CSE 2600
Intro. To Digital Logic & Computer Design

Bill Siever
&
Michael Hall

Reminder

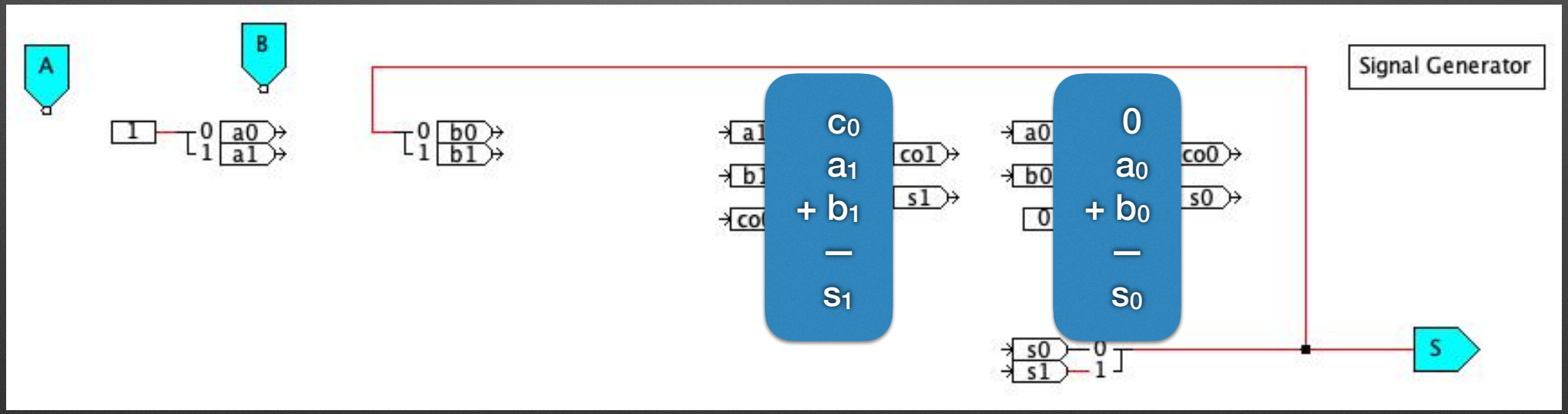
- Studios can (and should) be done collaboratively!
- Homework should be done *independently*
  - Ok to discuss *ideas* and related work from *studio*
  - Should not discuss / share details of specific solutions

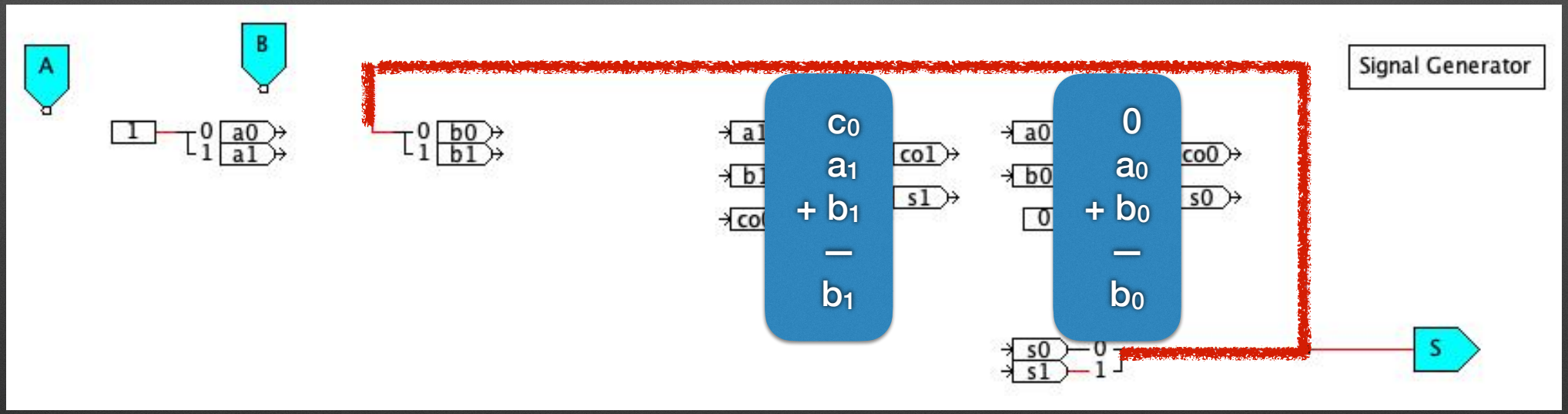
This week & misc.

- Hw1 returned: Regrade requests open for 1 week
- Hw #3B Posted / Due Sunday
  - Can re-submit autograder stuff
- Exam 1: Next Thursday during class
  - Exam 1 page: https://washu-cse2600-sp26.github.io/events/exam1
- Next Tuesday: Exam Review
  - TAs likely to have exam review ~7pm on Feb. 18th too (McDonnell 361?)

Last Lecture

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


A
B
1
0
1
a0
a1
0
1
b0
b1
a1
b1
co
c0
a1
+ b1
—
s1
co1
s1
a0
b0
0
0
a0
+ b0
—
s0
co0
s0
s0
s1
0
1
S
Signal Generator


A
B
1
0
1
a0
a1
0
1
b0
b1
a1
b1
co
c0
a1
+ b1
−
b1
co1
s1
a0
b0
0
0
a0
+ b0
−
b0
co0
s0
s0
s1
0
1
S
Signal Generator

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!

- Short pulse
- R & S dropping at the same time

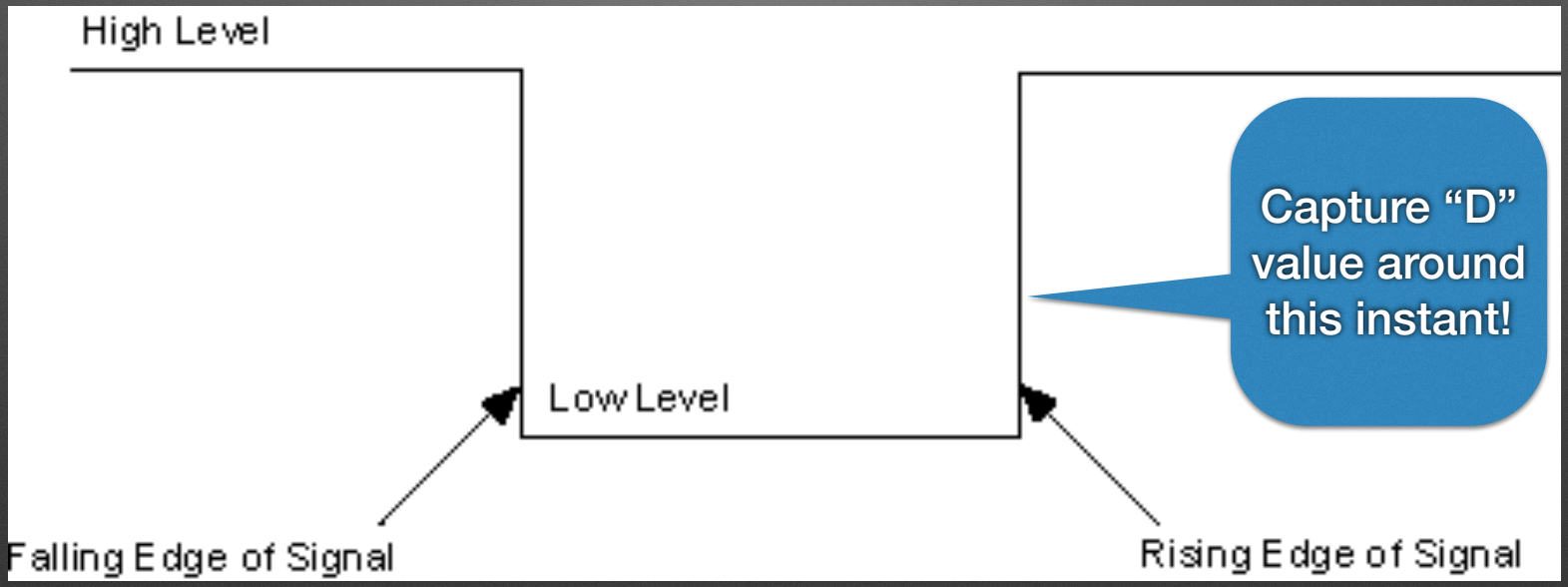
Last Lecture

- 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

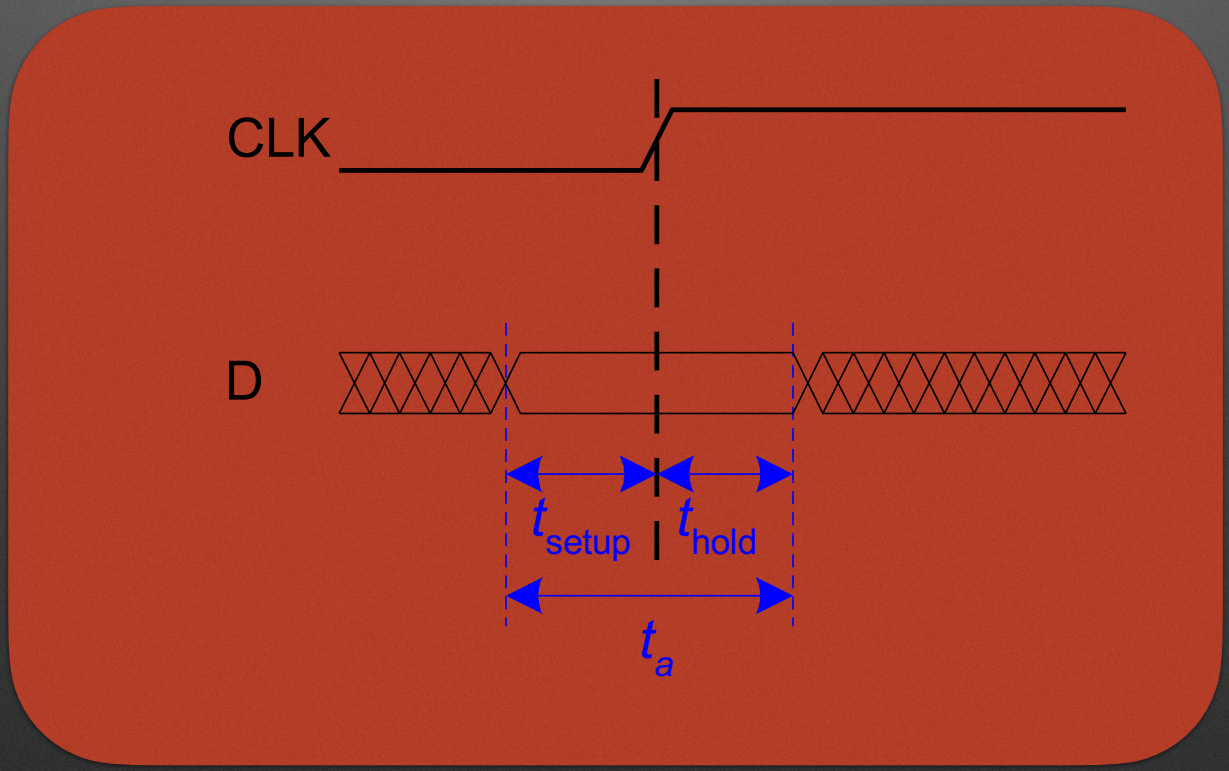
- Internal latch
  - Fails if pulses too short
  - Unstable if R/S drop at same time / too close
- Setup Time: Time needed before clock to ensure stable capture
- Hold Time: Time After clock edge value must be “held”

D-Flip-Flop Clock



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

Dff: Setup & Hold Time

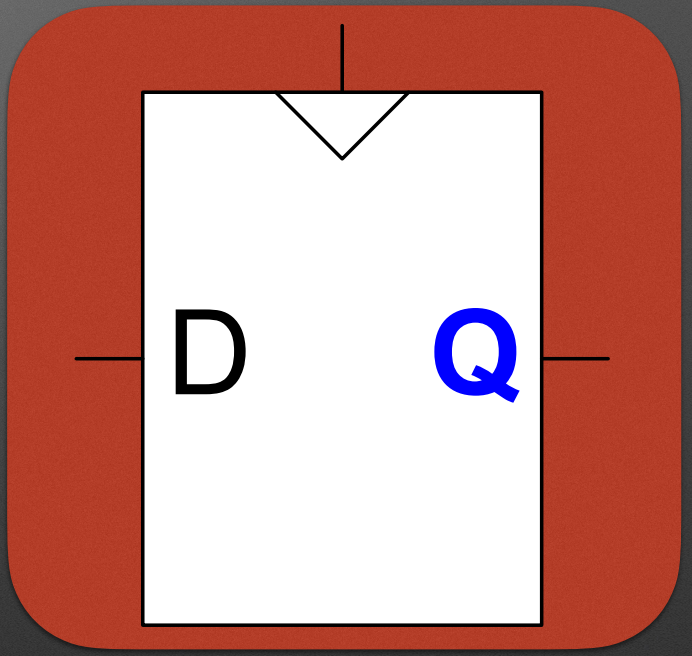


D Flip Flop Time Parameters

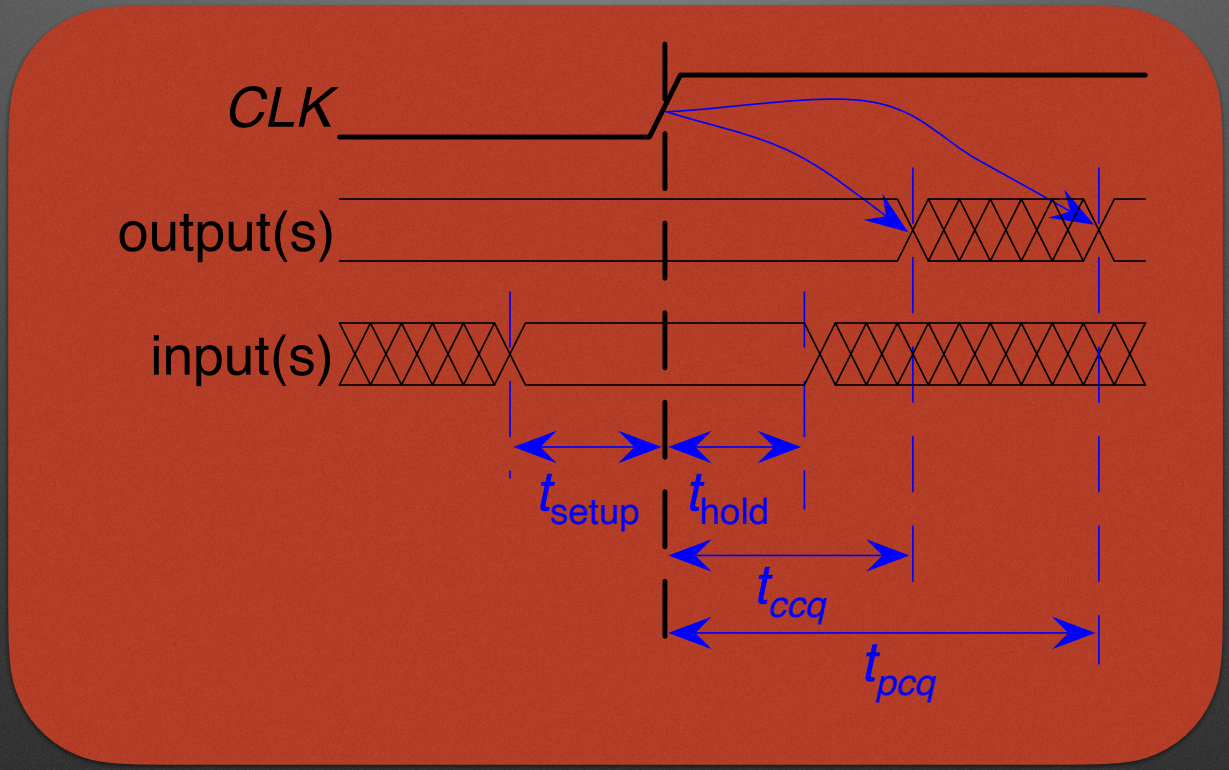
- t_{setup}: Time D must be stable before clock
- t_{hold}: Time D must be stable after clock
- t_a: Aperture time ($t_{setup} + t_{hold}$) –
  total window of time D needs to be stable around clock

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

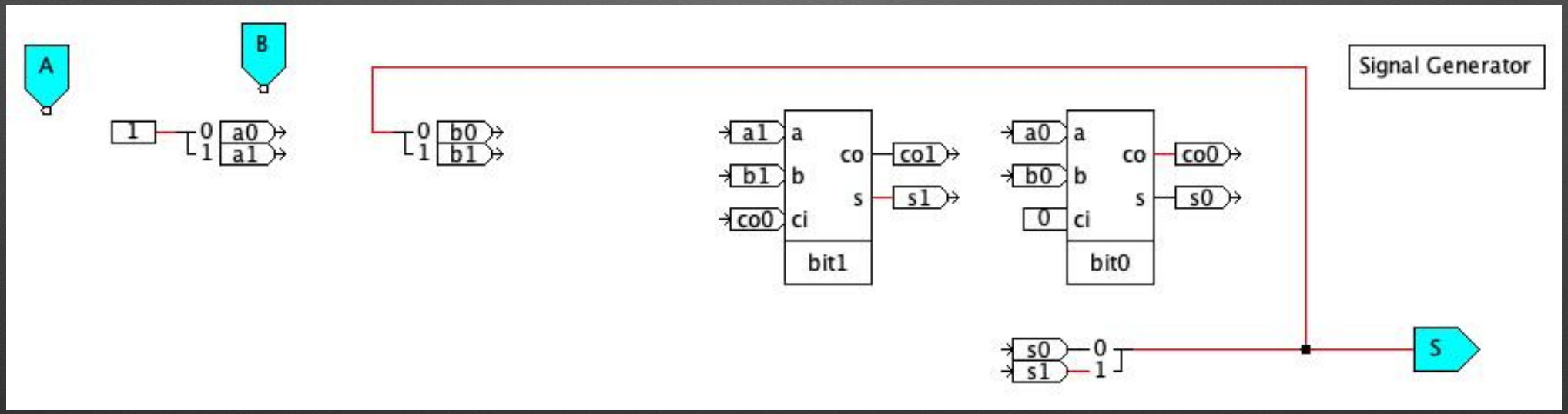


Sequential & Synchronous Logic

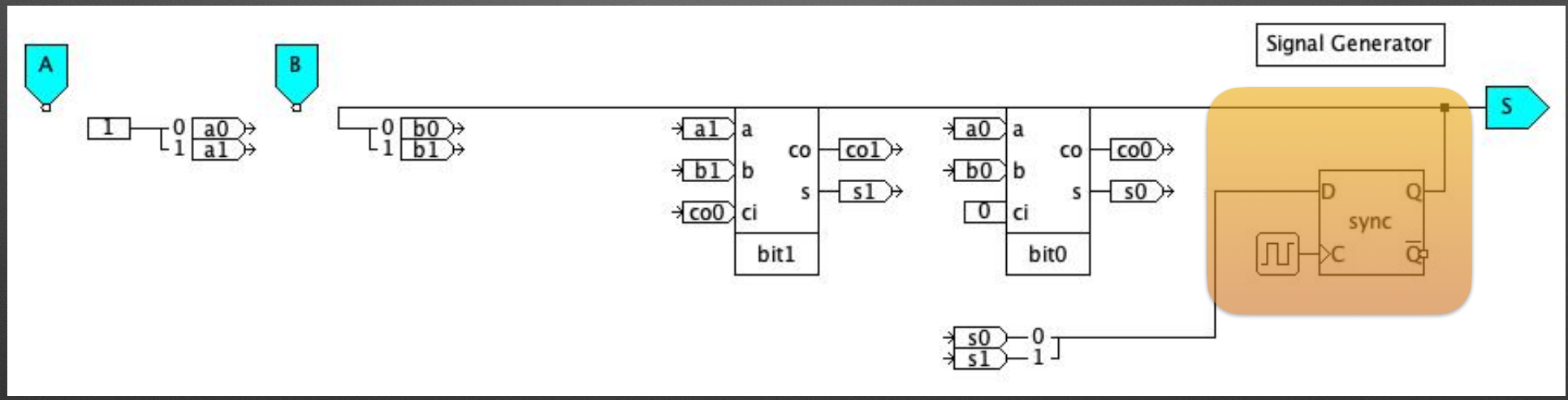
- *Sequential* circuits
  - Can’t be represented with a *simple* table of just inputs and outputs (Possibly a complex table of history of inputs and outputs)
  - Output depends on sequence of inputs and timing
  - *Synchronous* Sequential Circuits
    - Sequential circuits with additional restrictions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

Synchronous Sequential Circuits

- 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!


A
B
Signal Generator
1
0
1
a0
a1
0
1
b0
b1
a1
b1
co0
a
b
ci
co
s
co1
s1
bit1
a0
b0
0
a
b
ci
co
s
co0
s0
bit0
s0
s1
0
1
S

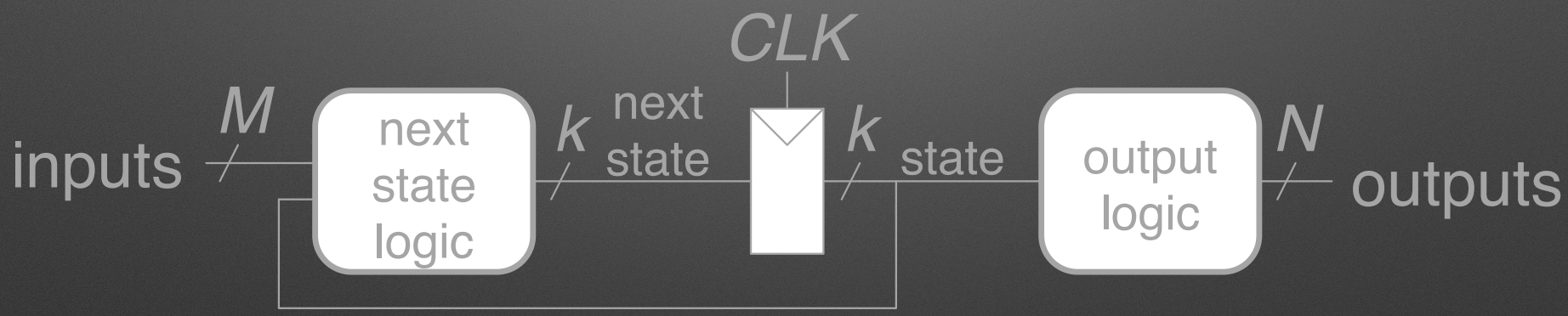
Loopy Adder w/ *Synchronous*



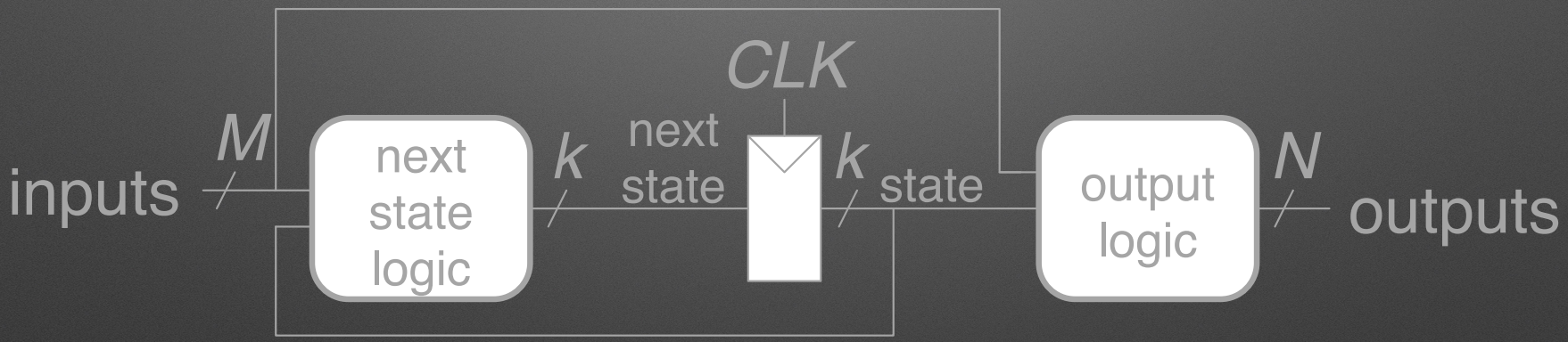
Finite State Machines

- 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: Mealy Machine



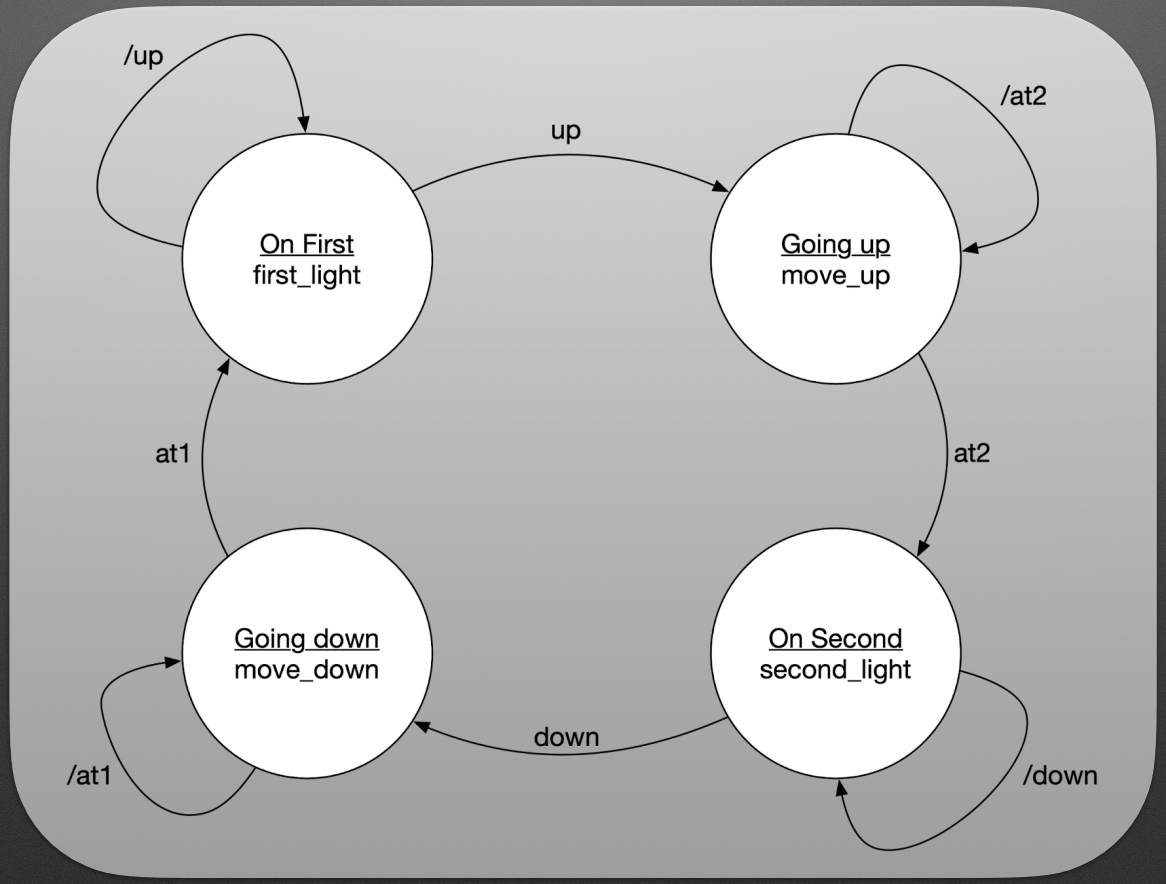
FSM Applications

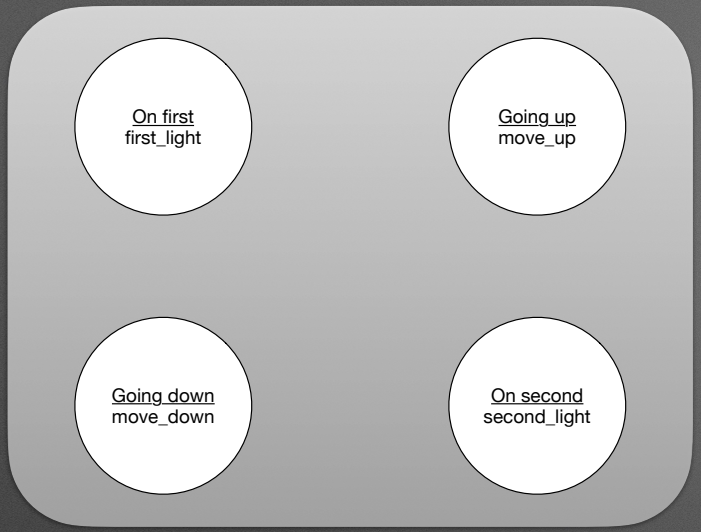
- Things with modes or sequences of steps. Examples:
  - Washing Machine (fill, agitate, rinse, spin)
  - Stop lights & Traffic control: Green, Yellow, Red
  - Locks: Locked & unlocked
  - Computer programs: Playing game vs. on menu
  - Elevator controls (state = floor)
  - …

Studio 3A Summary

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

Overall


/up
up
/at2
On First
first_light
Going up
move_up
at1
at2
Going down
move_down
On Second
second_light
down
/at1
/down



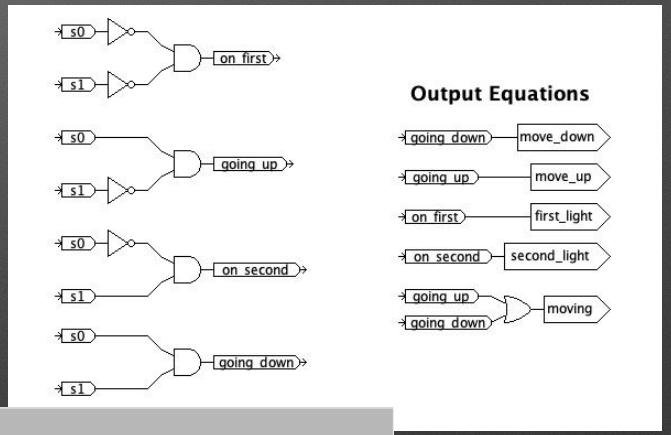
Overall

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

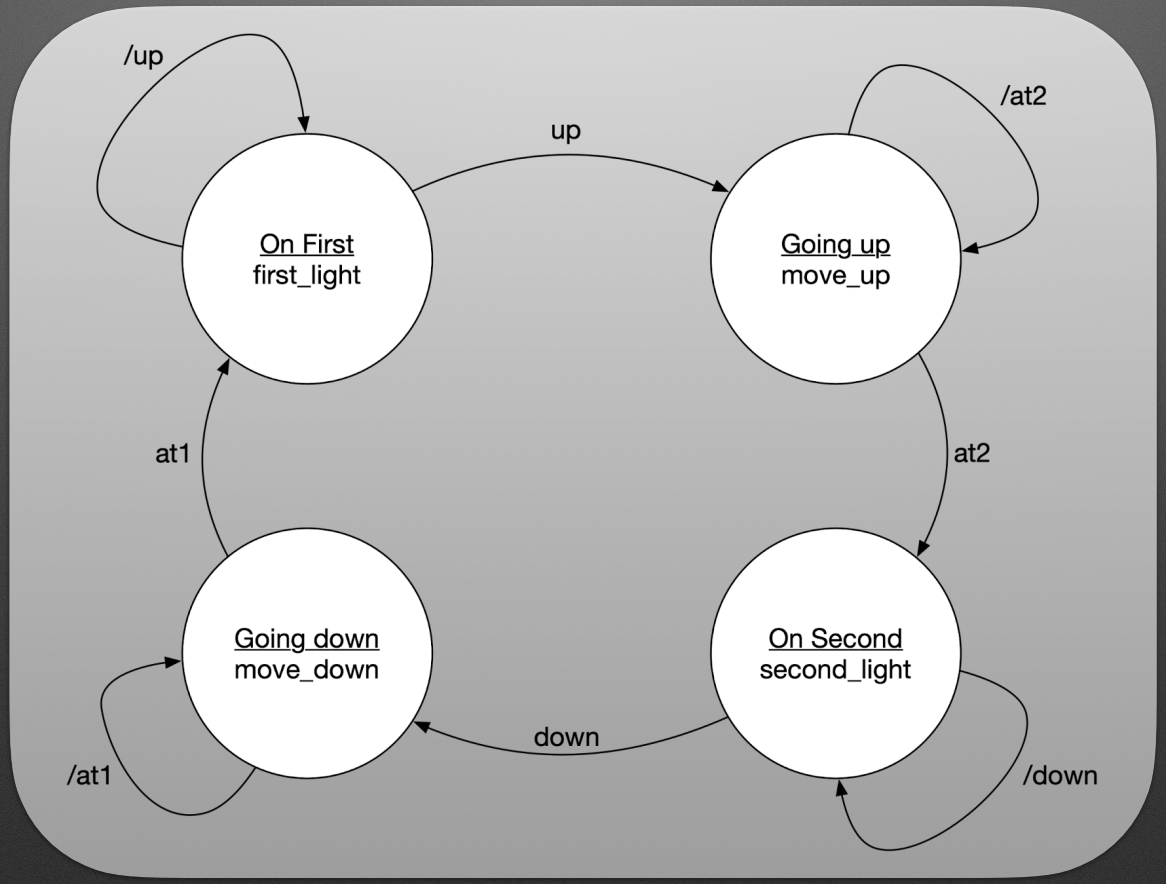
move_up = Going_up = /S1*S0
move_down = Going_down = S1*S0
first_light = On_First =/S1*/S0
second_light =On_Second = S1*/S0
moving = Going_up + Going_down = /S1*S0 + S1*S0

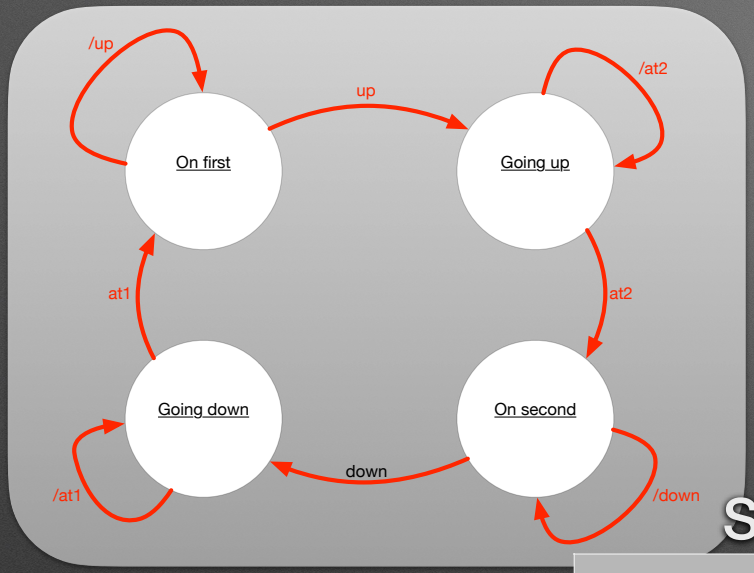


Output Decisions (based on state)

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

Overall


/up
up
/at2
On First
first_light
Going up
move_up
at1
at2
Going down
move_down
On Second
second_light
down
/at1
/down



Overall

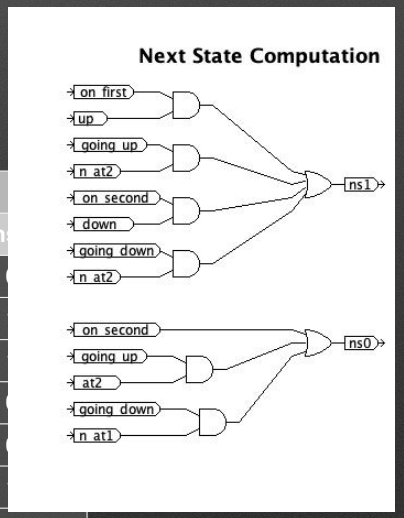
State Encoding

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

ns1 = On_Second*at2 +
On_second*/down +
Going_down*down +
Going_down*/at1

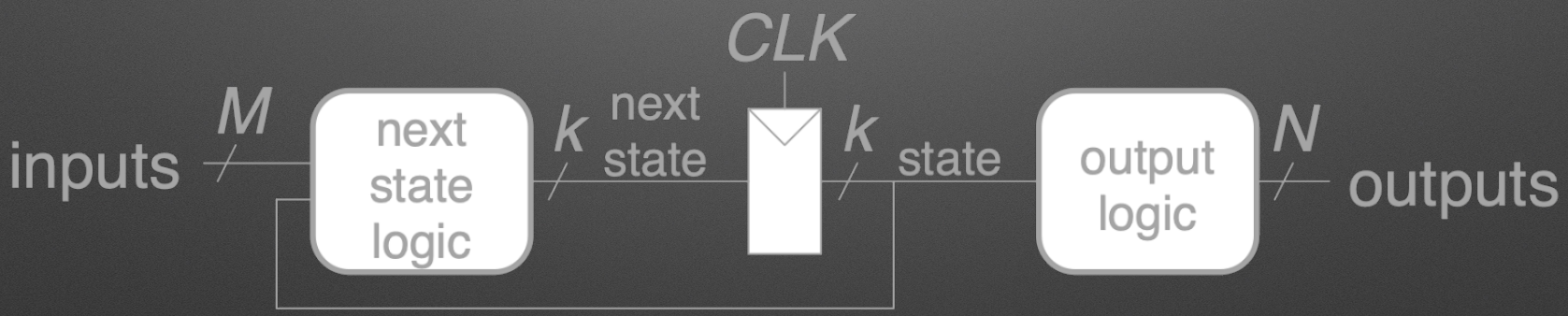
ns0 = Going_up*up +
Going_up*/at2 +
On_second*down +
Going_down*/at1



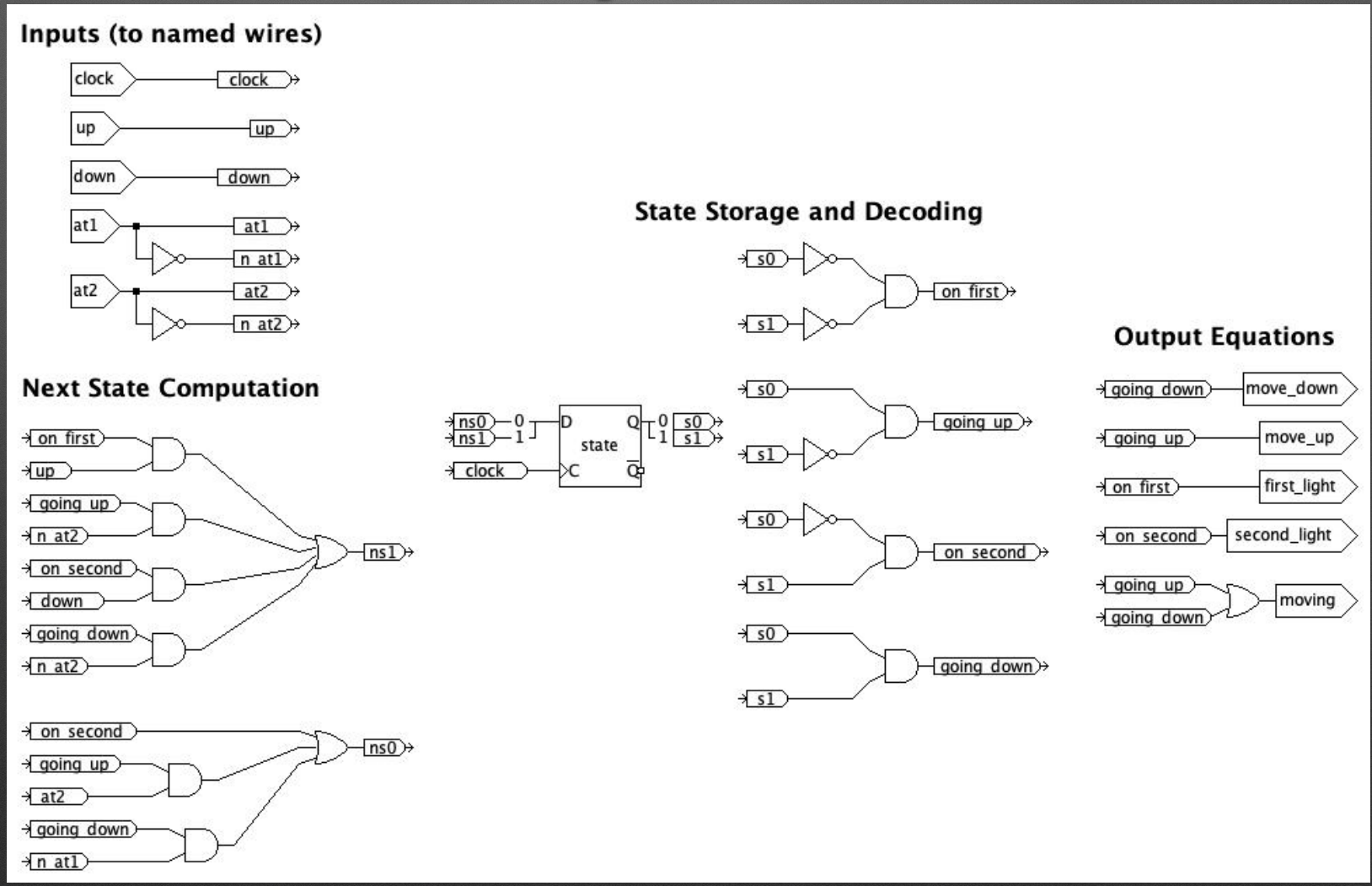
State Transitions (to “next state”)

State Name	Numeric Encoding	Inputs				Next State		
		up	down	at1	at2	Name	ns1	n
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	1
		X	X	1	X	On_first	0	0

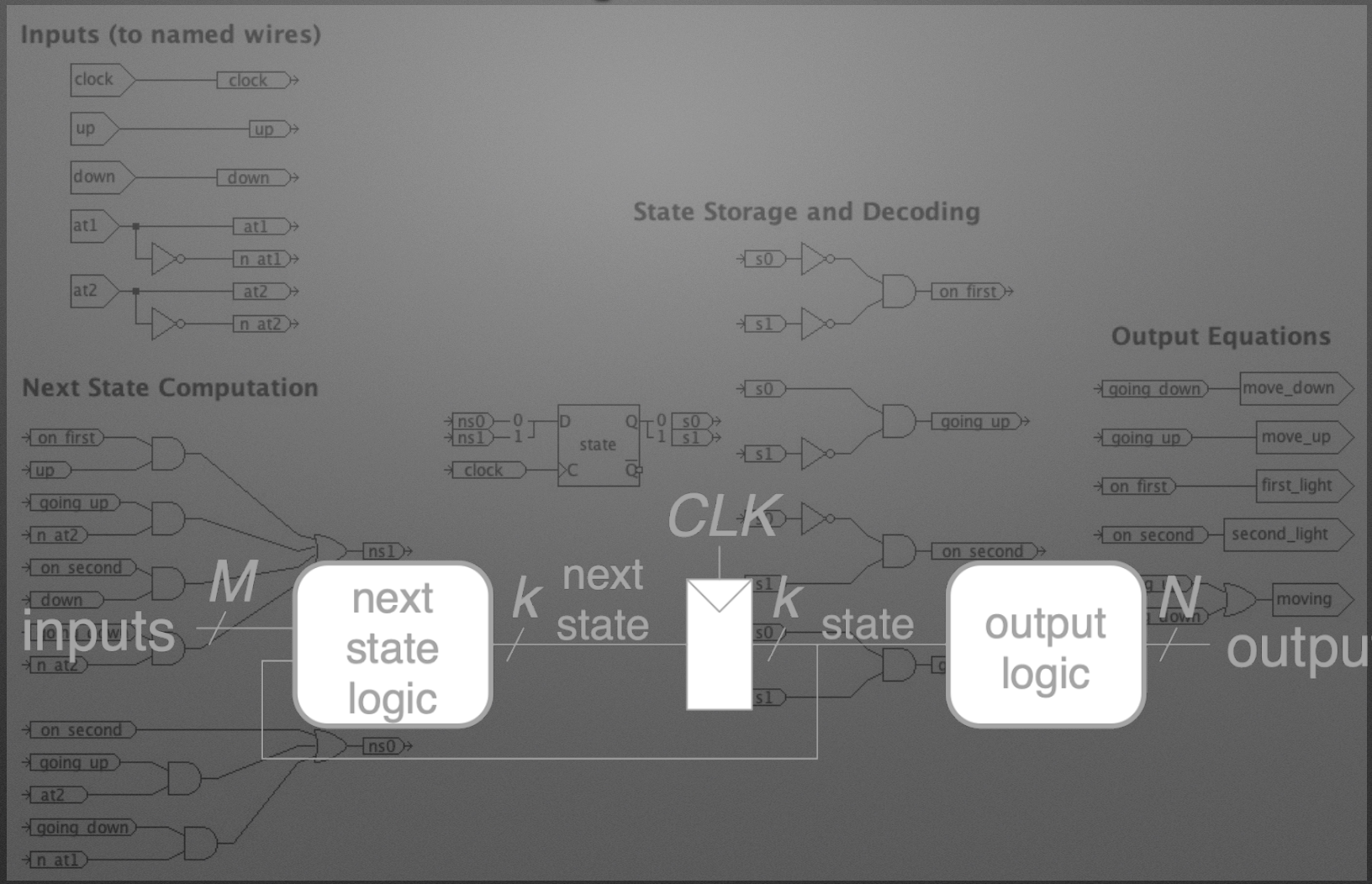
Overall Structure



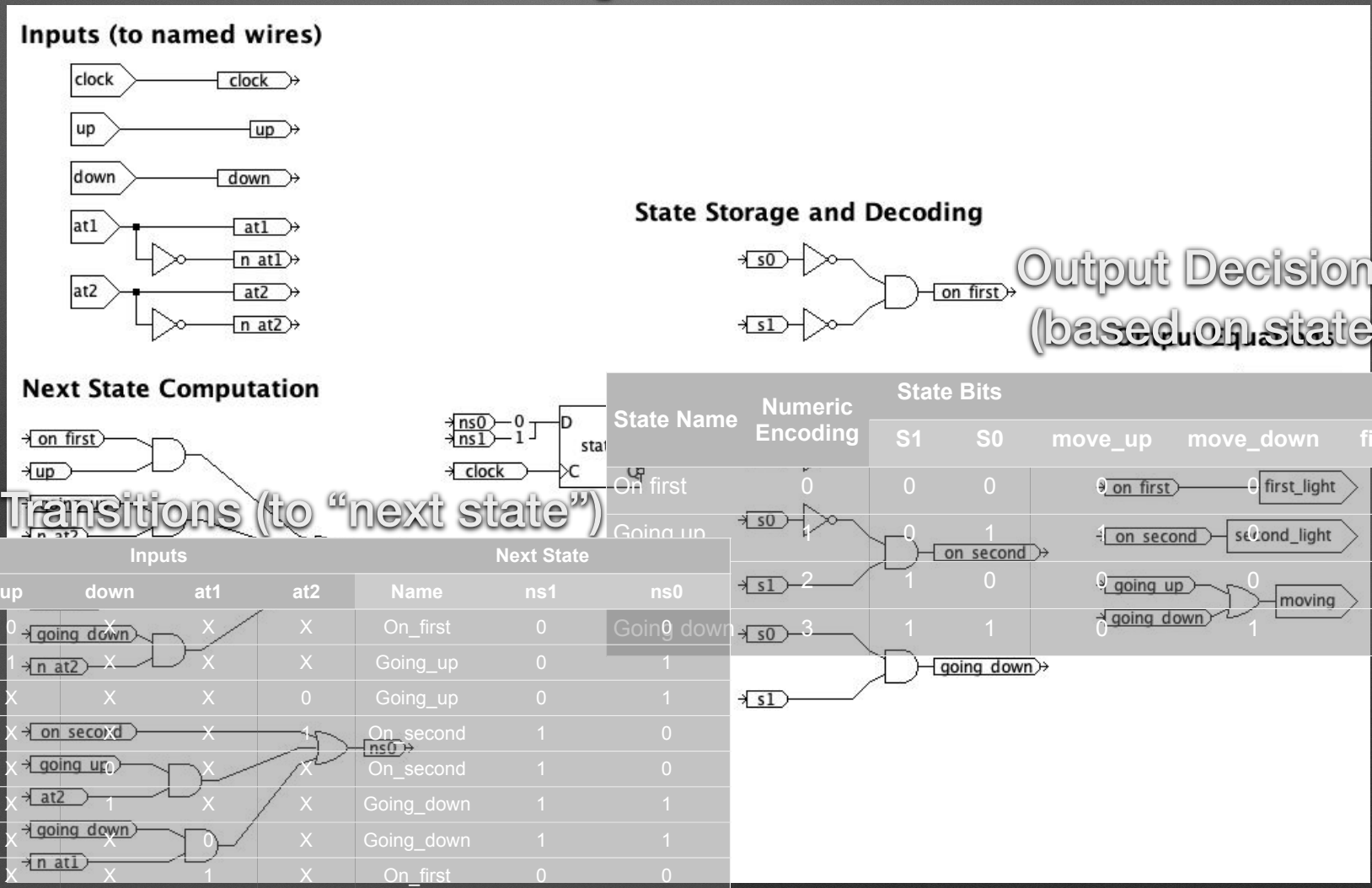
Overall Implementation



Overall Implementation



Overall Implementation



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	X	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

Output Decisions (based on state)

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

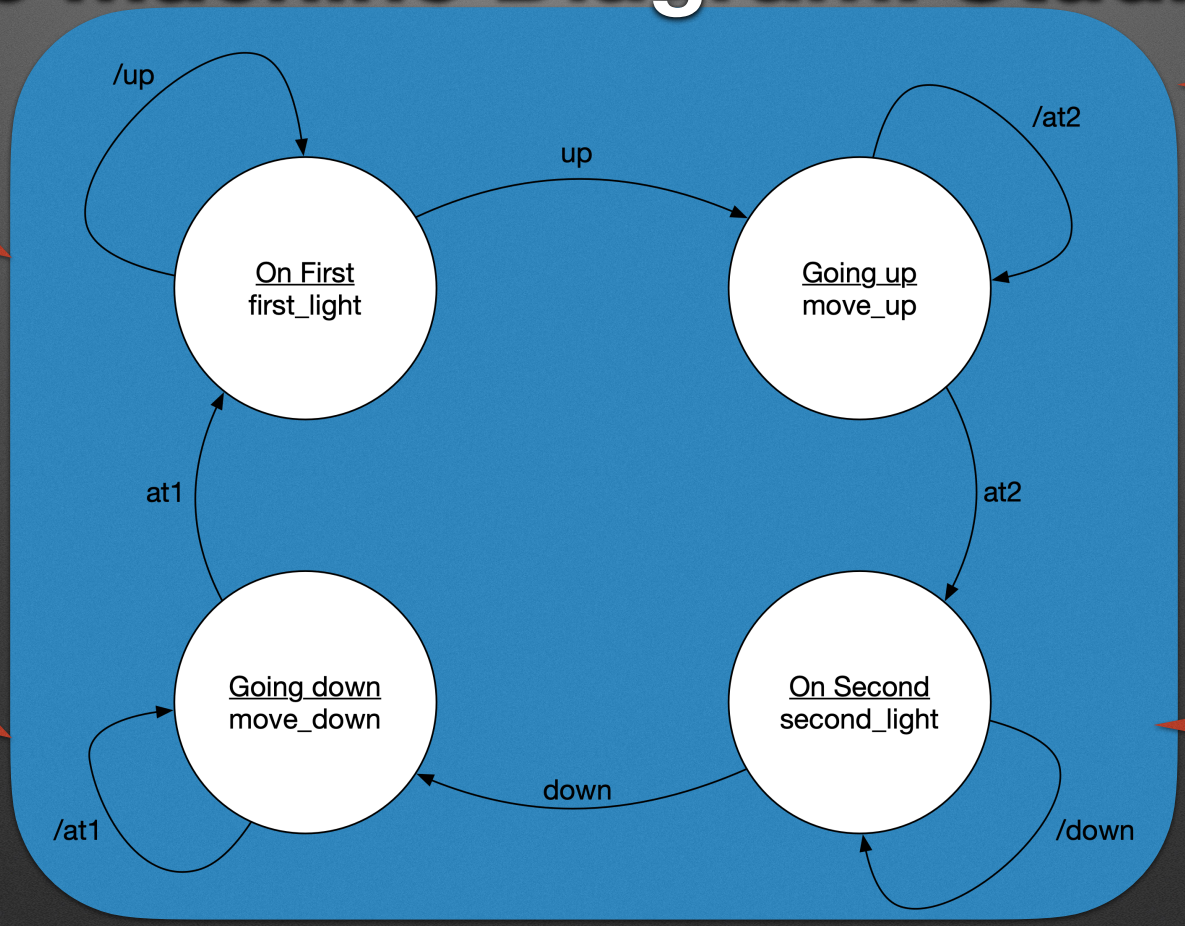
e Machine Diagram: Studic

Assume clock cycle of 10s?

Vs 240s?

What does “clock” do?

Can the clock be too short?



State Encodings

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

Elevator: Examples

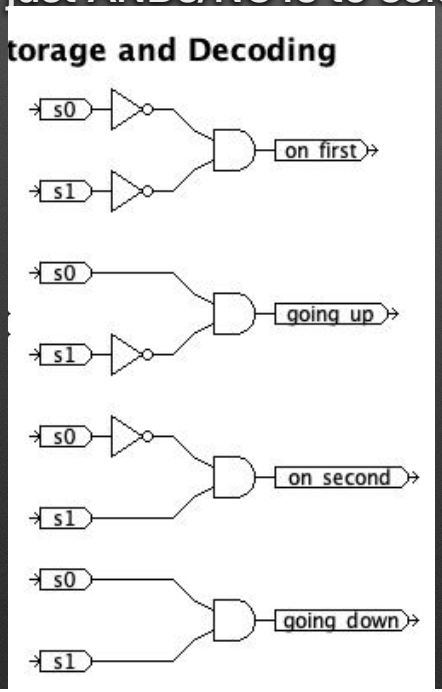
STATE NAME	BINARY COUNTING	ONE HOT
ON FIRST		
GOING UP		
ON SECOND		
GOING DOWN		

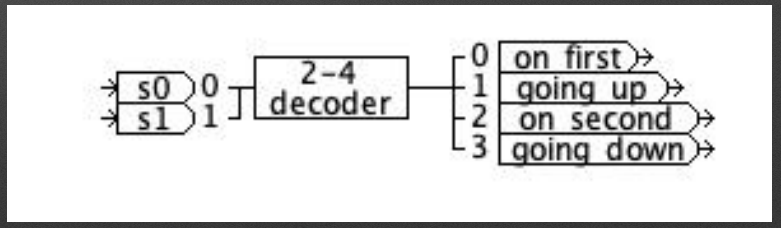
Elevator: Examples Part 2

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

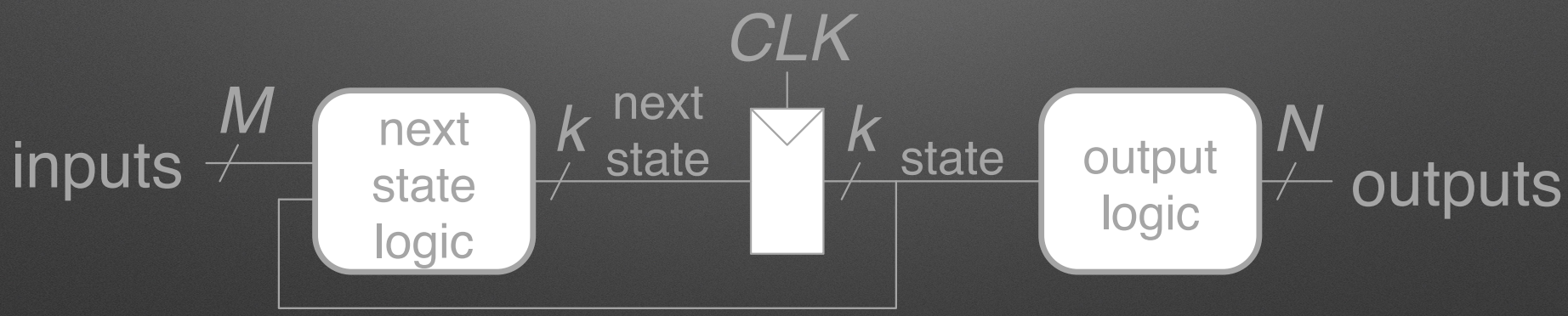
Decoder

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

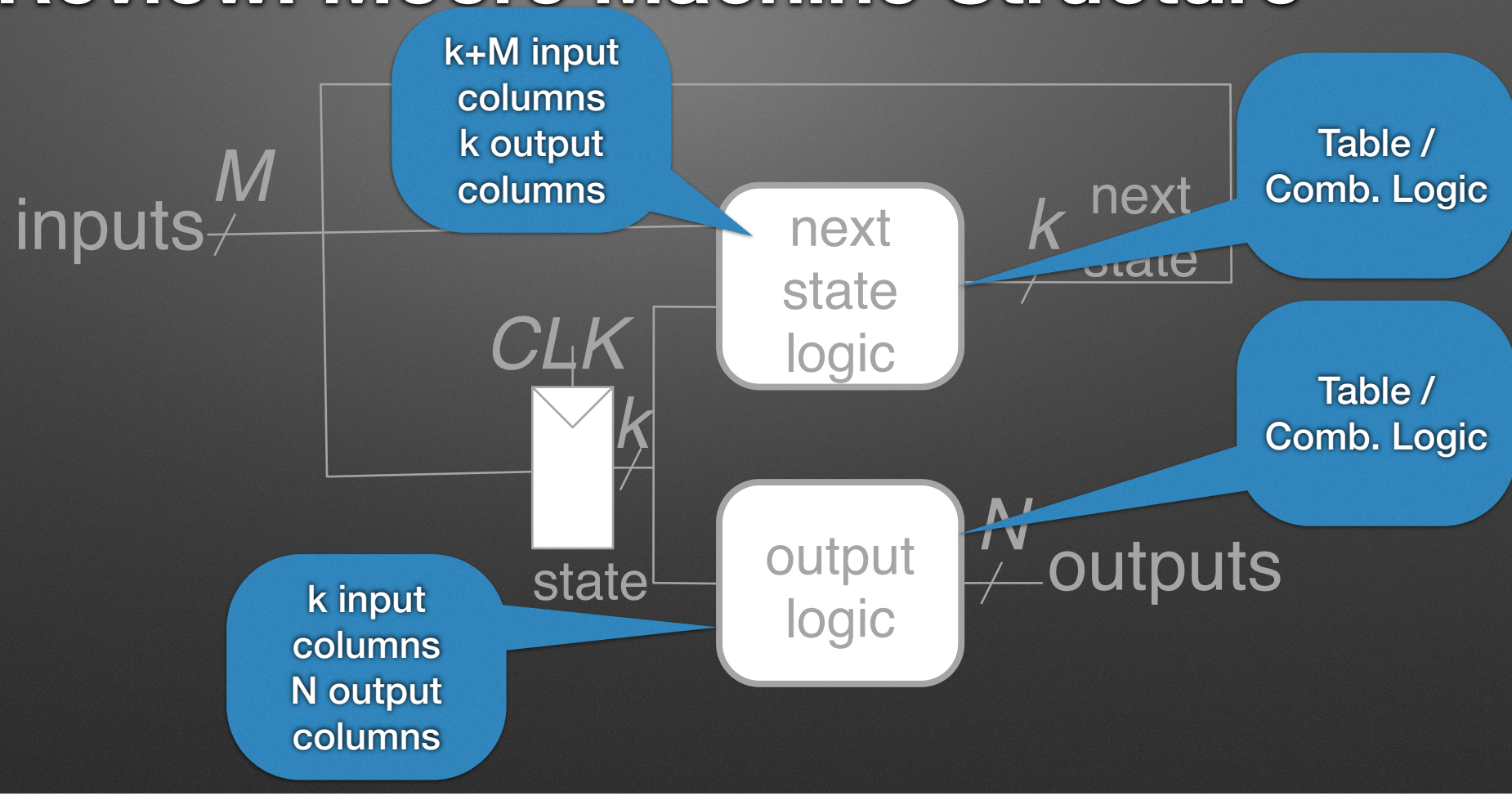




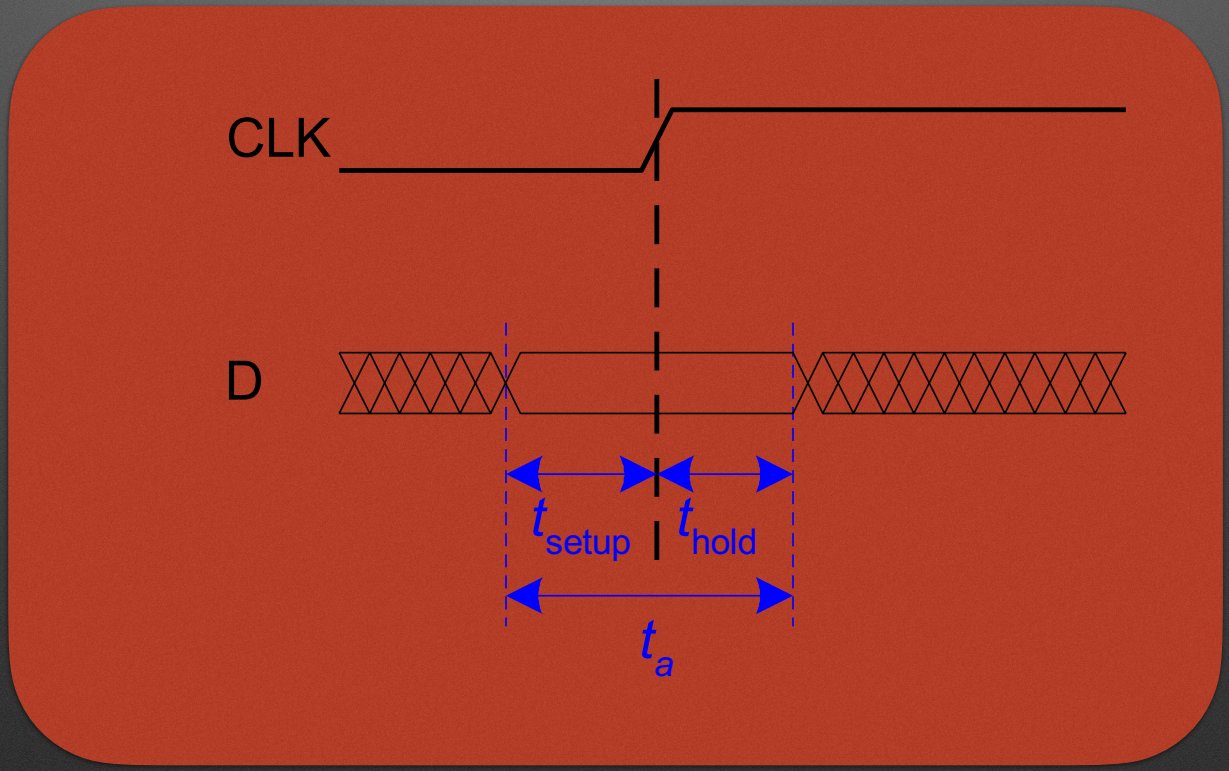
Review: Moore Machine Structure



Review: Moore Machine Structure

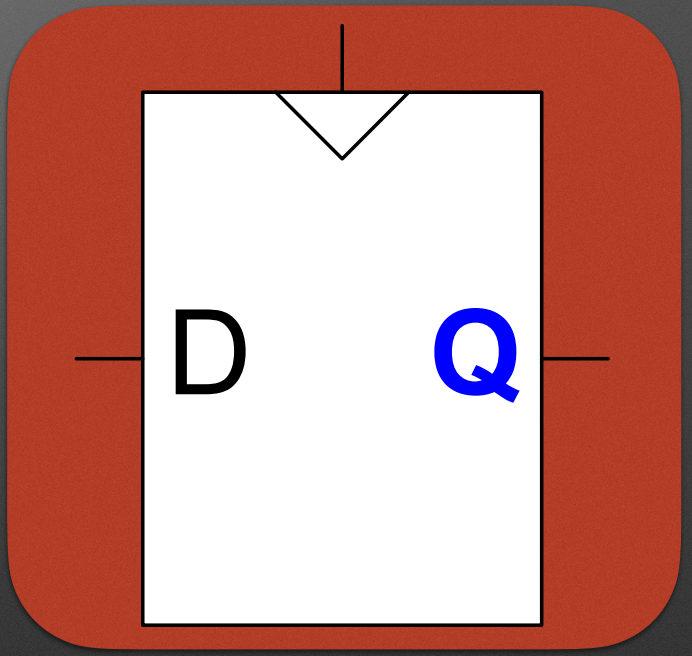


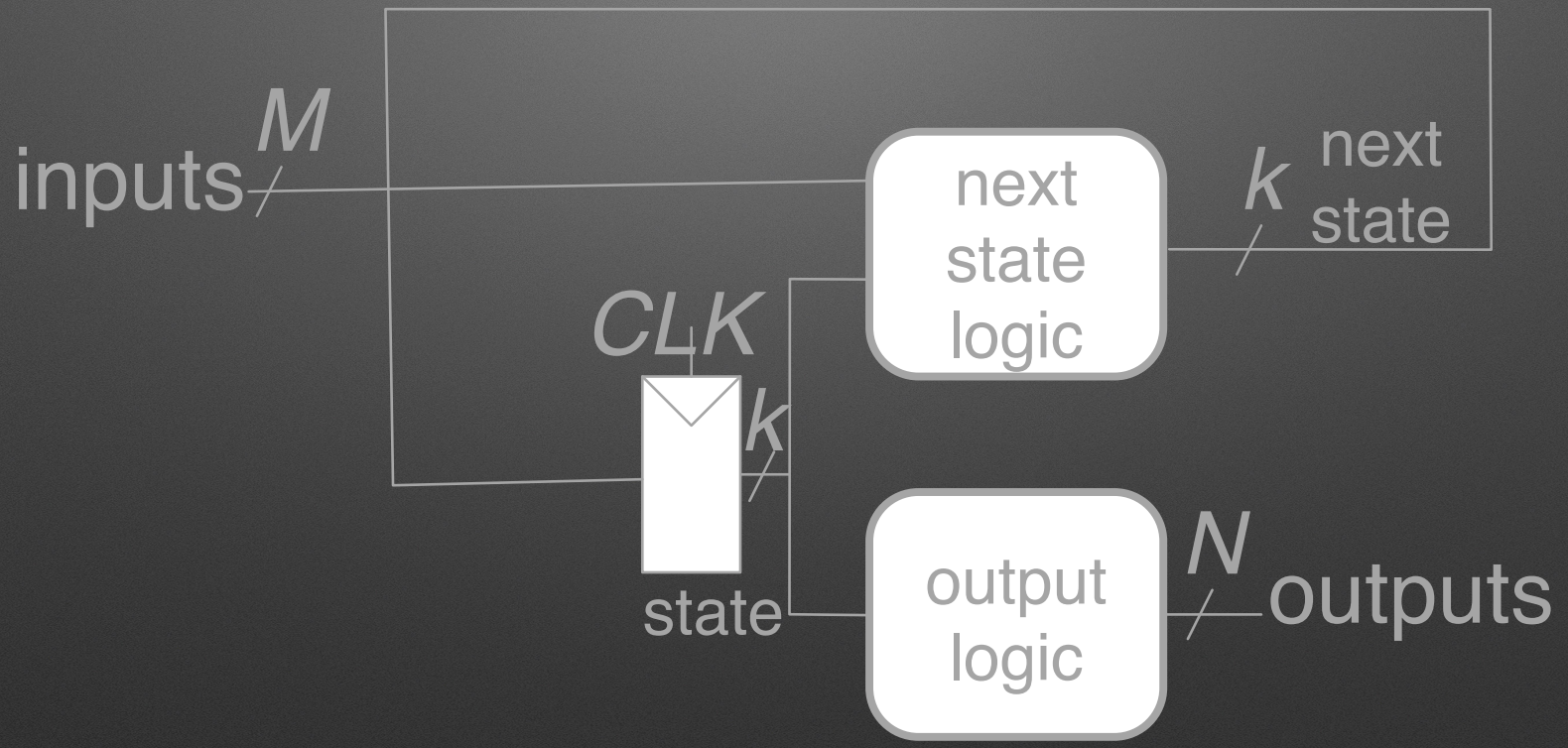
Review: Dff Setup & Hold Time



Dff Time Parameters

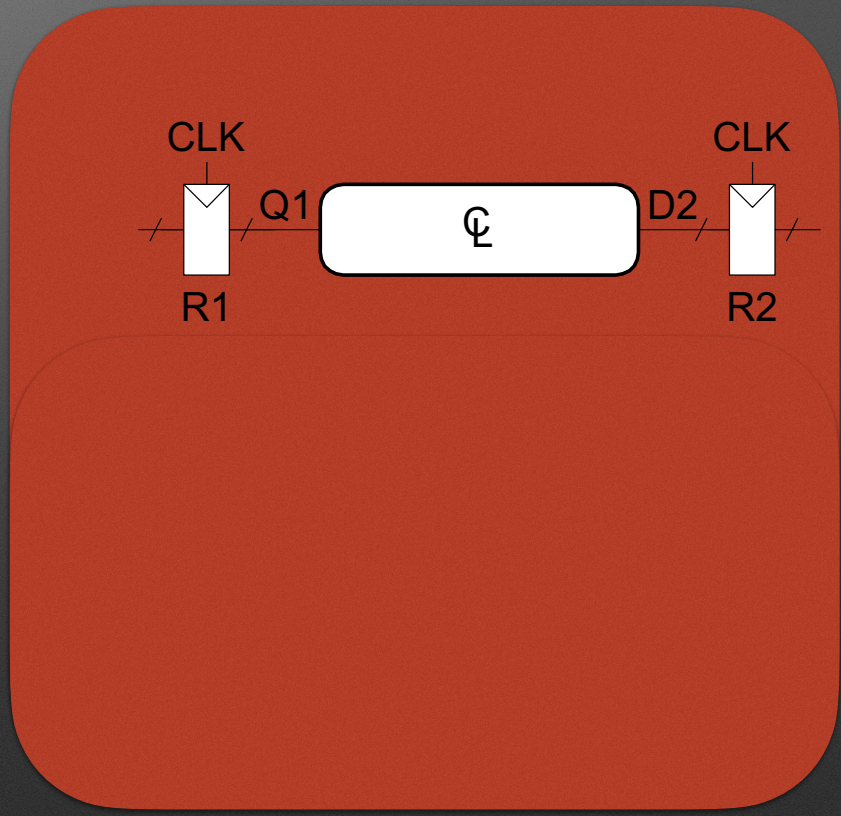
- t_{pcq}: Propagation delay from Clock to Q (pcq)
- t_{ccq}: Contamination delay from C to Q (ccq)




inputs
M
next
state
logic
k
next
state
CLK
k
state
output
logic
N
outputs

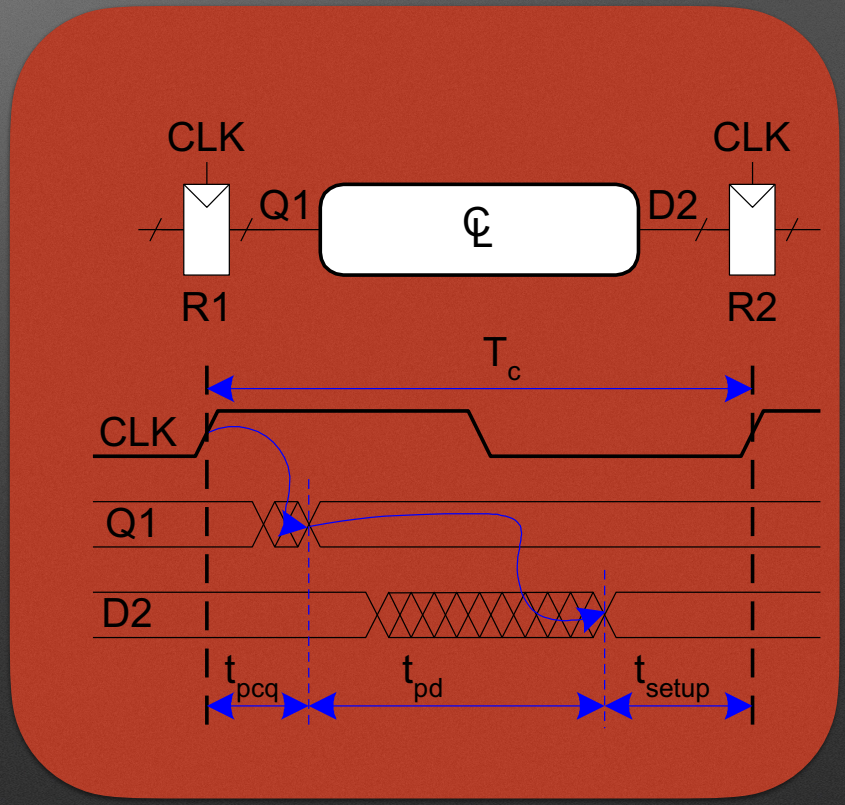
Setup Time Constraint

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



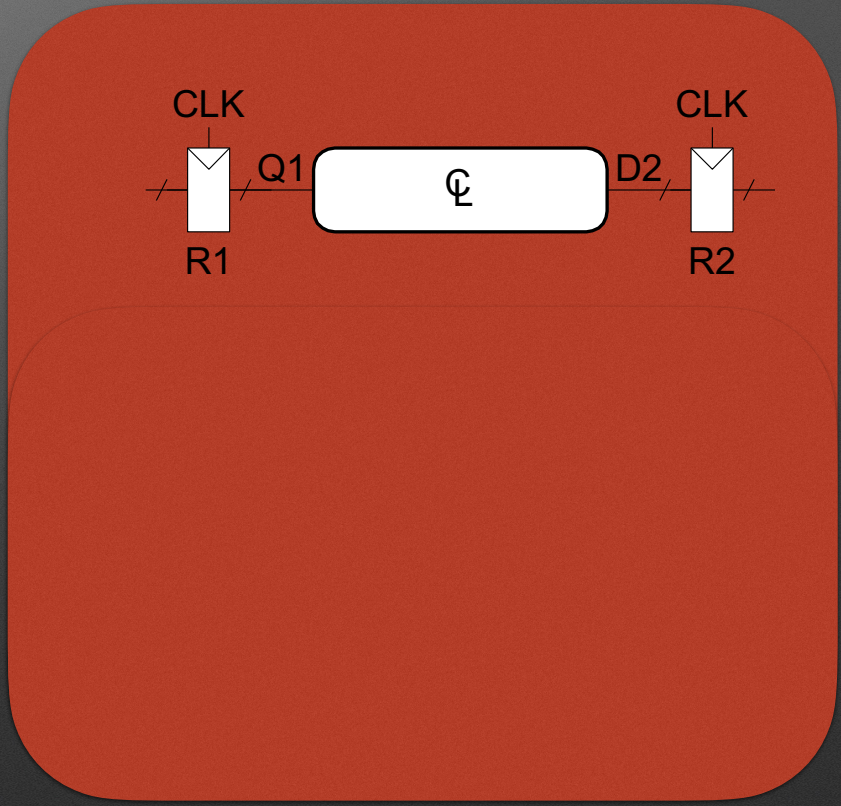
Setup Time Constraint

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



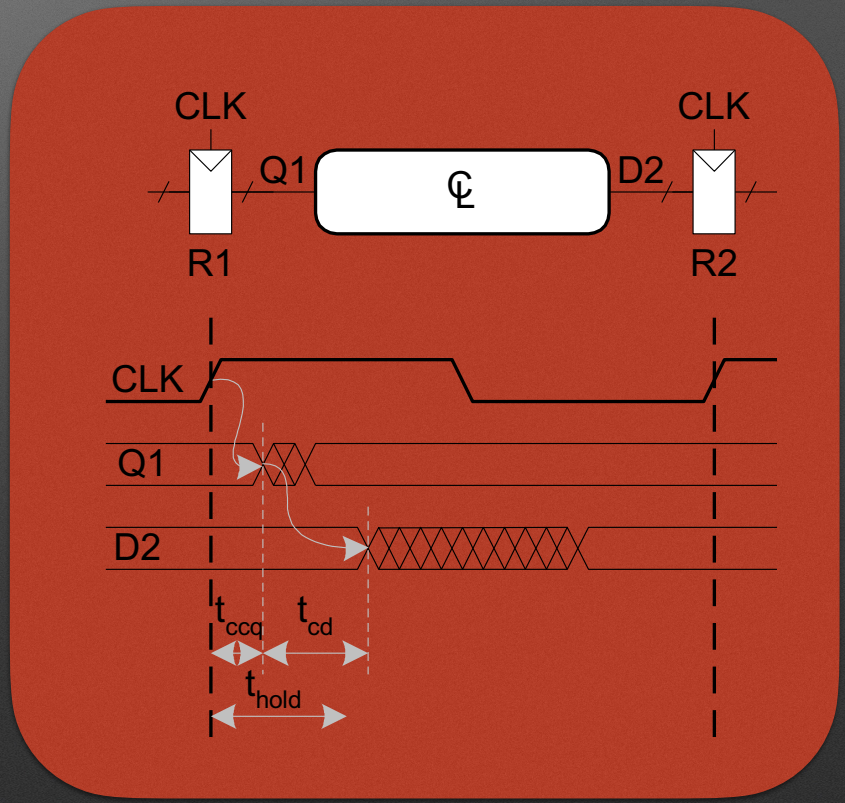
Hold Time Constraint

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

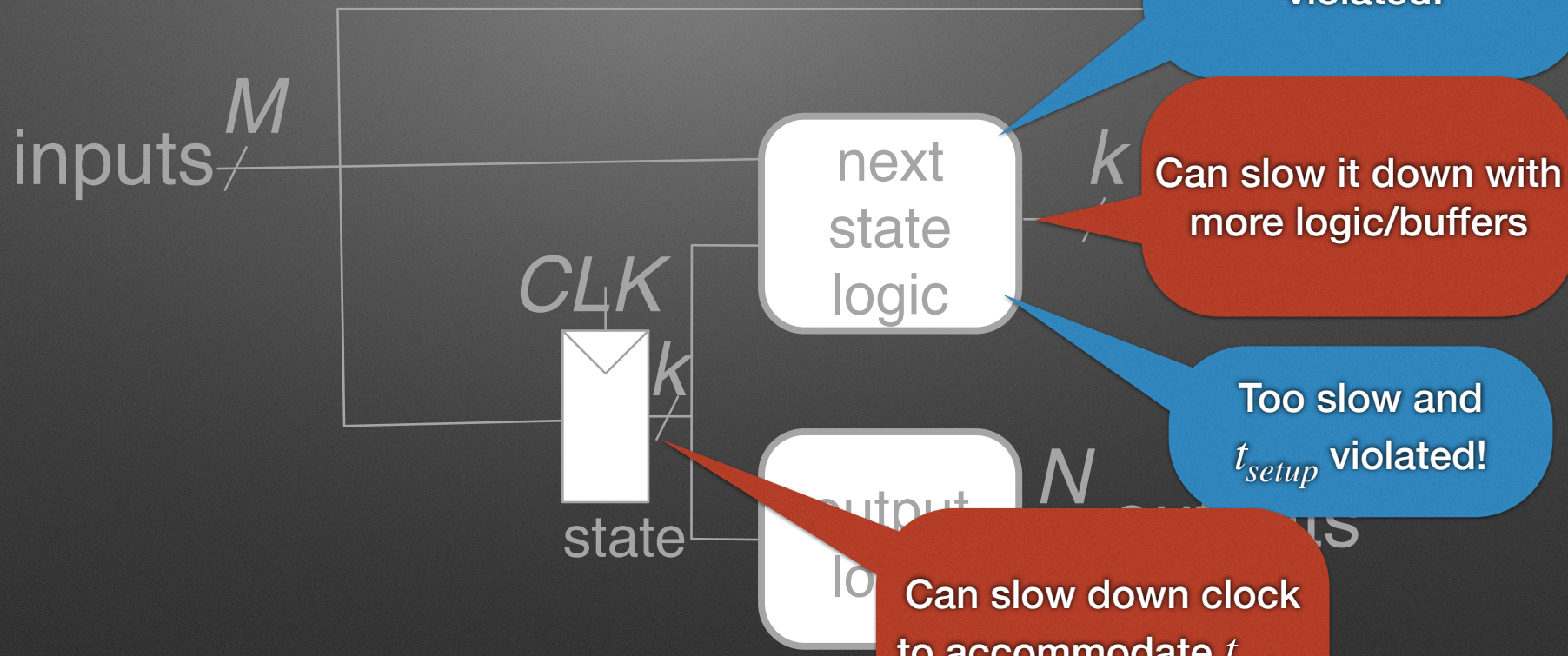


Hold Time Constraint

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



Review: Moore Machine Stru



Synchronous Timing

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

Next Time

- Studio