

CSE 2600

Intro. To Digital Logic & Computer Design

Bill Siever
&
Michael Hall

Reminder

- Studios can (and should) be done collaboratively!
 - Studio 4A (a little more than a week away) needs to be done in a group of 2-3; Hardware handout day / need USB cable
- Homework should be done independently
 - Ok to discuss ideas and related work from studio
 - Should not discuss / share details of specific solutions

This week & misc.

- Hw2A returned: Regrade requests open for 1 week
- Hw #3B Posted / Due Sunday
 - Can re-submit autograder stuff

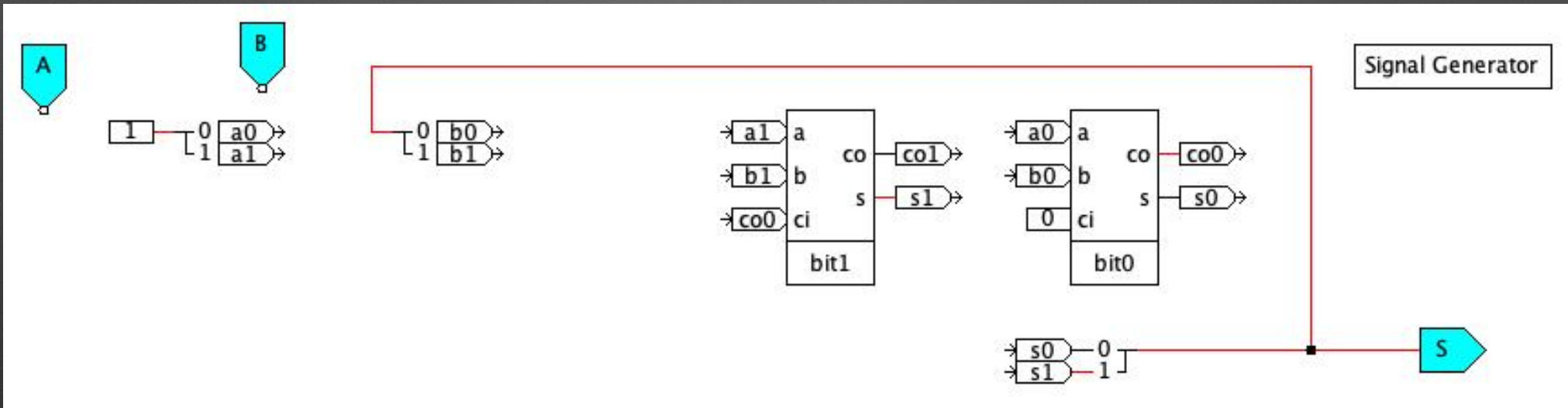
Last Lecture

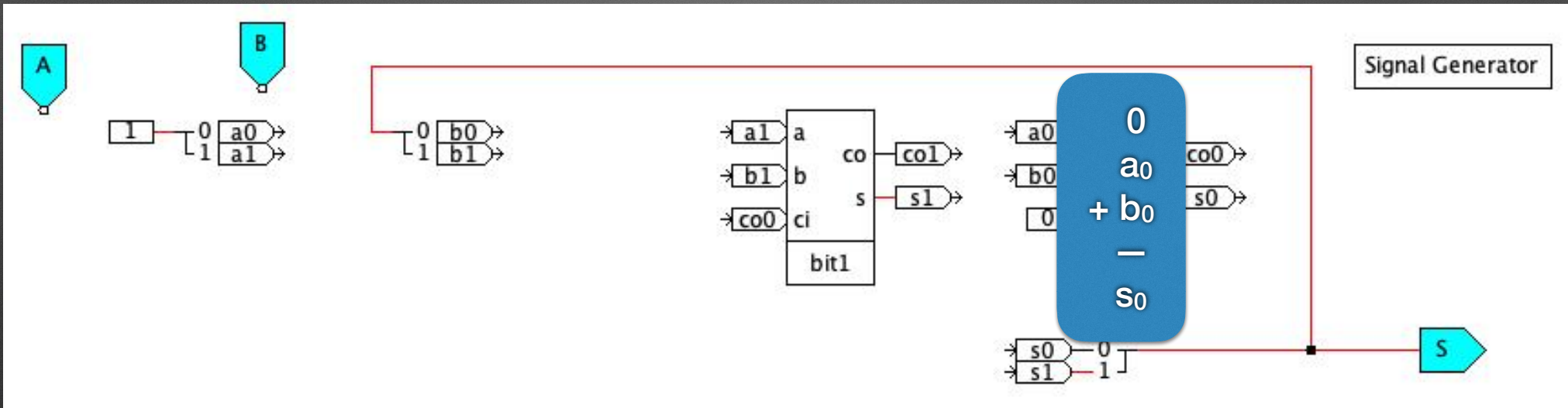
Last Lecture

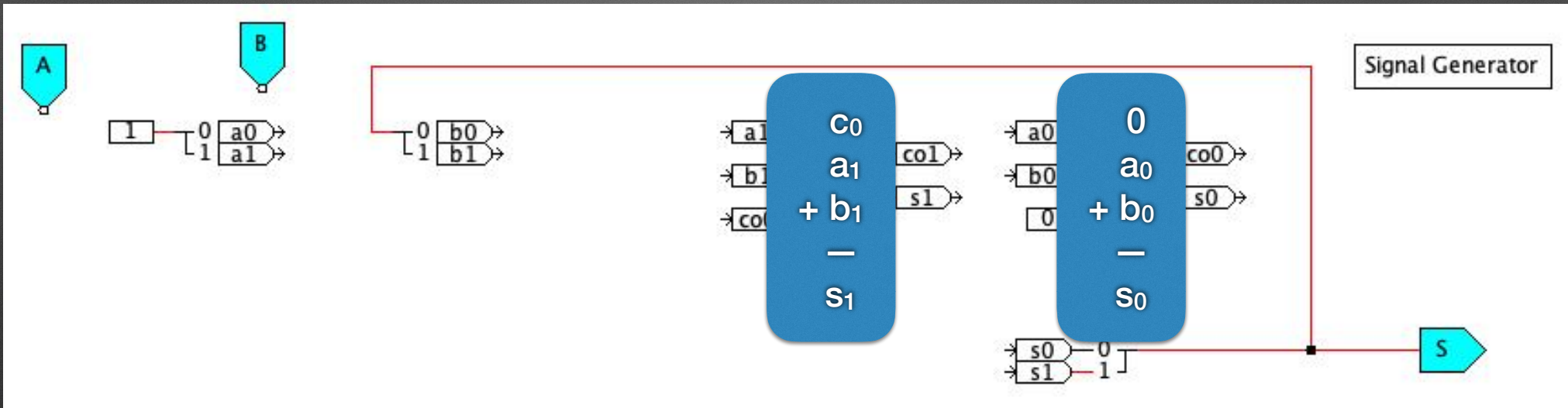
- Sequential Circuits: Have a loop and outputs impact the inputs

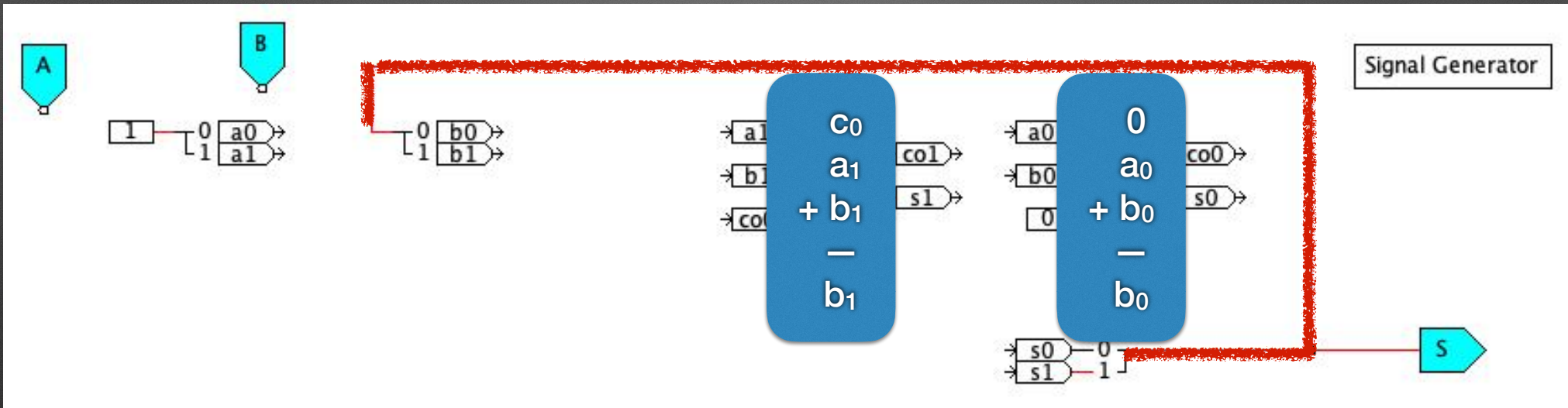
Last Lecture

- Sequential Circuits: Have a loop and outputs impact the inputs
 - Loopy adder









Last Lecture

Last Lecture

- Bistable: Will hold one of two different states

Last Lecture

- Bistable: Will hold one of two different states
 - May have some “metastable” condition (oscillating or between 1/0)

Last Lecture

- Bistable: Will hold one of two different states
 - May have some “metastable” condition (oscillating or between 1/0)
- SR-Latch: Cross-coupled NOR version

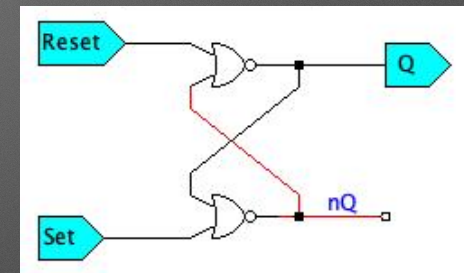
Last Lecture

- Bistable: Will hold one of two different states
 - May have some “metastable” condition (oscillating or between 1/0)
- SR-Latch: Cross-coupled NOR version
 - JLS version & time

Last Lecture

- Bistable: Will hold one of two different states
 - May have some “metastable” condition (oscillating or between 1/0)
- SR-Latch: Cross-coupled NOR version
 - JLS version & time
 - Racing and unpredictable behavior

SR Latch Issues: Hazardous Conditions!

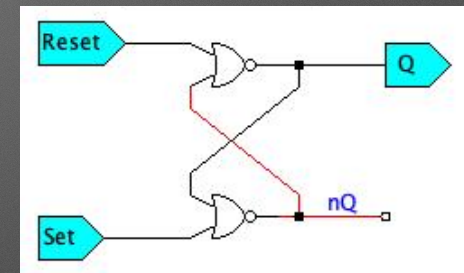


Assumptions

- NOR: delay of 5; initial output of 1
- Input pins are either set in signal generator or 0 by default

SR Latch Issues: Hazardous Conditions!

- Short 1 pulse on Reset

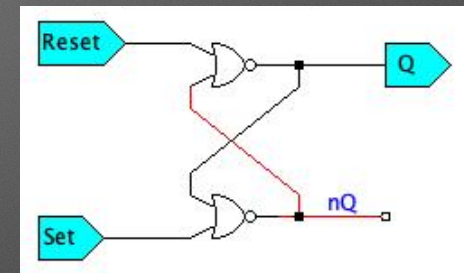


Assumptions

- NOR: delay of 5; initial output of 1
- Input pins are either set in signal generator or 0 by default

SR Latch Issues: Hazardous Conditions!

- Short 1 pulse on Reset

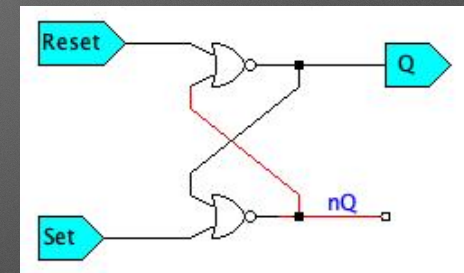


Assumptions

- NOR: delay of 5; initial output of 1
- Input pins are either set in signal generator or 0 by default

SR Latch Issues: Hazardous Conditions!

- Short 1 pulse on Reset

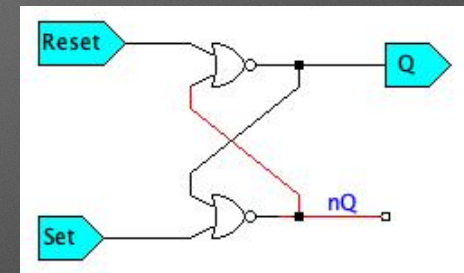


Assumptions

- NOR: delay of 5; initial output of 1
- Input pins are either set in signal generator or 0 by default

SR Latch Issues: Hazardous Conditions!

- Short 1 pulse on Reset
- R & S dropping at the same time



Assumptions

- NOR: delay of 5; initial output of 1
- Input pins are either set in signal generator or 0 by default

Last Lecture

Last Lecture

- D Latch: Transparent on high clock

Last Lecture

- D Latch: Transparent on high clock
- D Flip-Flop: Two D-latches that are transparent at opposite clock levels

Last Lecture

- D Latch: Transparent on high clock
- D Flip-Flop: Two D-latches that are transparent at opposite clock levels
 - Provide precise timing of data acquisition / storage

Last Lecture

- D Latch: Transparent on high clock
- D Flip-Flop: Two D-latches that are transparent at opposite clock levels
 - Provide precise timing of data acquisition / storage
 - General focus: positive/rising edge triggers

**D Flip Flop can be built from SR
Latches**

D Flip Flop can be built from SR Latches

- Internal latch

D Flip Flop can be built from SR Latches

- Internal latch
 - Fails if pulses too short

D Flip Flop can be built from SR Latches

- Internal latch
 - Fails if pulses too short
 - Unstable if R/S drop at same time / too close

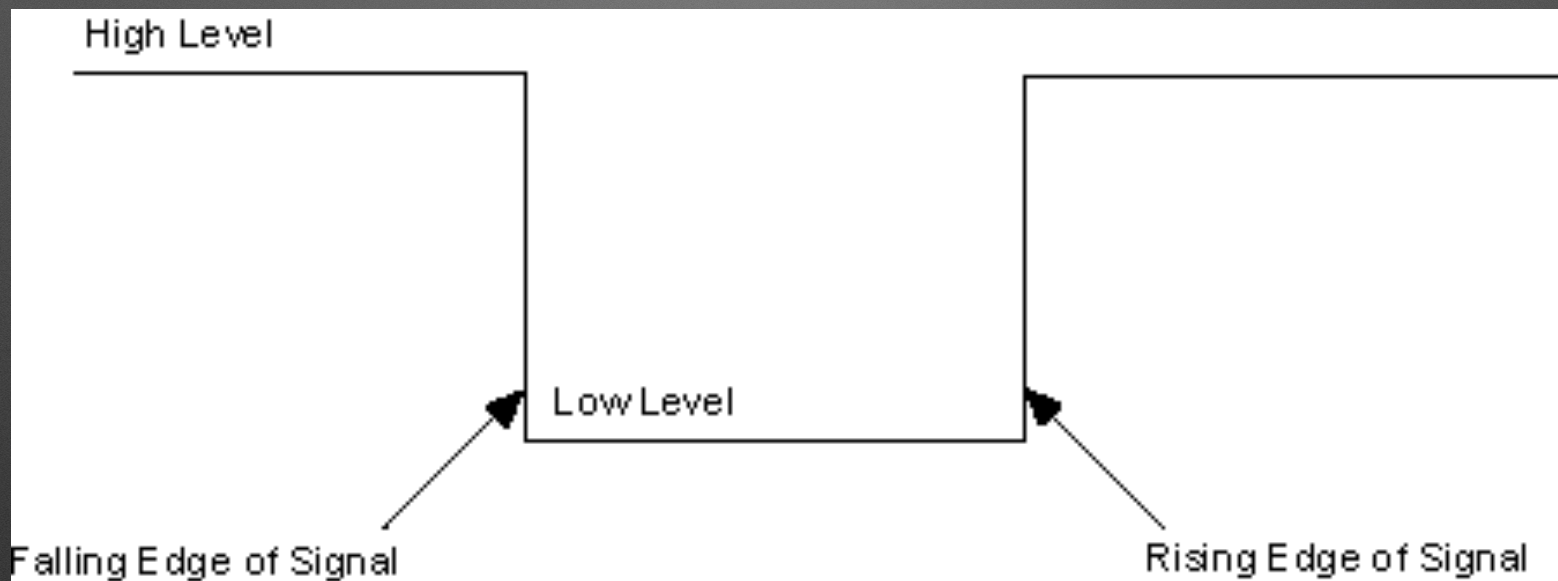
D Flip Flop can be built from SR Latches

- Internal latch
 - Fails if pulses too short
 - Unstable if R/S drop at same time / too close
- Setup Time: Time needed before clock to ensure stable capture

D Flip Flop can be built from SR Latches

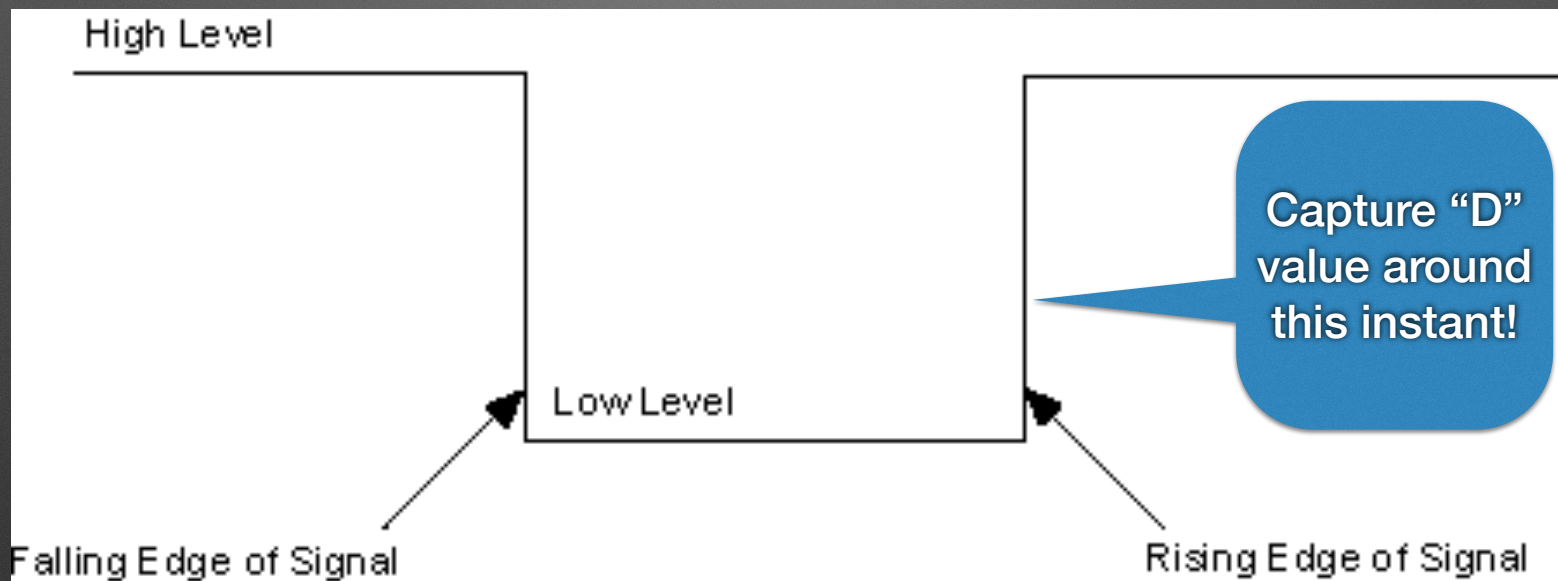
- Internal latch
 - Fails if pulses too short
 - Unstable if R/S drop at same time / too close
- Setup Time: Time needed before clock to ensure stable capture
- Hold Time: Time After clock edge value must be “held”
(Not accurately reflected in our pure-digital model)

D-Flip-Flop Clock



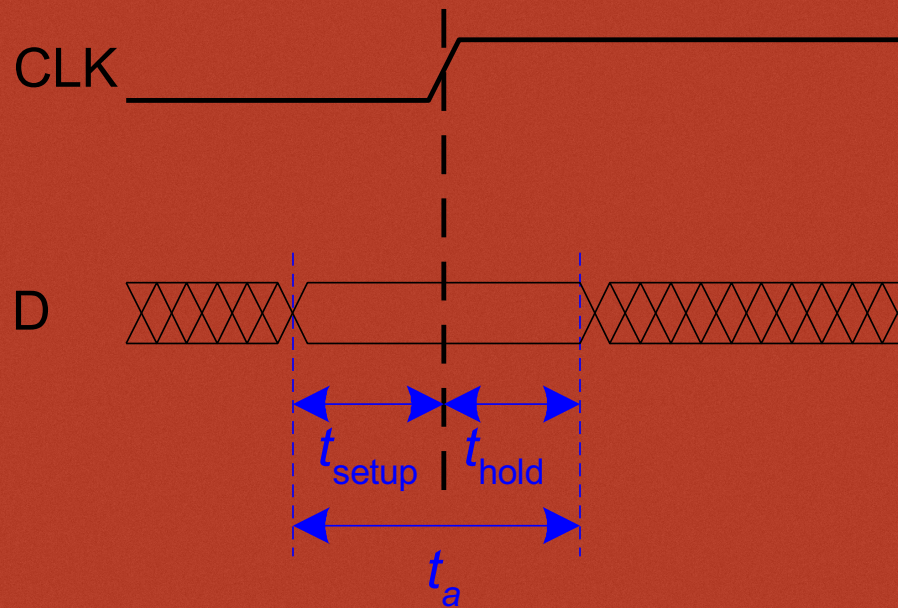
https://www.ni.com/docs/en-US/bundle/ni-hsdio/page/hsdio/fedge_trigger.html

D-Flip-Flop Clock



https://www.ni.com/docs/en-US/bundle/ni-hsdio/page/hsdio/fedge_trigger.html

Dff: Setup & Hold Time

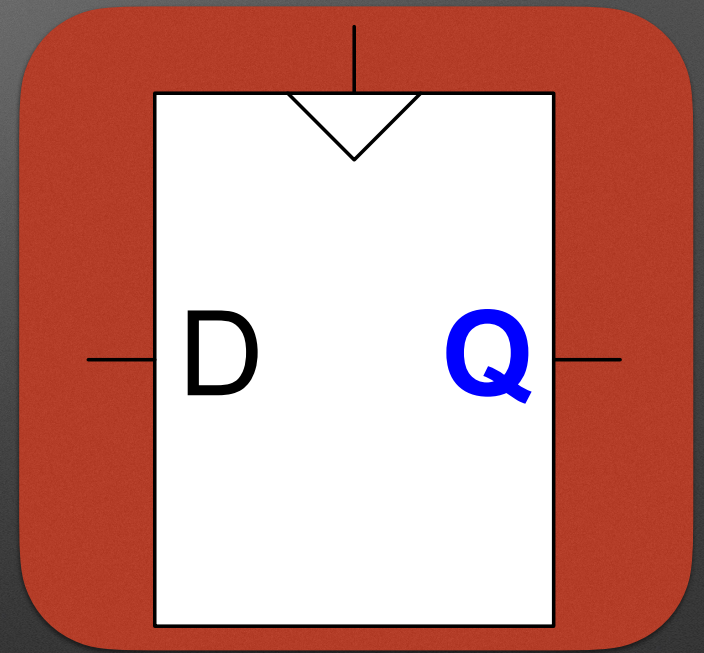


D Flip Flop Time Parameters

- t_{setup} : Time D must be stable before clock
- t_{hold} : Time D must be stable after clock
(Not accurately captured in JLS models)
- t_a : Aperture time ($t_{setup} + t_{hold}$) —
total window of time D needs to be stable around clock

Dff Time Parameters Relative to Clock

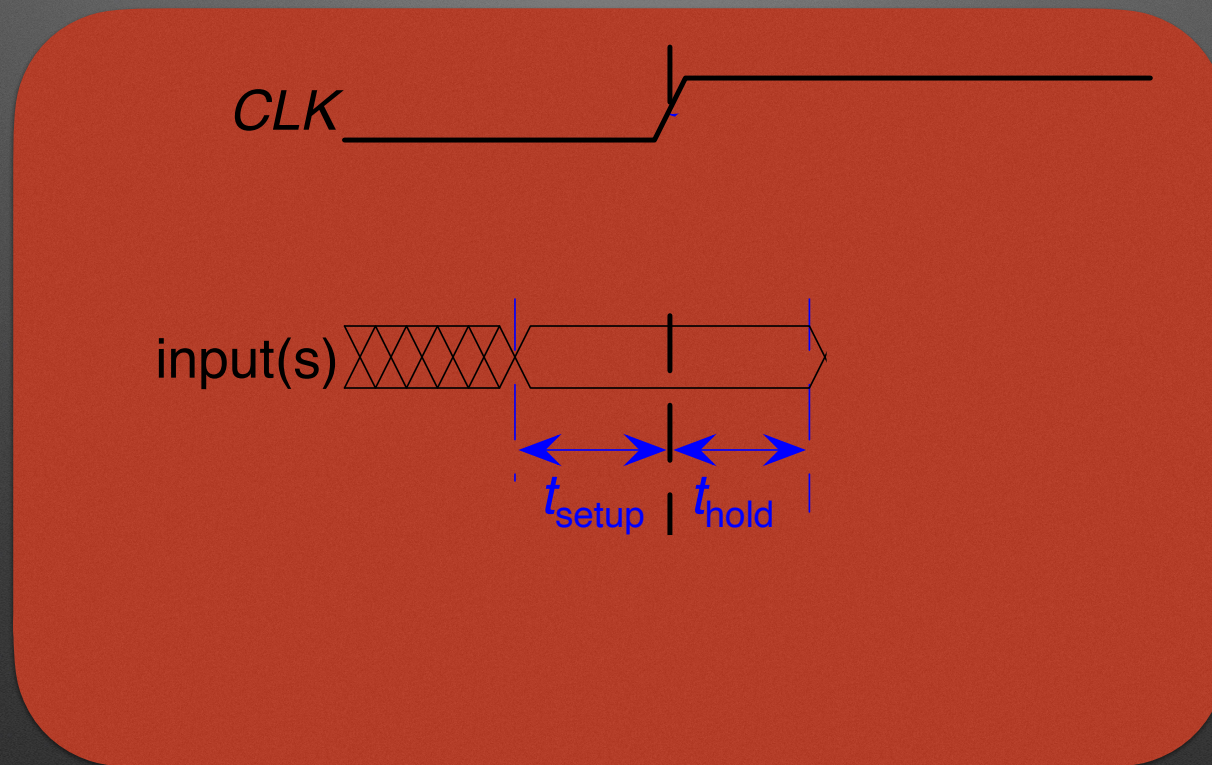
- t_{pcq} : Propagation delay from Clock to Q (pcq)
- t_{ccq} : Contamination delay from C to Q (ccq)



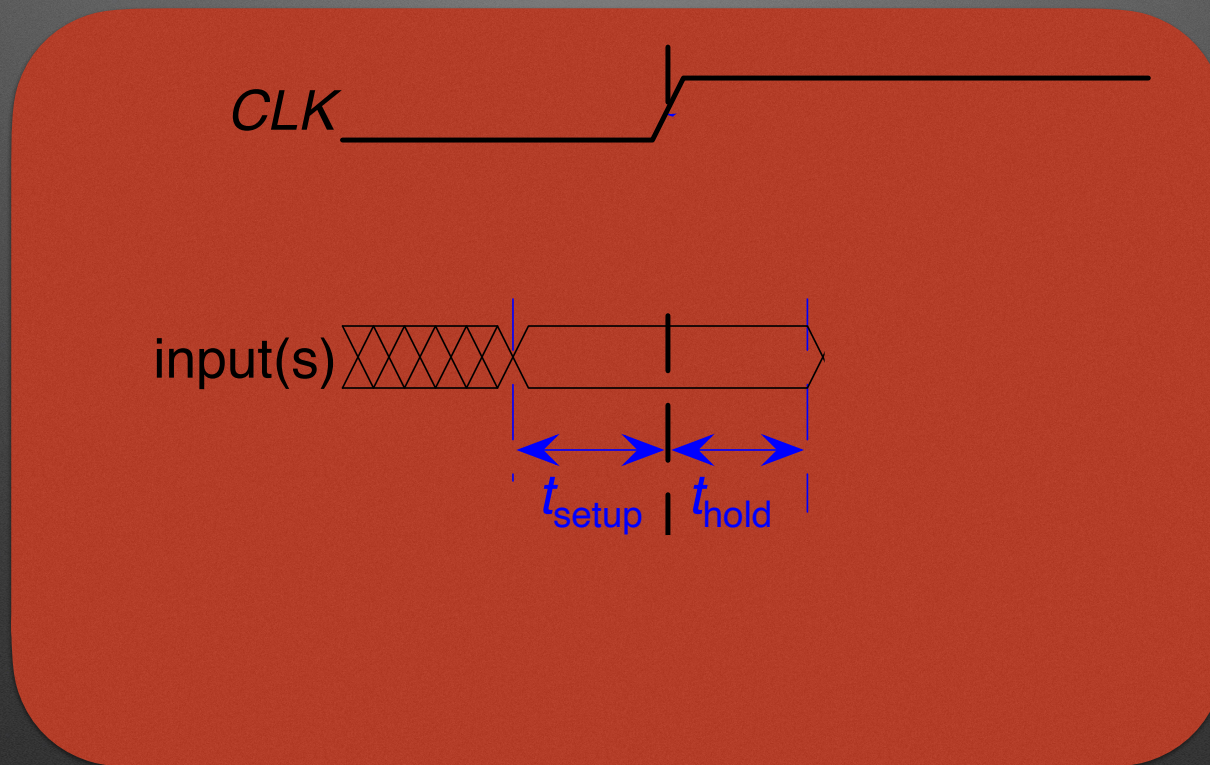
Flip-Flop Timing Parameters

CLK 

Flip-Flop Timing Parameters

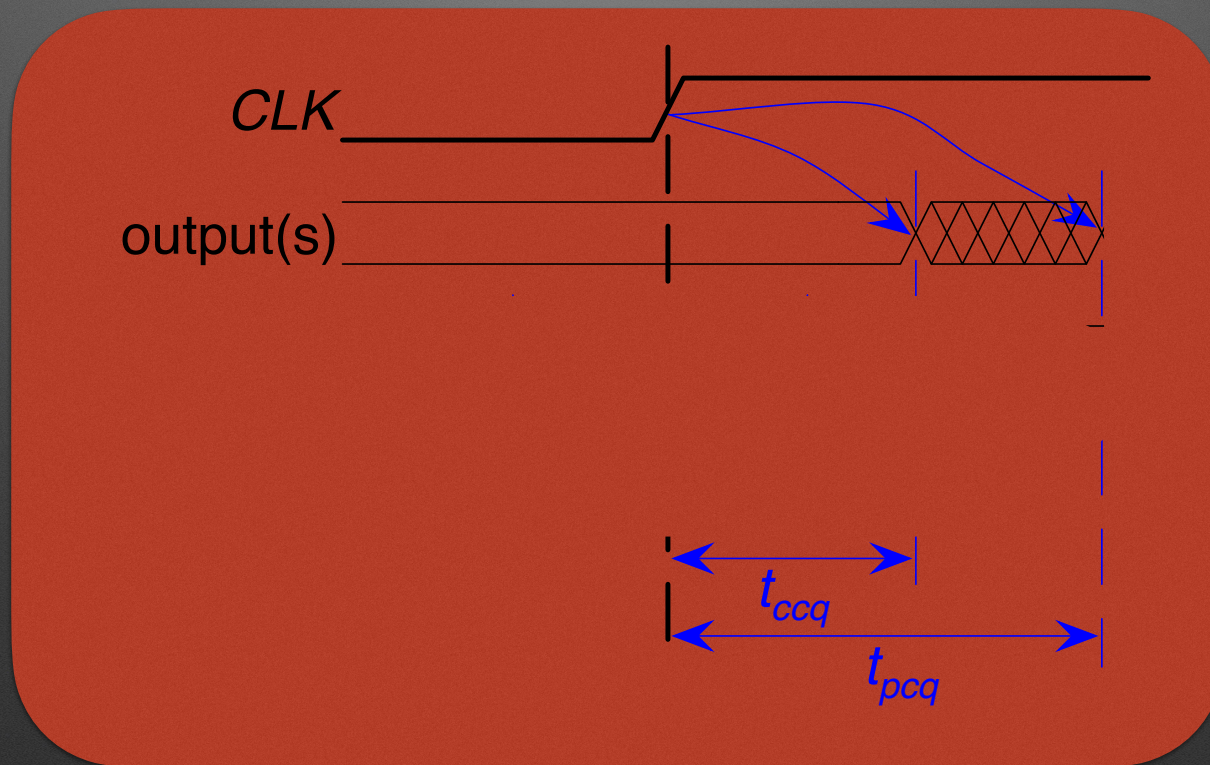


Flip-Flop Timing Parameters

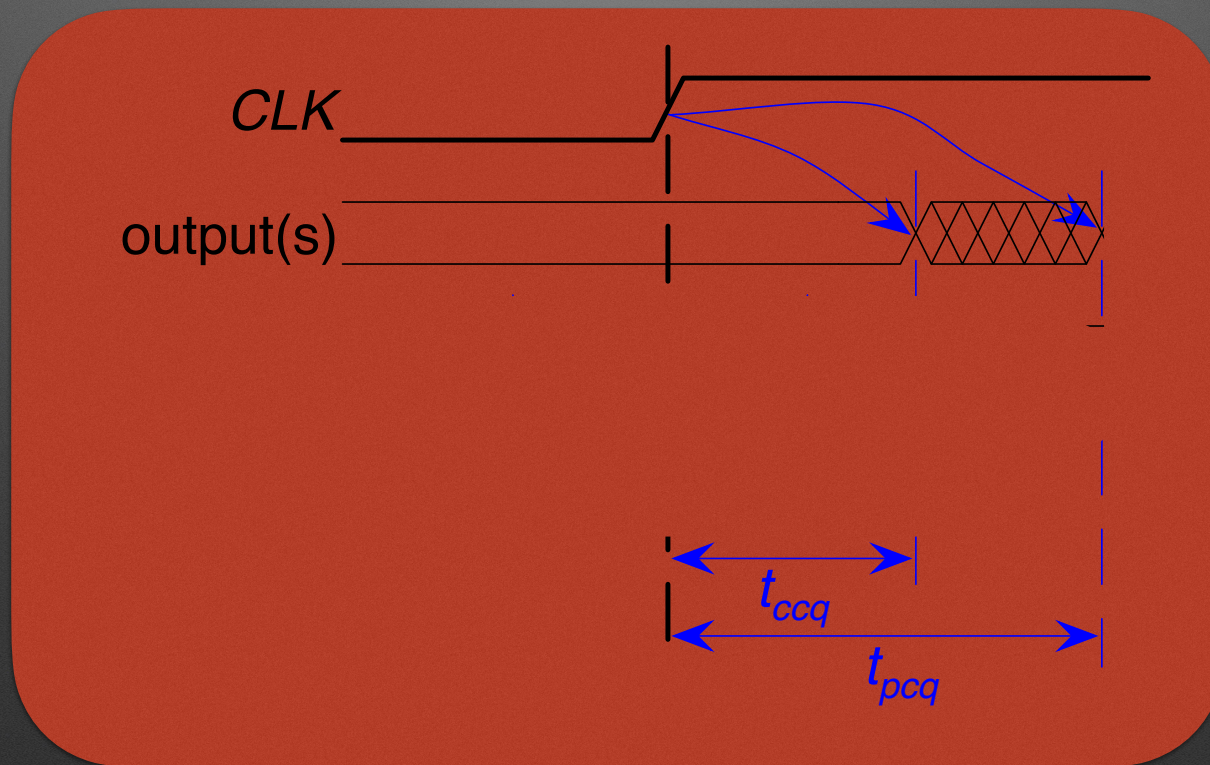


Have to make sure inputs are stable during aperture

Flip-Flop Timing Parameters

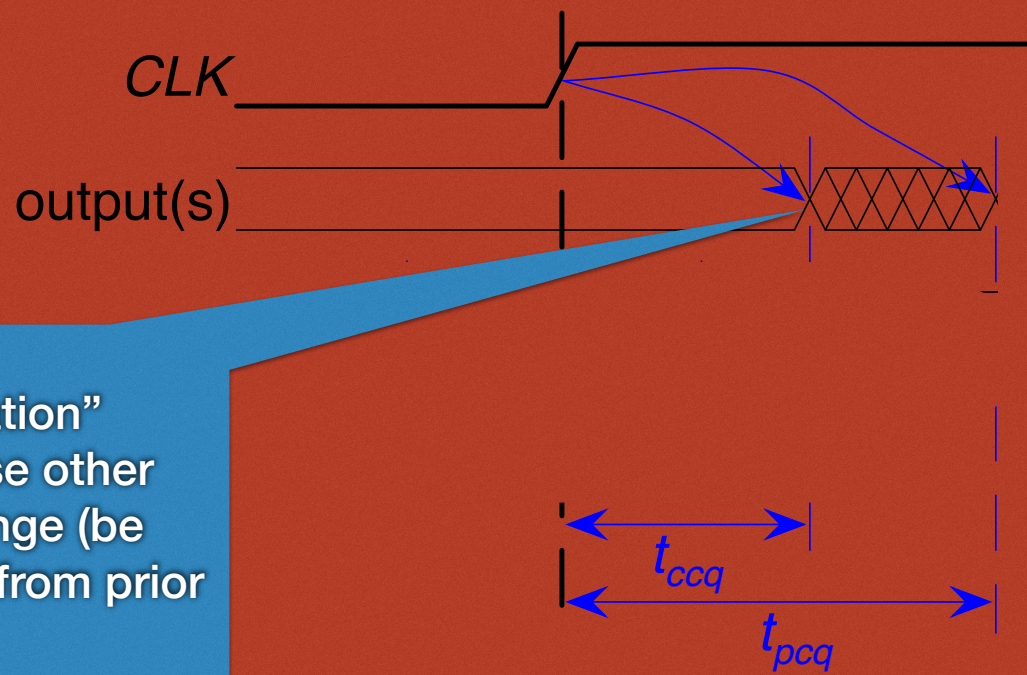


Flip-Flop Timing Parameters



Min & max time after clock until outputs change

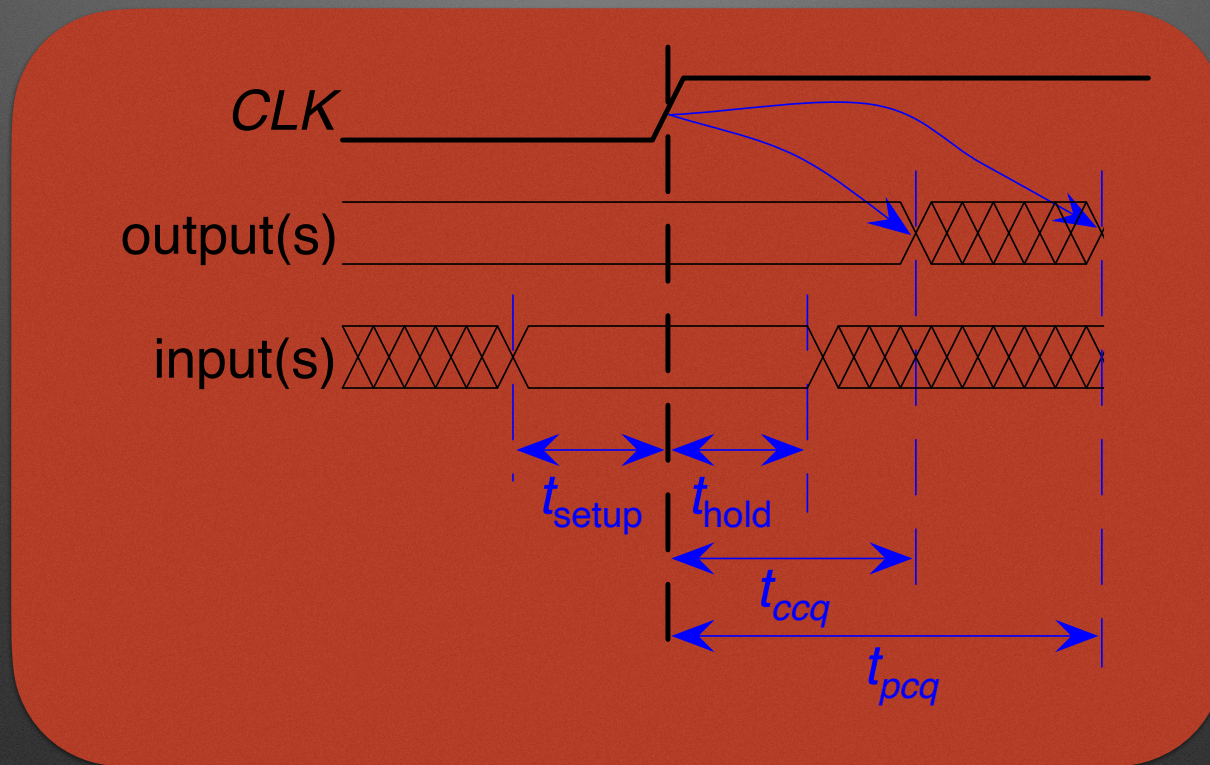
Flip-Flop Timing Parameters



“Contamination”
This may cause other
things to change (be
“contaminated” from prior
state)

Min & max time after clock until outputs change

Flip-Flop Timing Parameters



Sequential & Synchronous Logic

- Sequential circuits
 - Can't be represented with a *simple* table of just inputs and outputs (Possibly a complex table of history of inputs and outputs)
 - Output depends on sequence of inputs and timing
- Synchronous Sequential Circuits
 - Sequential circuits with additional restrictions on form to improve predictability

Synchronous Sequential Circuits

Synchronous Sequential Circuits

- Are synchronized by a common clock

Synchronous Sequential Circuits

- Are synchronized by a common clock
- Uses registers (*D Flip-Flops*)

Synchronous Sequential Circuits

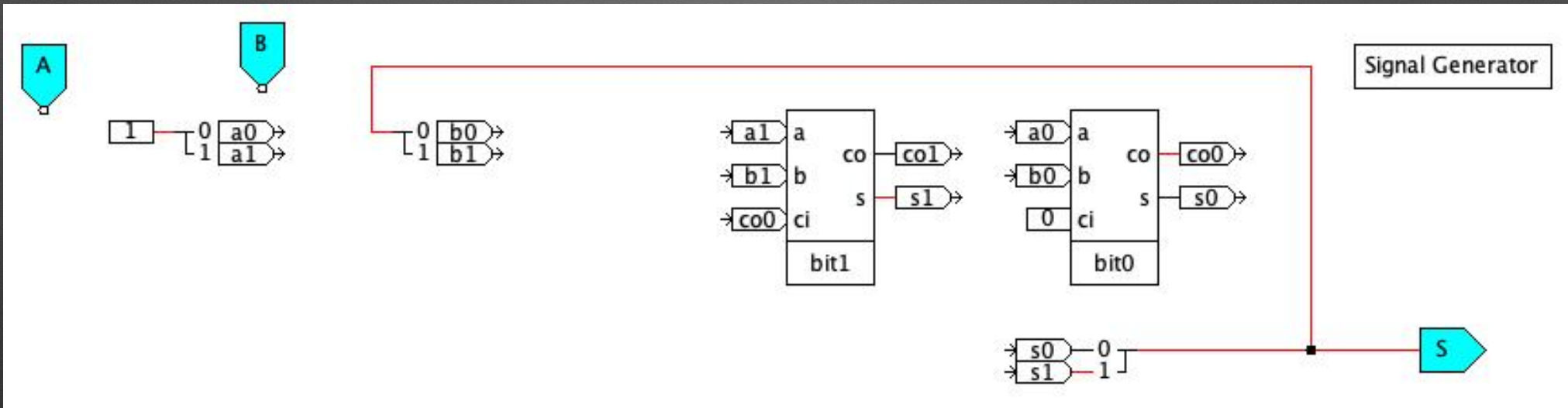
- Are synchronized by a common clock
- Uses registers (*D Flip-Flops*)
- Mix of registers and combinational logic

Synchronous Sequential Circuits

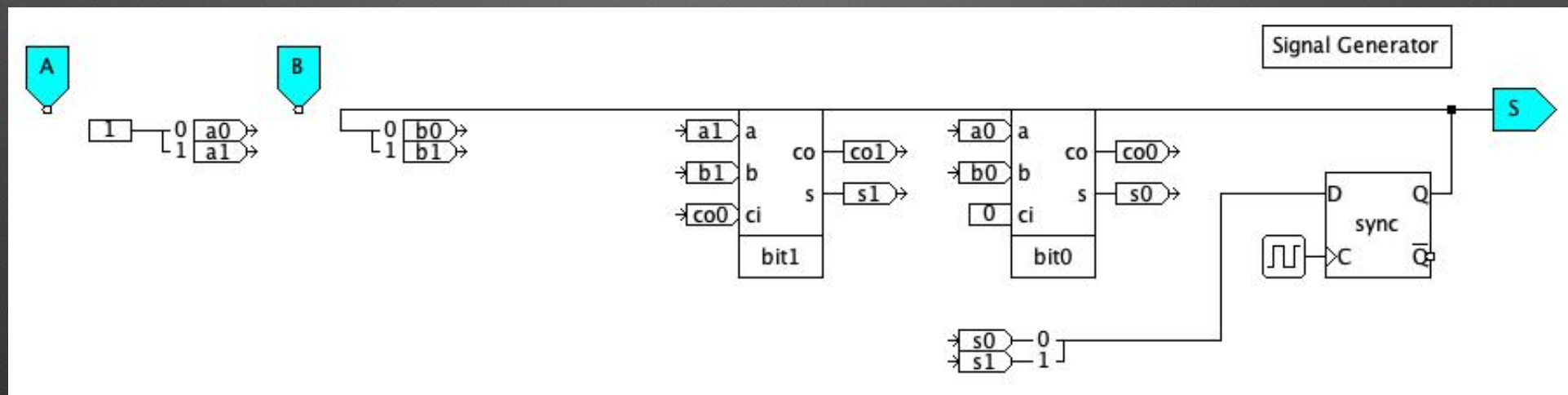
- Are synchronized by a common clock
- Uses registers (*D Flip-Flops*)
- Mix of registers and combinational logic
- All cycles in circuit include *at least one register*

Synchronous Sequential Circuits

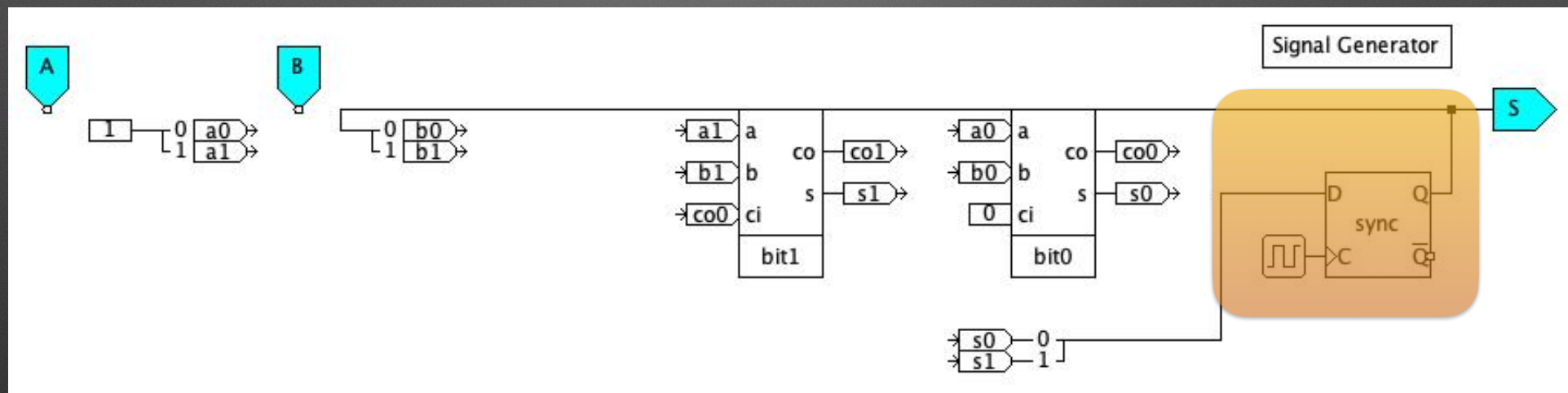
- Are synchronized by a common clock
- Uses registers (*D Flip-Flops*)
- Mix of registers and combinational logic
- All cycles in circuit include *at least one register*
- Goal: Impose predictable behavior!



Loopy Adder w/ Synchronous



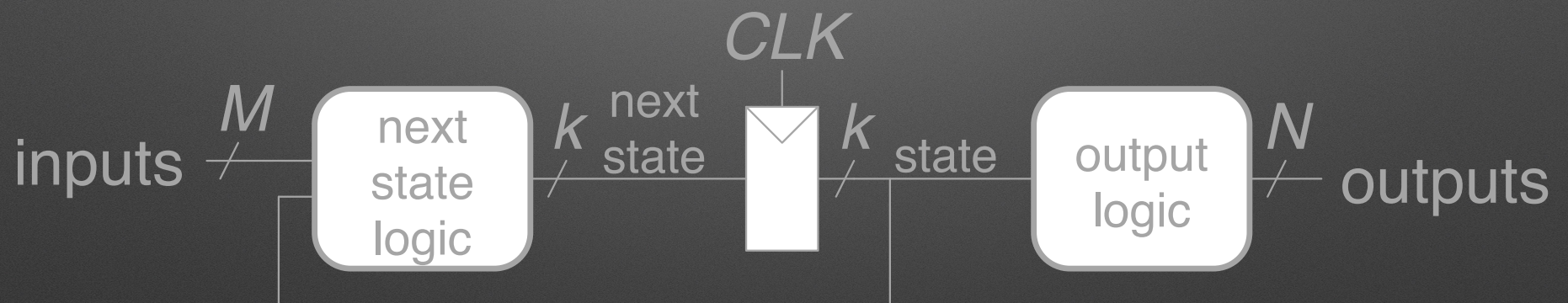
Loopy Adder w/ Synchronous



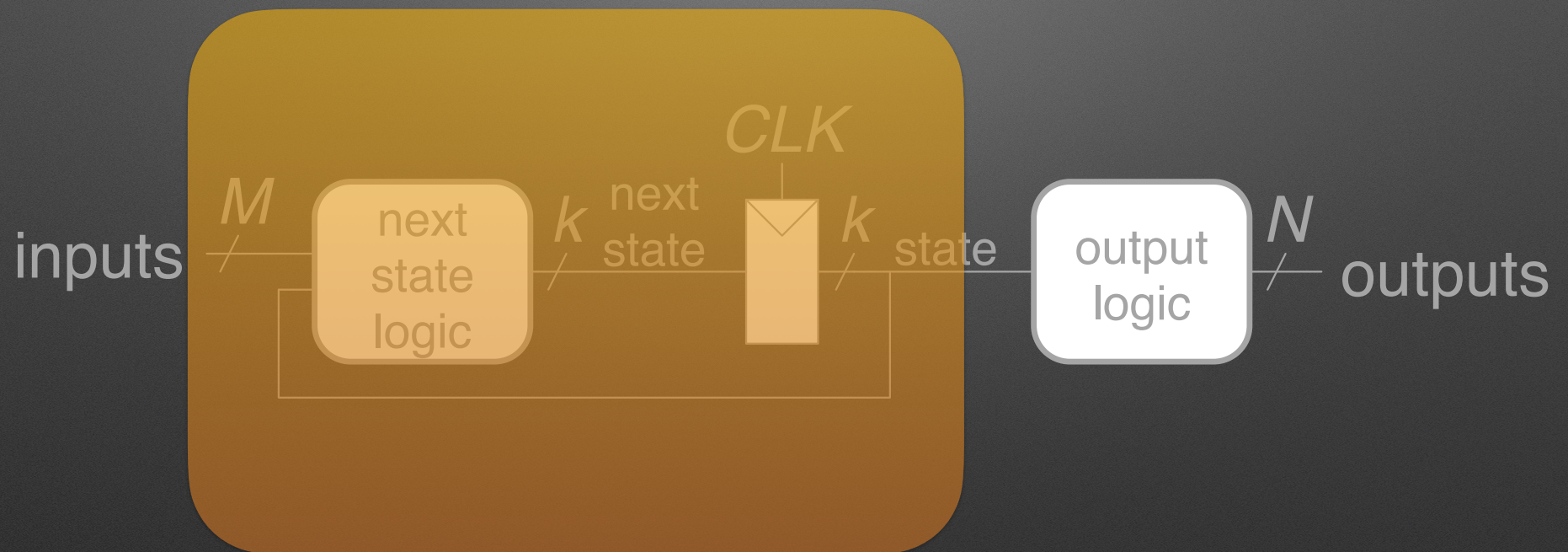
Finite State Machines

- Can be realized with synchronous sequential circuits
 - Memory (registers/flip-flops) hold encoded state (represents the location or current bubble in diagram)
 - Combinational logic for:
 - Output control (the outputs in the bubbles)
 - Determine next state (the arrows)

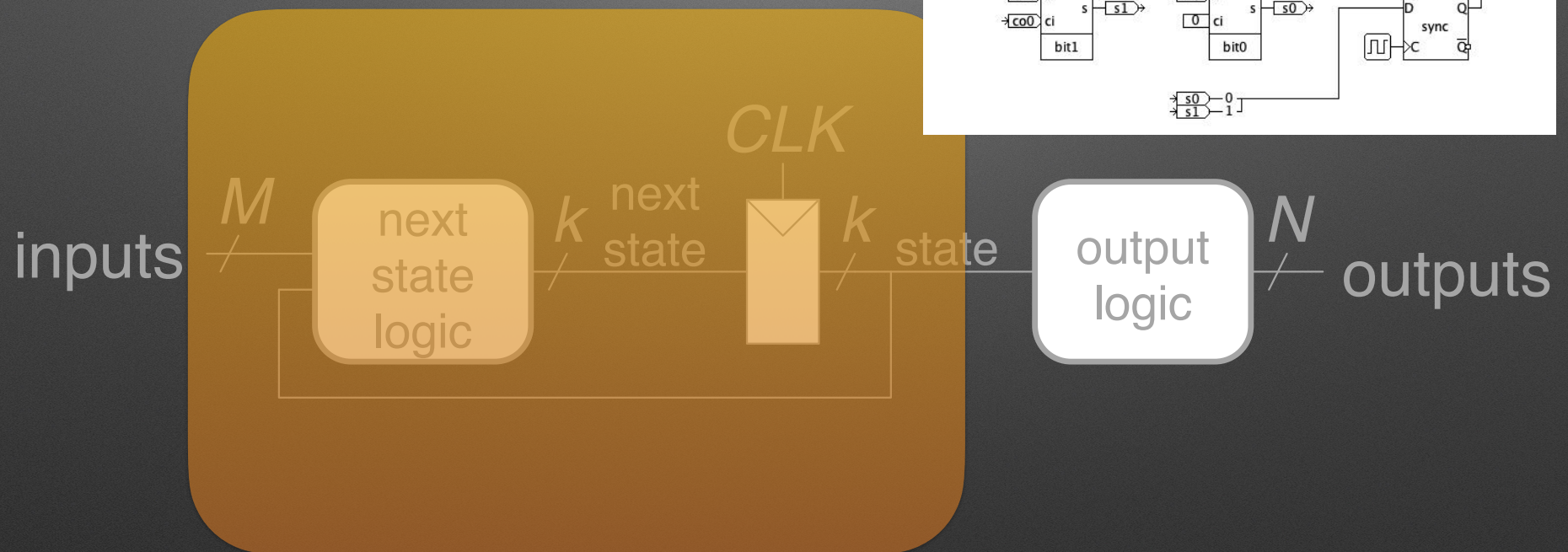
FSM: Moore Machine Structure



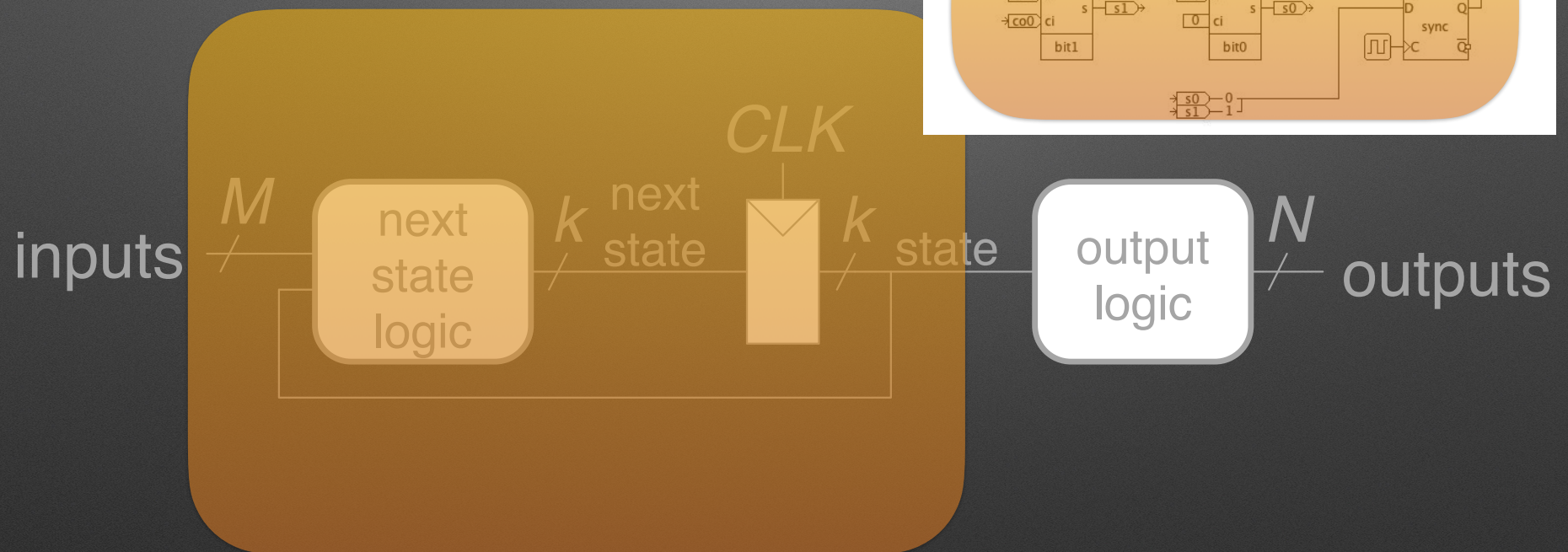
FSM: Moore Machine Structure



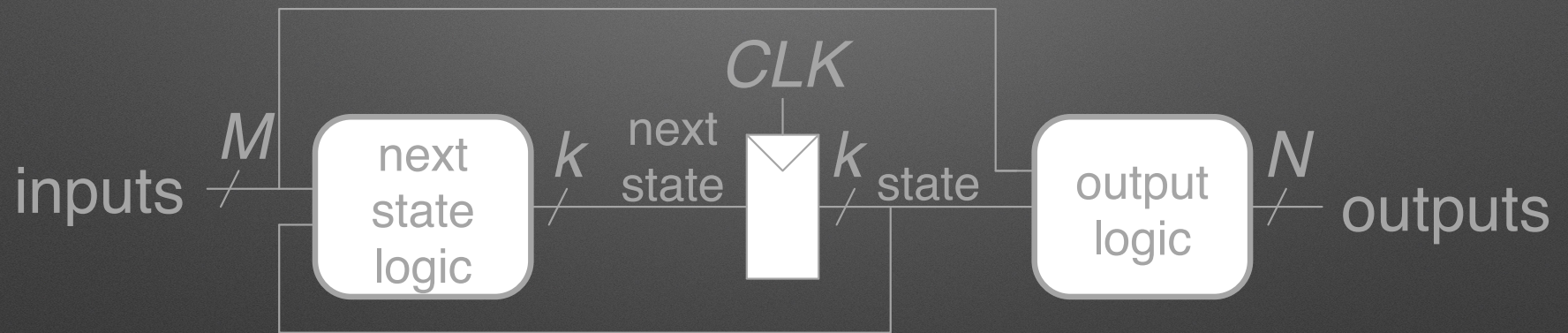
FSM: Moore Machine Structure



FSM: Moore Machine Structure



FSM: Mealy Machine



FSM Applications

FSM Applications

- Things with modes or sequences of steps. Examples:

FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)

FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)
 - Stop lights & Traffic control: Green, Yellow, Red

FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)
 - Stop lights & Traffic control: Green, Yellow, Red
 - Locks: Locked & unlocked

FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)
 - Stop lights & Traffic control: Green, Yellow, Red
 - Locks: Locked & unlocked
 - Computer programs: Playing game vs. on menu

FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)
 - Stop lights & Traffic control: Green, Yellow, Red
 - Locks: Locked & unlocked
 - Computer programs: Playing game vs. on menu
 - Elevator controls (state = floor)

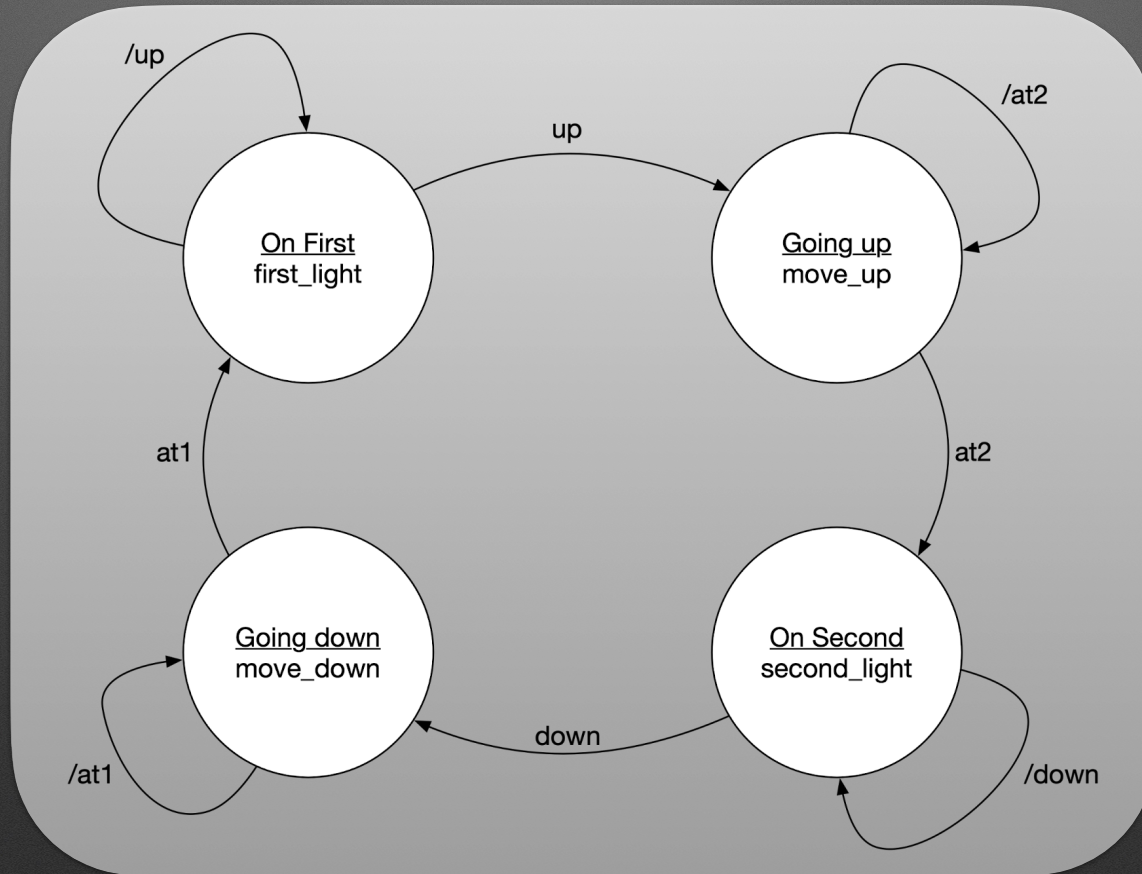
FSM Applications

- Things with modes or sequences of steps. Examples:
 - Washing Machine (fill, agitate, rinse, spin)
 - Stop lights & Traffic control: Green, Yellow, Red
 - Locks: Locked & unlocked
 - Computer programs: Playing game vs. on menu
 - Elevator controls (state = floor)
 - ...

Studio 3A Summary

- 74F175: Memory/storage, bits, and resets
- Elevator Example

Overall



Overall

On first
first_light

Going up
move_up

Going down
move_down

On second
second_light

Overall

On first
first_light

Going up
move_up

Going down
move_down

On second
second_light

Overall

State Encoding

On first
first_light

Going up
move_up

Going down
move_down

On second
second_light

Overall

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

On first
first_light

Going up
move_up

Going down
move_down

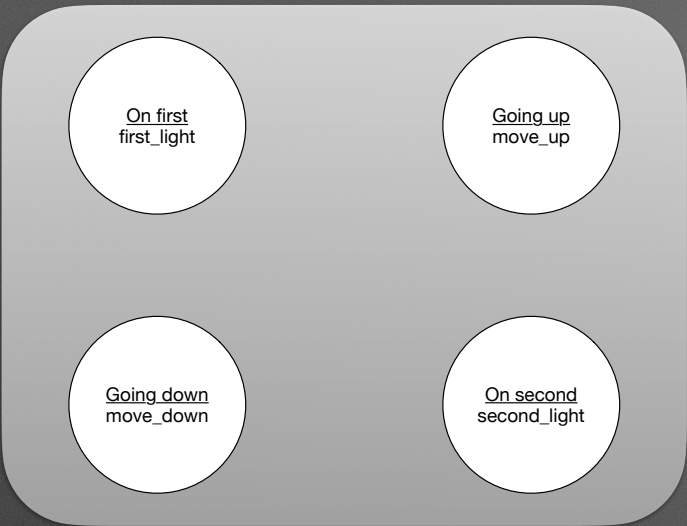
On second
second_light

Overall

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Output Decisions (based on state)



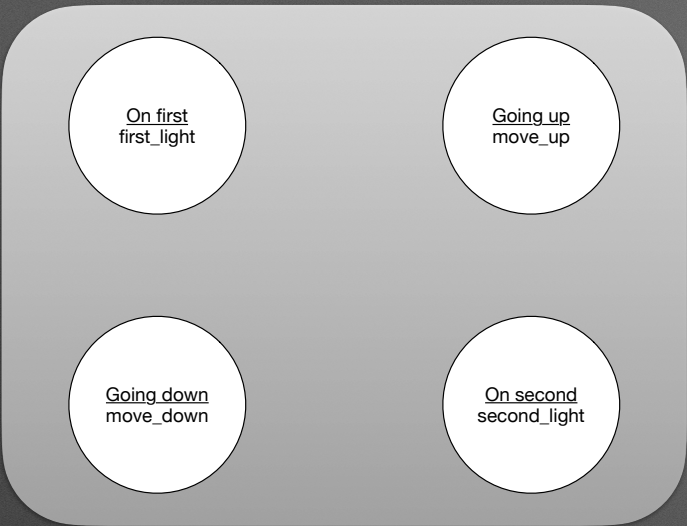
Overall

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Output Decisions (based on state)

State Name	Numeric Encoding	State Bits		Outputs				
		S1	S0	move_up	move_down	first_light	second_light	moving
On first	0	0	0	0	0	1	0	0
Going up	1	0	1	1	0	0	0	1
On second	2	1	0	0	0	0	1	0
Going down	3	1	1	0	1	0	0	1



Overall

Output Equations

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Output Decisions (based on state)

State Name	Numeric Encoding	State Bits		Outputs				
		S1	S0	move_up	move_down	first_light	second_light	moving
On first	0	0	0	0	0	1	0	0
Going up	1	0	1	1	0	0	0	1
On second	2	1	0	0	0	0	1	0
Going down	3	1	1	0	1	0	0	1

On first
first_light

Going up
move_up

Going down
move_down

On second
second_light

Overall

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Output Equations

$\text{move_up} = \text{Going_up} = \neg S1 * S0$
 $\text{move_down} = \text{Going_down} = S1 * S0$
 $\text{first_light} = \text{On_First} = \neg S1 * \neg S0$
 $\text{second_light} = \text{On_Second} = S1 * \neg S0$
 $\text{moving} = \text{Going_up} + \text{Going_down} = \neg S1 * S0 + S1 * S0$

Output Decisions (based on state)

State Name	Numeric Encoding	State Bits		Outputs				
		S1	S0	move_up	move_down	first_light	second_light	moving
On first	0	0	0	0	0	1	0	0
Going up	1	0	1	1	0	0	0	1
On second	2	1	0	0	0	0	1	0
Going down	3	1	1	0	1	0	0	1

On first
first_light

Going up
move_up

Going down
move_down

On second
second_light

Overall State Encoding

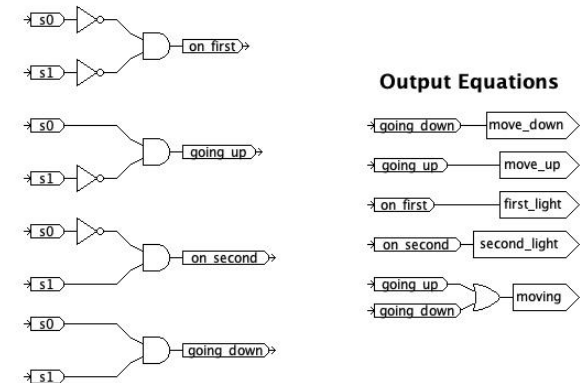
State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Output Decisions (based on state)

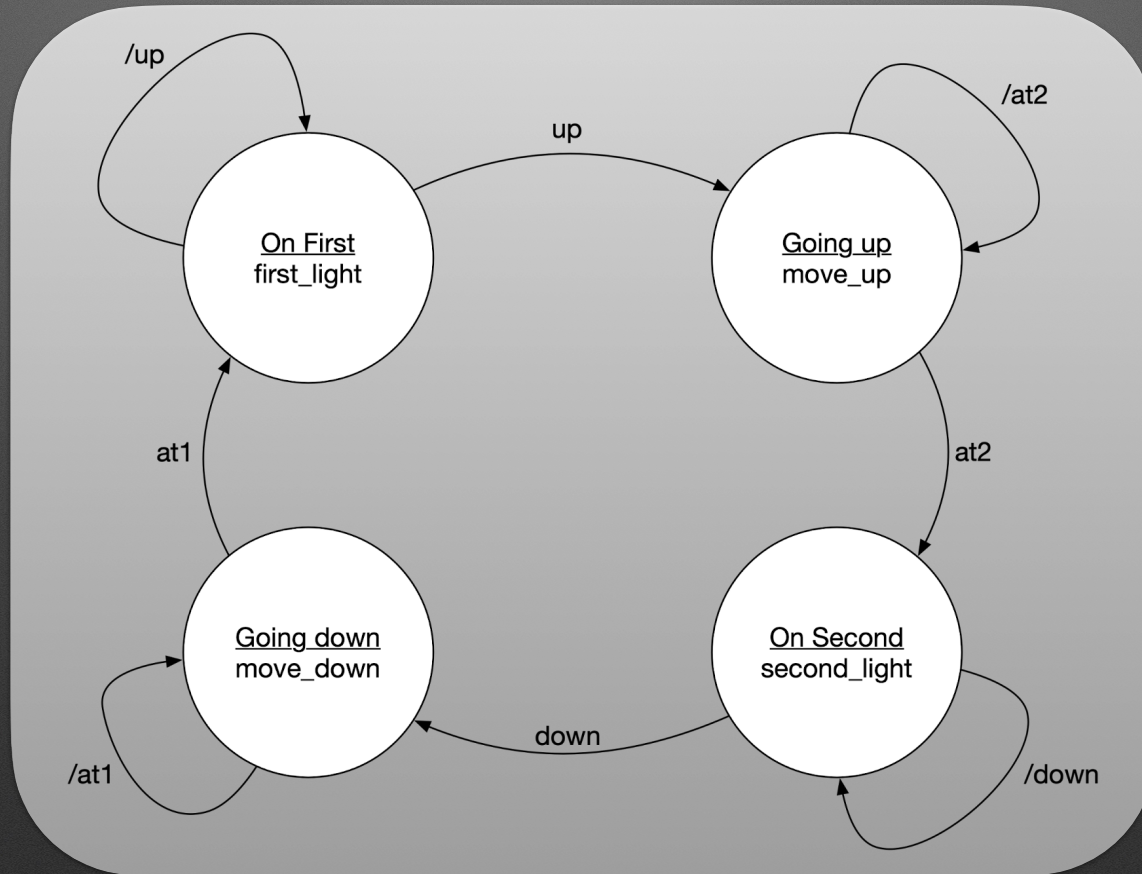
State Name	Numeric Encoding	State Bits		Outputs				
		S1	S0	move_up	move_down	first_light	second_light	moving
On first	0	0	0	0	0	1	0	0
Going up	1	0	1	1	0	0	0	1
On second	2	1	0	0	0	0	1	0
Going down	3	1	1	0	1	0	0	1

Output Equations

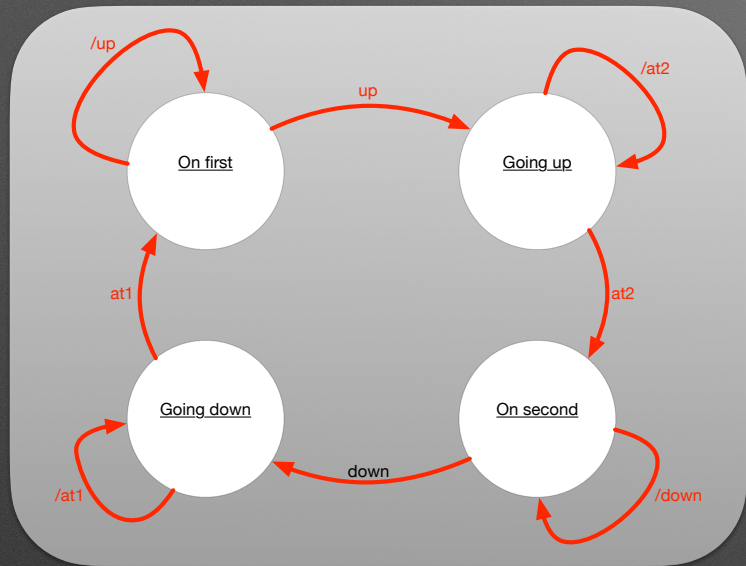
$\text{move_up} = \text{Going_up} = /S1 * S0$
 $\text{move_down} = \text{Going_down} = S1 * S0$
 $\text{first_light} = \text{On_First} = /S1 * /S0$
 $\text{second_light} = \text{On_Second} = S1 * /S0$
 $\text{moving} = \text{Going_up} + \text{Going_down} = /S1 * S0 + S1 * S0$

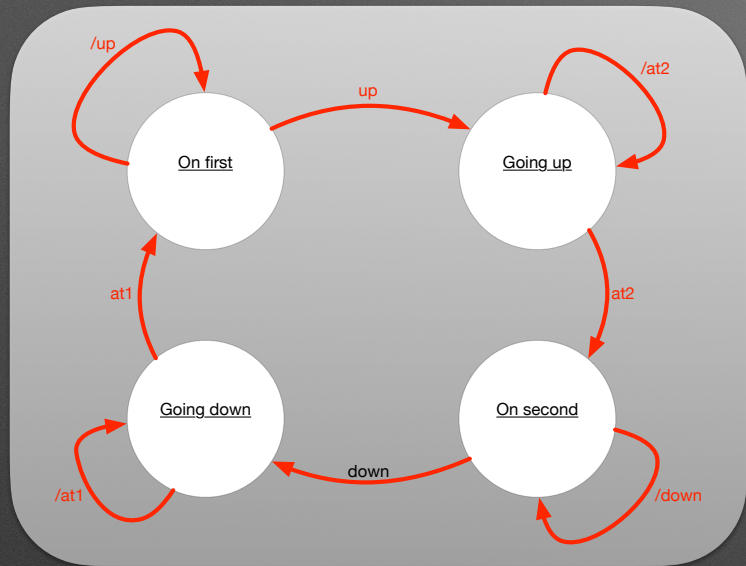


Overall



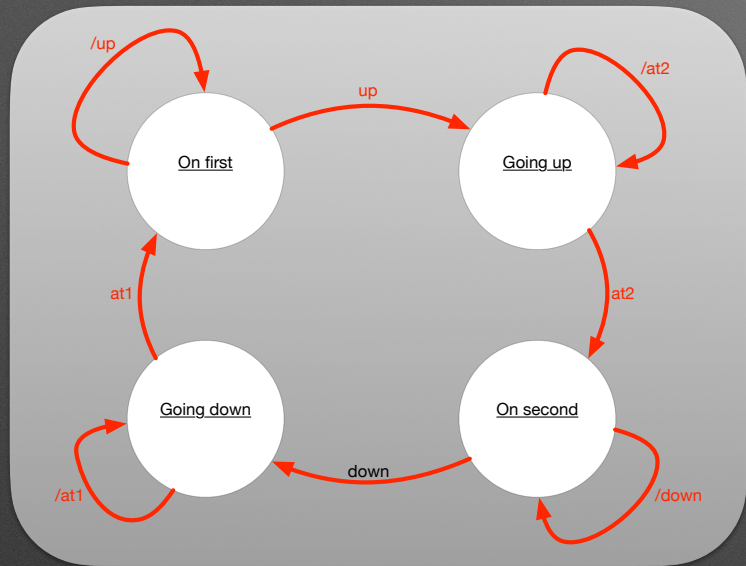
Overall





Overall

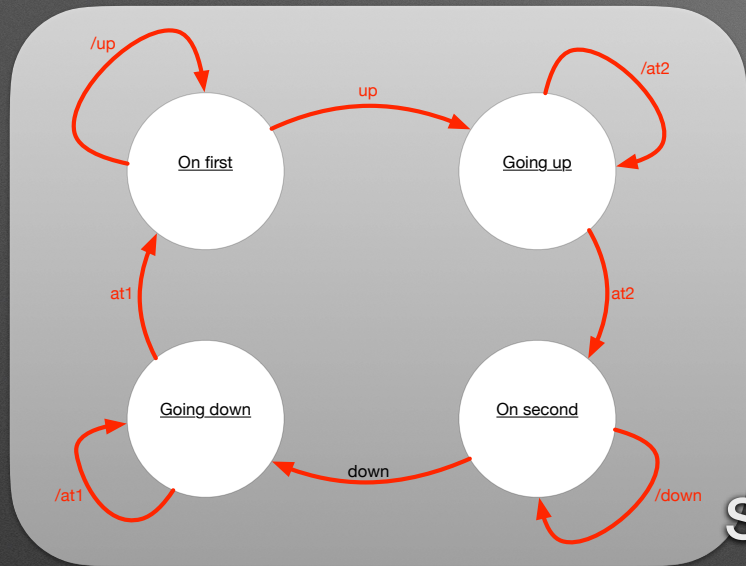
State Encoding



Overall

State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1



Overall

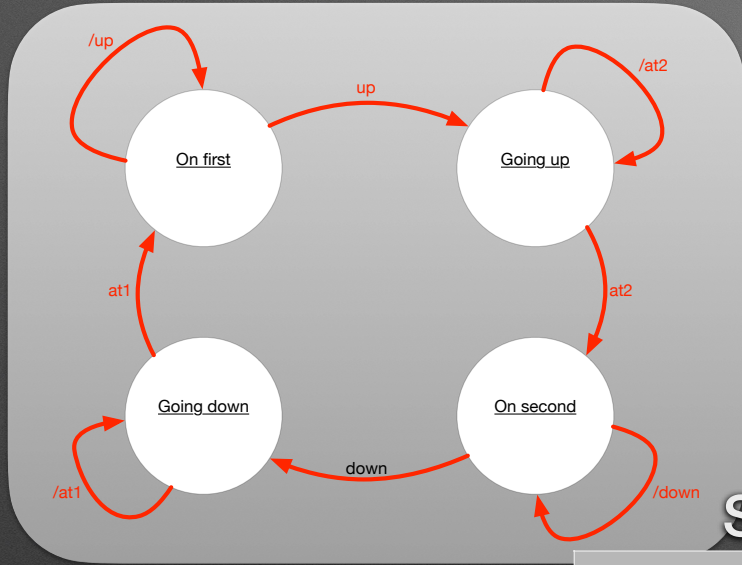
State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

State Transitions (to “next state”)

Overall

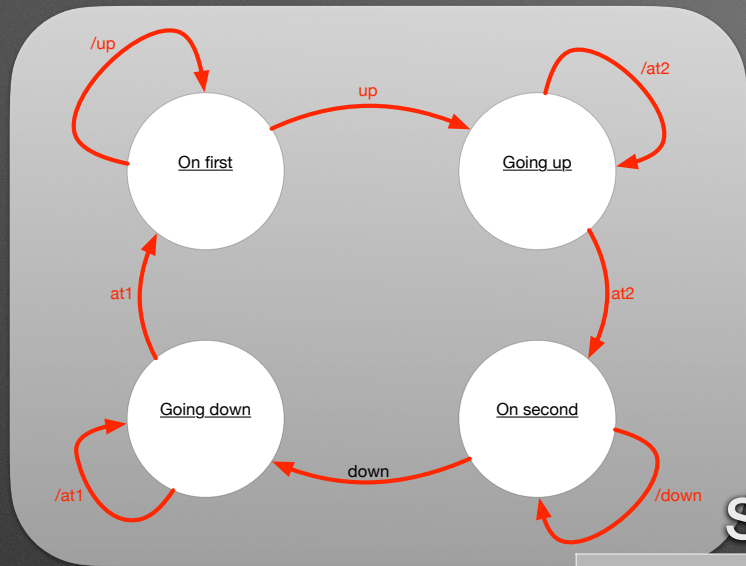
State Encoding



State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

State Transitions (to “next state”)

State Name	Numeric Encoding	Inputs				Next State		
		up	down	at1	at2	Name	ns1	ns0
On first	0	0	X	X	X	On_first	0	0
		1	X	X	X	Going_up	0	1
Going up	1	X	X	X	0	Going_up	0	1
		X	X	X	1	On_second	1	0
On second	2	X	0	X	X	On_second	1	0
		X	1	X	X	Going_down	1	1
Going down	3	X	X	0	X	Going_down	1	1
		X	X	1	X	On_first	0	0



Overall

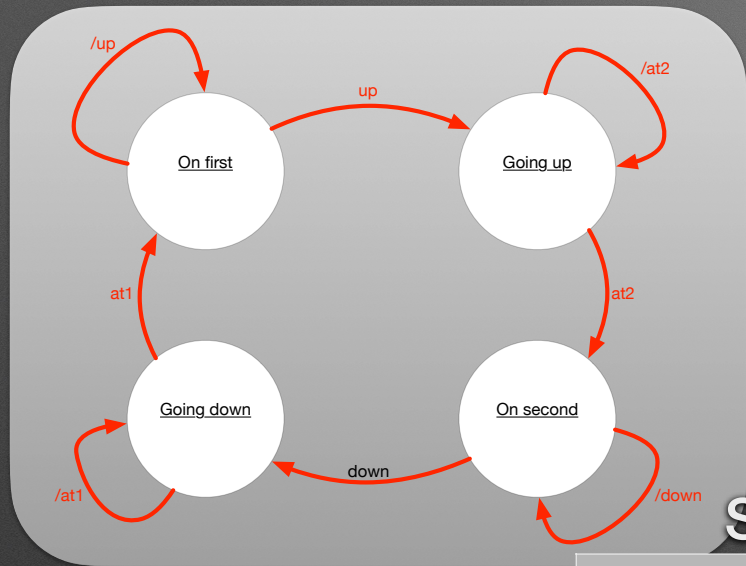
State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Next State Equations

State Transitions (to “next state”)

State Name	Numeric Encoding	Inputs				Next State		
		up	down	at1	at2	Name	ns1	ns0
On first	0	0	X	X	X	On_first	0	0
		1	X	X	X	Going_up	0	1
Going up	1	X	X	X	0	Going_up	0	1
		X	X	X	1	On_second	1	0
On second	2	X	0	X	X	On_second	1	0
		X	1	X	X	Going_down	1	1
Going down	3	X	X	0	X	Going_down	1	1
		X	X	1	X	On_first	0	0



Overall State Encoding

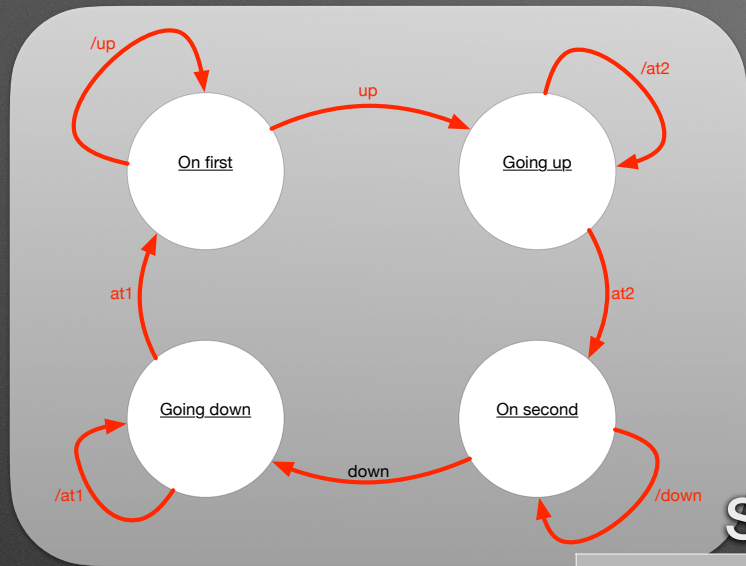
State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

Next State Equations

$$\begin{aligned}
 ns1 &= On_Second * at2 + \\
 &\quad On_second * /down + \\
 &\quad Going_down * down + \\
 &\quad Going_down * /at1 \\
 ns0 &= Going_up * up + \\
 &\quad Going_up * /at2 + \\
 &\quad On_second * down + \\
 &\quad Going_down * /at1
 \end{aligned}$$

State Transitions (to “next state”)

State Name	Numeric Encoding	Inputs				Next State		
		up	down	at1	at2	Name	ns1	ns0
On first	0	0	X	X	X	On_first	0	0
		1	X	X	X	Going_up	0	1
Going up	1	X	X	X	0	Going_up	0	1
		X	X	X	1	On_second	1	0
On second	2	X	0	X	X	On_second	1	0
		X	1	X	X	Going_down	1	1
Going down	3	X	X	0	X	Going_down	1	1
		X	X	1	X	On_first	0	0



Overall State Encoding

State Name	Numeric Encoding	State Bits	
		S1	S0
On first	0	0	0
Going up	1	0	1
On second	2	1	0
Going	3	1	1

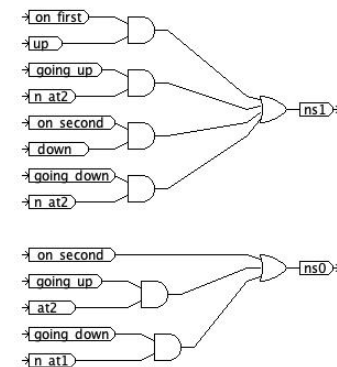
Next State Equations

$$\begin{aligned}
 ns1 &= \text{On_Second} * at2 + \\
 &\quad \text{On_second} * /down + \\
 &\quad \text{Going_down} * down + \\
 &\quad \text{Going_down} * /at1 \\
 ns0 &= \text{Going_up} * up + \\
 &\quad \text{Going_up} * /at2 + \\
 &\quad \text{On_second} * down + \\
 &\quad \text{Going_down} * /at1
 \end{aligned}$$

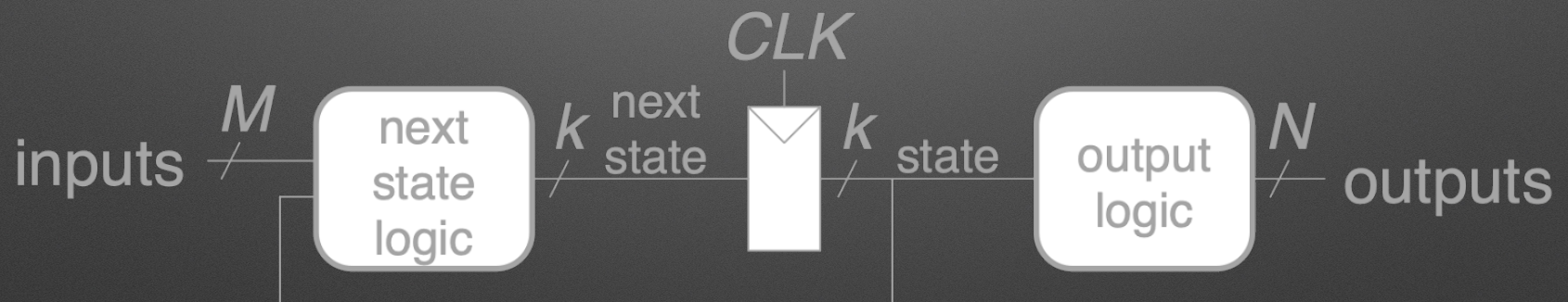
State Transitions (to “next state”)

State Name	Numeric Encoding	Inputs				Next State	
		up	down	at1	at2	Name	ns1
On first	0	0	X	X	X	On_first	0
		1	X	X	X	Going_up	0
Going up	1	X	X	X	0	Going_up	0
		X	X	X	1	On_second	1
On second	2	X	0	X	X	On_second	1
		X	1	X	X	Going_down	1
Going down	3	X	X	0	X	Going_down	1
		X	X	1	X	On_first	0

Next State Computation

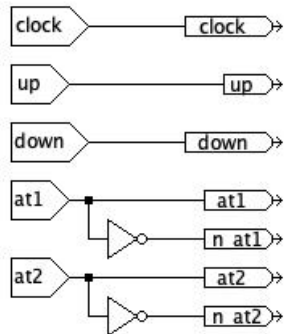


Overall Structure

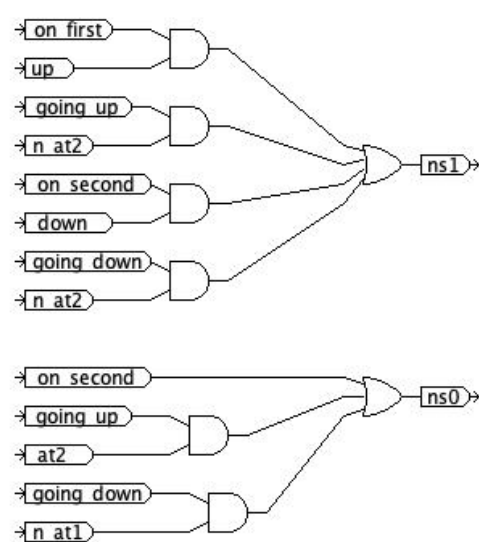


Overall Implementation

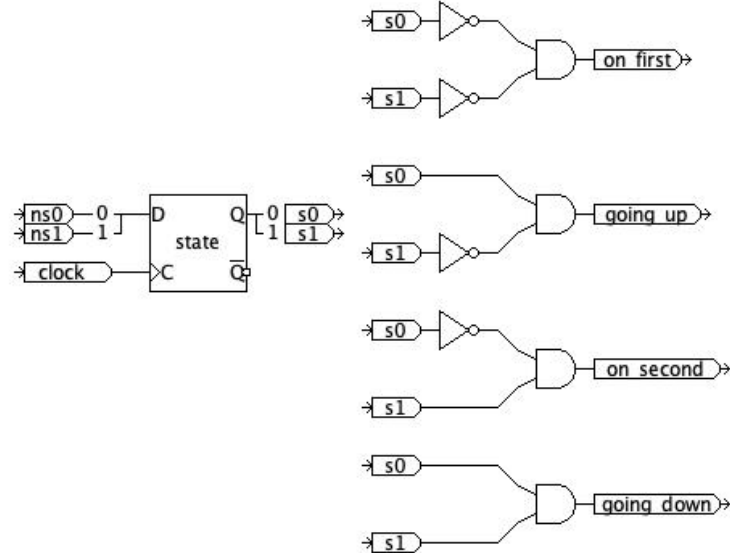
Inputs (to named wires)



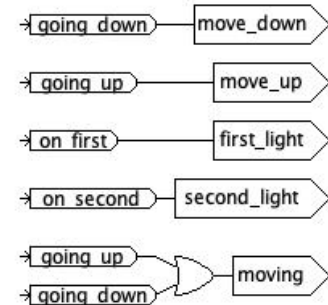
Next State Computation



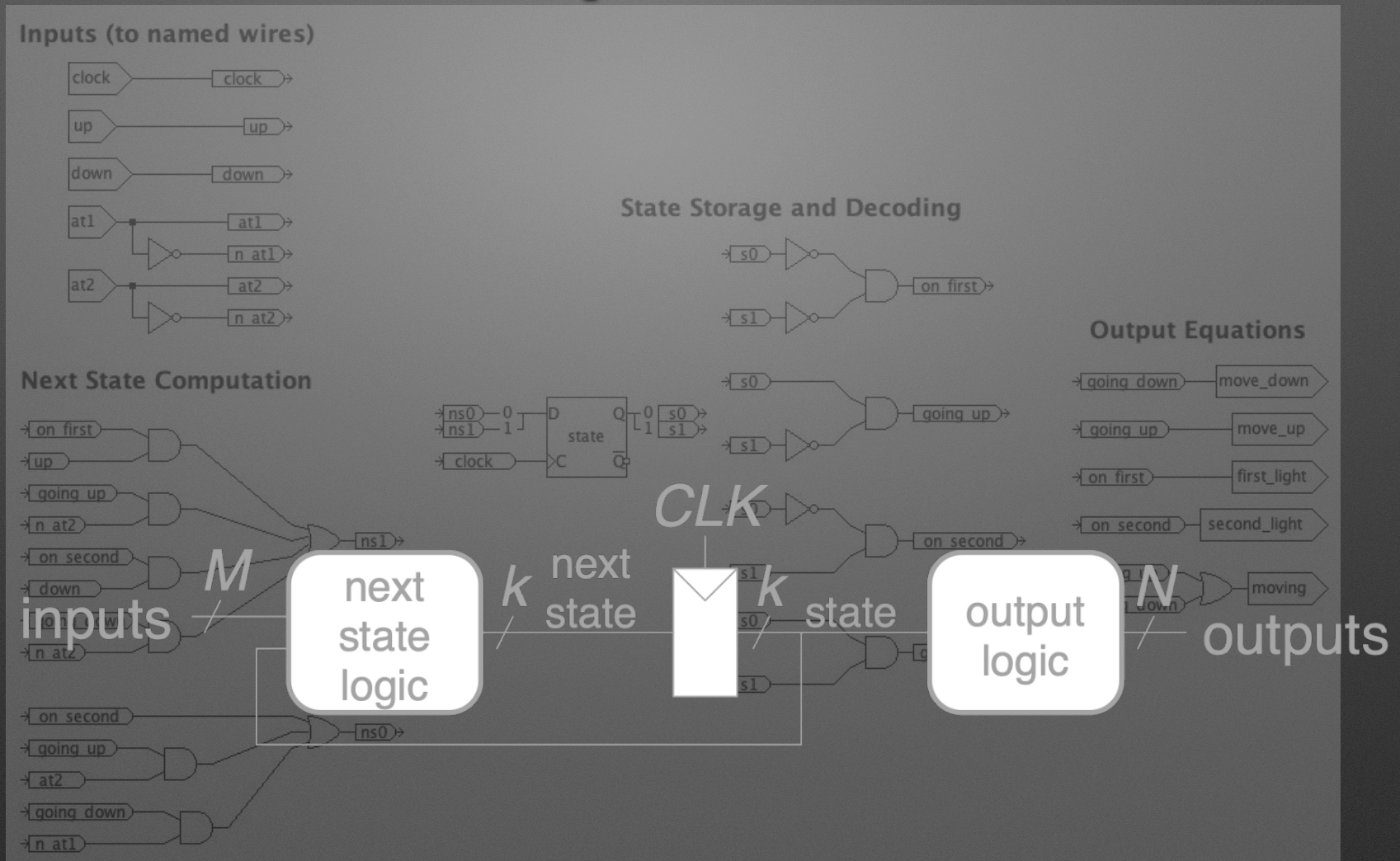
State Storage and Decoding



Output Equations

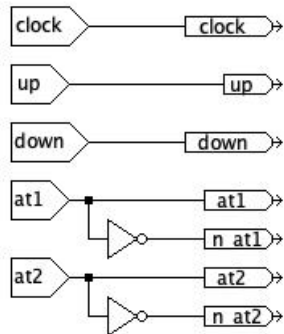


Overall Implementation

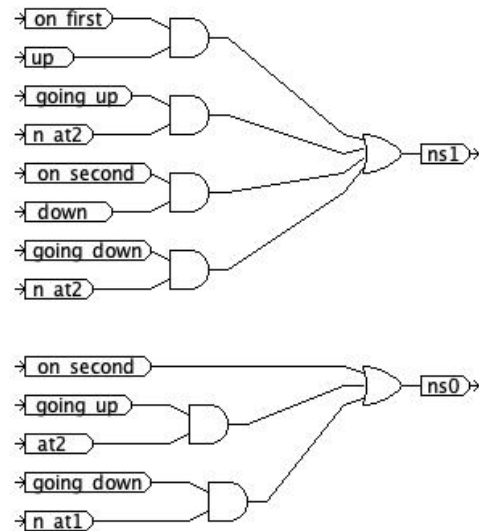


Overall Implementation

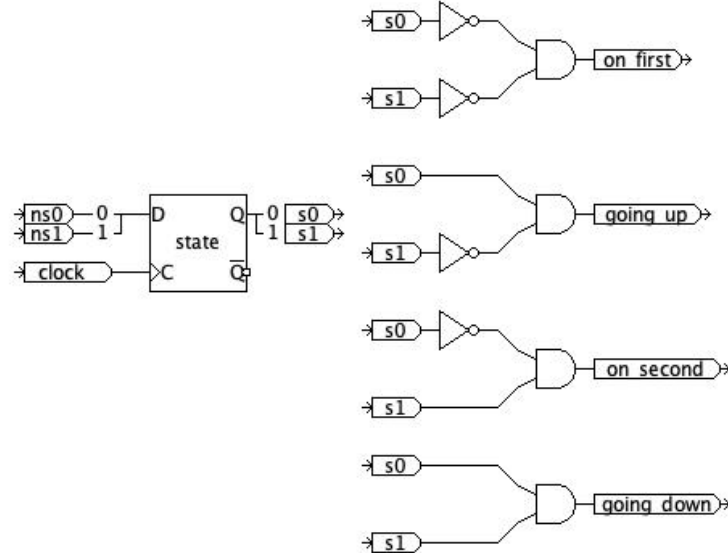
Inputs (to named wires)



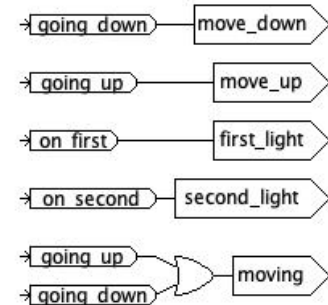
Next State Computation



State Storage and Decoding

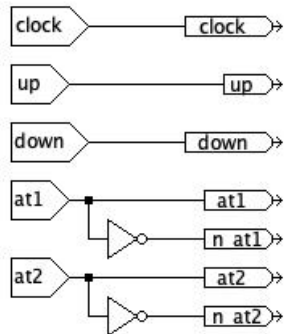


Output Equations

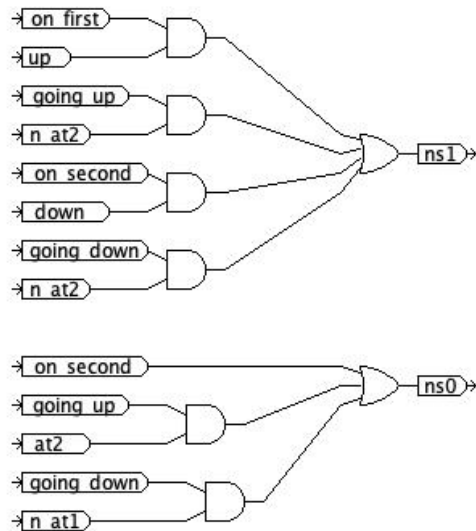


Overall Implementation

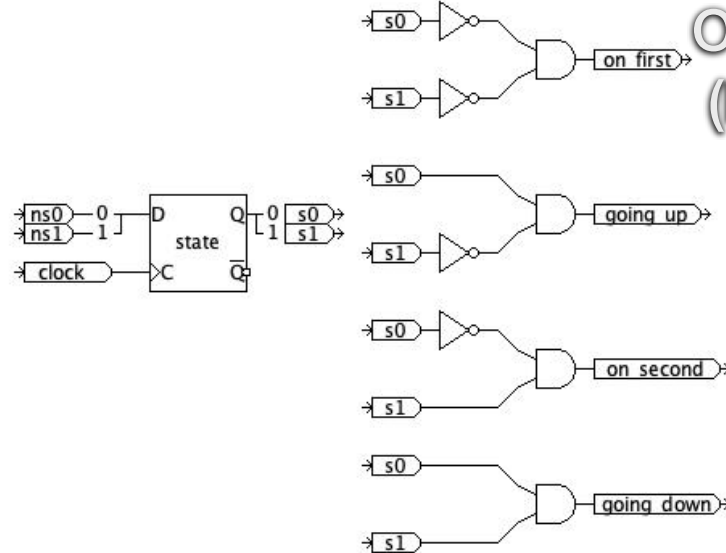
Inputs (to named wires)



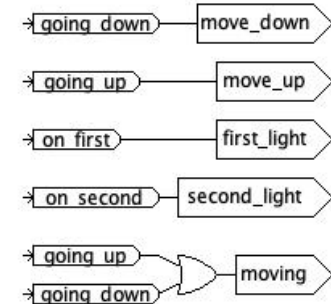
Next State Computation



State Storage and Decoding

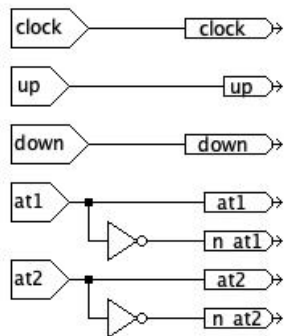


Output Decisions (based on state)

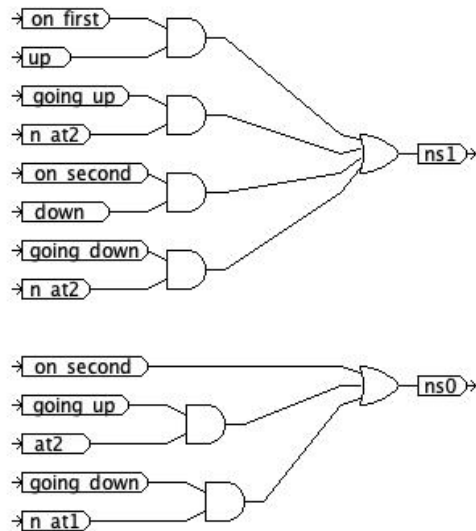


Overall Implementation

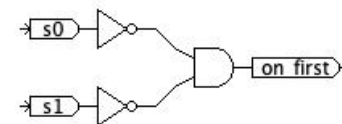
Inputs (to named wires)



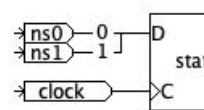
Next State Computation



State Storage and Decoding



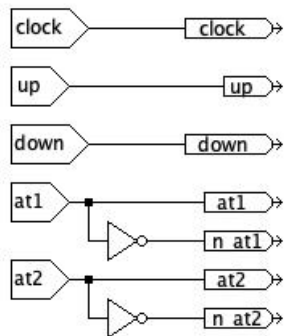
Output Decisions
(based on state)



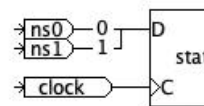
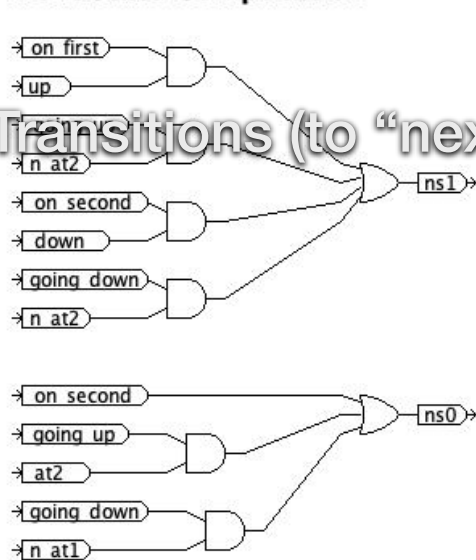
State Name	Numeric Encoding	State Bits		Outputs			
		S1	S0	move_up	move_down	first_light	second_light
On first	0	0	0	on first	0	1	0
Going up	1	0	1	on second	0	0	0
On second	2	1	0	going up	0	0	0
Going down	3	1	1	going down	1	0	0

Overall Implementation

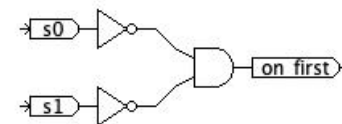
Inputs (to named wires)



Next State Computation



State Storage and Decoding



Output Decisions
(based on state)

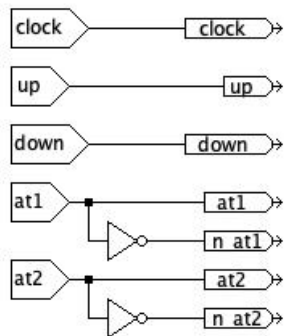
State Name	Numeric Encoding	State Bits		Outputs			
		S1	S0	move_up	move_down	first_light	second_light
On first	0	0	0	on first	0	1	0
Going up	1	0	1	on second	0	0	0
On second	2	1	0	going up	0	0	0
Going down	3	1	1	going down	1	0	0



State Transitions (to "next state")

Overall Implementation

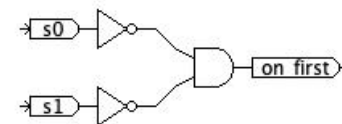
Inputs (to named wires)



Next State Computation



State Storage and Decoding



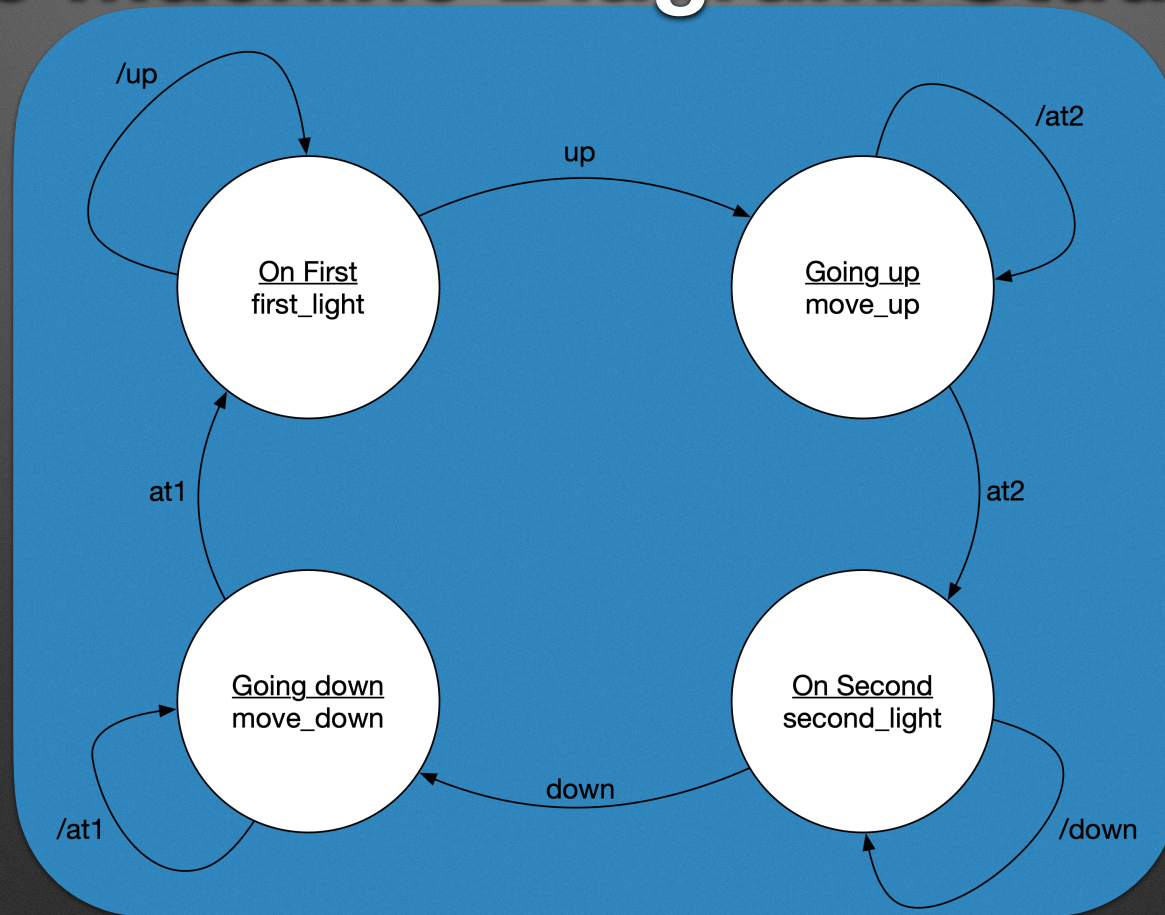
Output Decisions
(based on state)

State Transitions (to "next state")

State Name	Numeric Encoding	Inputs				Next State		
		up	down	at1	at2	Name	ns1	ns0
On first	0	0	X	X	X	On_first	0	0
		1	X	X	X	Going_up	0	1
Going up	1	X	X	X	0	Going_up	0	1
		X	0	X	1	On_second	1	0
On second	2	X	0	X	X	On_second	1	0
		X	1	X	X	Going_down	1	1
		X	0	0	X	Going_down	1	1
Going down	3	X	0	1	X	On_first	0	0

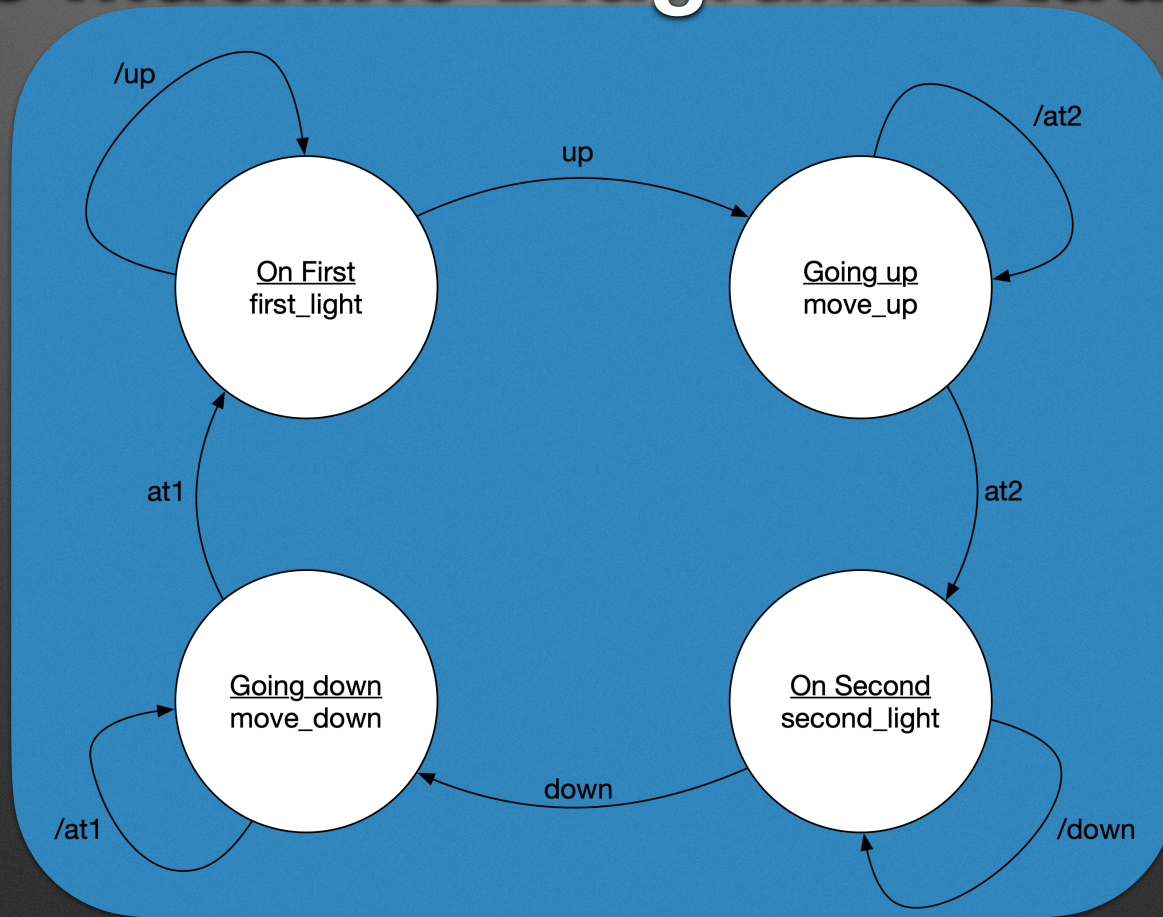
State Name	Numeric Encoding	State Bits				Outputs	
		S1	S0	move_up	move_down	first_light	second_light
On first	0	0	0	on first	0	1	0
Going up	1	0	1	on second	0	0	0
On second	2	1	0	going up	0	0	0
Going down	3	1	1	going down	1	0	0

State Machine Diagram: Studio 3A



State Machine Diagram: Studio

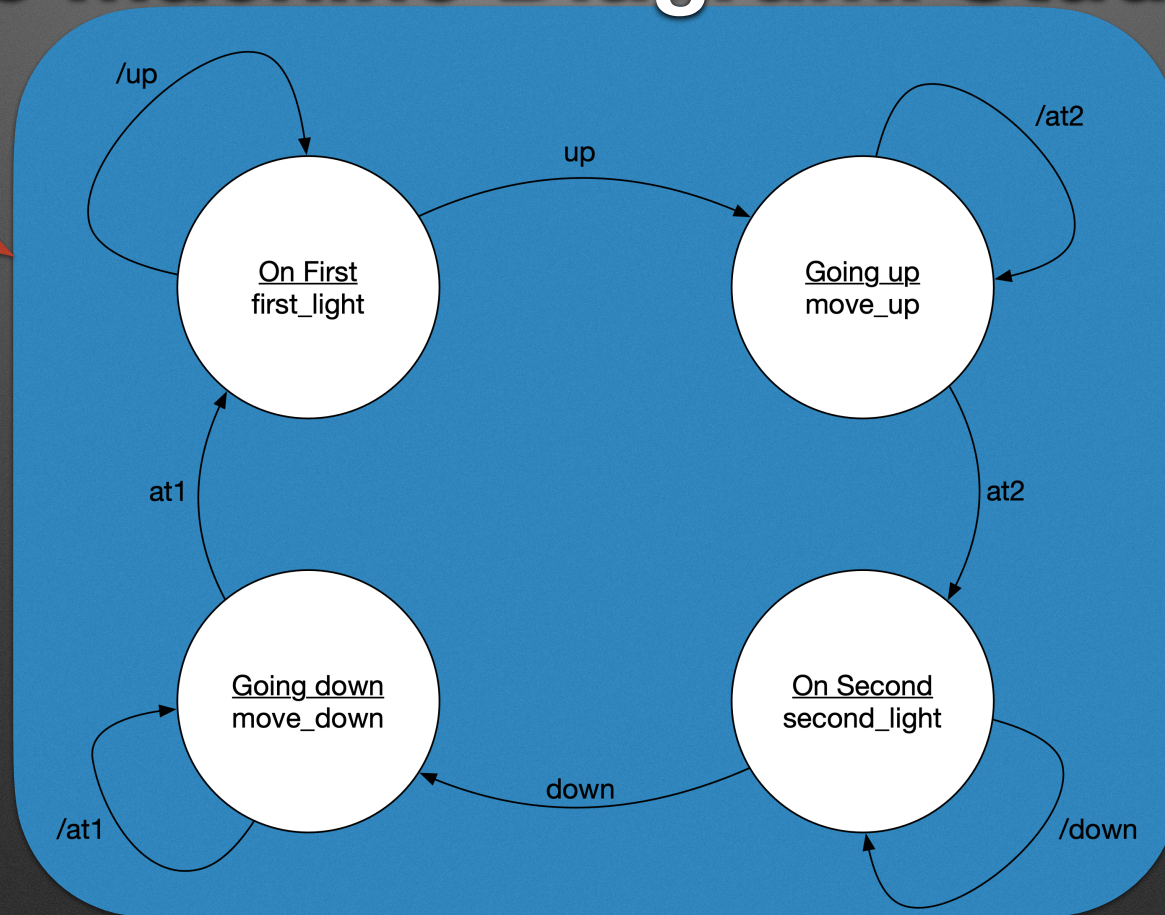
What does
“clock” do?



Assume
clock cycle of
10s?

State Machine Diagram: Studio

What does
“clock” do?

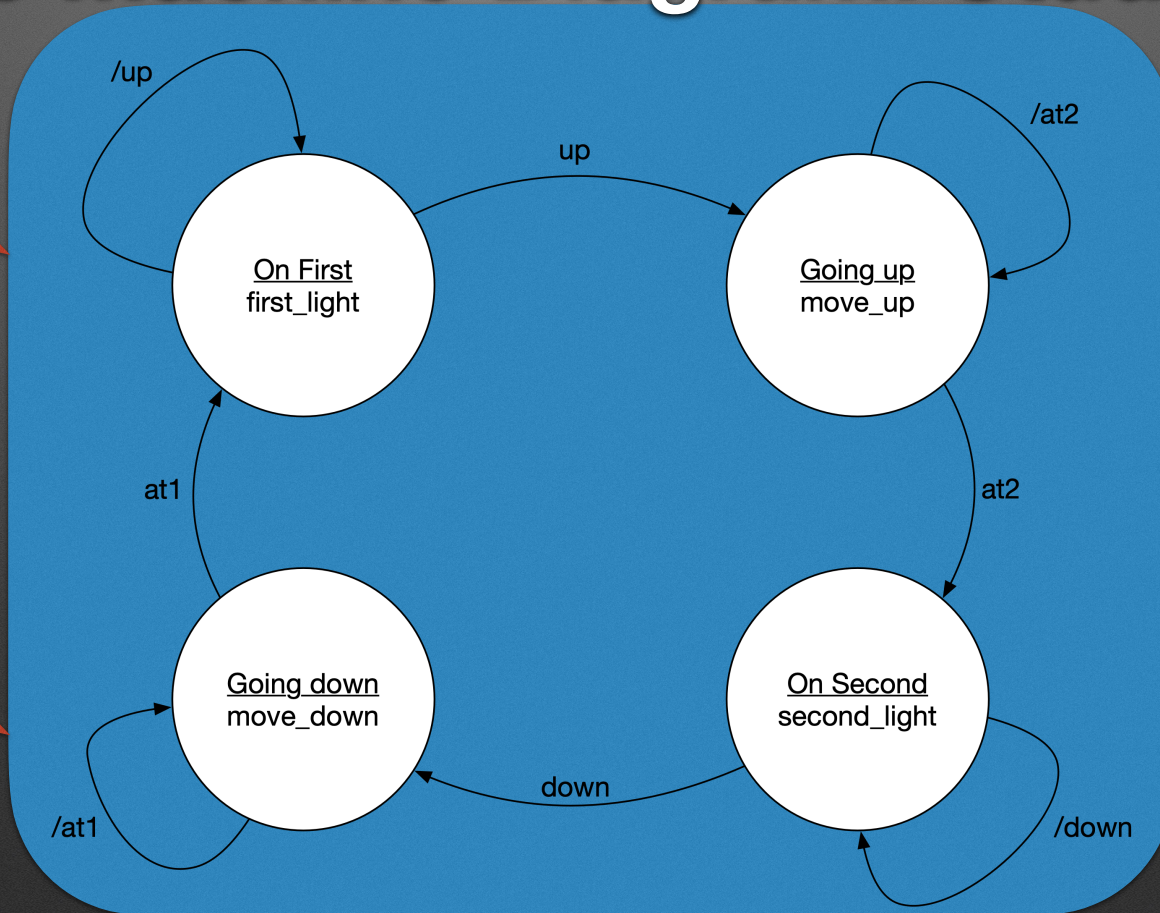


Assume
clock cycle of
10s?

Vs 240s?

State Machine Diagram: Studio

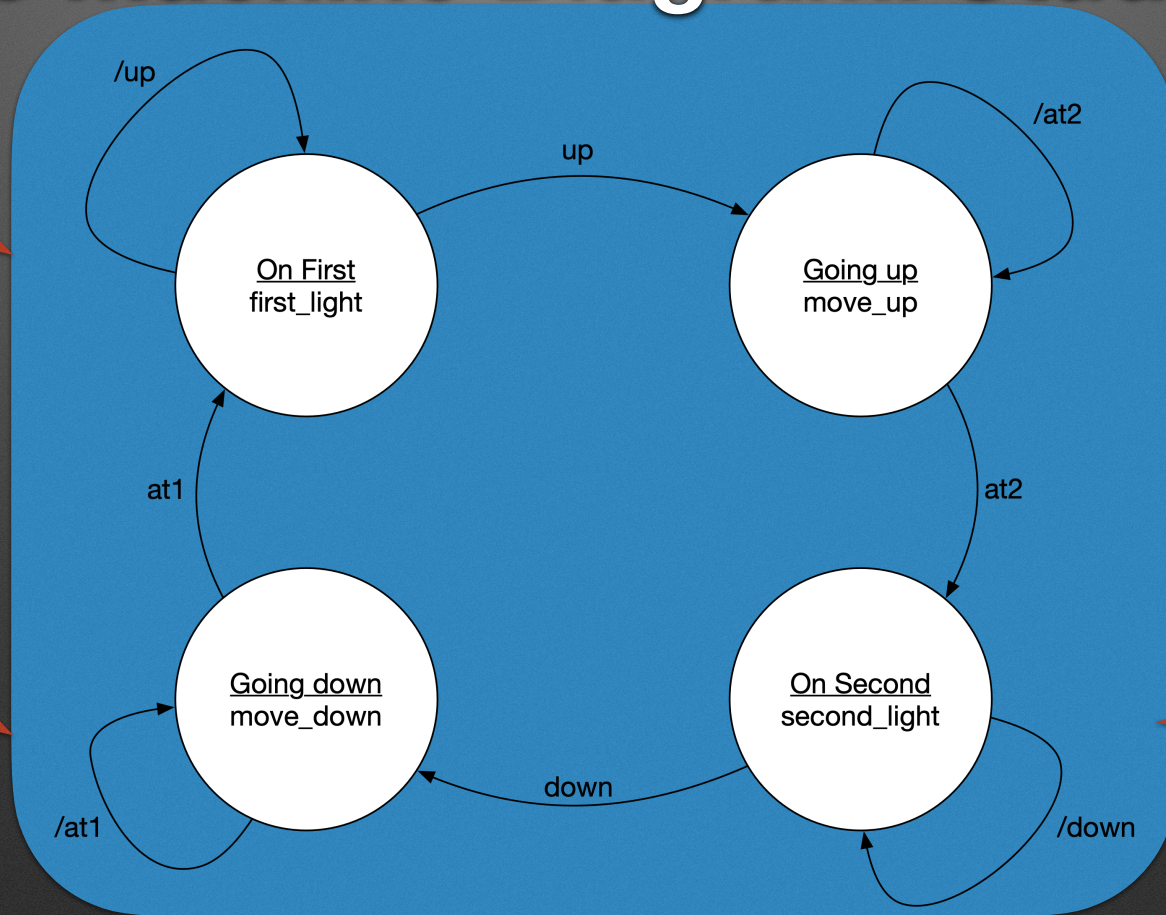
What does
“clock” do?



Assume
clock cycle of
10s?

State Machine Diagram: Studio

What does
“clock” do?



Vs 240s?

Can the
clock be too
short?

State Encodings

State Encodings

- “Art”

State Encodings

- “Art”
 - Encoding has impact on equations used

State Encodings

- “Art”
 - Encoding has impact on equations used
- Major flavors

State Encodings

- “Art”
 - Encoding has impact on equations used
- Major flavors
 - Counting

State Encodings

- “Art”
 - Encoding has impact on equations used
- Major flavors
 - Counting
 - One Hot

Elevator: Examples

STATE NAME	BINARY COUNTING	ONE HOT
ON FIRST		
GOING UP		
ON SECOND		
GOING DOWN		

Elevator: Examples Part 2

STATE NAME	BINARY COUNTING	ONE HOT
ON FIRST		
GOING UP		
ON SECOND		
GOING DOWN		

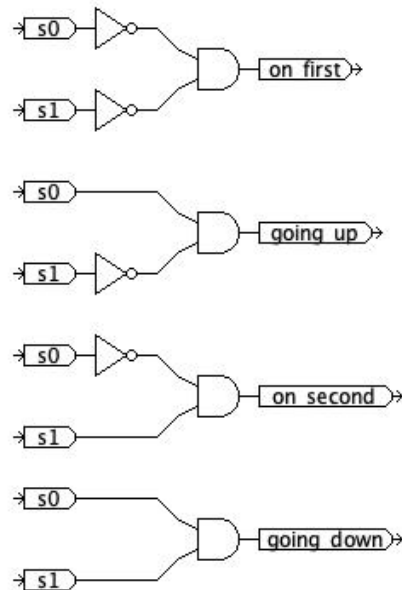
Decoder

- Decoders...decode
 - n bits of input to 2^n distinct, mutually exclusive outputs
 - It's just ANDs/NOTs to select each distinct binary value

Decoder

- Decoders...decode
 - n bits of input to 2^n distinct, mutually exclusive outputs
 - It's just ANDs/NOTs to select each distinct binary value

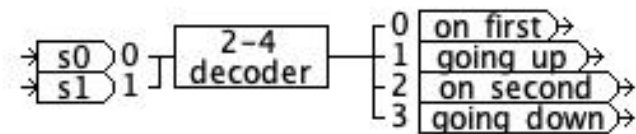
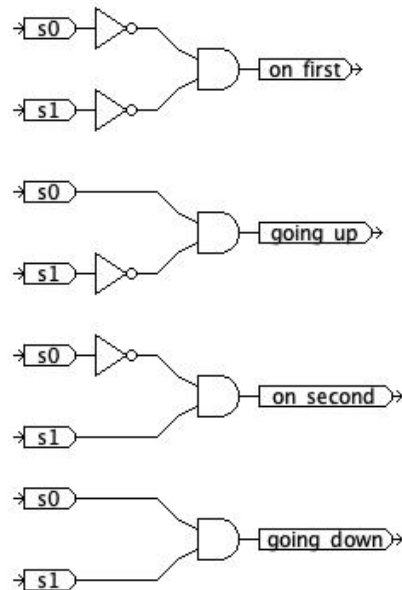
Storage and Decoding



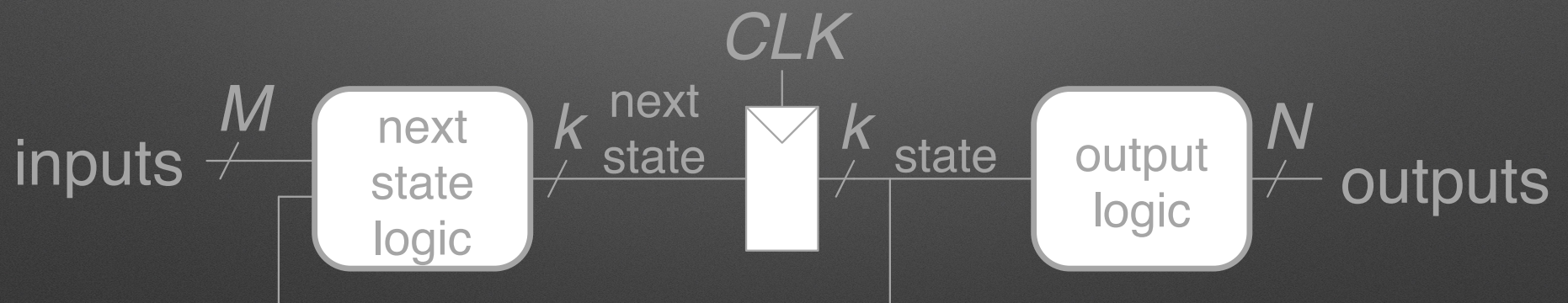
Decoder

- Decoders...decode
 - n bits of input to 2^n distinct, mutually exclusive outputs
 - It's just ANDs/NOTs to select each distinct binary value

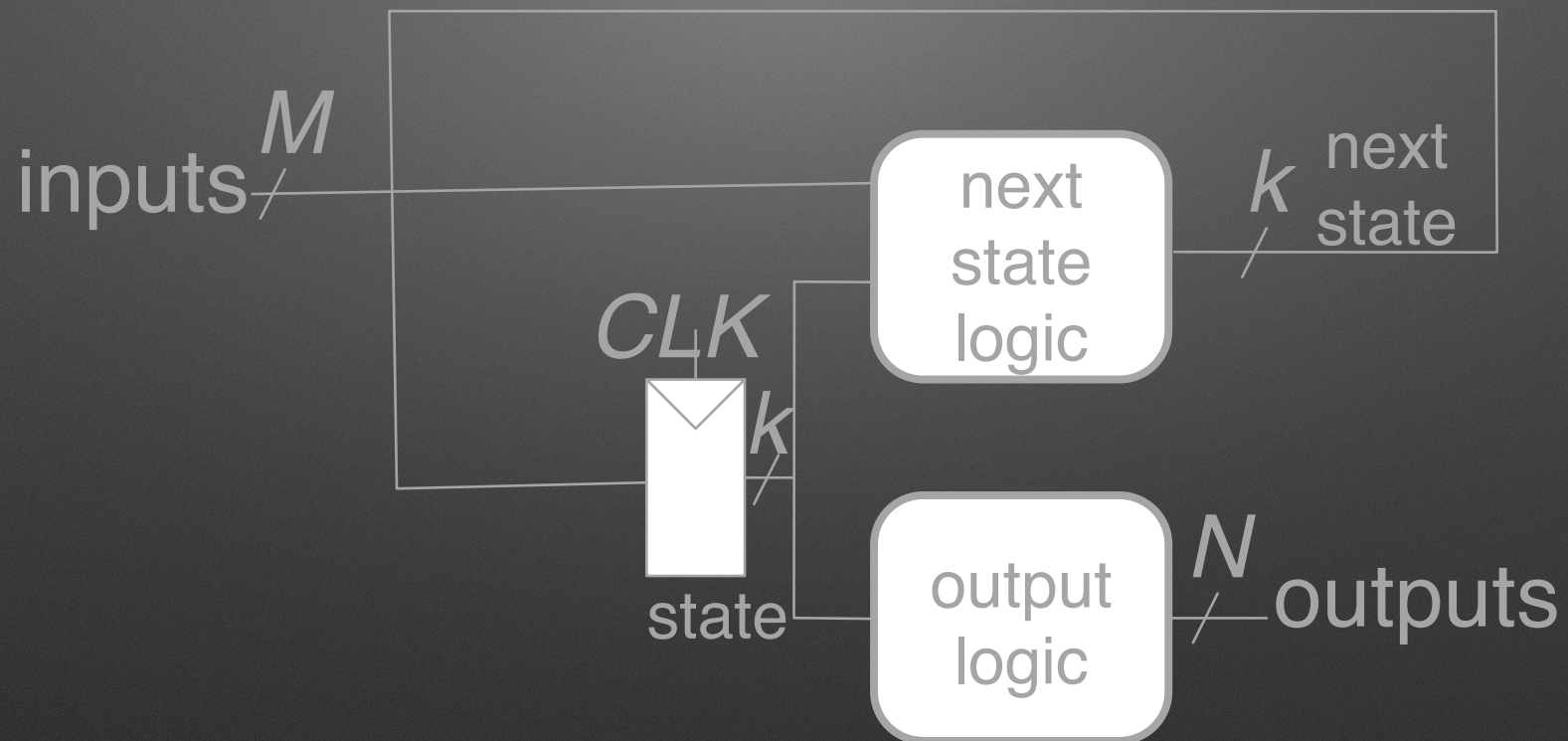
Storage and Decoding



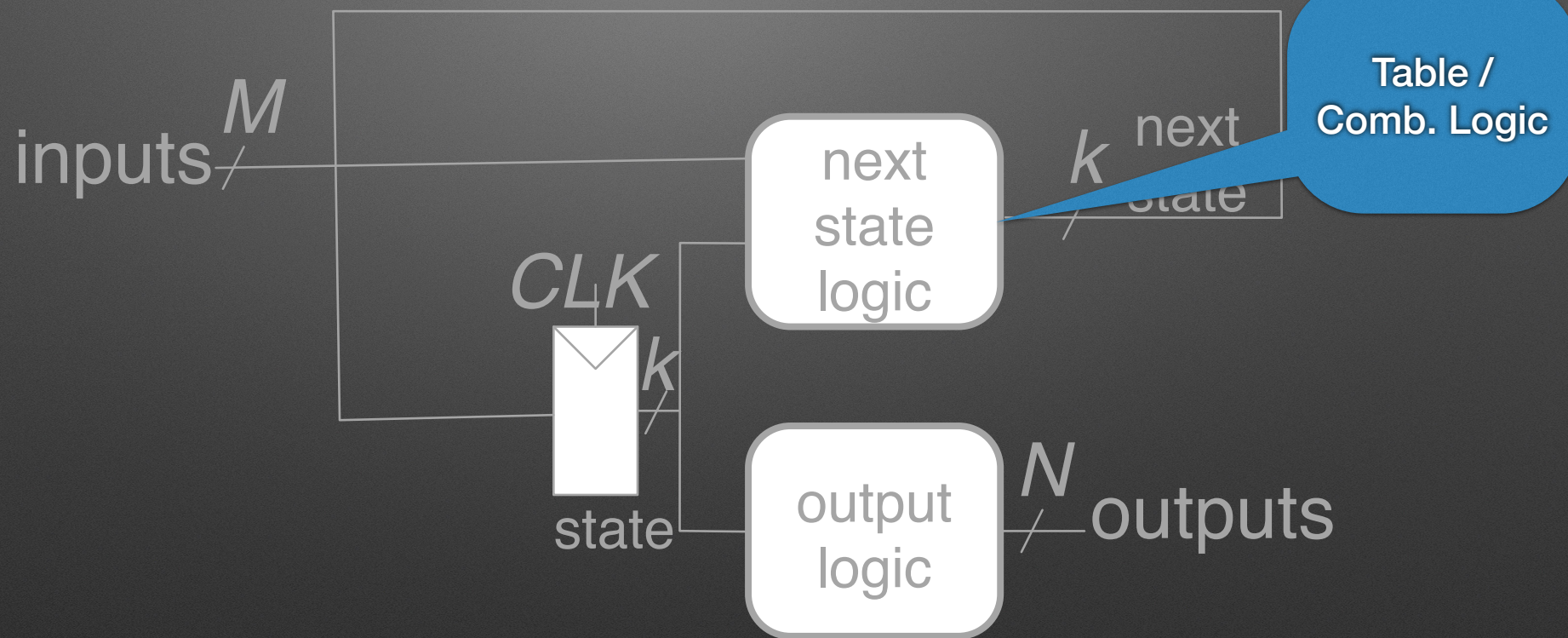
Review: Moore Machine Structure



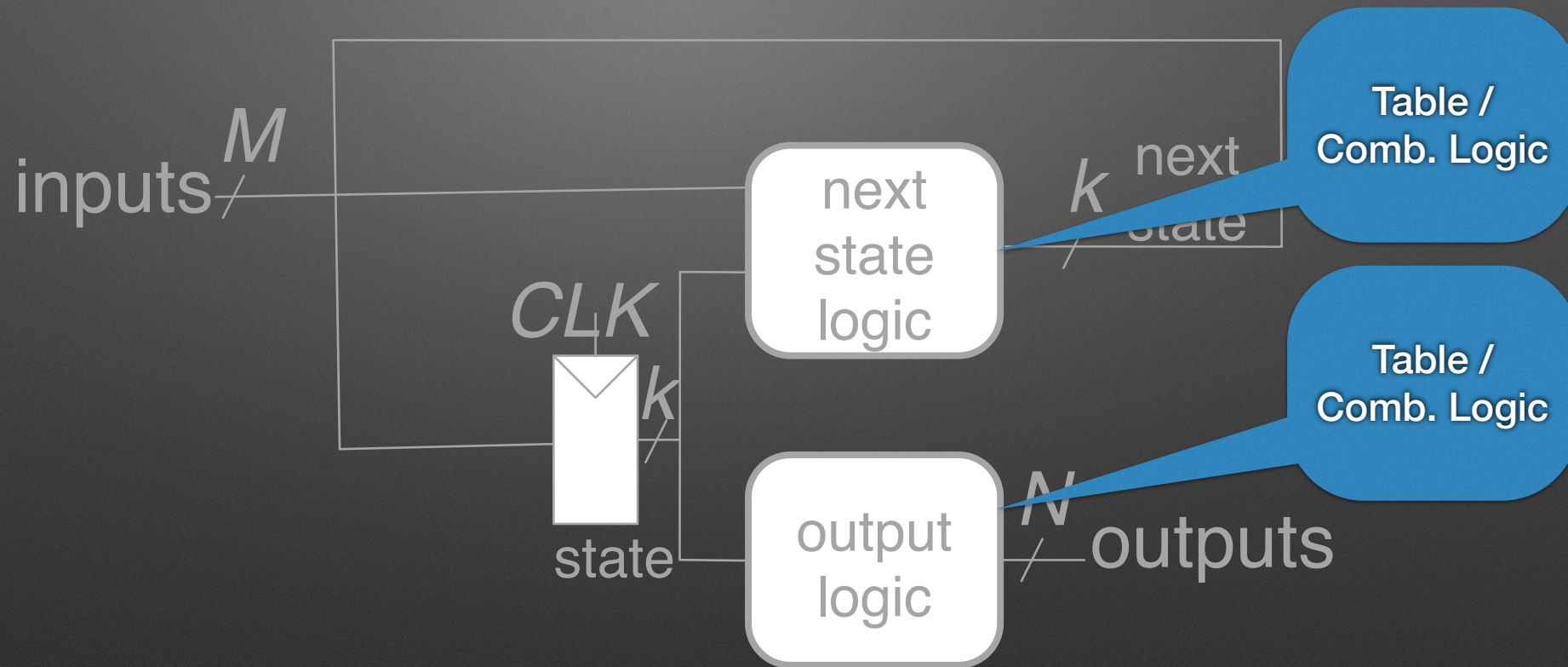
Review: Moore Machine Structure



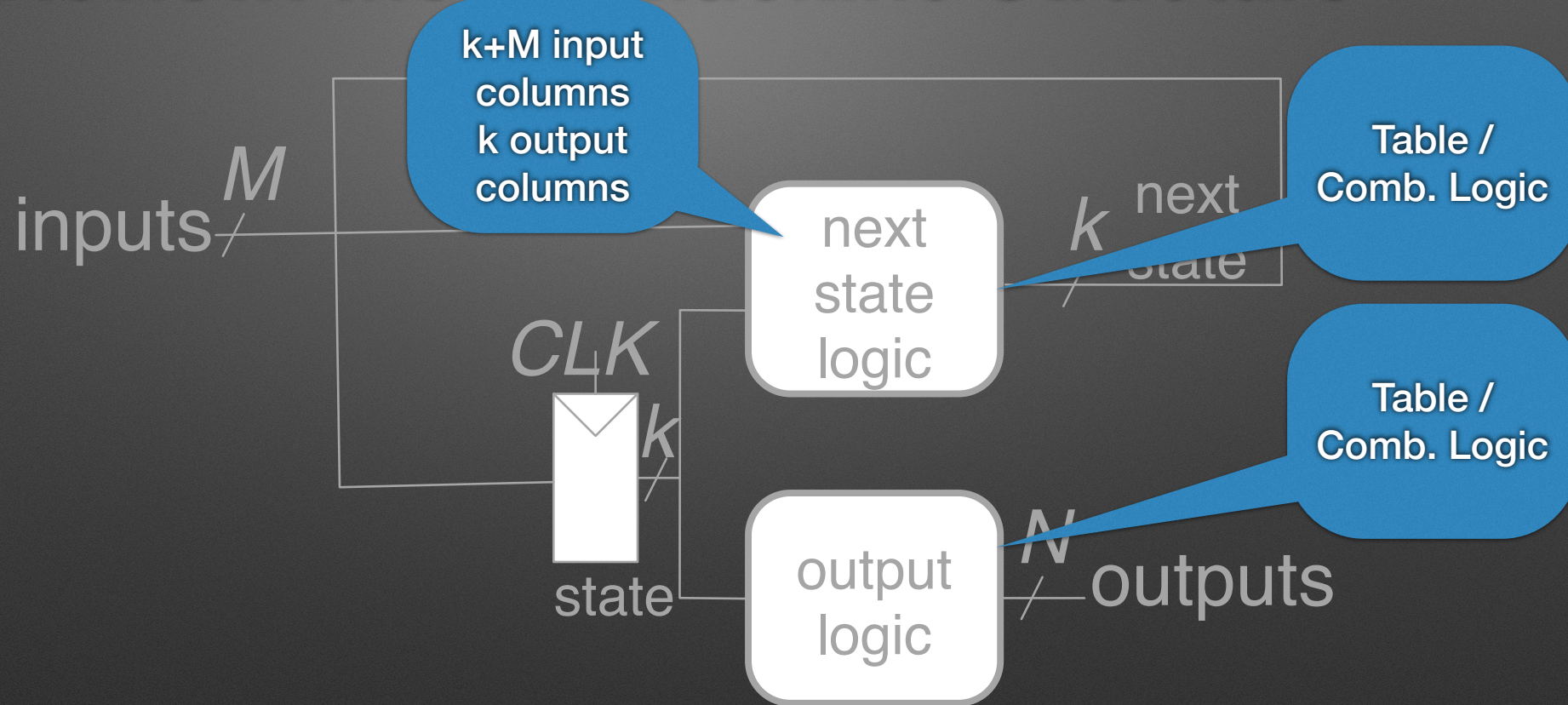
Review: Moore Machine Structure



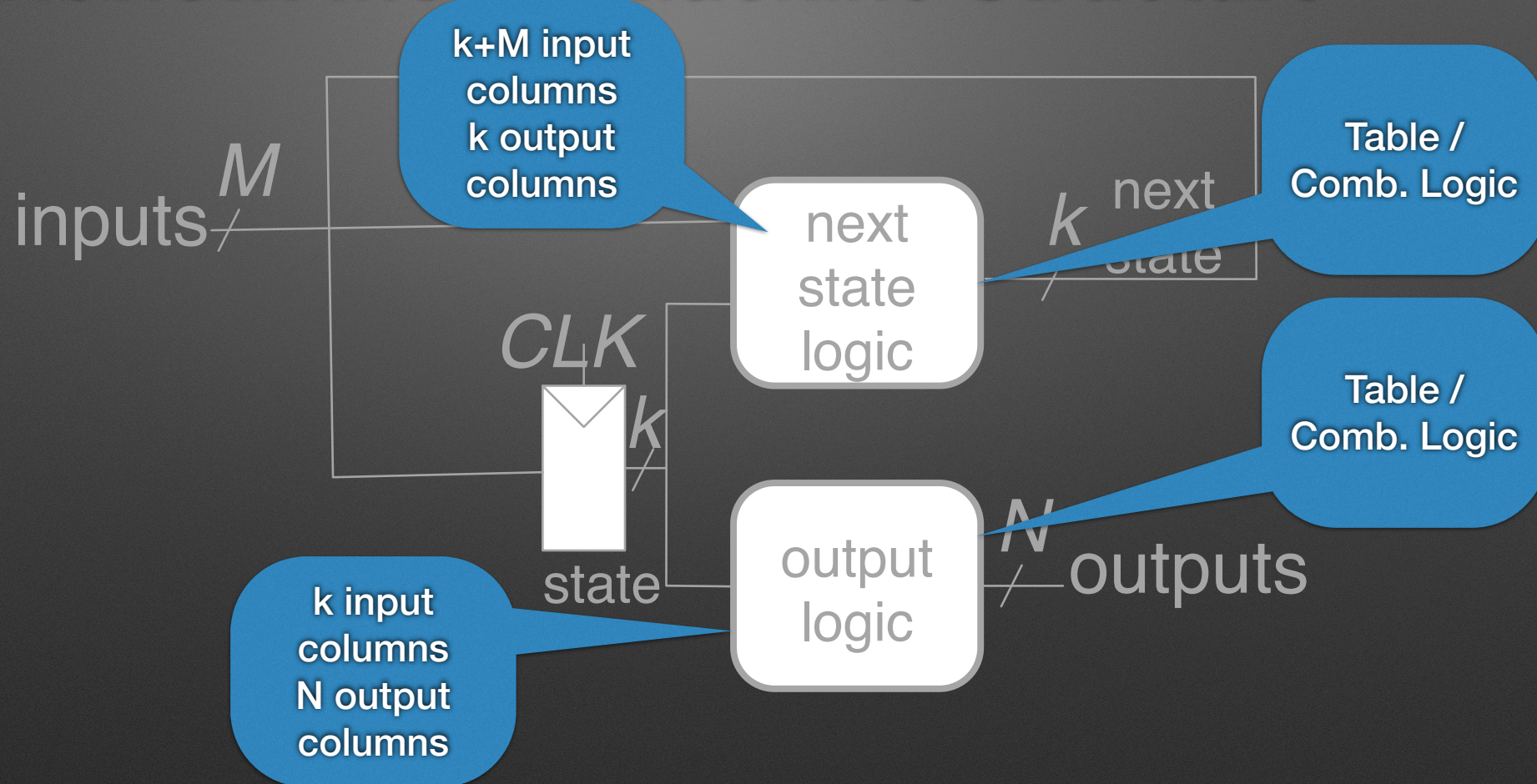
Review: Moore Machine Structure



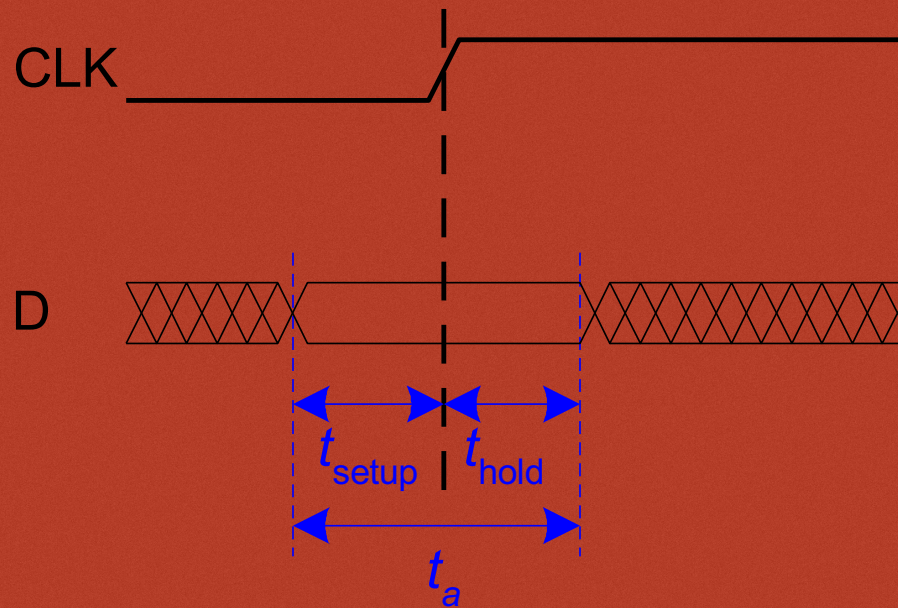
Review: Moore Machine Structure



Review: Moore Machine Structure

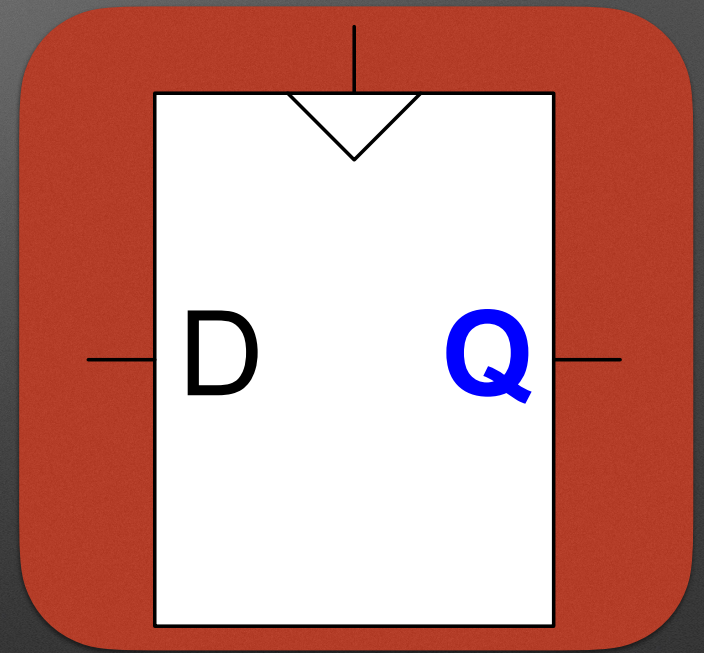


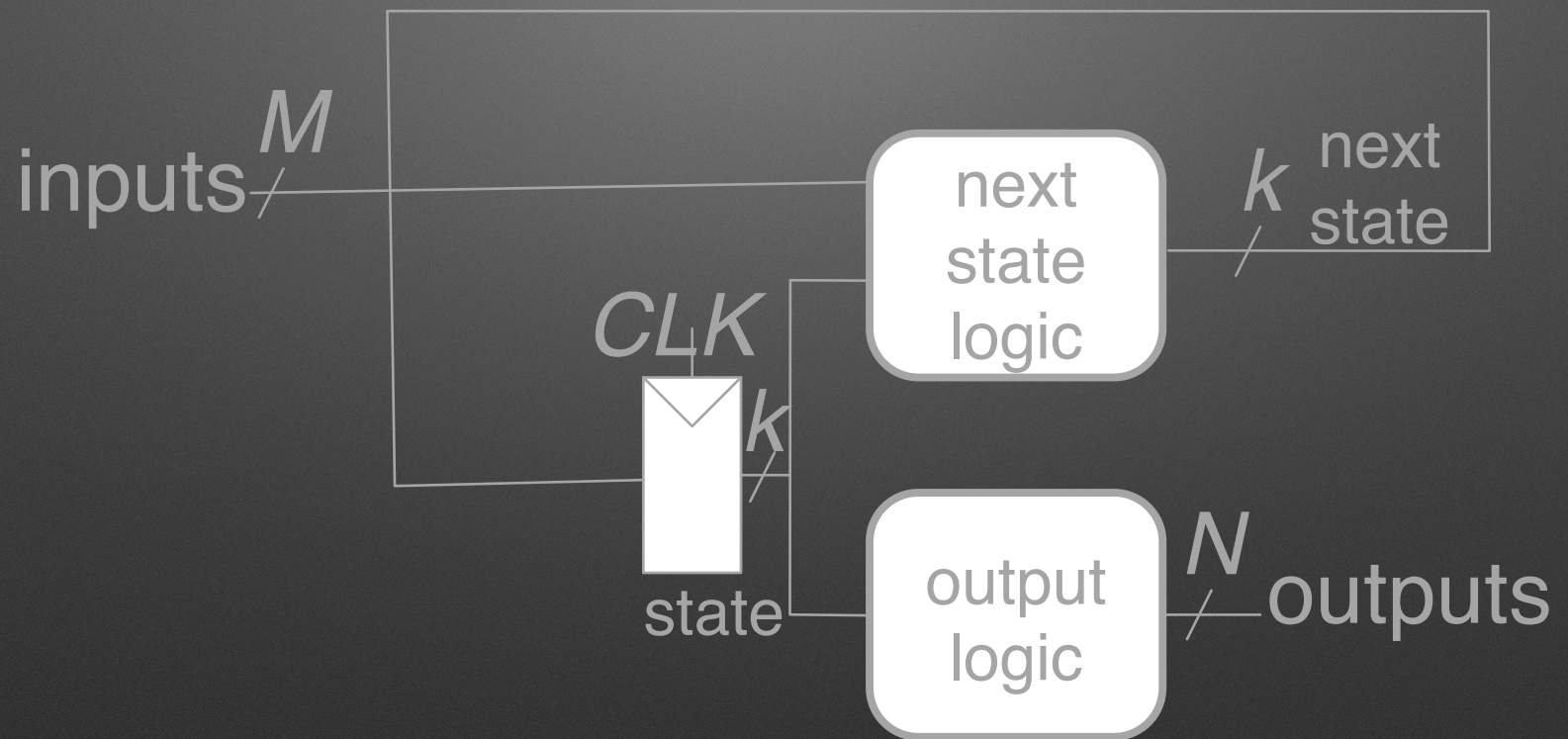
Review: Dff Setup & Hold Time



Dff Time Parameters

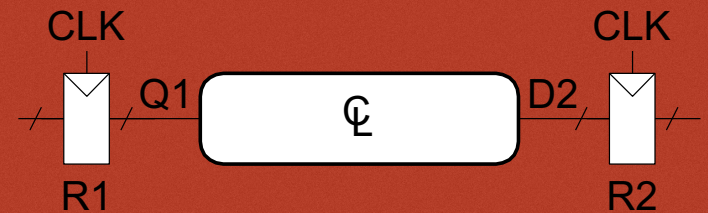
- t_{pcq} : Propagation delay from Clock to Q (pcq)
- t_{ccq} : Contamination delay from C to Q (ccq)





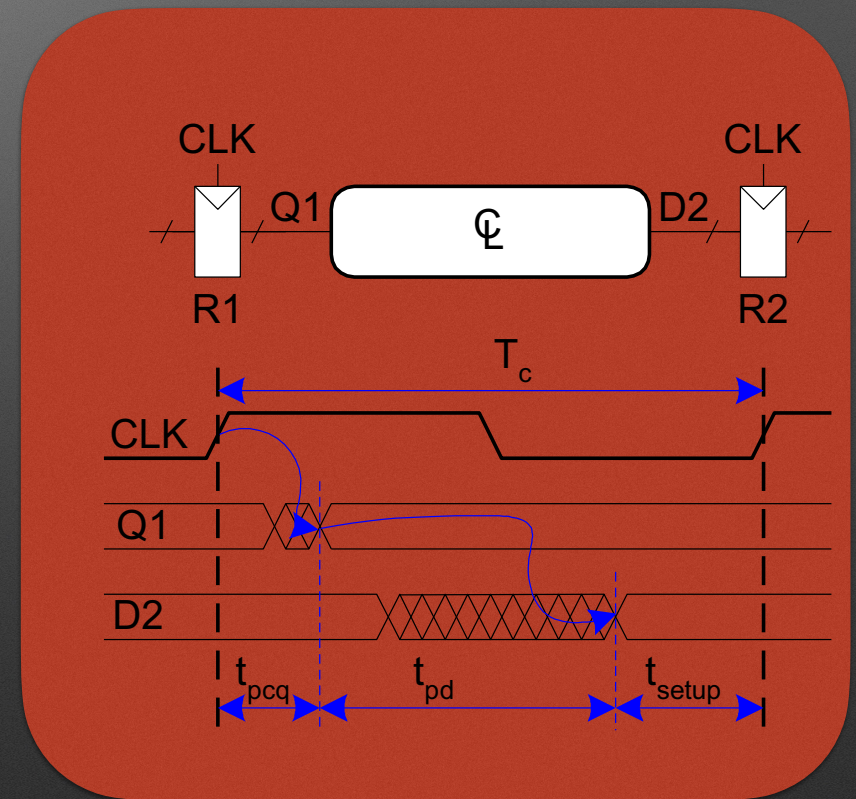
Setup Time Constraint

- Max time from R1 through CL
- R2's input (D2) needs to be stable t_{setup} before clock



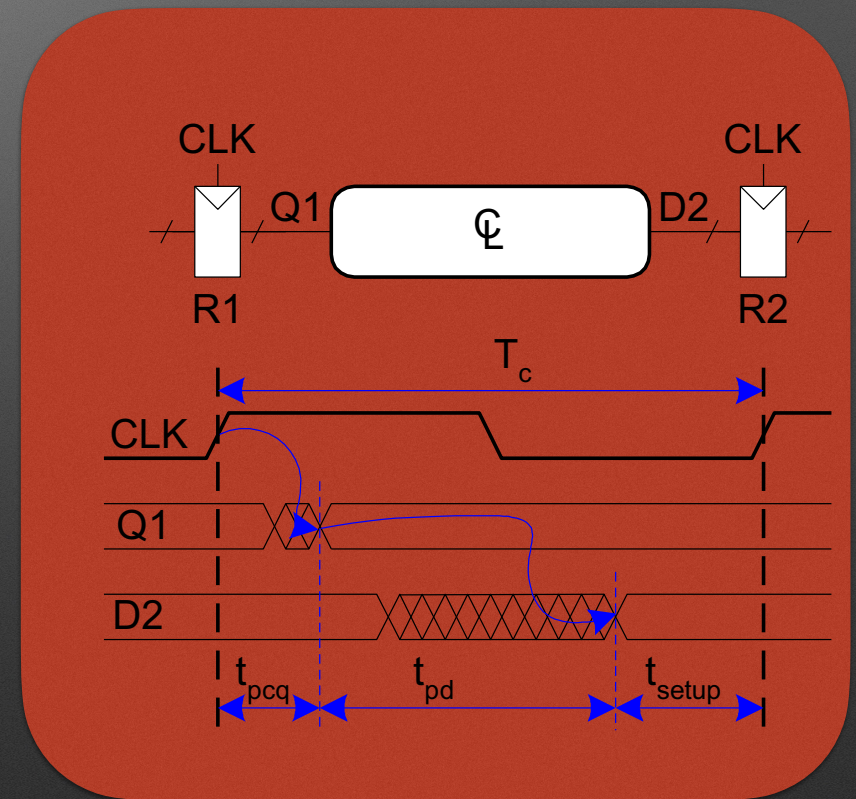
Setup Time Constraint

- Max time from R1 through CL
- R2's input (D2) needs to be stable t_{setup} before clock



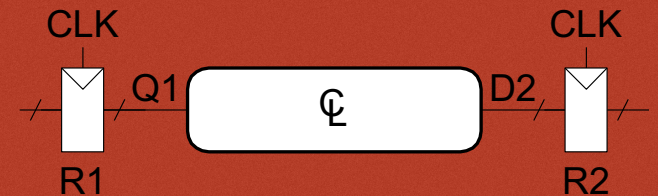
Setup Time Constraint

- Max time from R1 through CL
- R2's input (D2) needs to be stable t_{setup} before clock



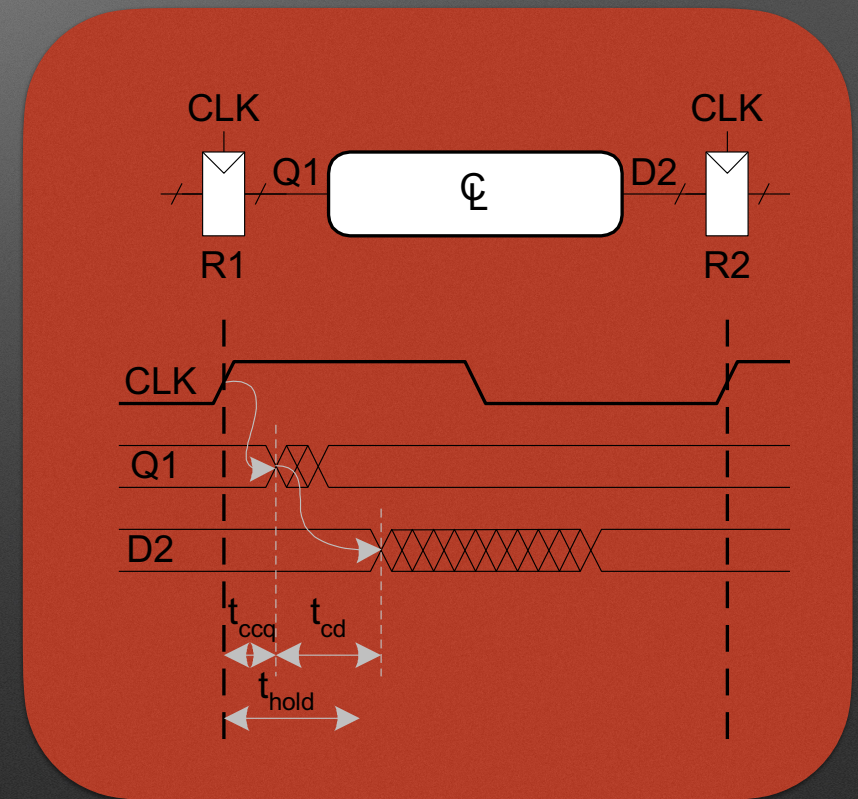
Hold Time Constraint

- Min time from R1 through CL
- R2's input (D2) must be stable t_{hold} after the clock



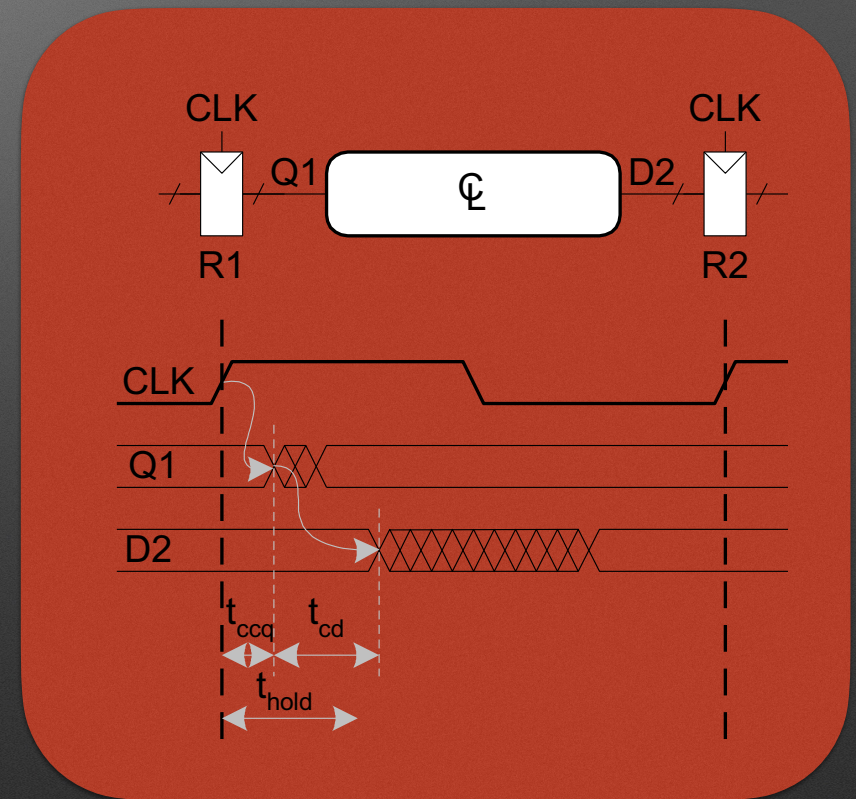
Hold Time Constraint

- Min time from R1 through CL
- R2's input (D2) must be stable t_{hold} after the clock

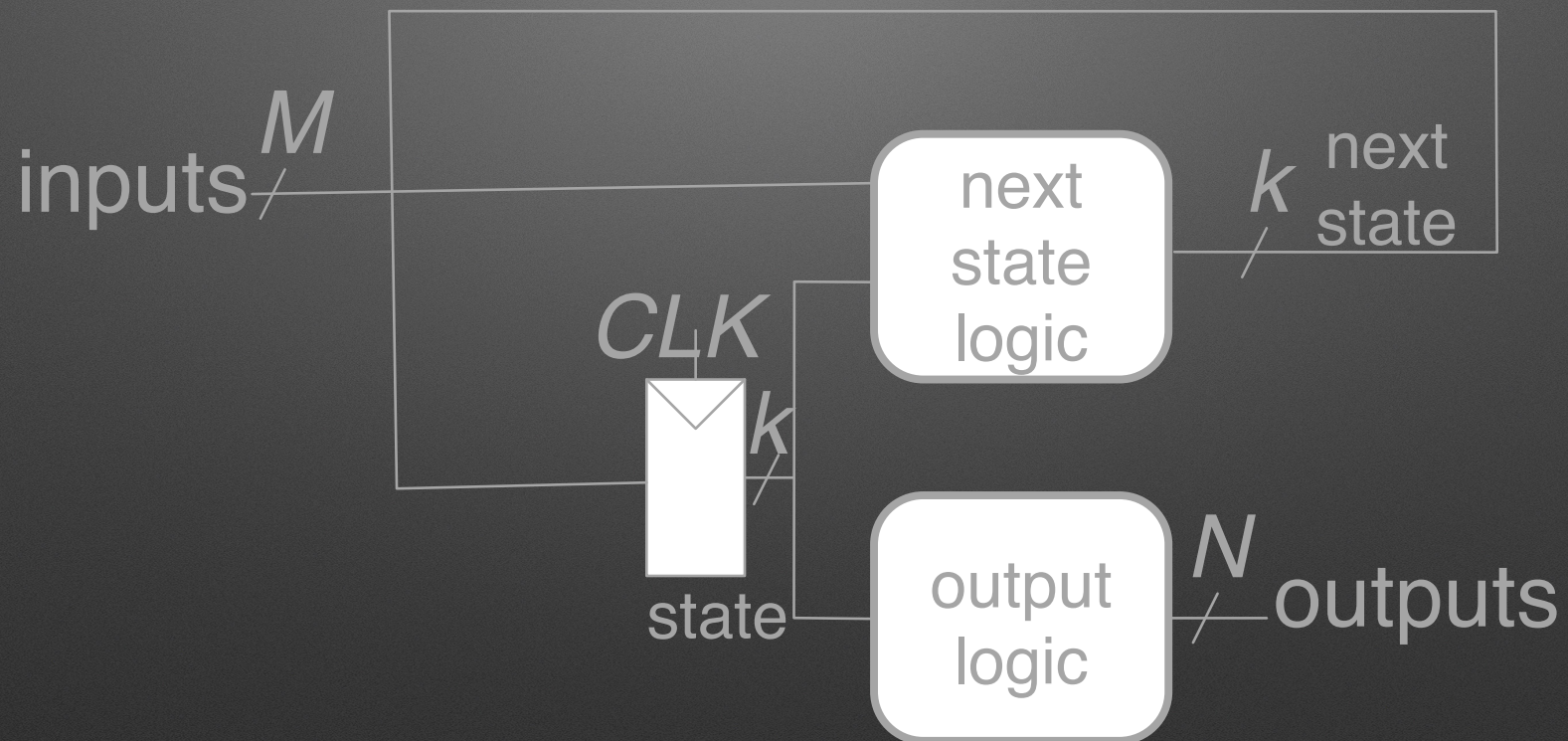


Hold Time Constraint

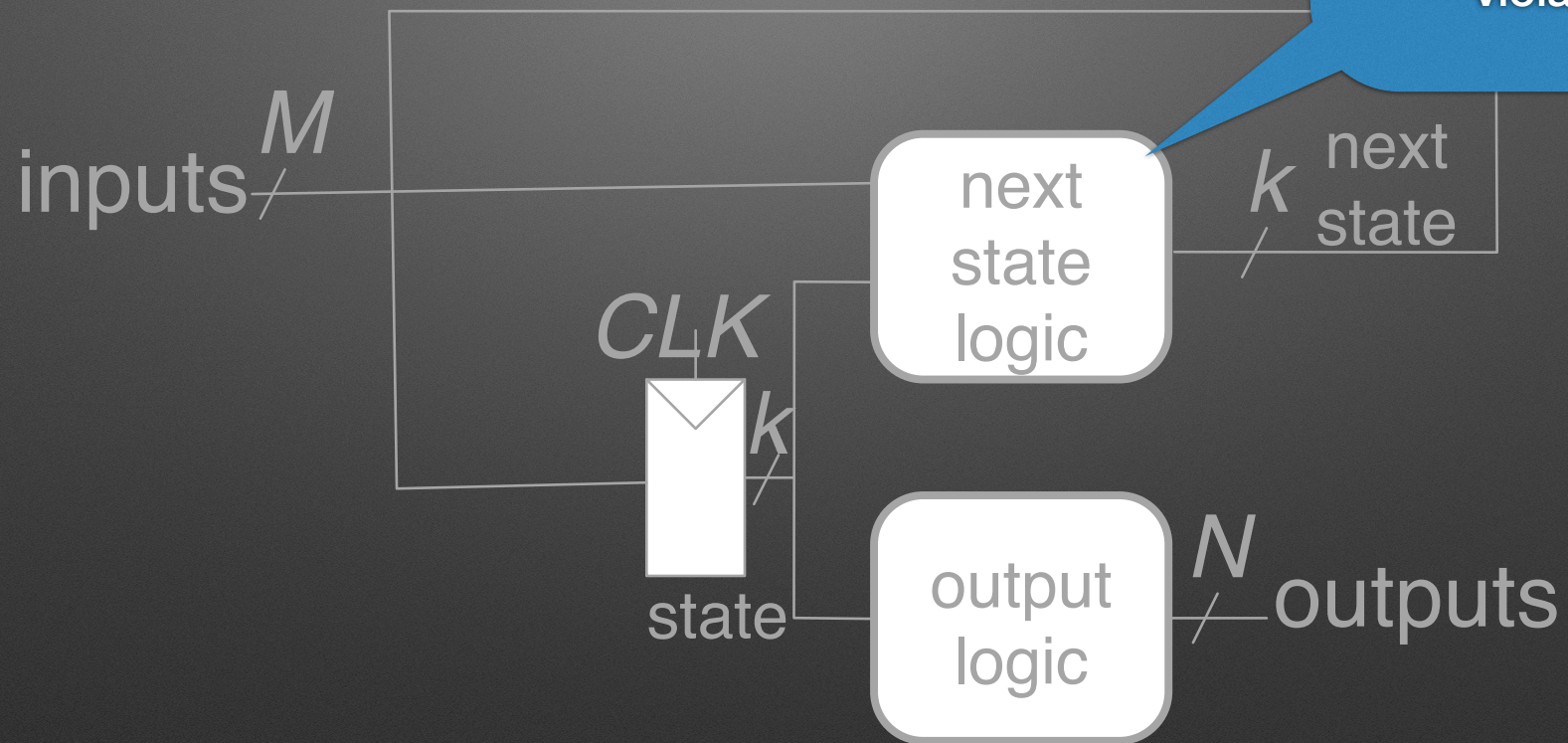
- Min time from R1 through CL
- R2's input must be stable t_{hold} after the clock



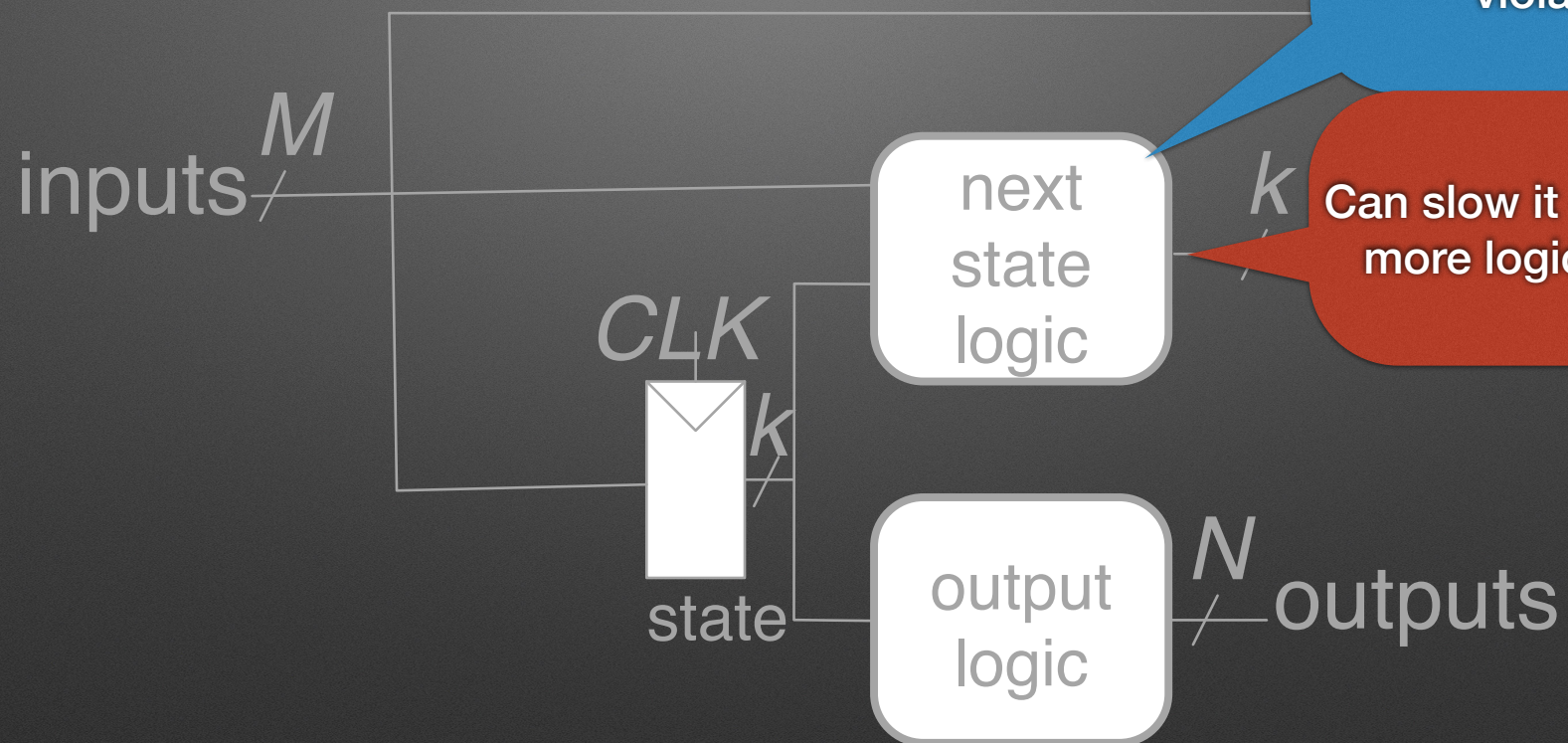
Review: Moore Machine Structure



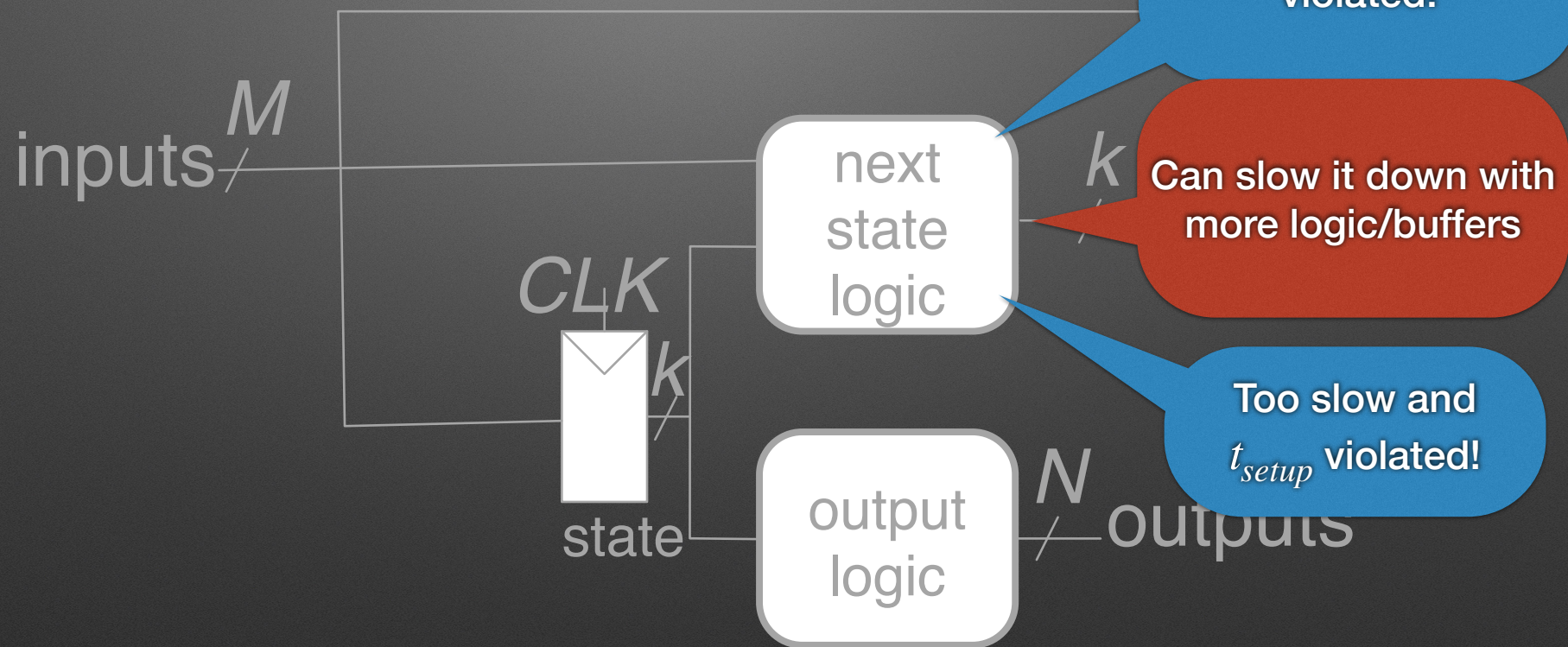
Review: Moore Machine Structure



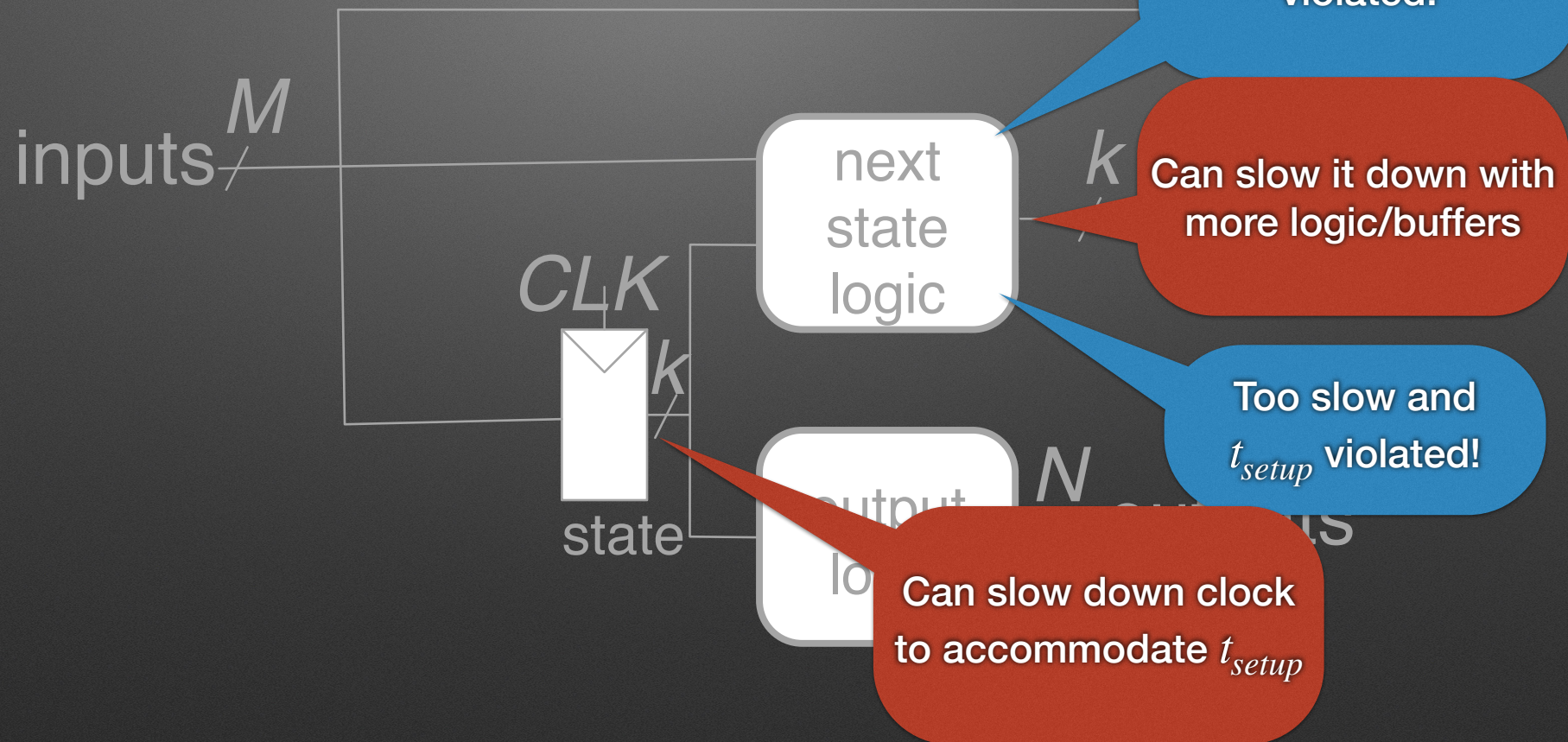
Review: Moore Machine Structure



Review: Moore Machine Structure



Review: Moore Machine Structure



Synchronous Timing

- Must meet both
 - Setup Time Constraint
 - Hold Time Constraint

Next Time

- Studio