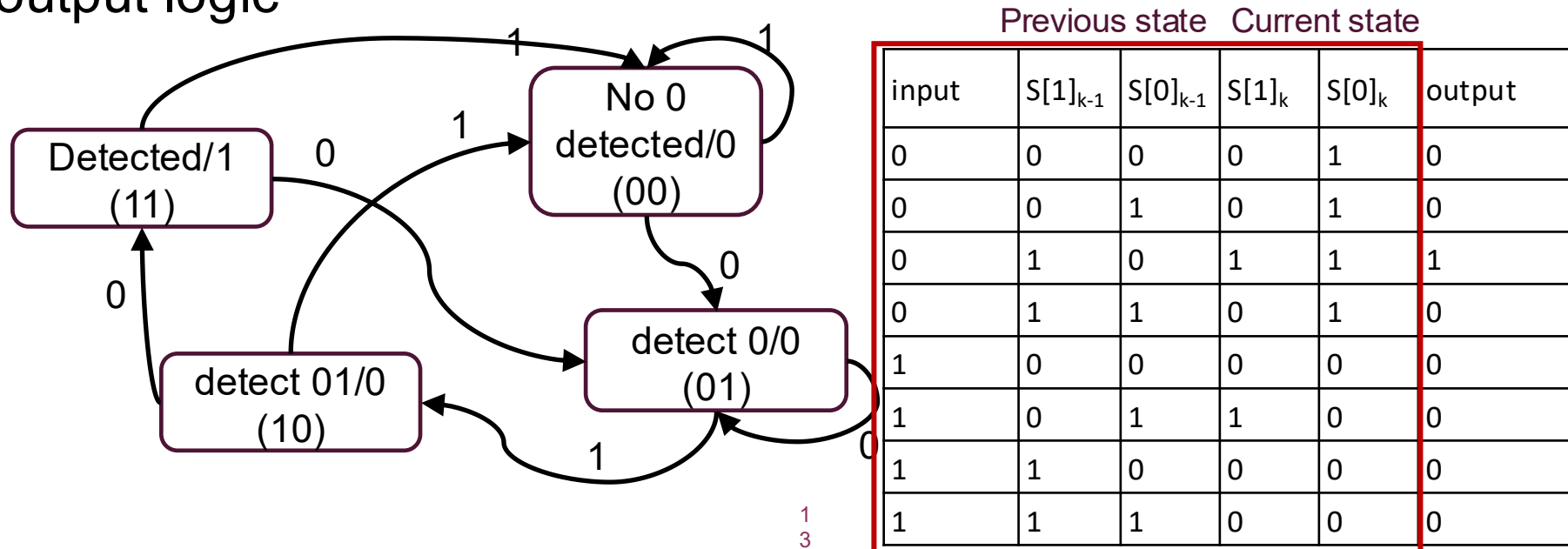
- Step 4: Use template and decide the combinational block for state transition and output logic



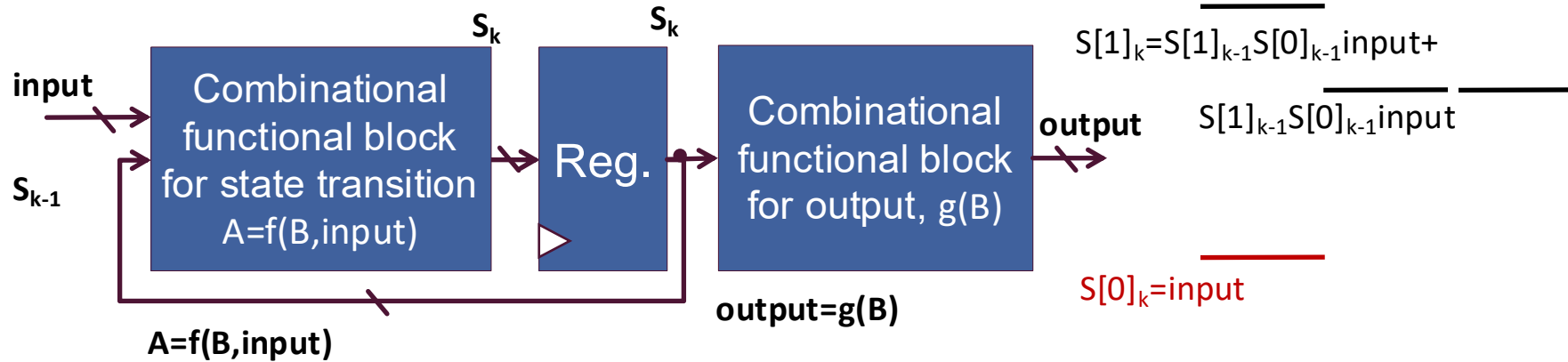
An example of FSM



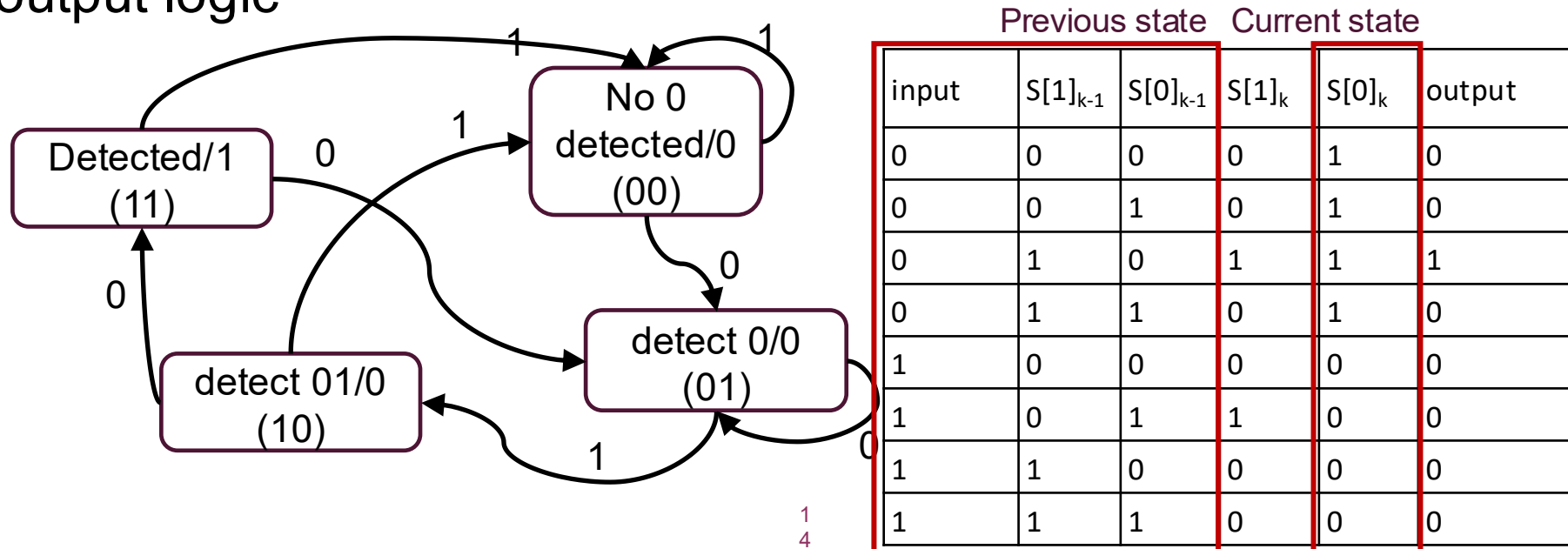
- Step 4: Use template and decide the combinational block for state transition and output logic

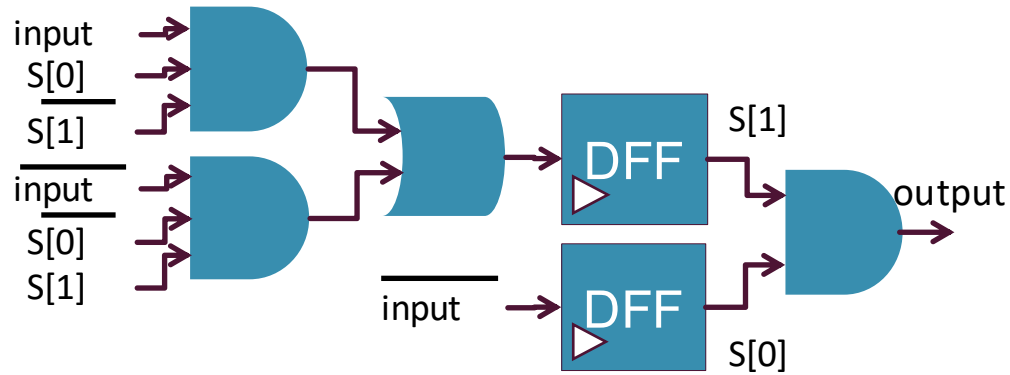


An example of FSM



- Step 4: Use template and decide the combinational block for state transition and output logic





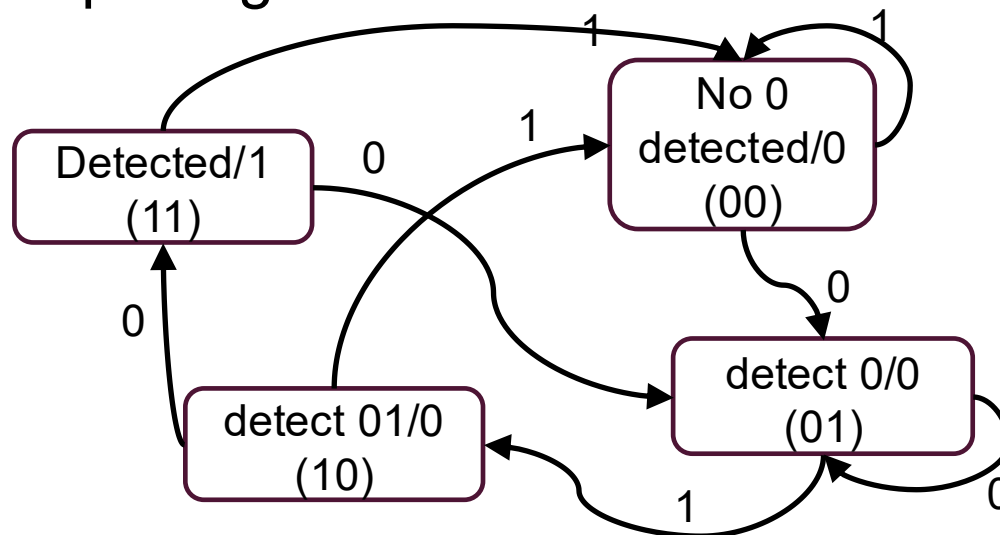
$$\text{output} = S[1]_k S[0]_k$$

$$S[1]_k = S[1]_{k-1} S[0]_{k-1} \text{input} +$$

$$S[1]_{k-1} S[0]_{k-1} \text{input}$$

$$S[0]_k = \text{input}$$

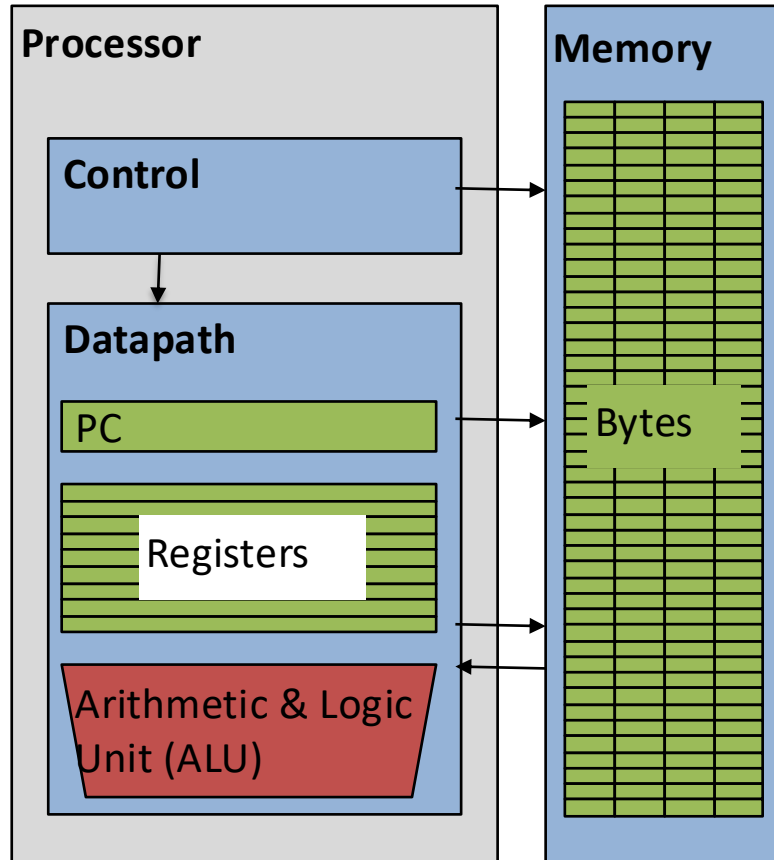
- Step 4: Use template and decide the combinational block for state transition and output logic



Previous state Current state

input	$S[1]_{k-1}$	$S[0]_{k-1}$	$S[1]_k$	$S[0]_k$	output
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

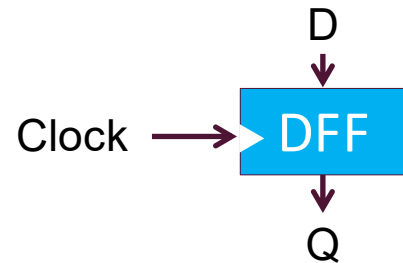
ISA as an FSM



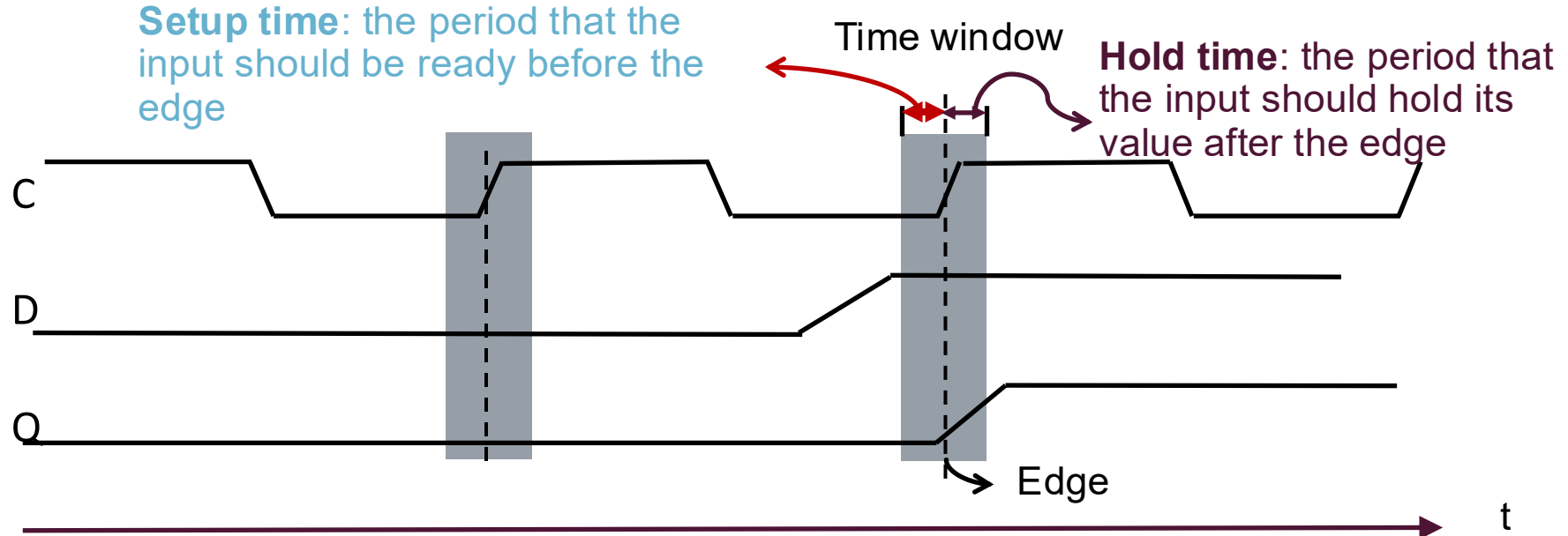
- States: all registers & memory
- Transition: register/memory value change
- Entrance: power on/reset
- Input: instructions, can change registers/memory value

Timing in synchronous circuits

- Can the frequency of the clock be infinity?

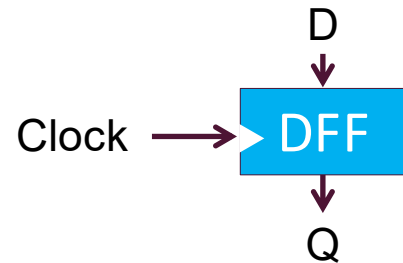


DFFs require a time window to correctly capture a signal, i.e., the signal should be stable during this time window

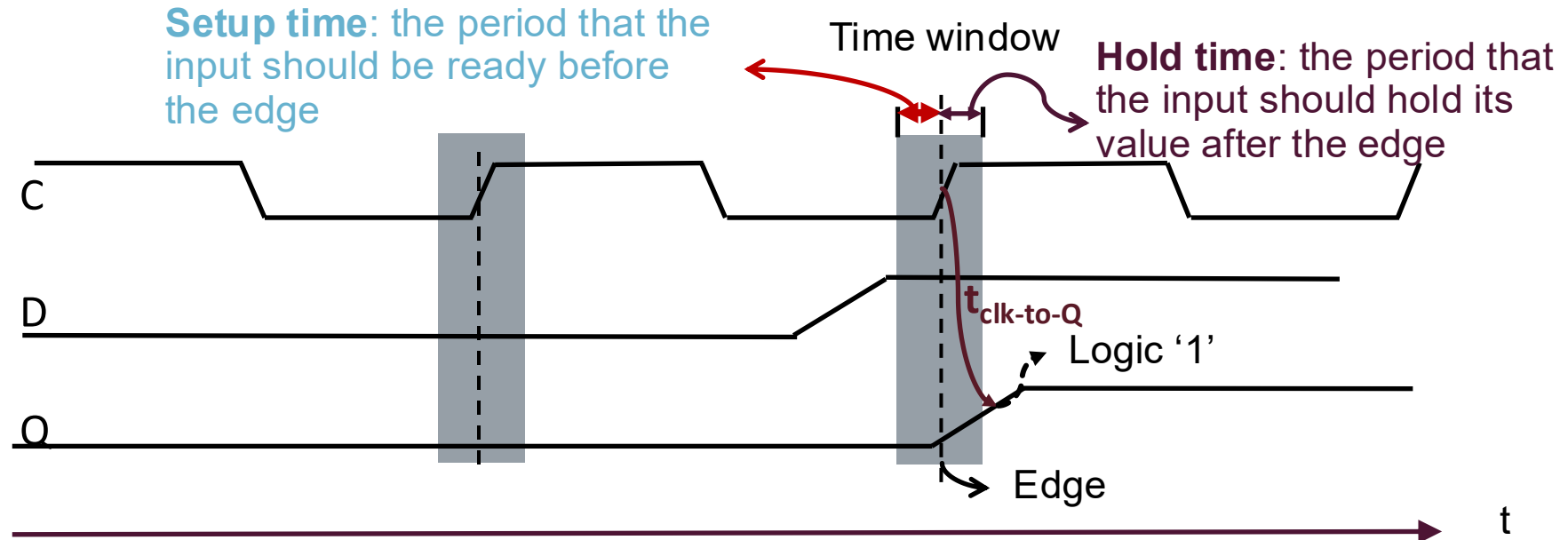


Timing in synchronous circuits

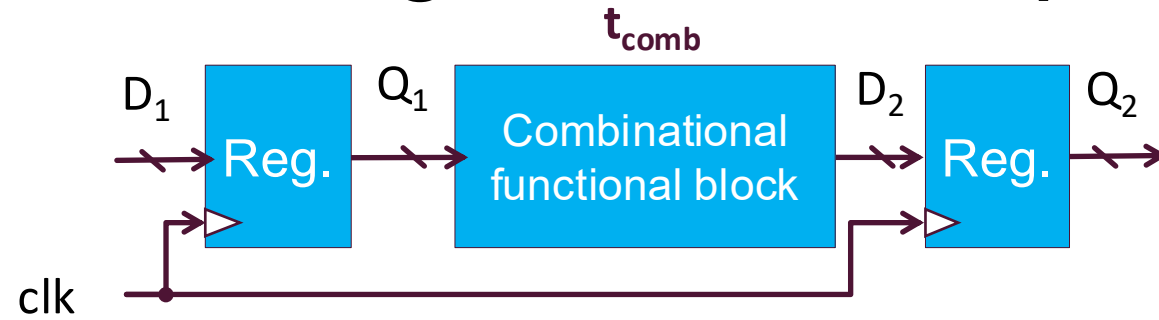
- DFF also has delay: $t_{\text{clk-to-Q}}$



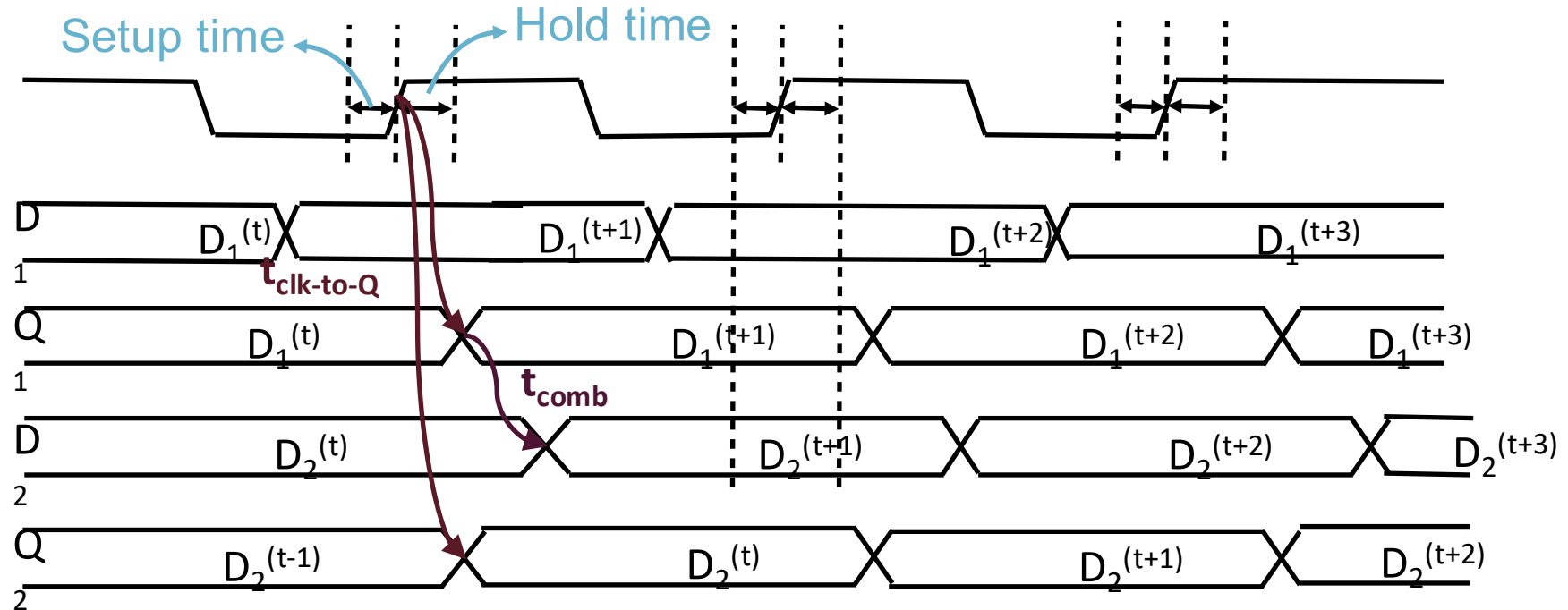
DFFs require a time window to correctly capture a signal, i.e., the signal should be stable during this time window



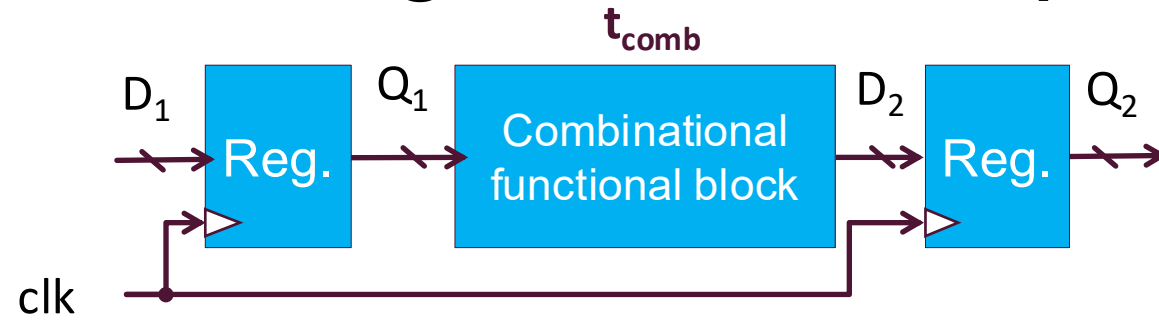
Estimating the max frequency



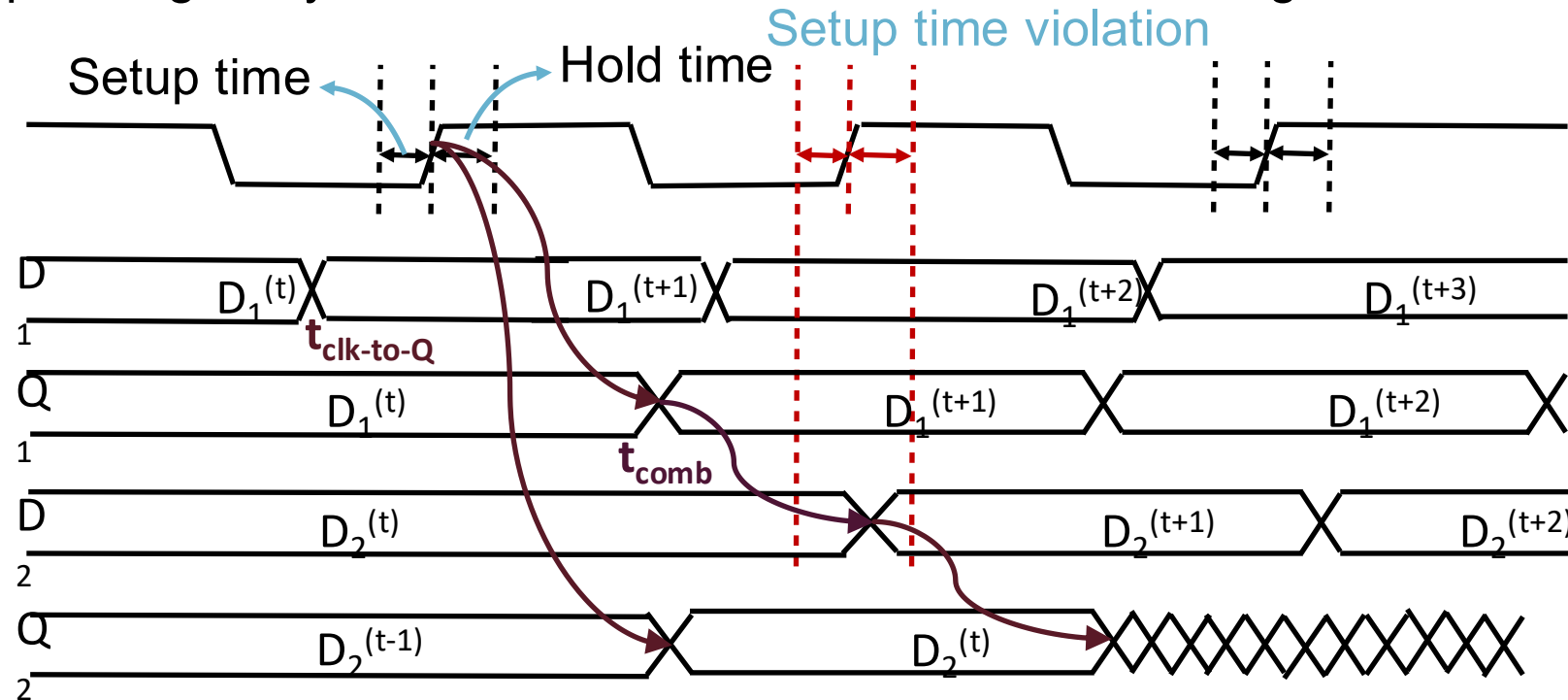
Typical digital system: a mix of combinational circuits & registers



Estimating the max frequency



Typical digital system: a mix of combinational circuits & registers

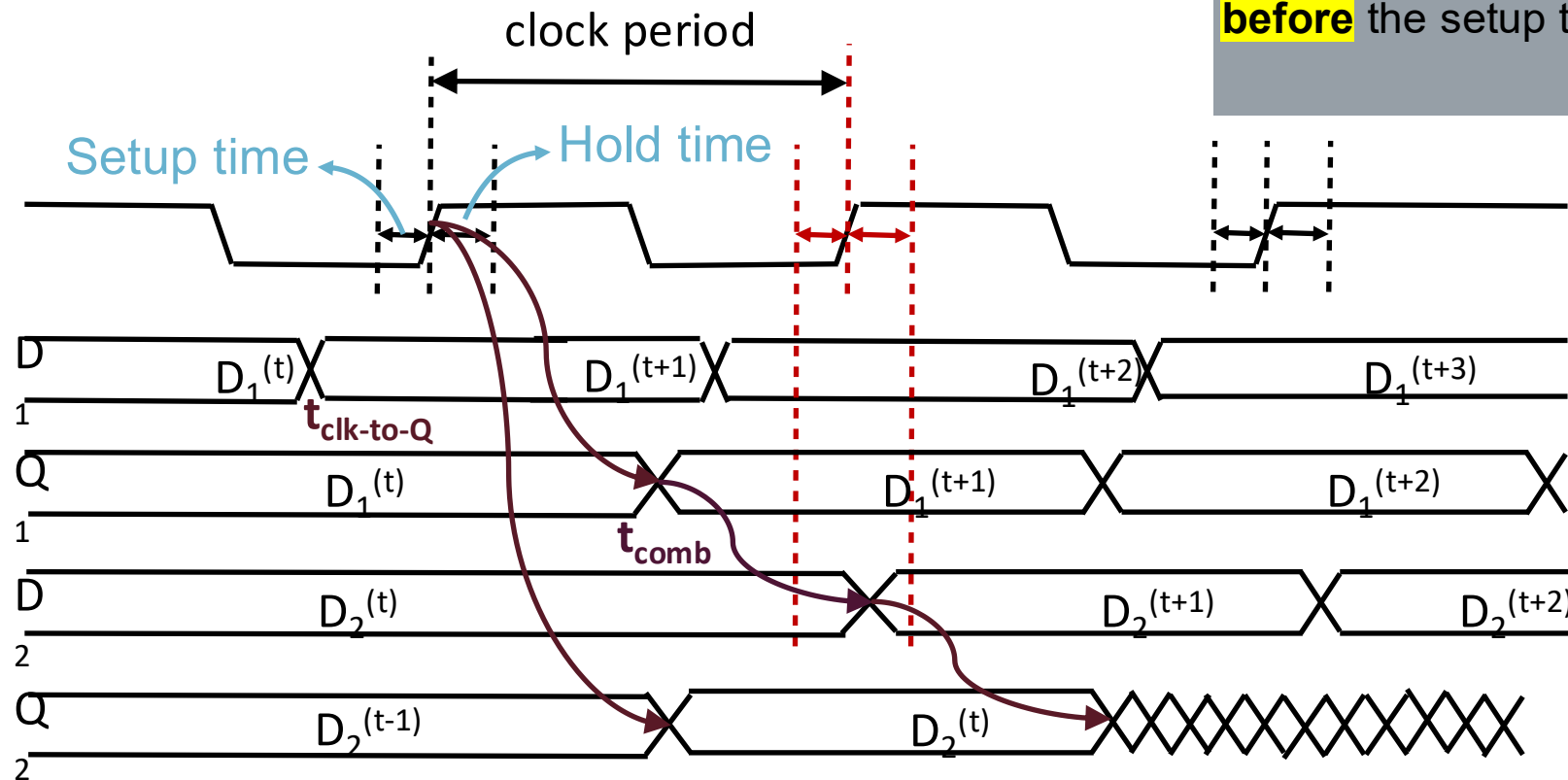


Estimating the max frequency

Max frequency = 1/min clock period

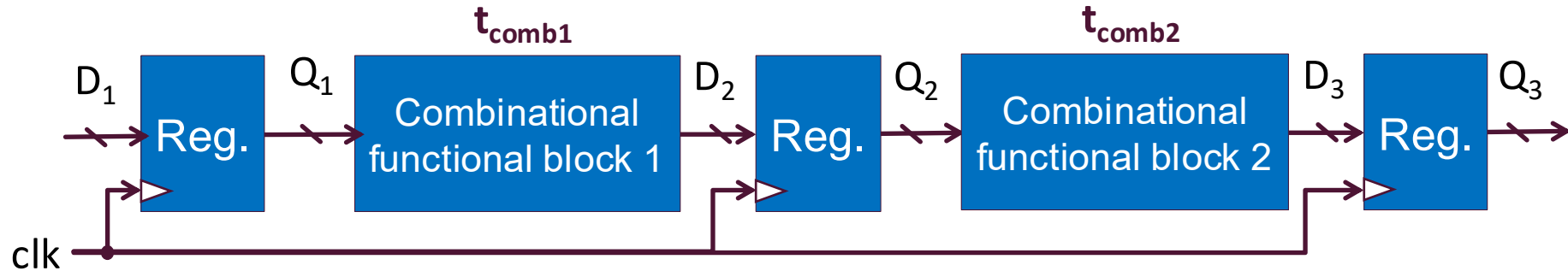
$$t_{\text{clk-to-Q}} + t_{\text{comb}} \leq \text{min clock period} - \text{setup time}$$

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 decides the max frequency.
- The corresponding path is also called **critical path**.



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

- 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.

Summary

- From FSM to SDS