inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture #15
Synchronous Digital Systems II

2007-7-19

Scott Beamer, Instructor

Review of SDS

Synchronous - Means all operations are coordinated by a central clock.

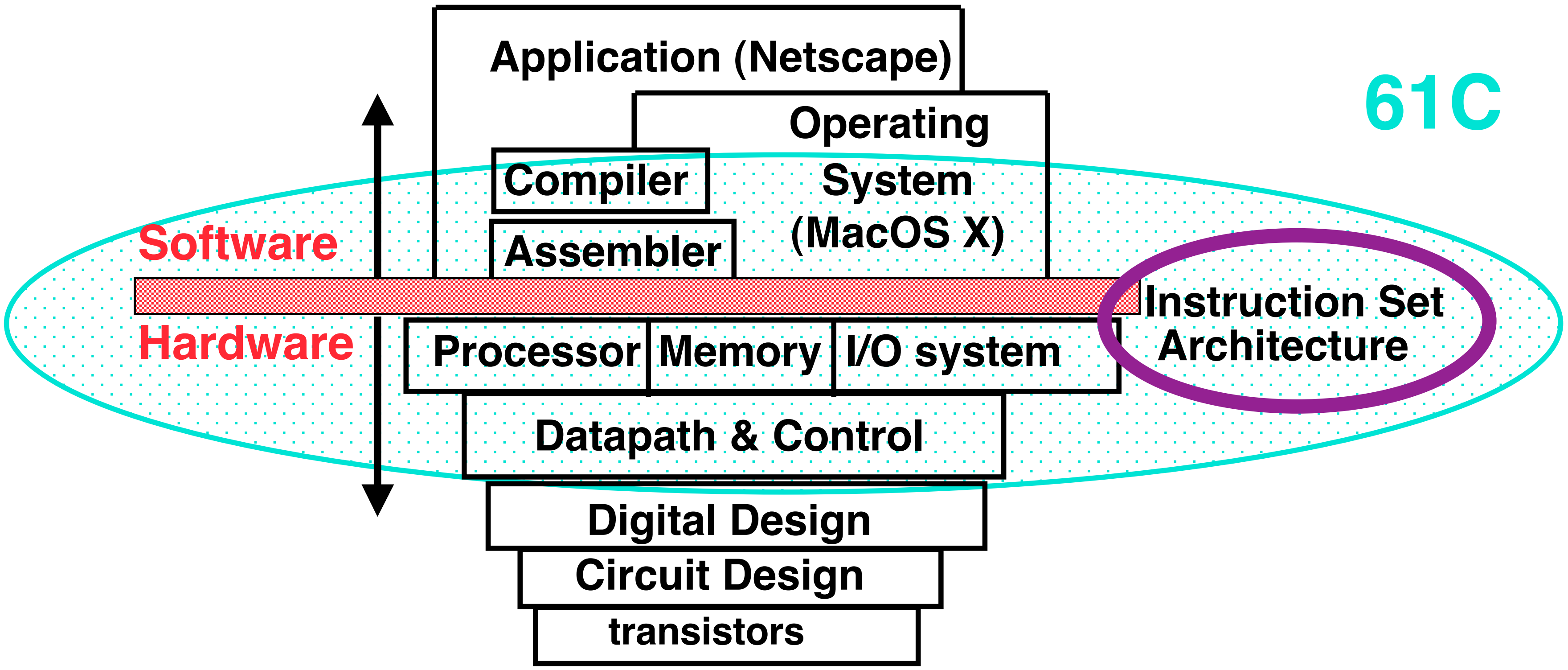
Digital - Means all values are represented by discrete values

- Electrical signals are treated as 1's and 0's and grouped together to form words.

Combinational Logic - functional block built only from logic gates

State Element - anything capable of storing data (ie registers, flip-flops…)

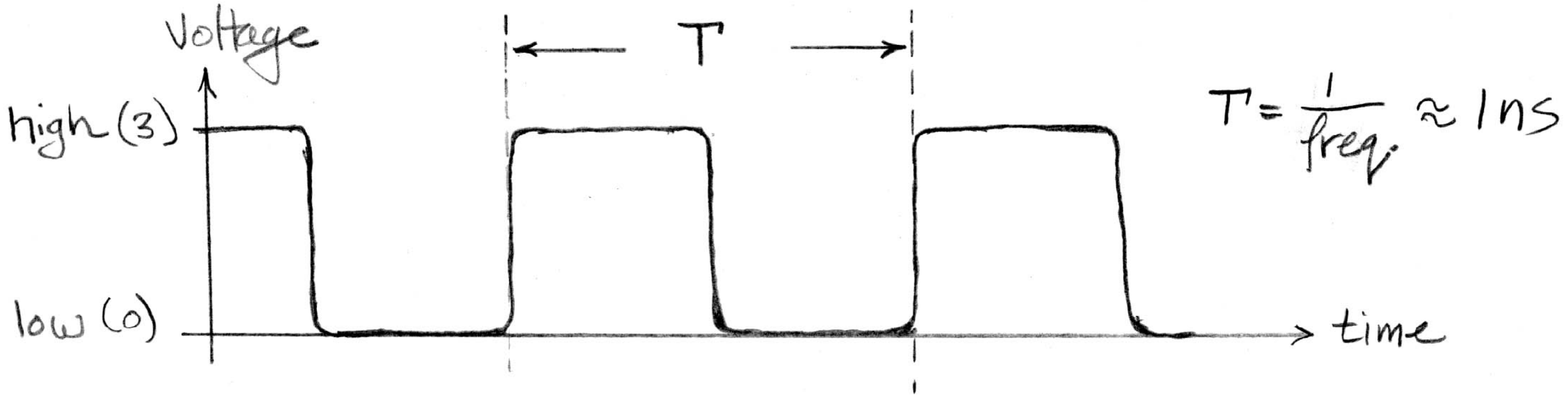
What are "Machine Structures"?



Coordination of many *levels of abstraction*

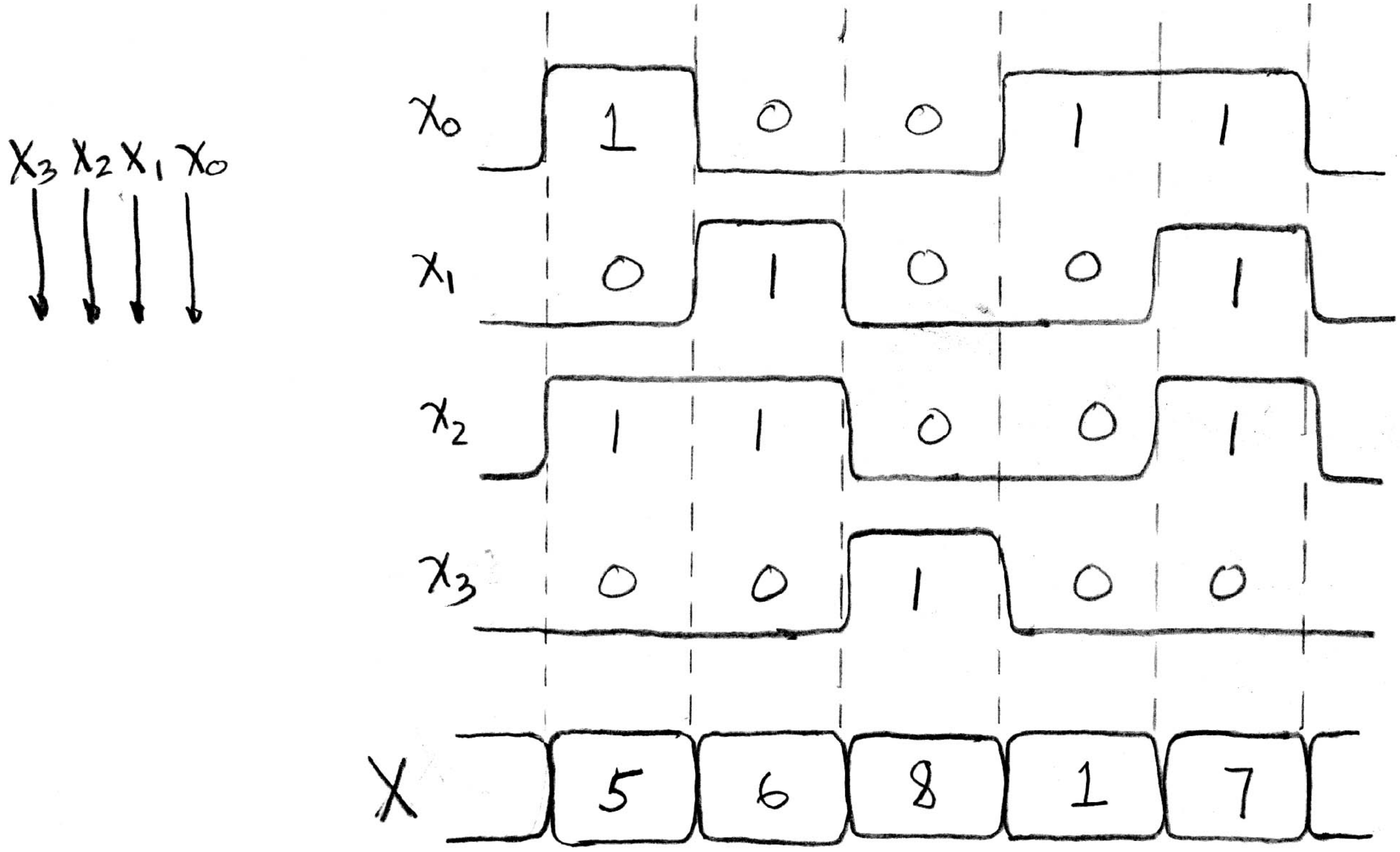
We'll investigate lower abstraction layers! (contract between HW & SW)

Signals and Waveforms: Clocks



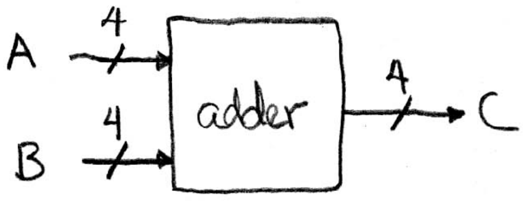
- **Signals**
  - **When digital (this class) is only treated as 1 or 0**
  - **Is transmitted over wires continuously**
  - **Transmission is effectively instant**
    - **Implies that any wire only contains 1 value at a time**

Signals and Waveforms: Grouping



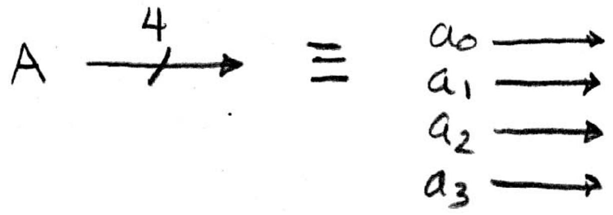
Bus - more than one signal treated as a unit

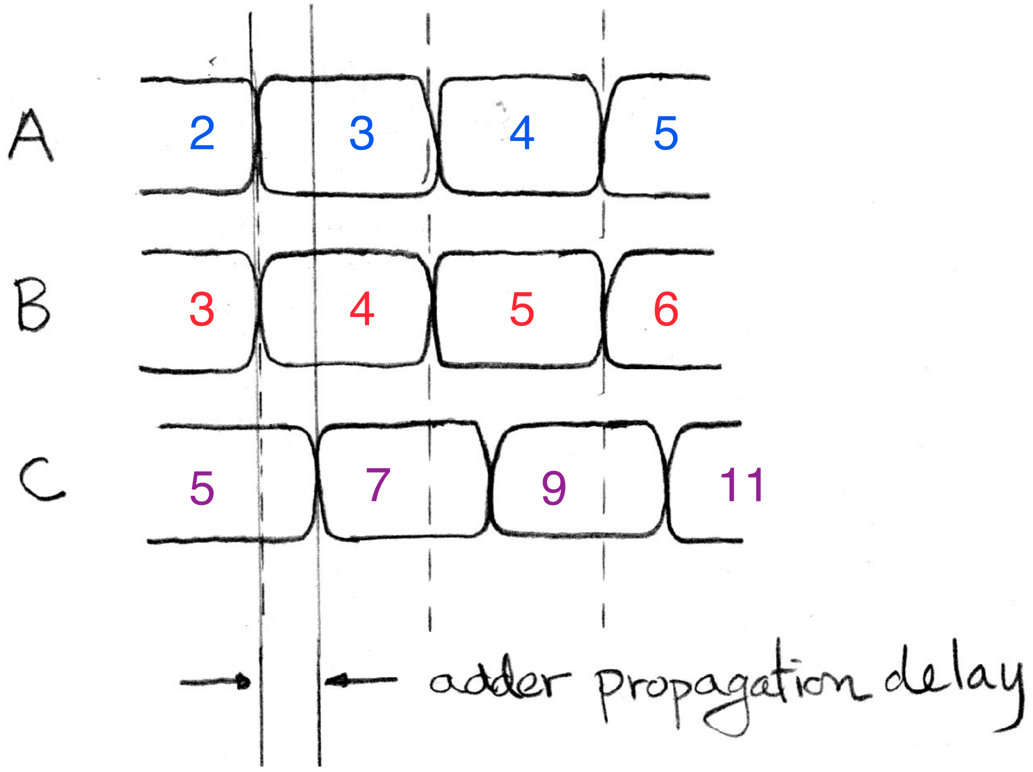
Signals and Waveforms: Circuit Delay



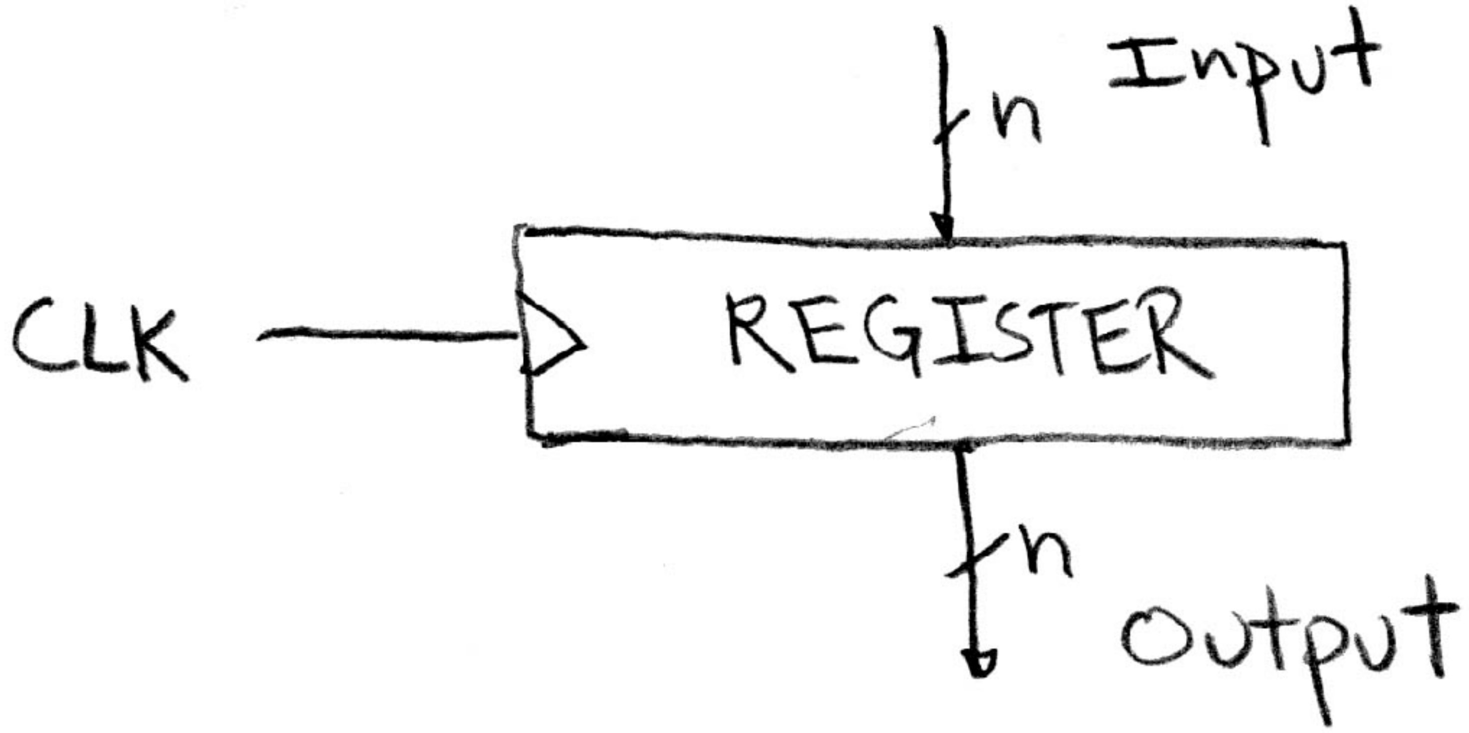
$A = [a_3, a_2, a_1, a_0]$

$B = [b_3, b_2, b_1, b_0]$



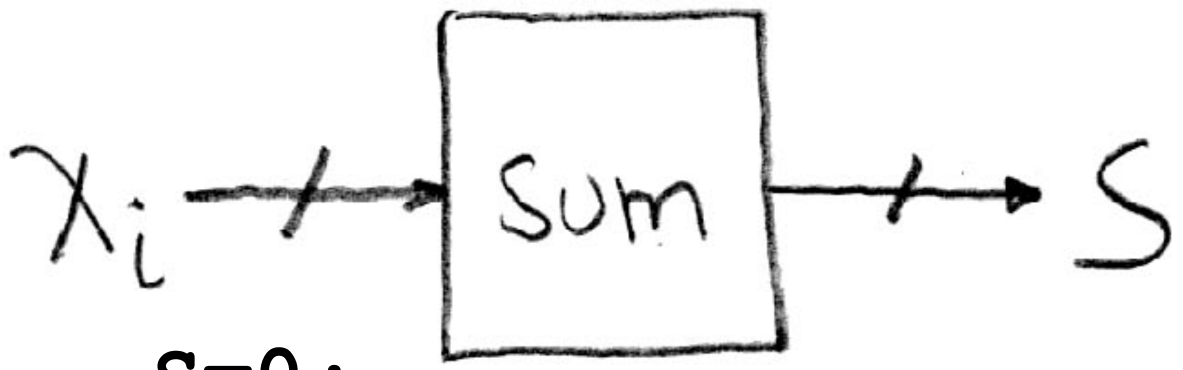


Circuits with STATE (e.g., register)



Accumulator Example

Why do we need to control the flow of information?



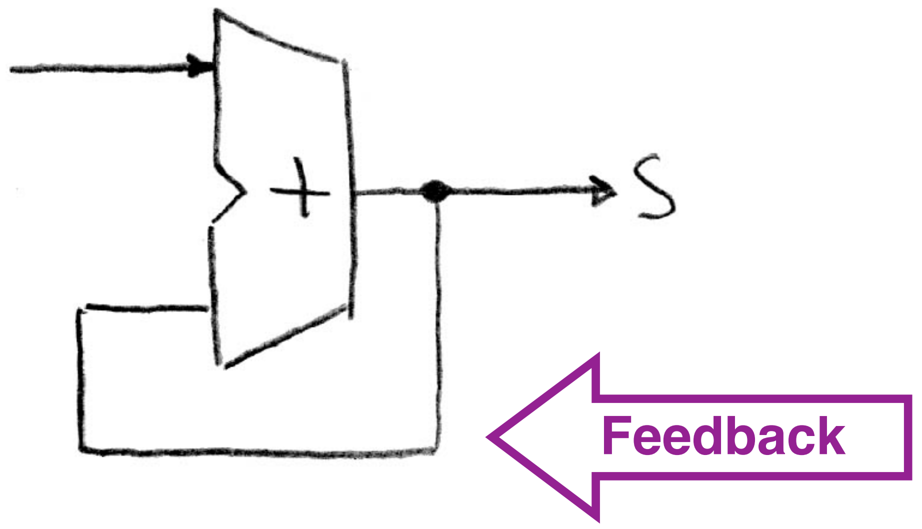
Want:

```
S=0;
for (i=0;i<n;i++)
   S = S + Xi
```

Assume:

- Each X value is applied in succession, one per cycle.
- After n cycles the sum is present on S.

First try…Does this work?

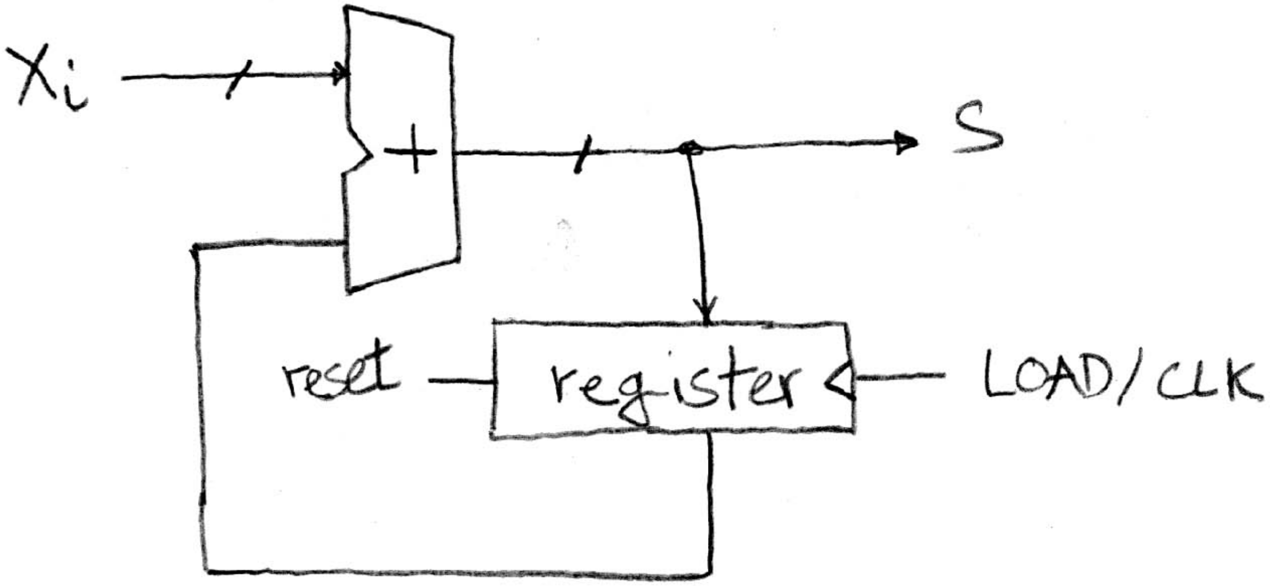


Nope!

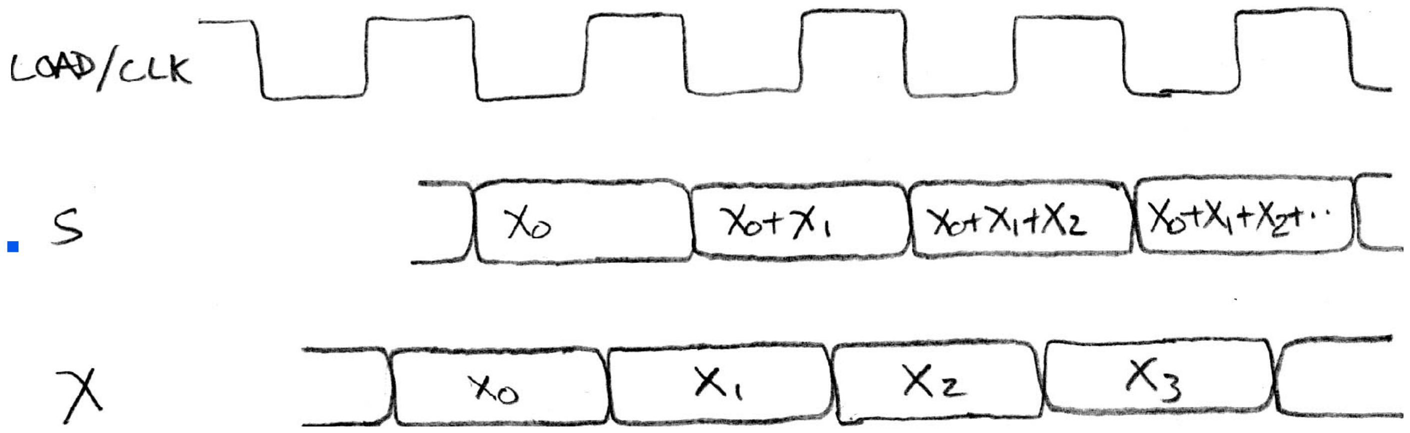
Reason #1… What is there to control the next iteration of the ‘`for`’ loop?

Reason #2… How do we say: ‘`S=0`’?

Second try…How about this?

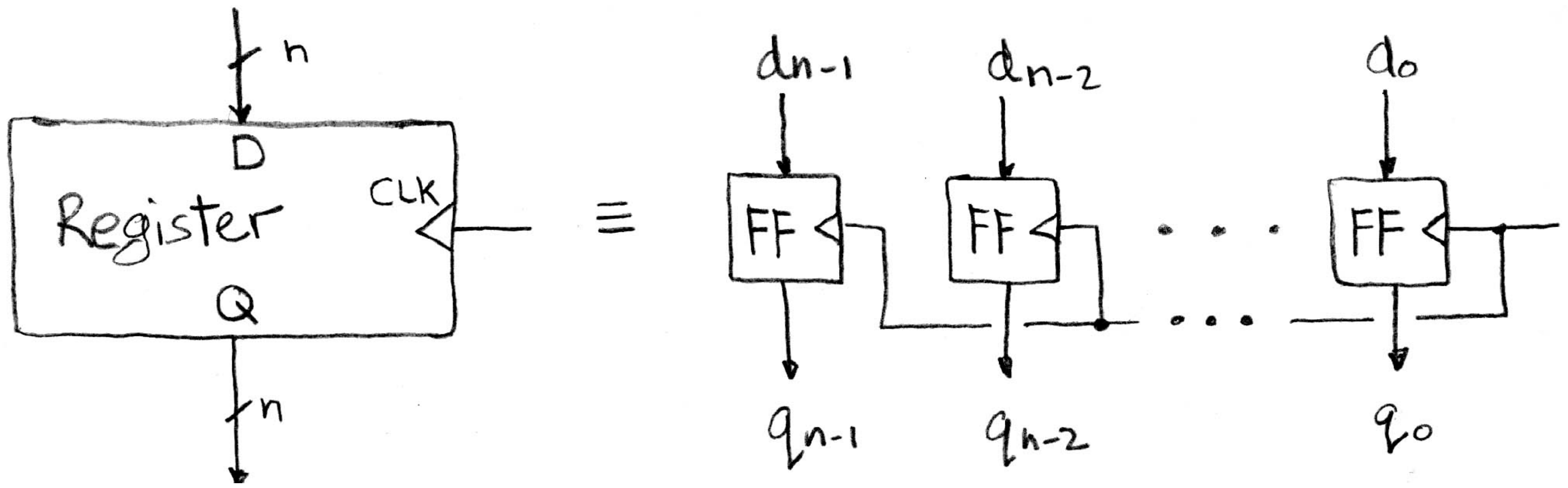


Rough timing…



Register is used to hold up the transfer of data to adder.

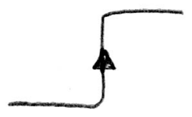
Register Details…What's inside?



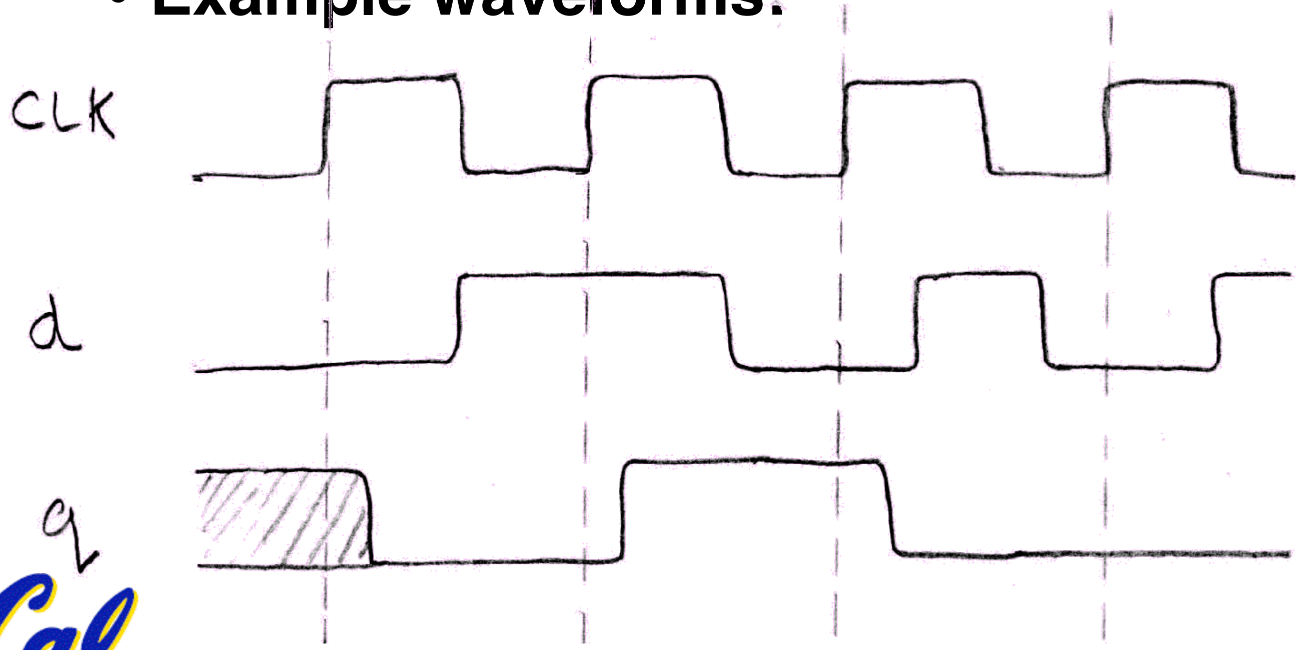
- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between and 0,1
- D is "data", Q is "output"
- Also called "d-type Flip-Flop"

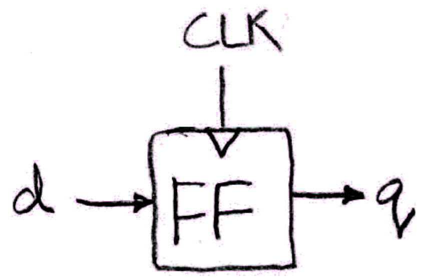
What's the timing of a Flip-flop? (1/2)

- Edge-triggered d-type flip-flop
  - This one is "positive edge-triggered"

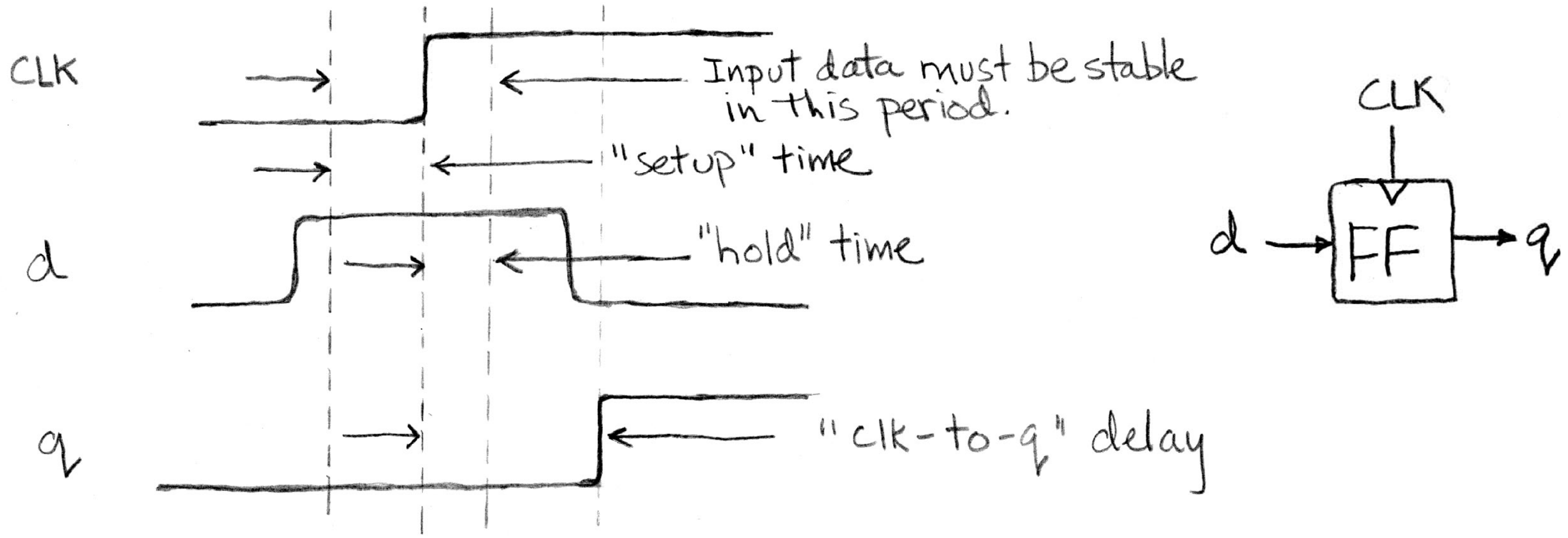


- "On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."
- Example waveforms:

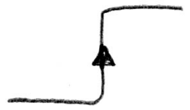




What’s the timing of a Flip-flop? (2/2)

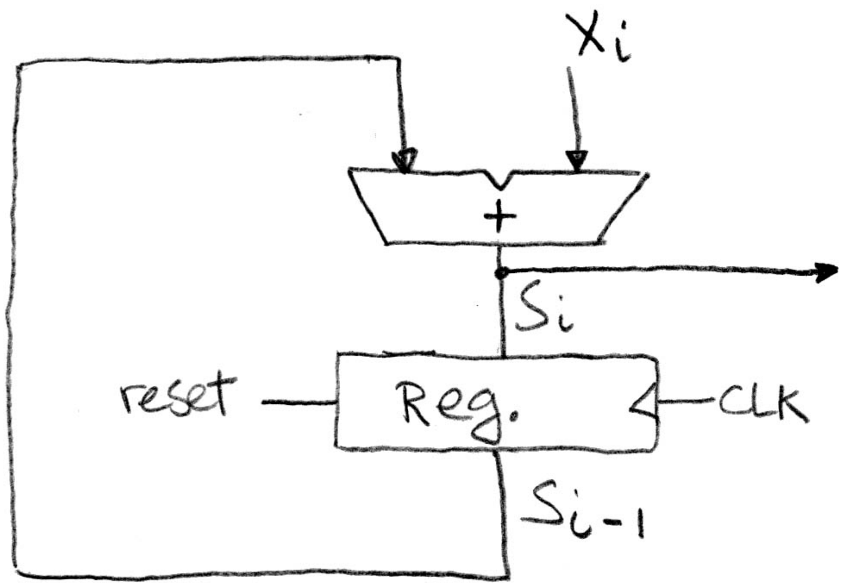


- **Edge-triggered d-type flip-flop**
  - **This one is “positive edge-triggered”**

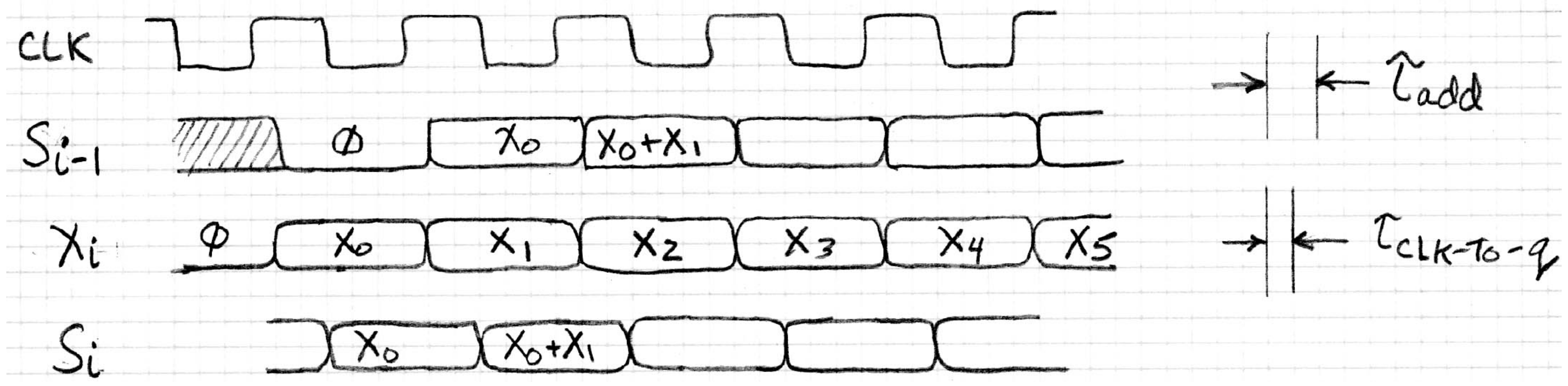


- **“On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”**

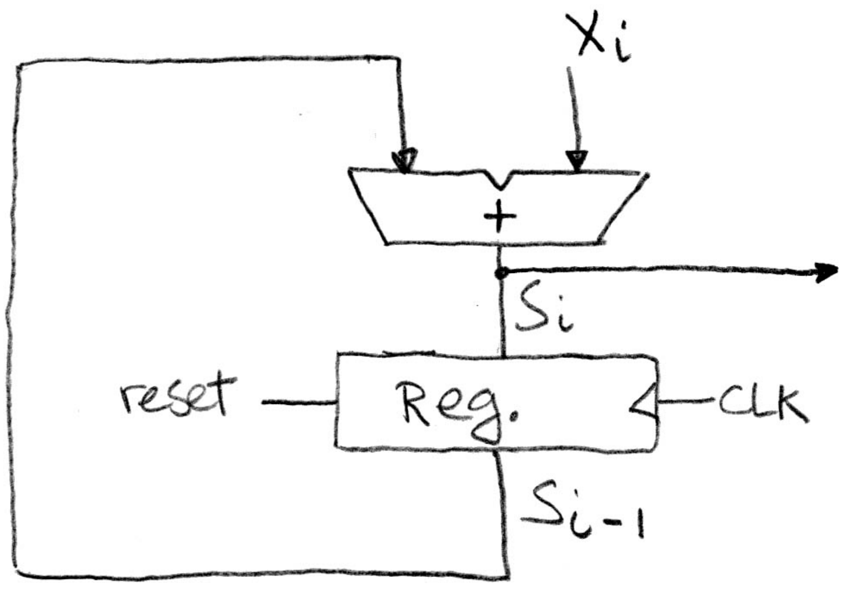
Accumulator Revisited (proper timing 1/2)



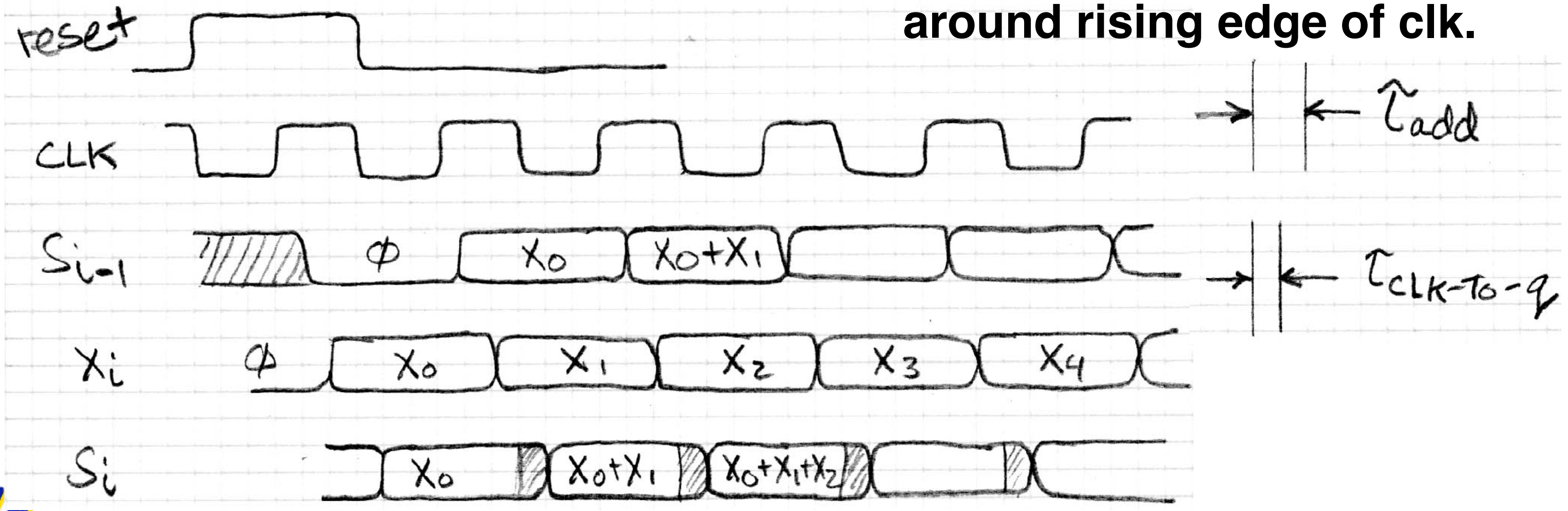
- Reset input to register is used to force it to all zeros (takes priority over D input).
- S_{i-1} holds the result of the ith-1 iteration.
- Analyze circuit timing starting at the output of the register.



Accumulator Revisited (proper timing 2/2)



- reset signal shown.
- Also, in practice X might not arrive to the adder at the same time as S_{i-1}
- S_i temporarily is wrong, but register always captures correct value.
- In good circuits, instability never happens around rising edge of clk.

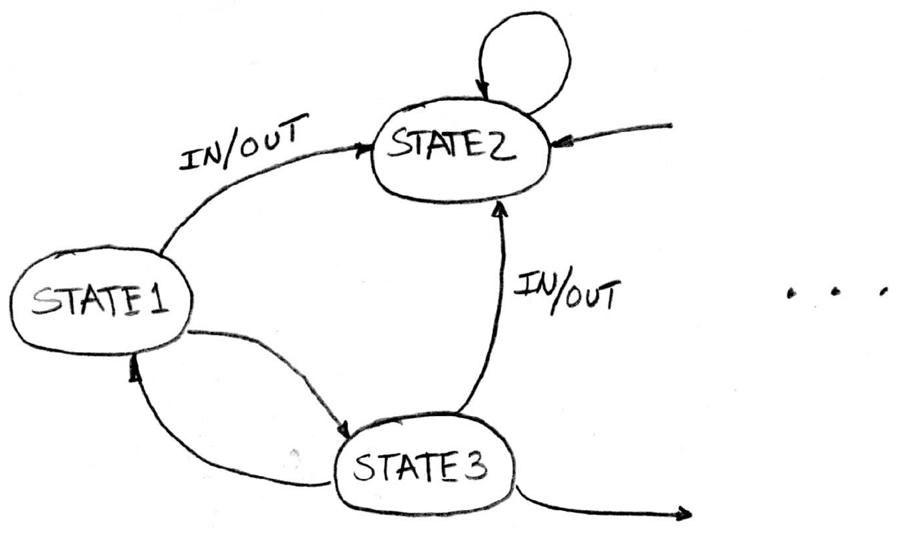


Administrivia

- Proj2 due Friday
- Midterm 7-10p on Monday in 10 Evans
- Midterm Review 11-2 on Friday, probably in 10 or 60 Evans
- Scott is not holding OH on Monday, but is holding extra OH on Friday 3-5
- Reading: For what the textbook lacks, there are handouts on the website

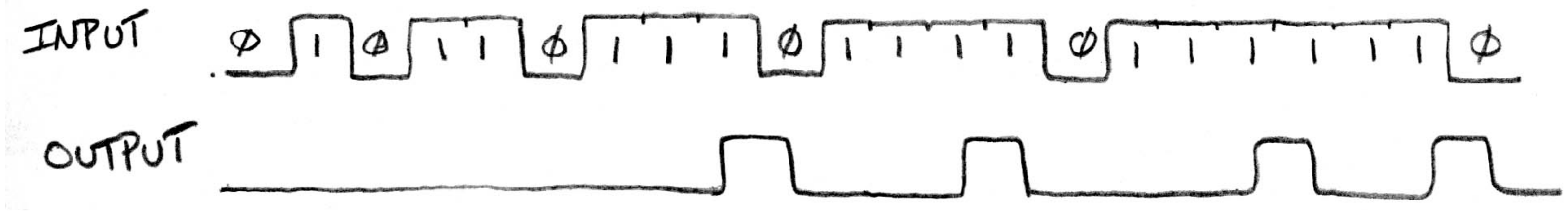
Finite State Machines (FSM) Introduction

- You have seen FSMs in other classes.
- Same basic idea.
- The function can be represented with a “state transition diagram”.
- With combinational logic and registers, any FSM can be implemented in hardware.

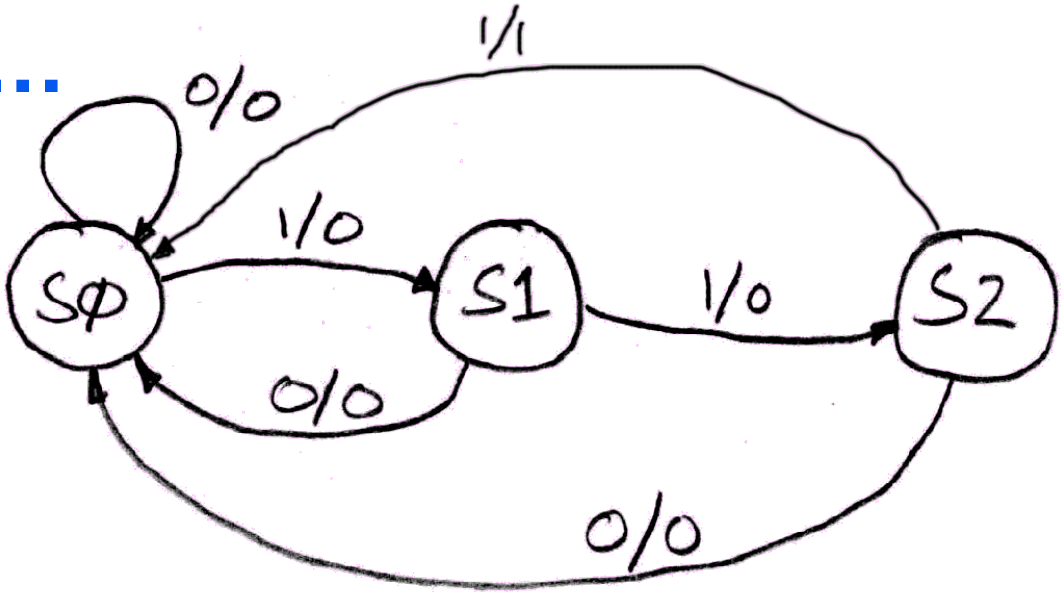


Finite State Machine Example: 3 ones…

FSM to detect the occurrence of 3 consecutive 1's in the input.



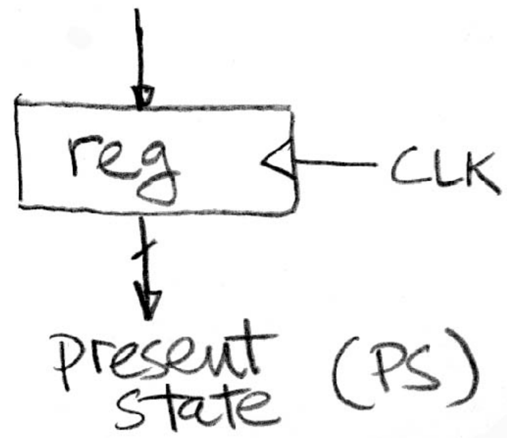
Draw the FSM…



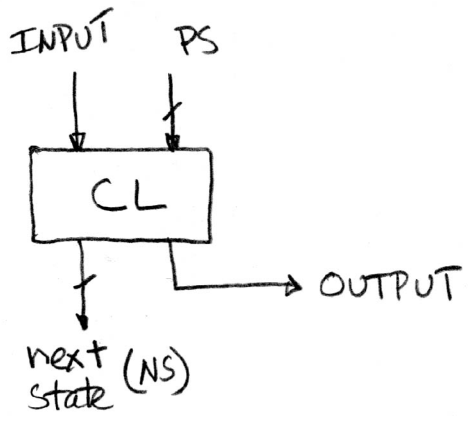
Assume state transitions are controlled by the clock:
on each clock cycle the machine checks the inputs and moves to a new state and produces a new output…

Hardware Implementation of FSM

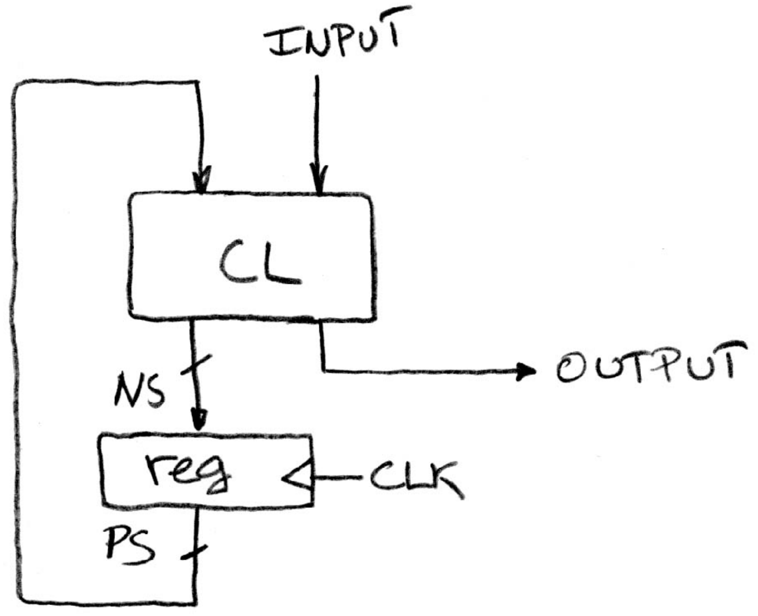
… Therefore a register is needed to hold the a representation of which state the machine is in. Use a unique bit pattern for each state.



+



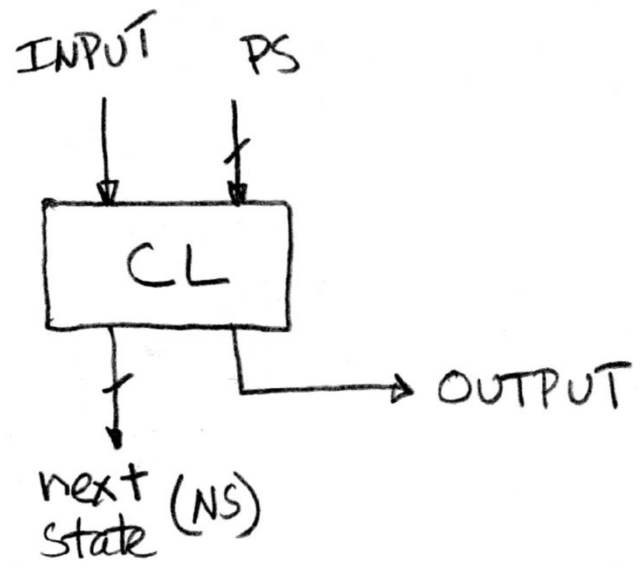
=



Combinational logic circuit is used to implement a function maps from *present state and input* to *next state and output.*

Hardware for FSM: Combinational Logic

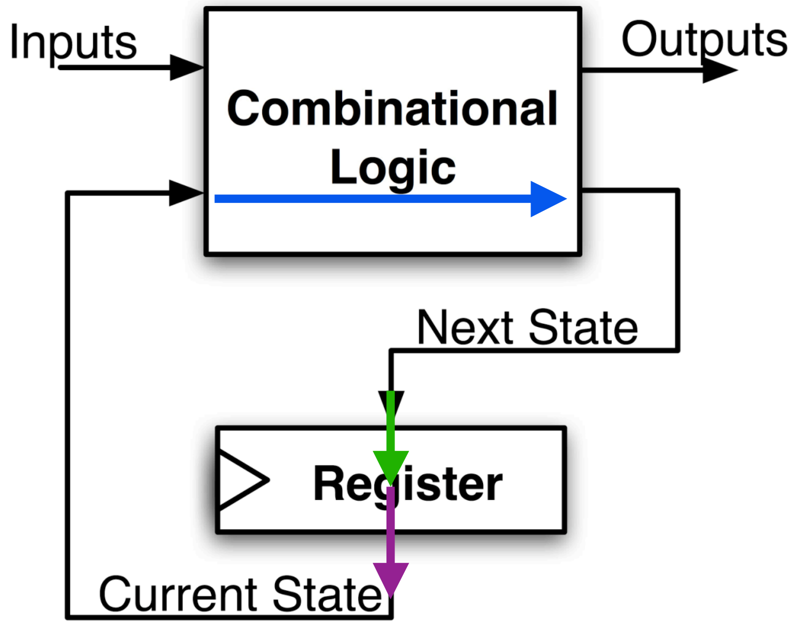
Next lecture we will discuss the detailed implementation, but for now can look at its functional specification, truth table form.



Truth table…

PS	Input	NS	Output
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1

Maximum Clock Frequency



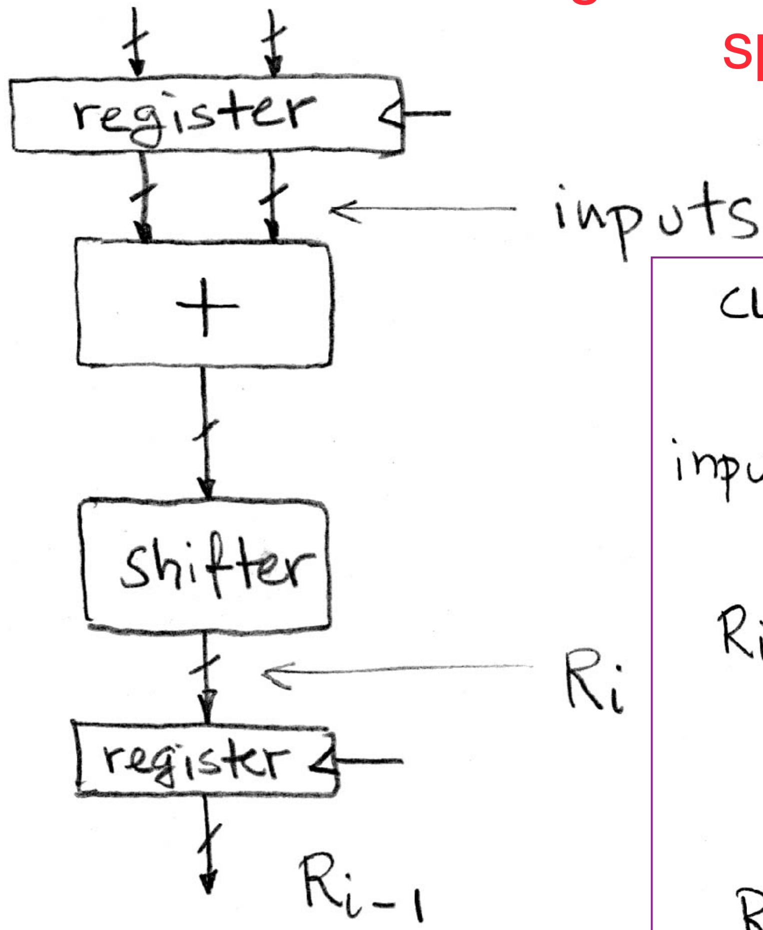
Hint…
Frequency = 1/Period

- **What is the maximum frequency of this circuit?**

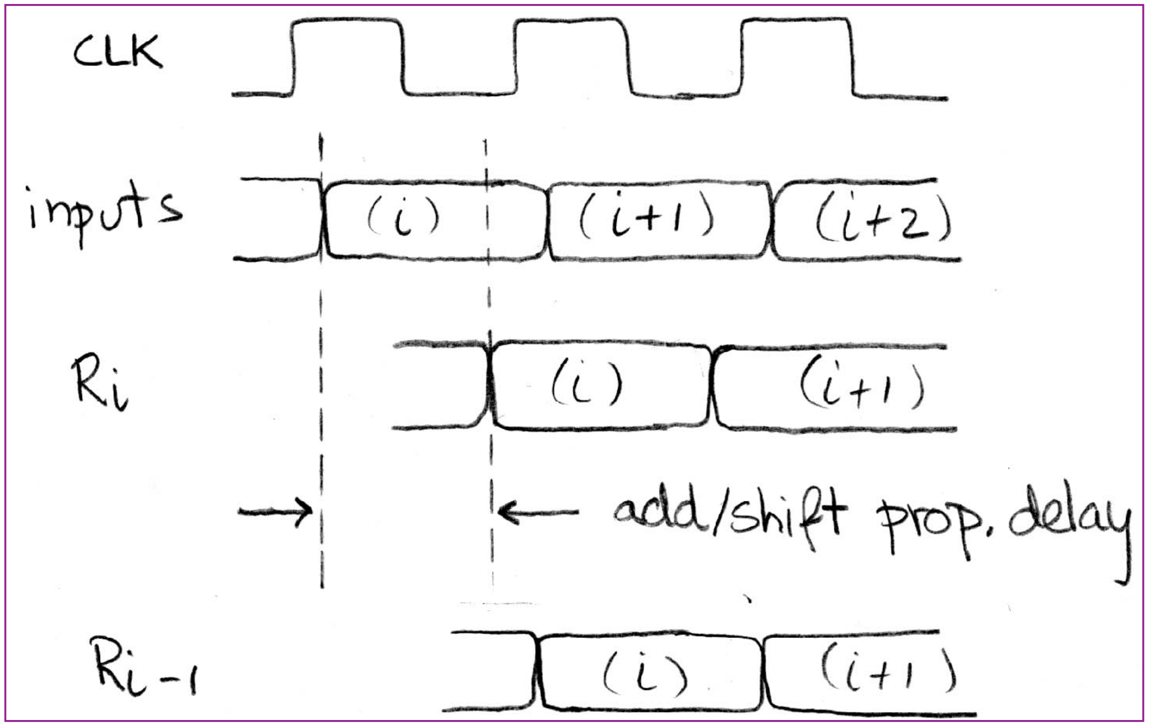
Max Delay = Setup Time + CLK-to-Q Delay + CL Delay

Pipelining to improve performance (1/2)

Extra Register are often added to help speed up the clock rate.

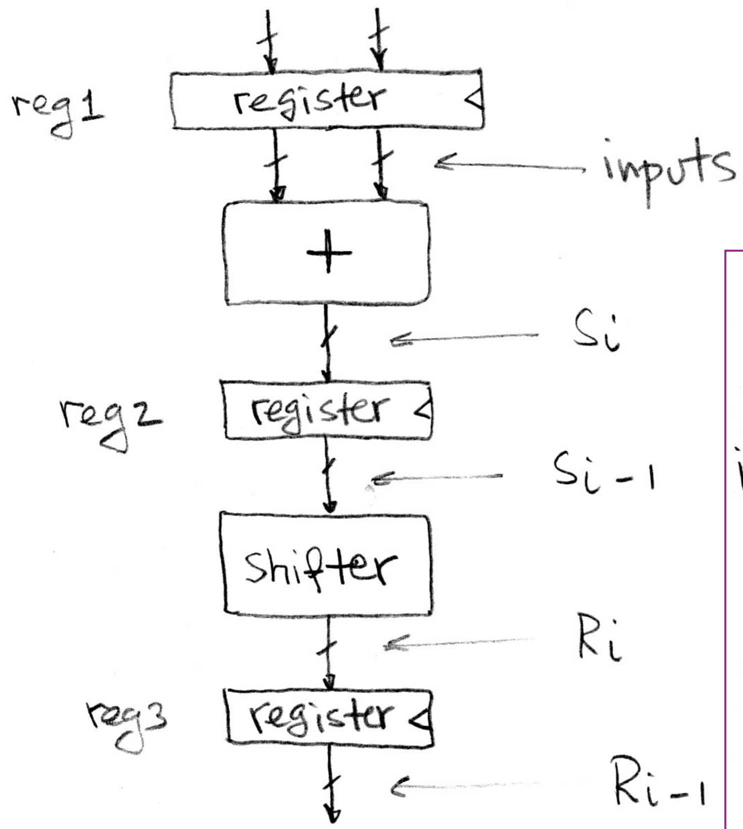


Timing…



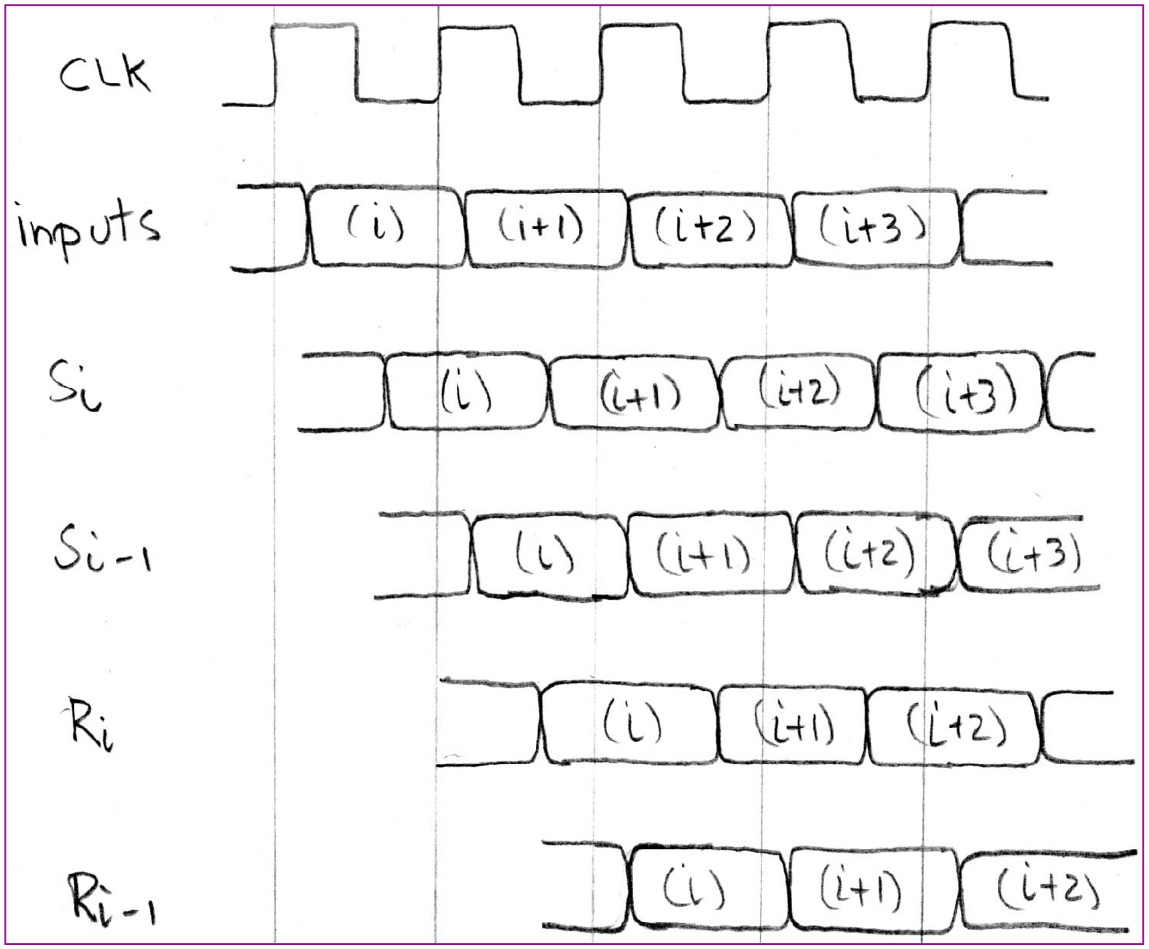
Note: delay of 1 clock cycle from input to output.
Clock period limited by propagation delay of adder/shifter.

Pipelining to improve performance (2/2)



- Insertion of register allows higher clock frequency.
- More outputs per second.

Timing…



Peer Instruction

A. HW feedback akin to SW recursion

B. We can implement a D-Q flipflop as simple CL (And, Or, Not gates)

C. You can build a FSM to signal when an equal number of 0s and 1s has appeared in the input.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Peer Instruction

A. It needs 'base case' (reg reset), way to step from i to i+1 (use register + clock).
- True!

B. D-Q has <u>state</u>, CL <u>never</u> has state!
- False!

C. How many states would it have? Say it's n. How does it know when n+1 bits have been seen?
- False!

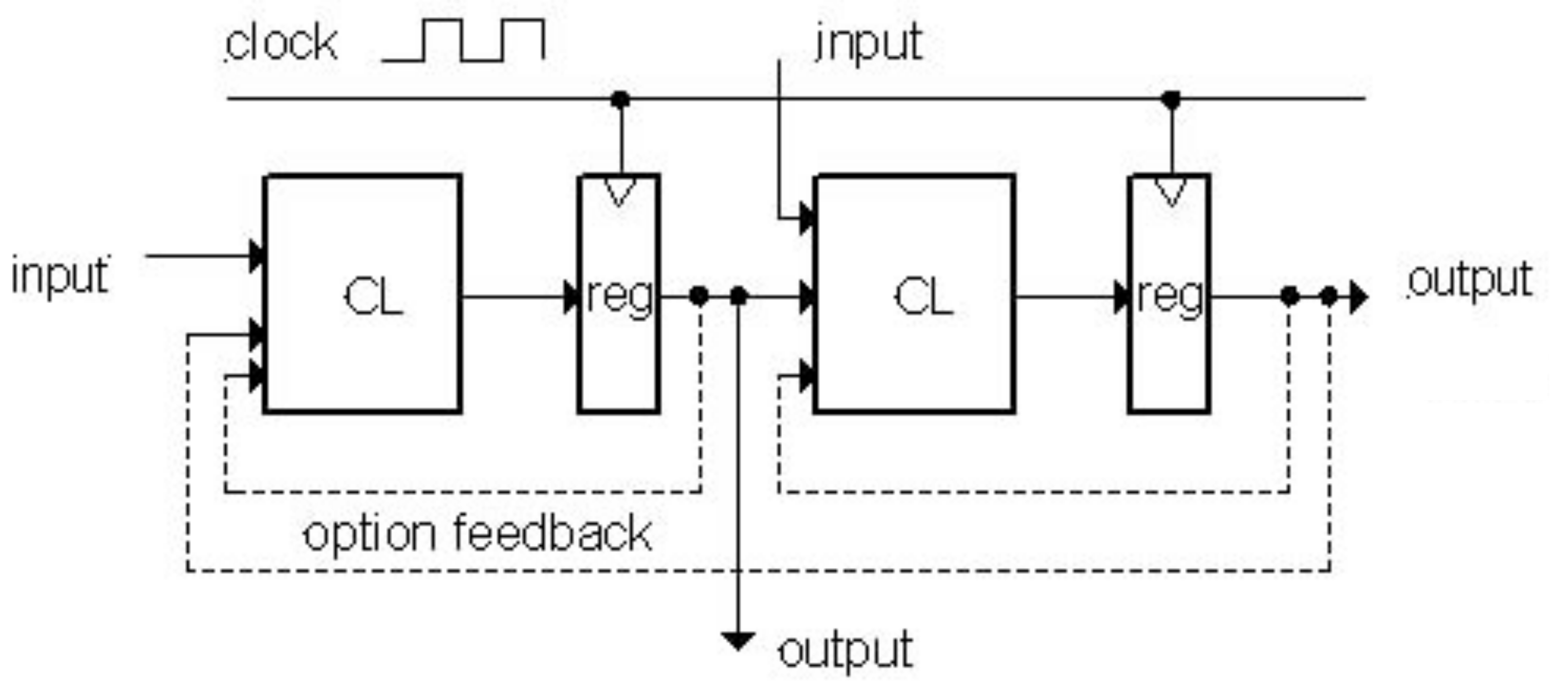
A. HW feedback akin to SW recursion

B. We can implement a D-Q flipflop as simple CL (And, Or, Not gates)

C. You can build a FSM to signal when an equal number of 0s and 1s has appeared in the input.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
[✓ 5:]	[✓ TFF]
6:	TFT
7:	TTF
8:	TTT

General Model for Synchronous Systems



- Collection of CL blocks separated by registers.
- Registers may be back-to-back and CL blocks may be back-to-back.
- Feedback is optional.
- Clock signal(s) connects only to clock input of registers.

"And In conclusion…"

- State elements are used to:
  - Build memories
  - Control the flow of information between other state elements and combinational logic
- D-flip-flops used to build registers
- Clocks tell us when D-flip-flops change
  - Setup and Hold times important
- We pipeline long-delay CL for faster clock
- Finite State Machines extremely useful
  - You'll see them again (150,152) & 164