
Computer Aided Digital Systems Design - EE 4743/6743

Sherif Abdelwahed

Sequential Logic Review

**Department of Electrical and Computer Engineering
Mississippi State University**

Combinational vs. Sequential Systems

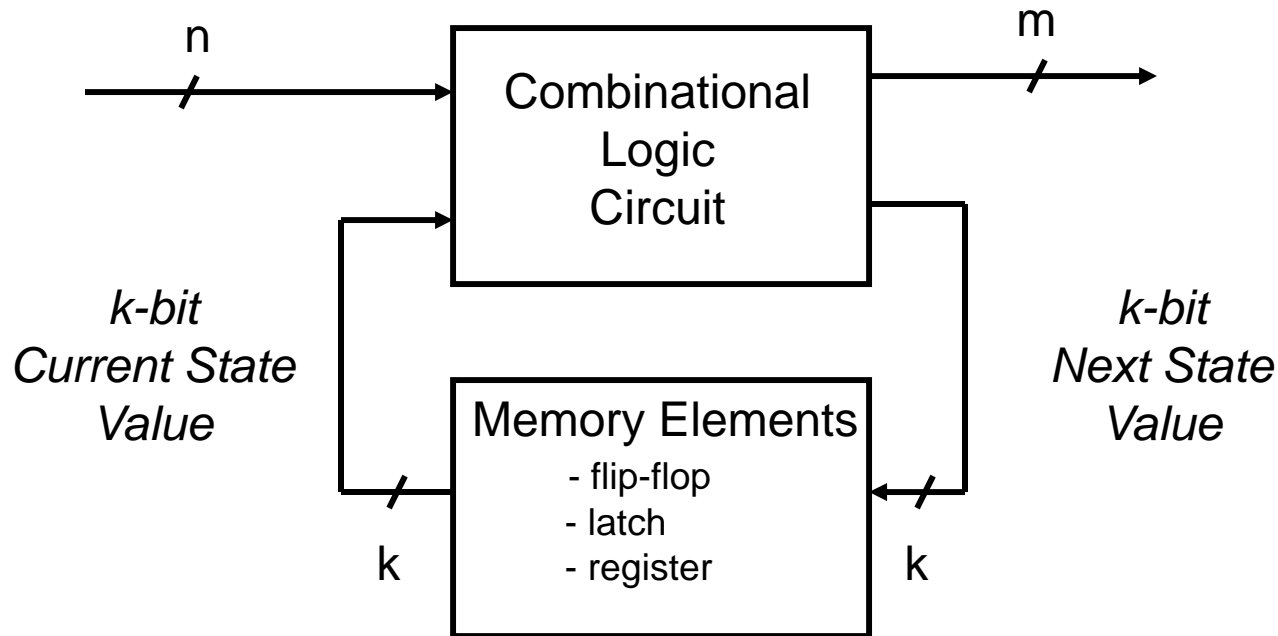
- **Combinational network**

- Output value only depends on input value

- **Sequential network**

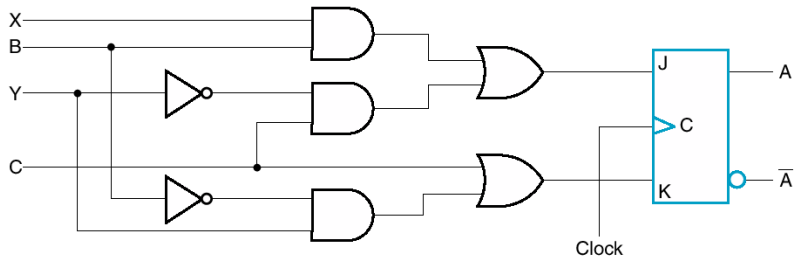
- Output Value depends on input value and present state value
 - Sequential network must have some way of retaining state via memory devices.
 - Use a clock signal in a synchronous sequential system to control changes between states
-

Sequential System Diagram

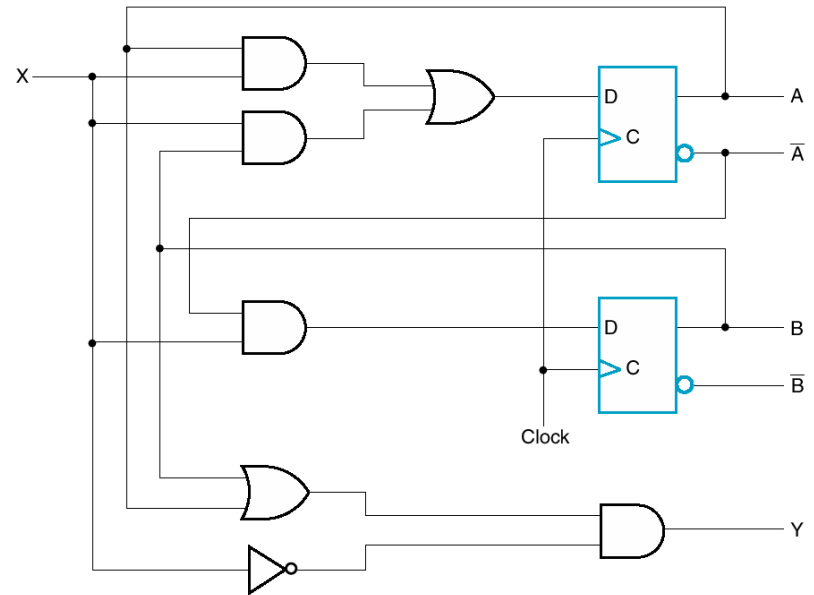


- m outputs only depend on k current state bits - Moore Machine
 - REMEMBER: Moore is Less !!
 - m outputs depend on k current state bits AND n inputs - Mealy Machine
-

Sequential Circuit Types

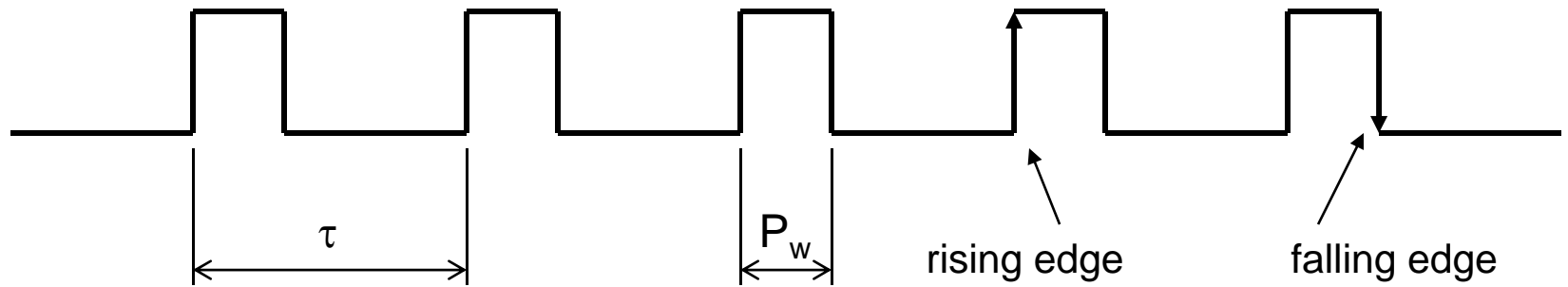


Moore model: outputs depend on states only.

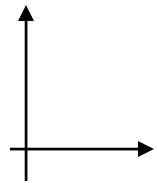


Mealy model: outputs depend on inputs & states

Clock Signal Review



voltage



time

τ - period (in seconds)

P_w - pulse width (in seconds)

f - frequency pulse width (in Hertz)

$$f = 1/\tau$$

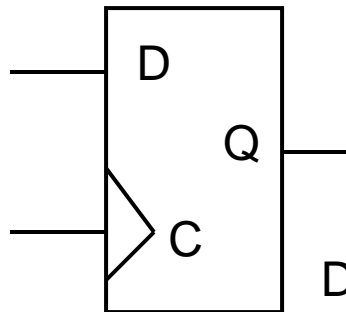
duty cycle - ratio of pulse width to period (in %)

$$\text{duty cycle} = P_w / \tau$$

millisecond (ms) 10^{-3}	Kilohertz (KHz) 10^3
microsecond (μ s) 10^{-6}	Megahertz (MHz) 10^6
nanosecond (ns) 10^{-9}	Gigahertz (GHz) 10^9

Memory Elements

Memory elements used in sequential systems are flip-flops and latches.

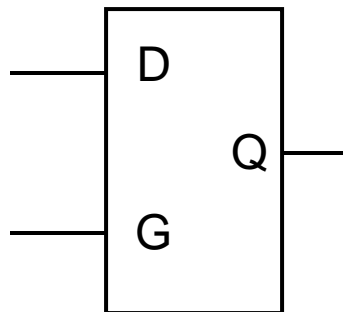


D flip flop (DFF)

D	Q(t+1)
0	0
1	1

Q(t+1) or Q+ is Q next state

Flip-flops are edge triggered (either rising or falling edge).



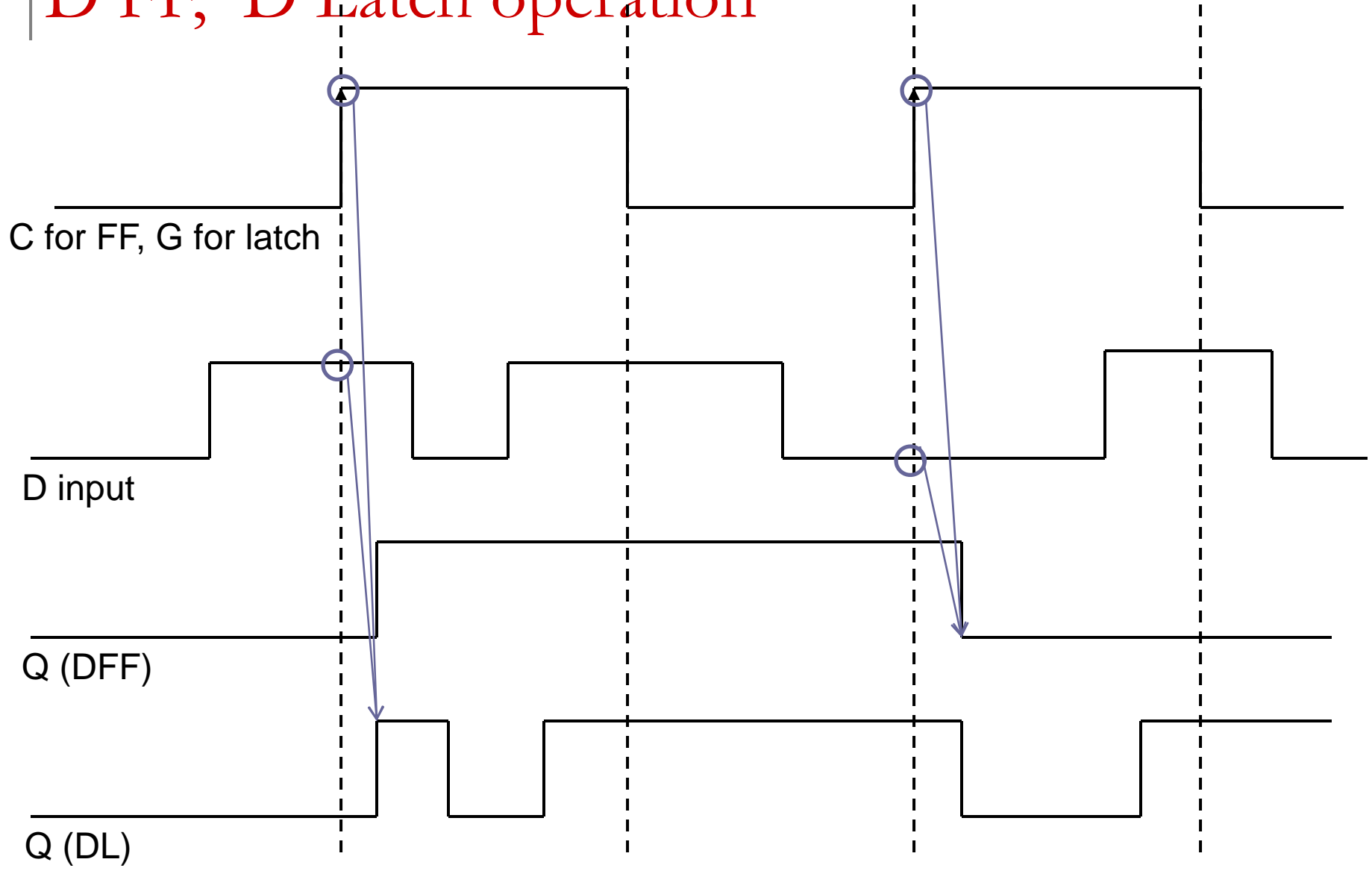
D latch (DL)

Latches are level sensitive.

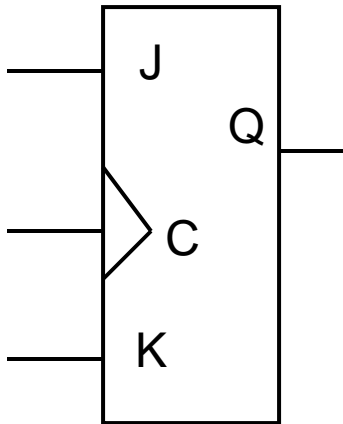
Q follows D when G=1, latches when G goes from 1 to 0.

Quiz: which design uses more gates?

D FF, D Latch operation

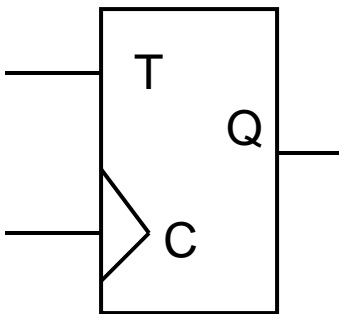


Other State Elements



J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

JK FF: used for single bit flags with separate set(J) and reset(K) control



T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

TFF: Used for counter design

DFFs are most common

- Most FPGA families only have DFFs
 - DFF is fastest, simplest (fewest transistors) of FFs
 - Other FF types (T, JK) can be built from DFFs
 - Latches are used to build Registers (using the Master-Slave Configuration), but are almost NEVER used by itself in a standard digital design flow.
 - Quite often, latches are inserted in the design by mistake (e.g., an error in your Verilog code). Make sure you understand the difference between the two.
 - We will use DFFs almost exclusively in this class
 - Will always use edge-triggered state elements (FFs), not level sensitive elements (latches).
-

Sequential Circuit Design

- Remember that a synchronous sequential circuit is made up of flip flops and combinational gates.
 - Part of the design is to choose the flip-flop type and combinational circuit structure which, together with the flip-flops produce a circuit that fulfills the stated specification.
 - How many FFs?
 - The number of flip-flops is determined from the number of states in the circuit
 - n flip-flops can represent up to 2^n binary states.
-

Realizing Circuits with Different Kinds of FF

Characteristic Equations

R-S: $Q^+ = S + \bar{R} Q$

D: $Q^+ = D$

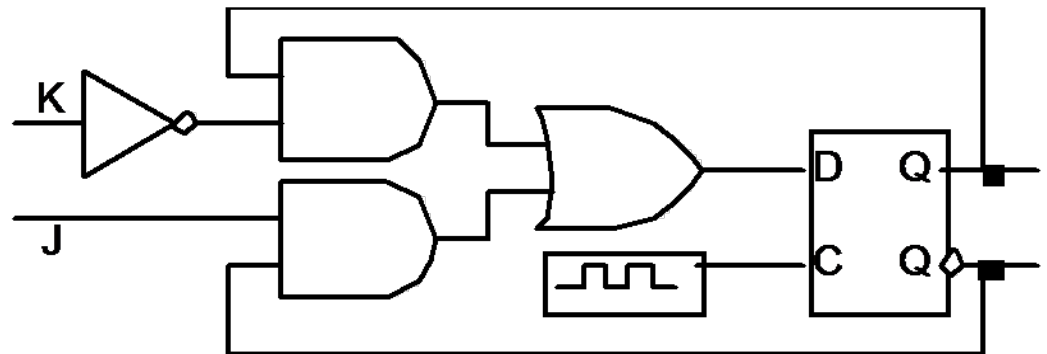
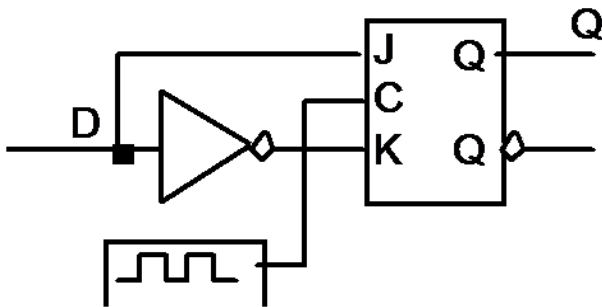
J-K: $Q^+ = J \bar{Q} + \bar{K} Q$

T: $Q^+ = T \bar{Q} + \bar{T} Q$

Derived from the K-maps
for $Q^+ = f(\text{Inputs}, Q)$

E.g., J=0, K=0, then $Q^+ = Q$
J=1, K=0, then $Q^+ = 1$
J=0, K=1, then $Q^+ = \underline{0}$
J=1, K=1, then $Q^+ = \bar{Q}$

Implementing One FF in Terms of Another



Excitation Tables

JK Flip Flop			
J	K	Q(t+1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

→

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

T Flip Flop		
T	Q(t+1)	
0	Q(t)	No change
1	Q'(t)	Complement

→

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Design Procedure

Start with the excitation table

Q	Q ⁺	J	K	T	D
0	0	0	X	0	0
0	1	1	X	1	1
1	0	X	1	1	0
1	1	X	0	0	1

Implementing D FF with a J-K FF:

- 1) Start with K-map of $Q^+ = f(D, Q)$
- 2) Create K-maps for J and K with same inputs (D, Q)
- 3) Fill in K-maps with appropriate values for J and K to cause the same state changes as in the original K-map

D \ Q	0	1
0	0	1
1	0	1

$Q^+ = D$

E.g., $D = Q = 0$, $Q^+ = 0$
then $J = 0$, $K = X$

D \ Q	0	1
0	0	1
1	X	X

$J = D$

D \ Q	0	1
0	X	X
1	1	0

$K = \bar{D}$

Design Procedure

Implementing J-K FF with a D FF:

1. K-Map of $Q^+ = F(J, K, Q)$
2. Revised K-map using D's excitation table its the same! that is why design procedure with D FF is simple!

Q	Q ⁺	J	K	T	D
0	0	0	X	0	0
0	1	1	X	1	1
1	0	X	1	1	0
1	1	X	0	0	1

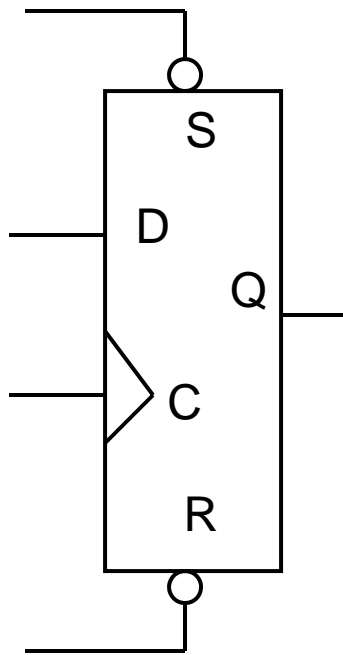
Q	JK		J	
	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$Q^+ = D = J\bar{Q} + \bar{K}Q$$

Resulting equation is the combinational logic input to D to cause same behavior as J-K FF. Of course it is identical to the characteristic equation for a J-K FF.

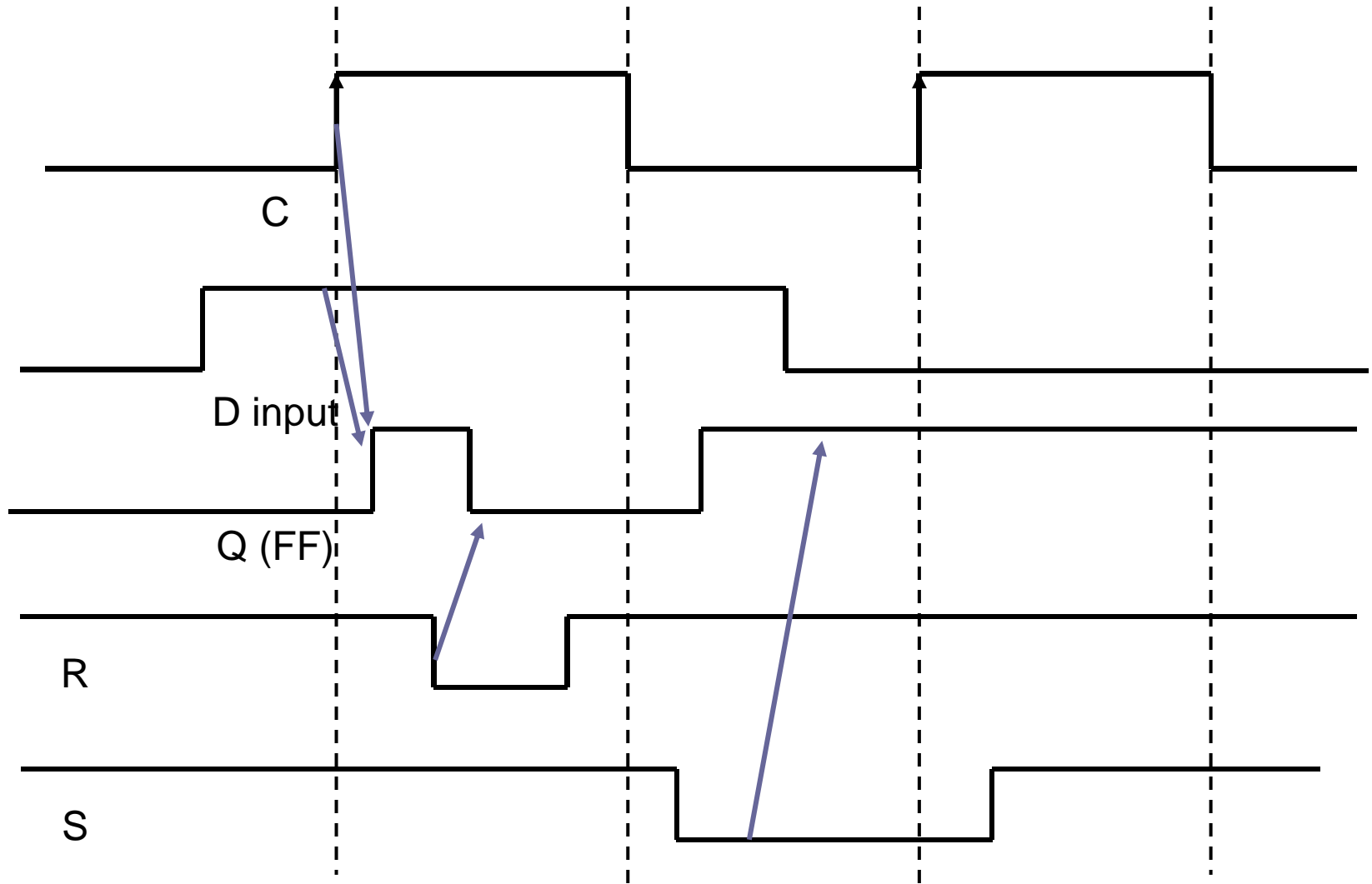
Synchronous vs. Asynchronous Inputs

- **Synchronous input:** Output will change after active clock edge
- **Asynchronous input:** Output changes independent of clock



- State elements often have async set, reset control
- D input is synchronous with respect to Clk
- S, R are asynchronous. Q output affected by S, R independent of Clk
- Async. inputs are dominant over Clk.
- Asynchronous inputs are **dangerous**, since they take effect immediately, glitches can be disastrous
- Synchronous inputs are greatly preferred!

D FF with async control



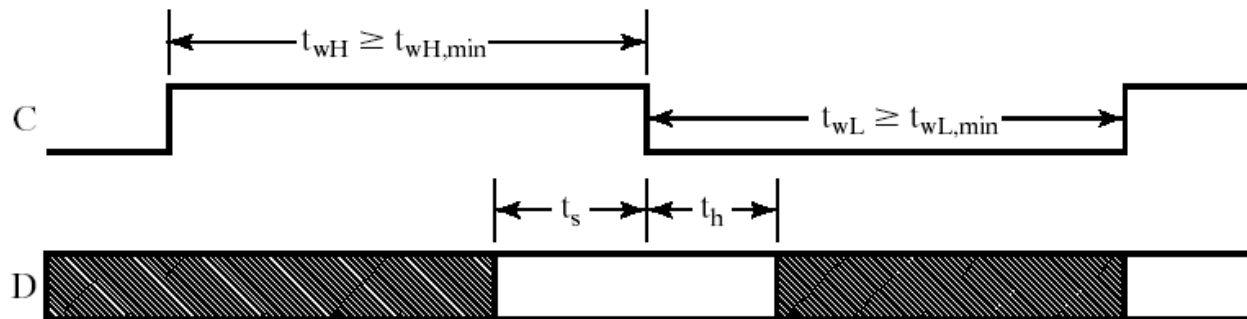
FF Timing

- Propagation Delay

- C2Q: Q will change some propagation delay after change in C. Value of Q is based on D input for DFF.
 - S2Q, R2Q: Q will change some propagation delay after change on S input, R input
 - Note that there is NO propagation delay D2Q for DFF.
 - D is a Synchronous INPUT, no prop delay value for synchronous inputs
-

Setup and Hold Times

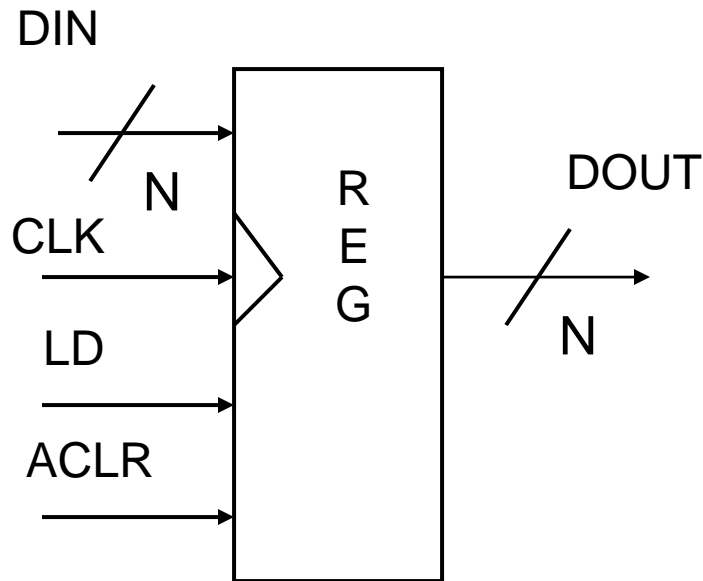
- Synchronous inputs (e.g. D) have Setup, Hold time specification with respect to the CLOCK input
- Setup Time: the amount of time the synchronous input (D) must be stable **before** the active edge of clock
- Hold Time: the amount of time the synchronous input (D) must be stable **after** the active edge of clock



- If changes on D input violate either setup or hold time, then correct FF operation is not guaranteed
- Setup/Hold measured around active clock edge

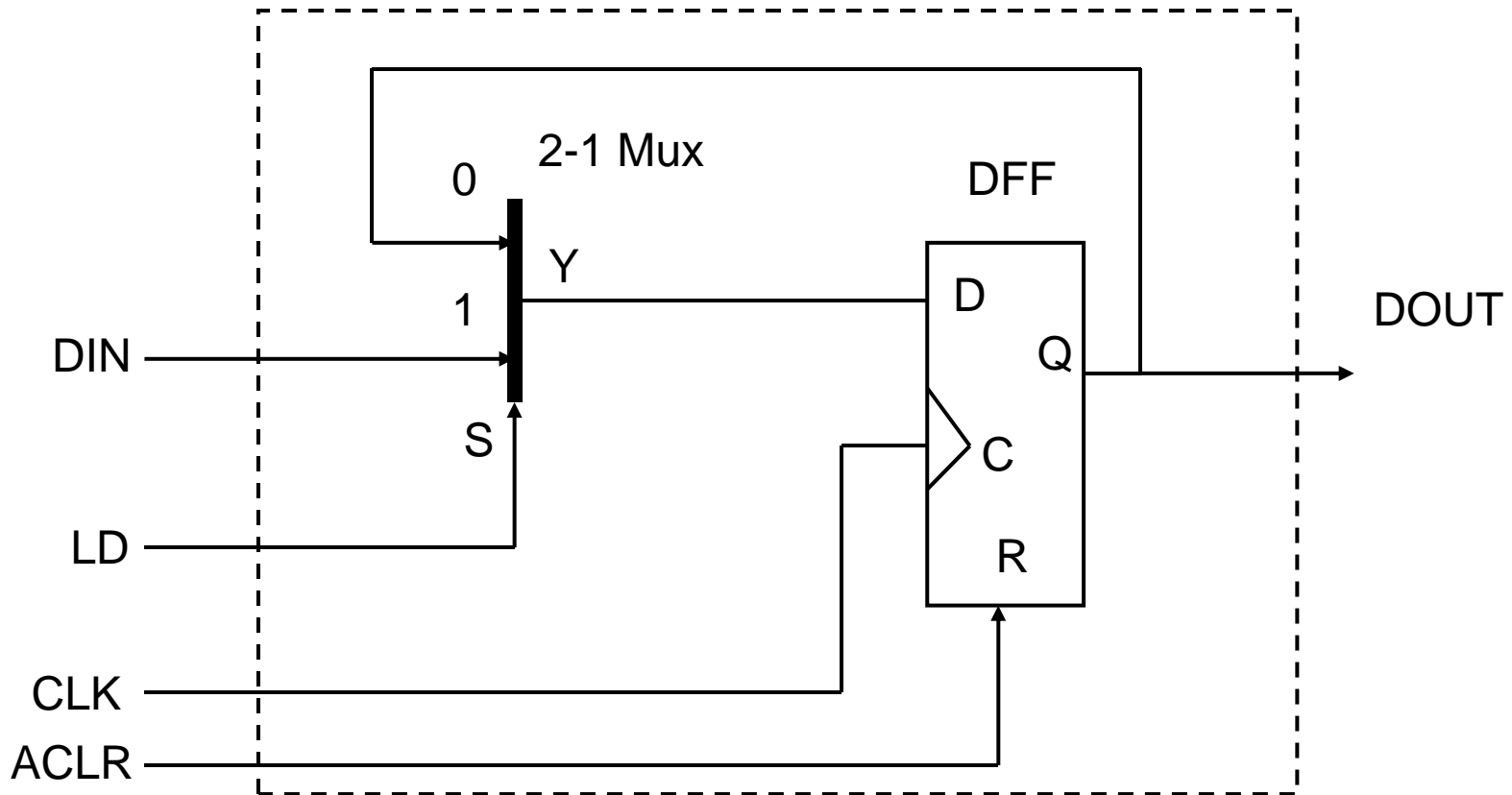
Registers

- The most common sequential building block is the register. A register is N bits wide, and has a load line for loading in a new value into the register.



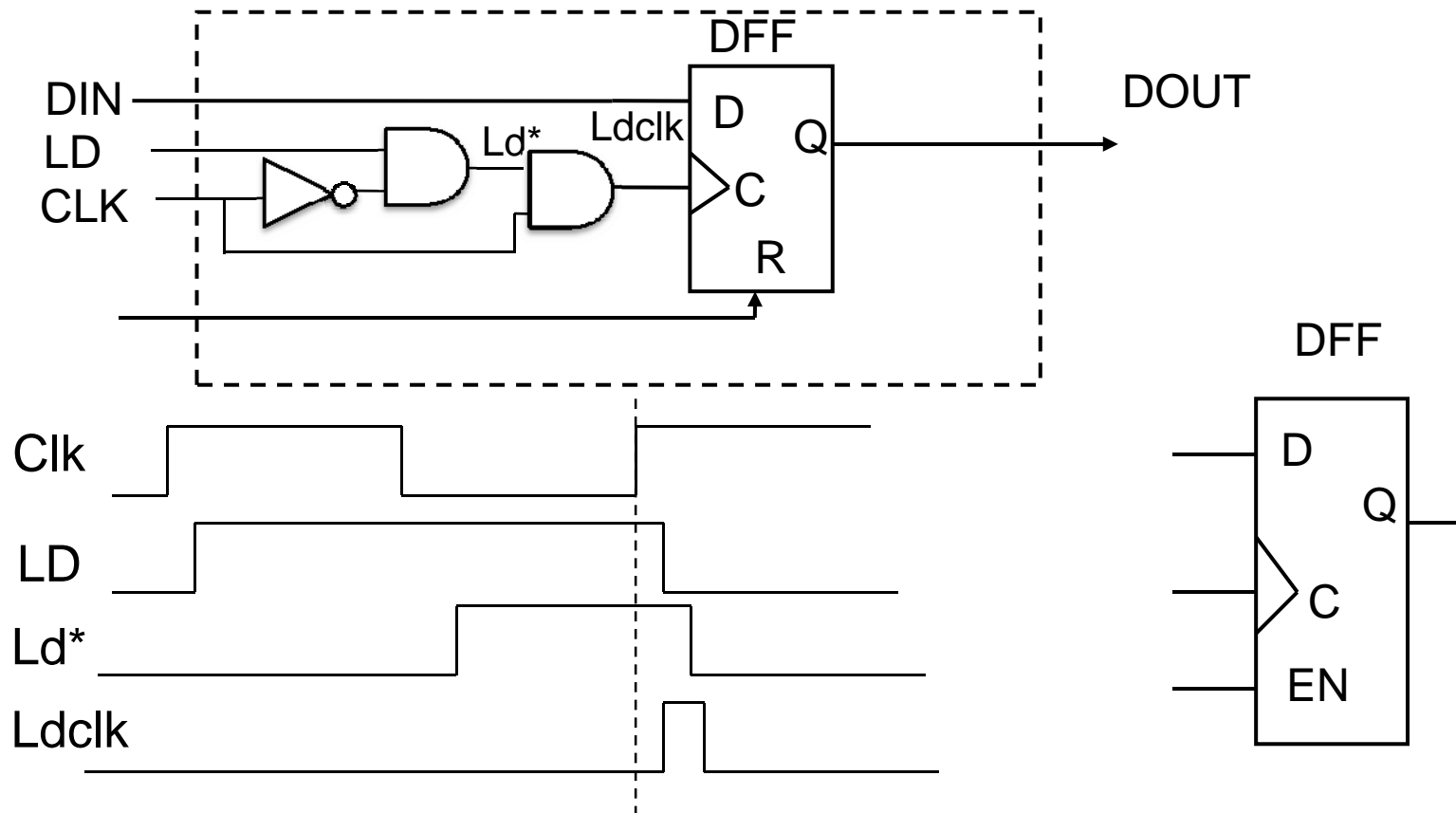
- Register contents do not change unless $LD = 1$ on active edge of clock.
- A DFF is NOT a register! DFF contents change every clock edge.
- ACLR used to asynchronously clear the register

1 Bit Register using DFF, Mux



Note that DFF simply loads old value when $LD = 0$. DFF is loaded every clock cycle.

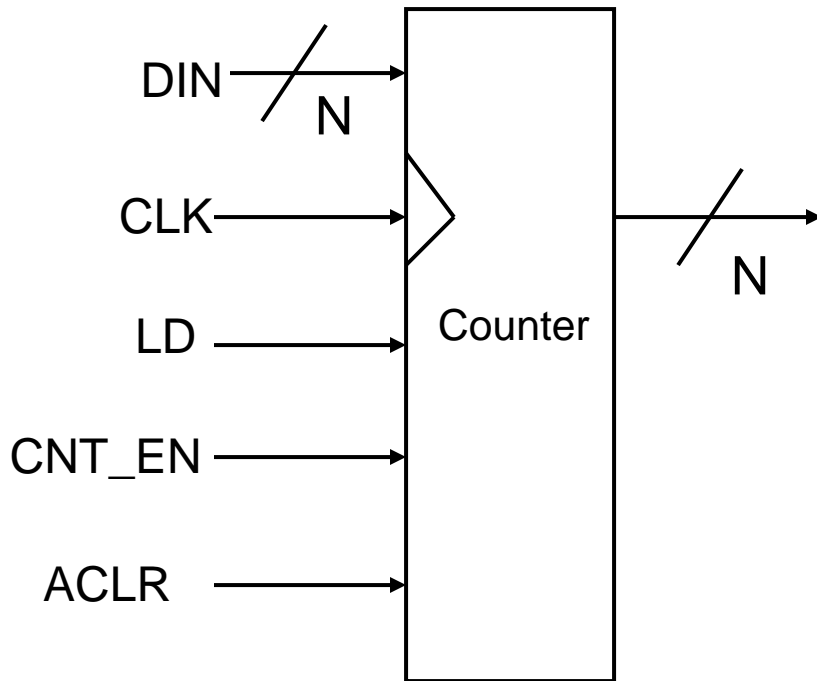
1 Bit Register using Gated Clock



Saves power over previous design since DFF is not clocked every clock cycle. Many FPGAs offer an 'enabled' DFF as an integrated unit. Gating can be optimized at transistor level in 'enabled' DFF.

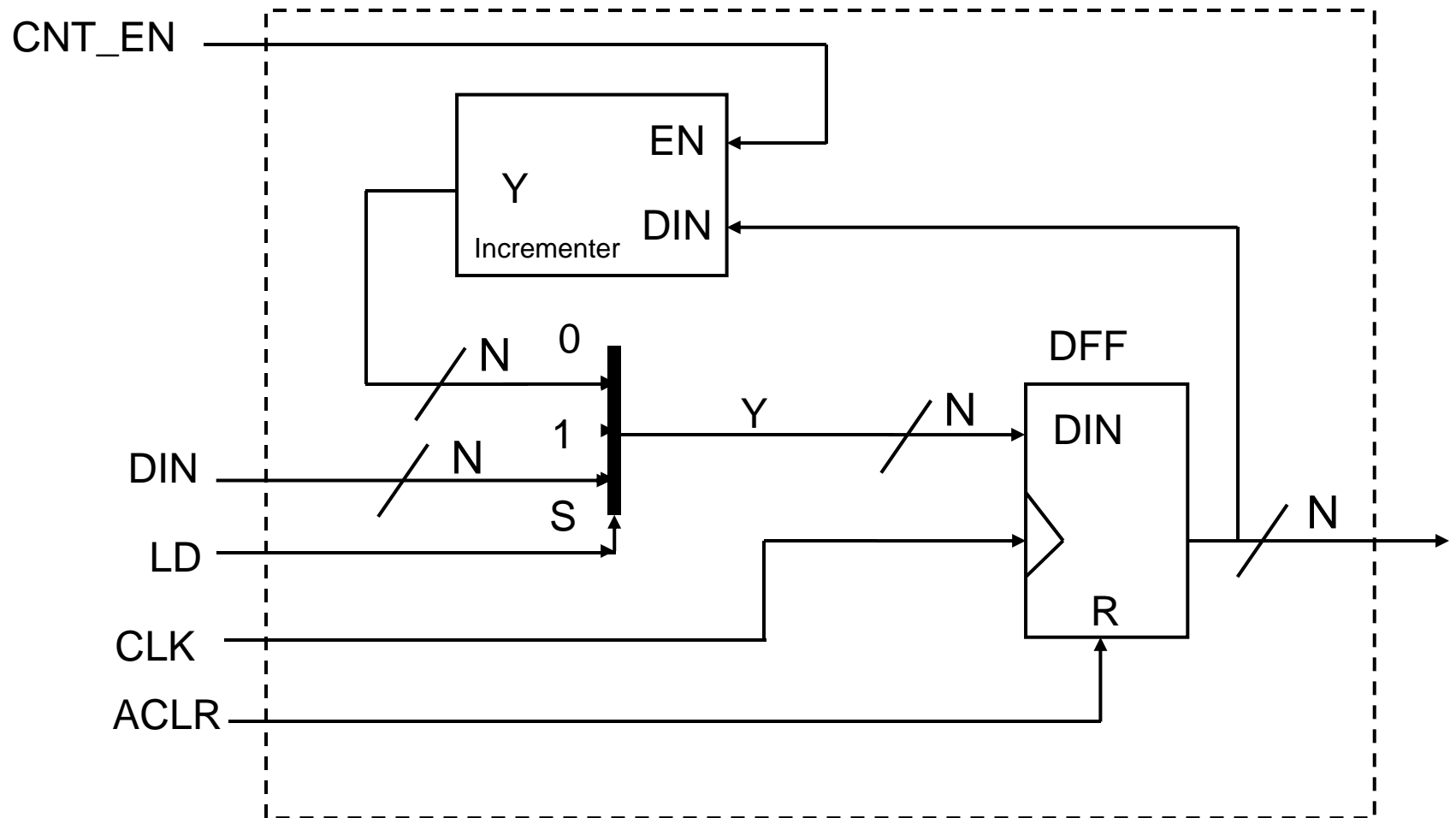
Counters

- Very common sequential building block. Used to generate memory addresses, or keep track of the number of times a datapath operation is performed.

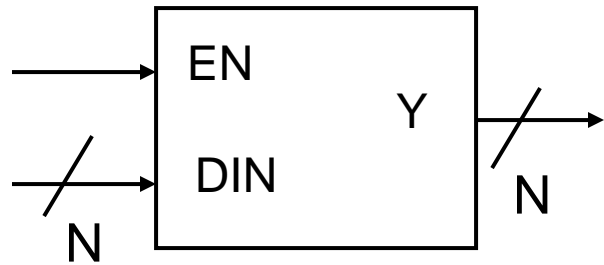


- LD asserted loads counter with DIN value.
- CNT_EN asserted will increment counter on next active clock edge.
- ACLR will asynchronously clear the counter.

One way to build a Counter “Counter A”

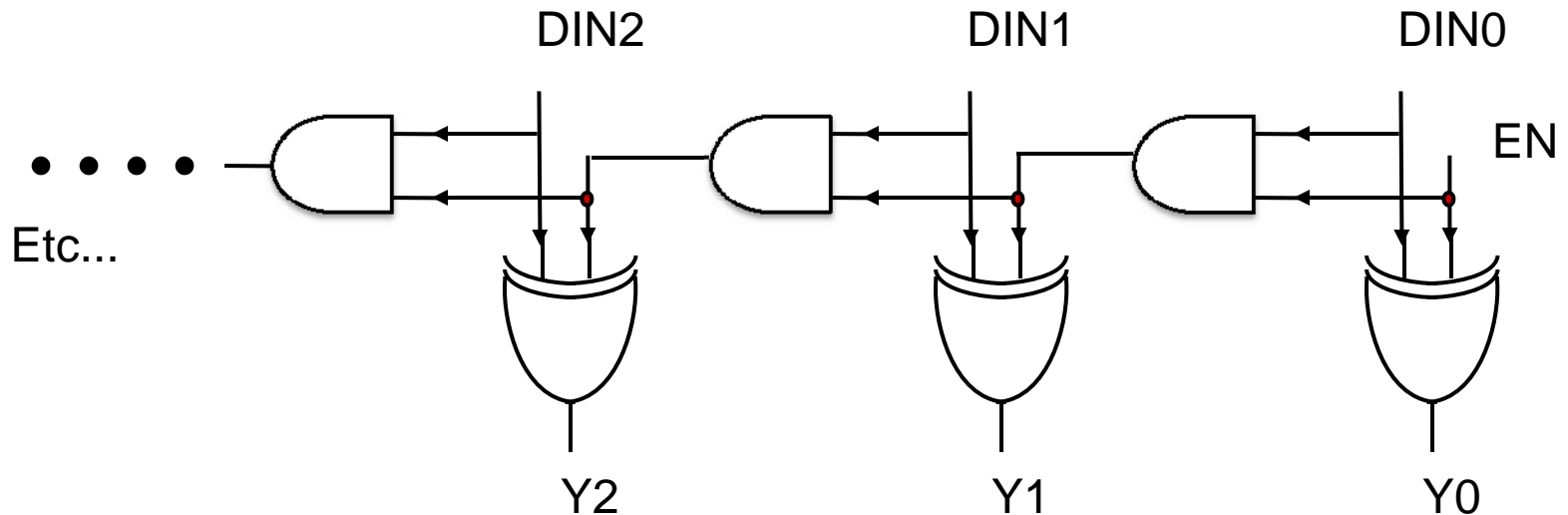


Incrementer: Combinational Building Block



When $EN=1$, $Y = DIN + 1$

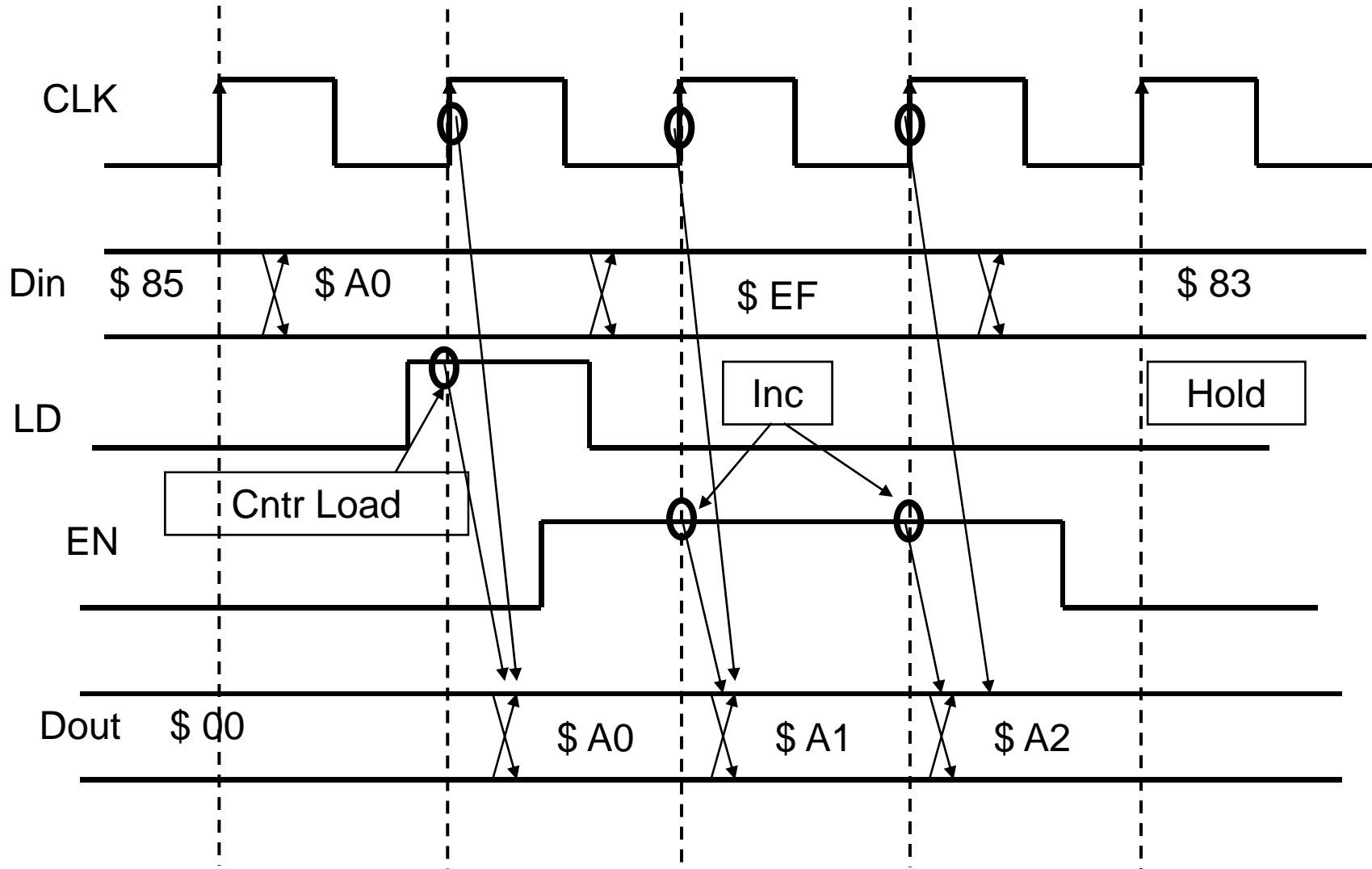
When $EN=0$, $Y = DIN$



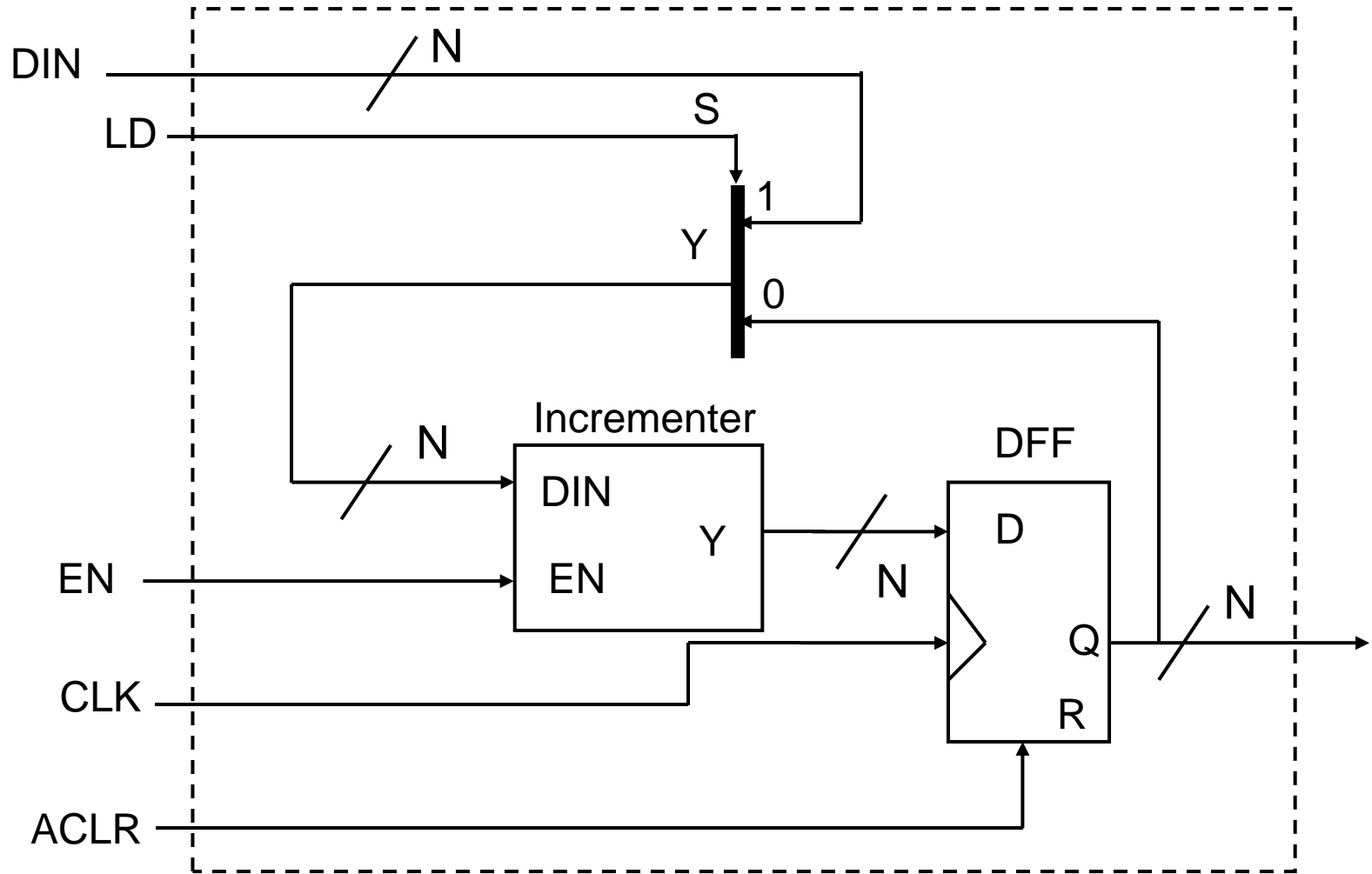
Counter Operation

Counter A							
Aclr	Clk	En	LD	Q	Q+	Op	
H	X	X	X	X	0	Async CLR	
L	↑	X	H	X	Din	Load	
L	↑	H	L	Q	Q+1	Increment	
L	X	L	L	Q	Q	Hold	

Counter Timing (8 Bit register)



Another Counter (Counter 'B')



Counter Operation

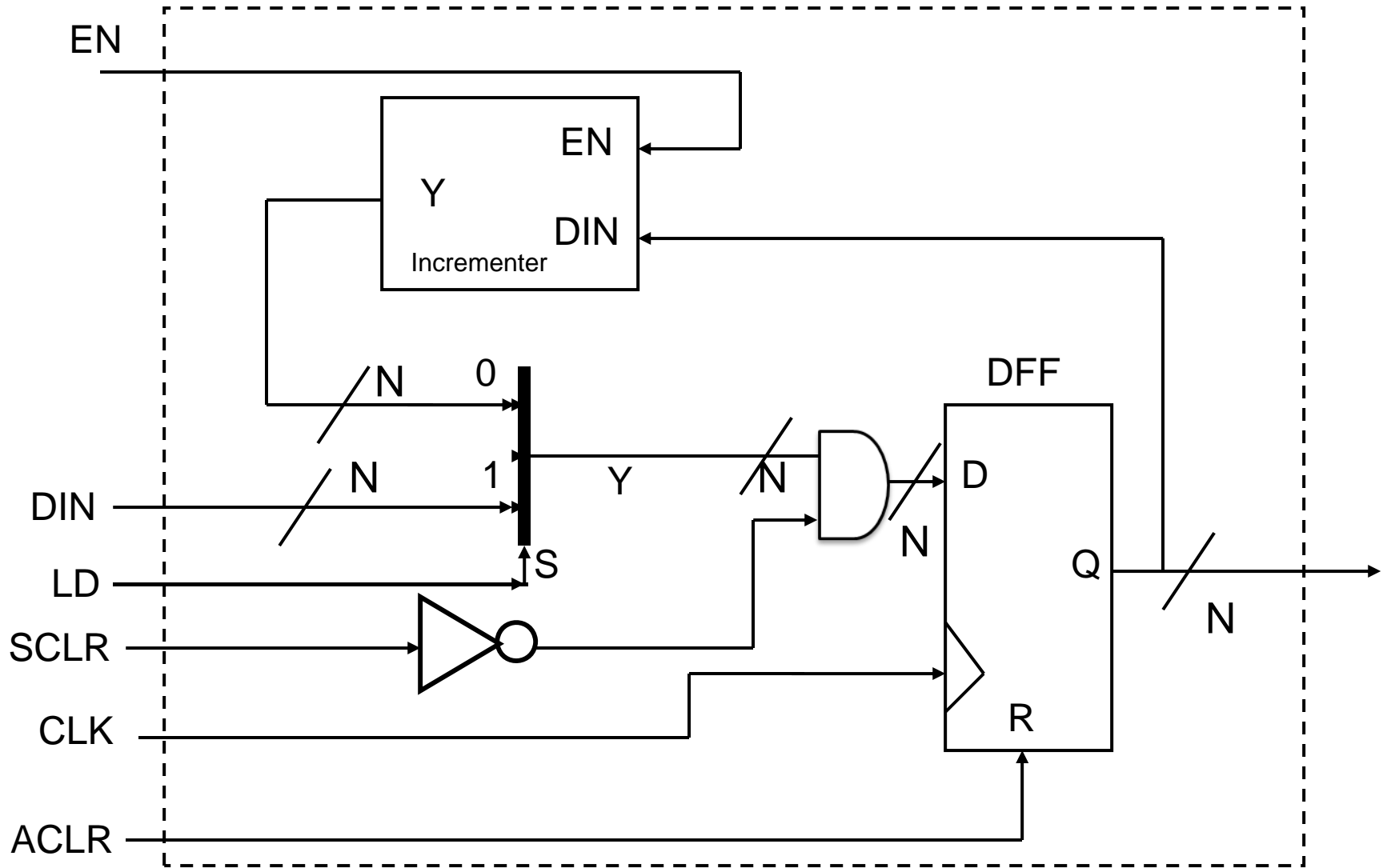
Counter B						
Aclr	CLK	En	LD	Q	Q+	Op
H	X	X	X	X	0	Async CLR
L	↑	L	H	X	Din	Load
L	↑	H	L	Q	Q+1	Increment
L	X	L	L	Q	Q	Hold
L	↑	H	H	Q	Din+1	Load Inc

EN=H, LD=H will load an incremented version of Din

Synchronous vs. Asynchronous Clear

- The ACLR line is tied to the asynchronous reset of the DFF
 - Asynchronous clear is independent of clock, will occur anytime clear is asserted
 - Usually tied to Power-On-Reset (POR) circuit
 - Not very useful for normal operation since any glitch on ACLR will clear the counter
 - Would like a Synchronous Clear input (SCLR) in which the clear operation takes place on the next active clock edge.
-

Counter 'A' with SCLR Input

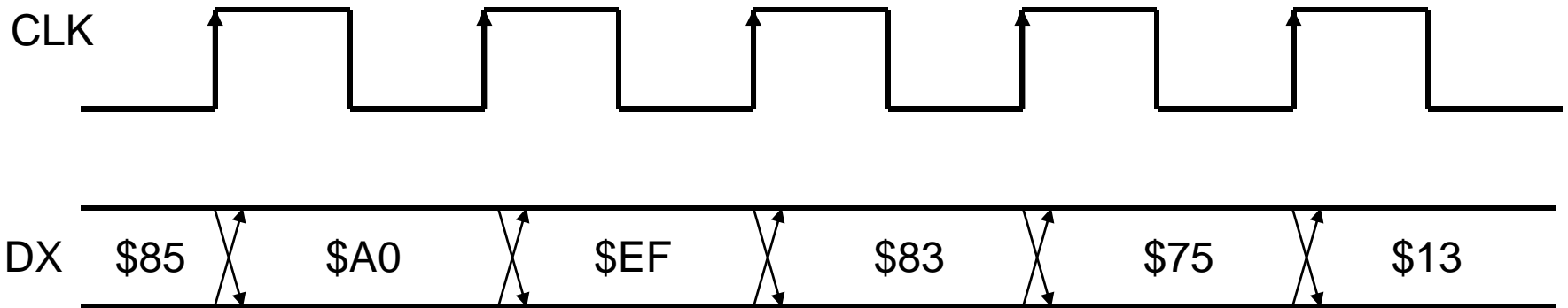
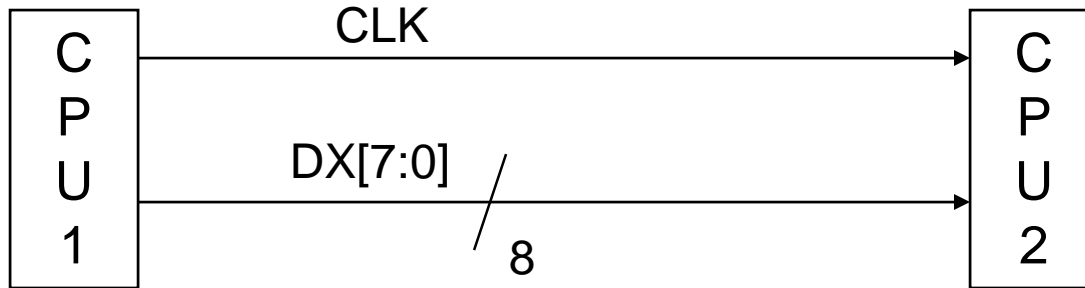


Counter Operation

Counter A with SCLR							
Aclr	Sclr	Clk	En	LD	Q	Q+	Op
H	X	X	X	X	X	0	Async Clr
L	H	↑	X	X	X	0	Sync Clr
L	L	↑	X	H	X	Din	Load
L	L	↑	H	L	Q	Q+1	Increment
L	L	X	L	L	Q	Q	Hold

Parallel Data Transfer

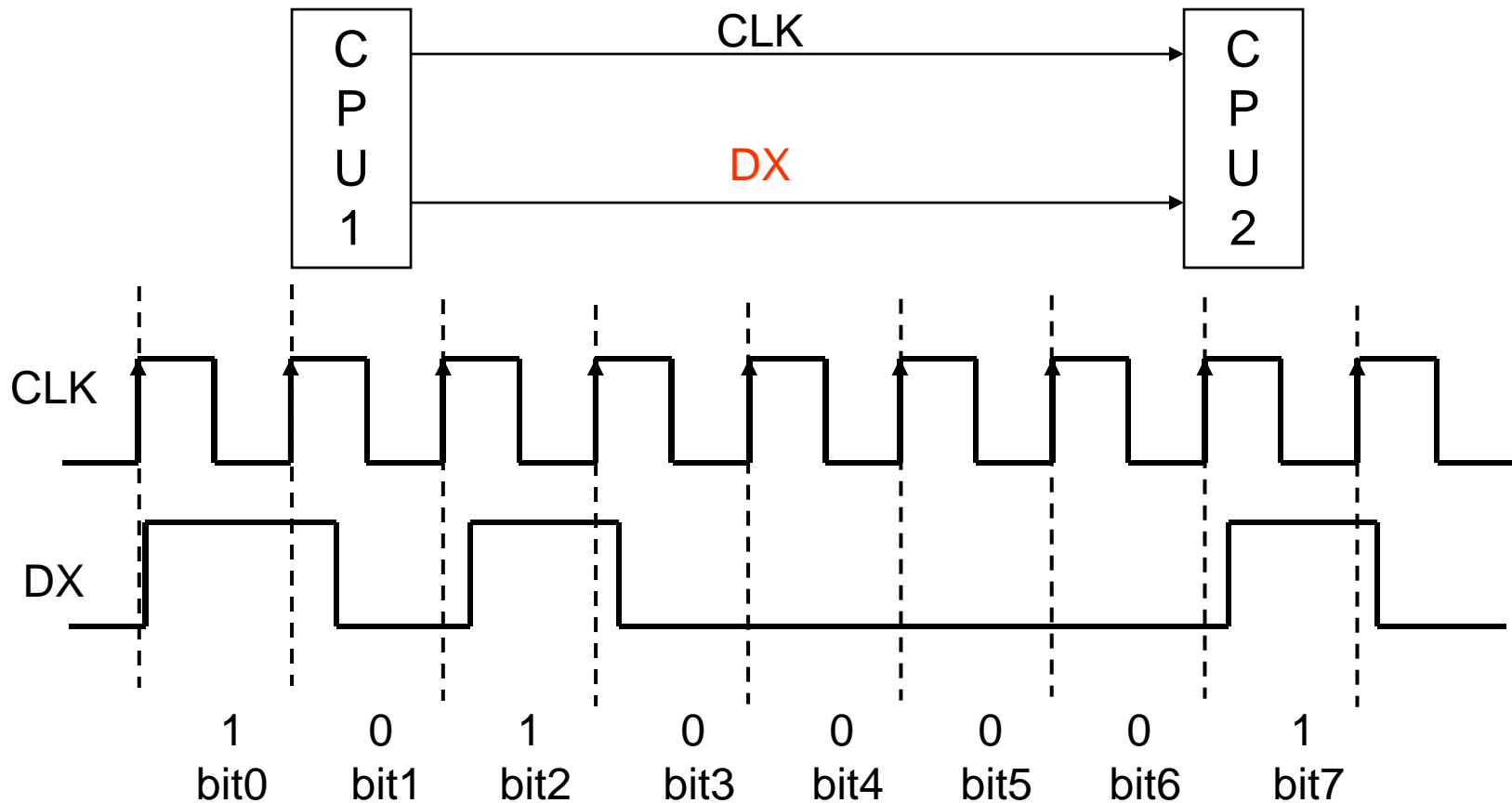
To transfer data between two computers, we can do it in parallel:



Parallel Data transfer requires a lot of lines to be run between computers; cabling be expensive, and bulky. Not practical for long distances.

Serial Data Transfer

We can transfer data in **serial** fashion, e.g., one bit at a time.



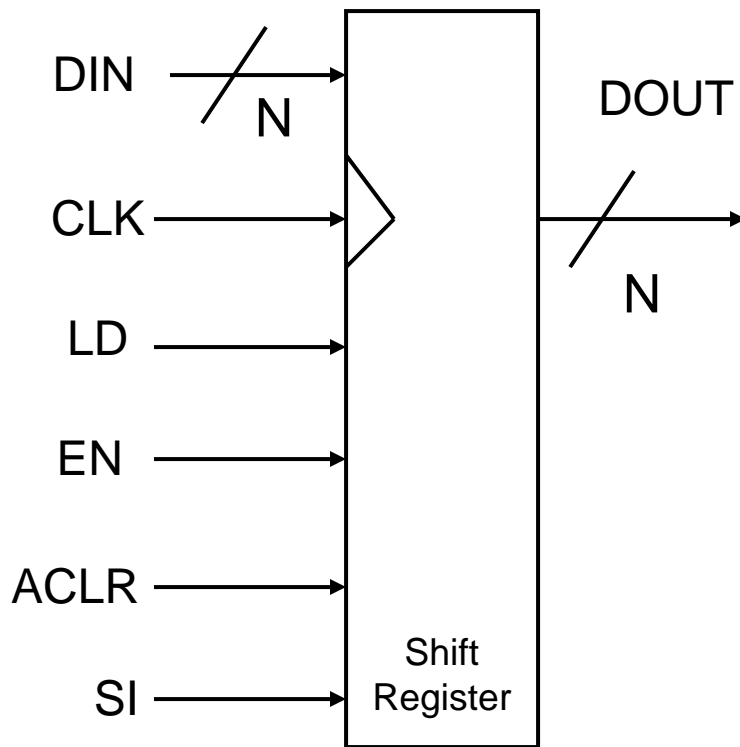
\$85 = 10000101, data transmitted LSB to MSB

More on Serial Data Transfer?

- Serial data transfer is more common than data parallel communication because less wires than parallel data transfer, can be run longer distances
 - Data can be transferred either LSB (least significant bit) to MSB (most significant bit) or vice-versa
 - Most common is LSB to MSB
 - To implement serial data transfer we need a sequential building block that is called a **SHIFT register**.
-

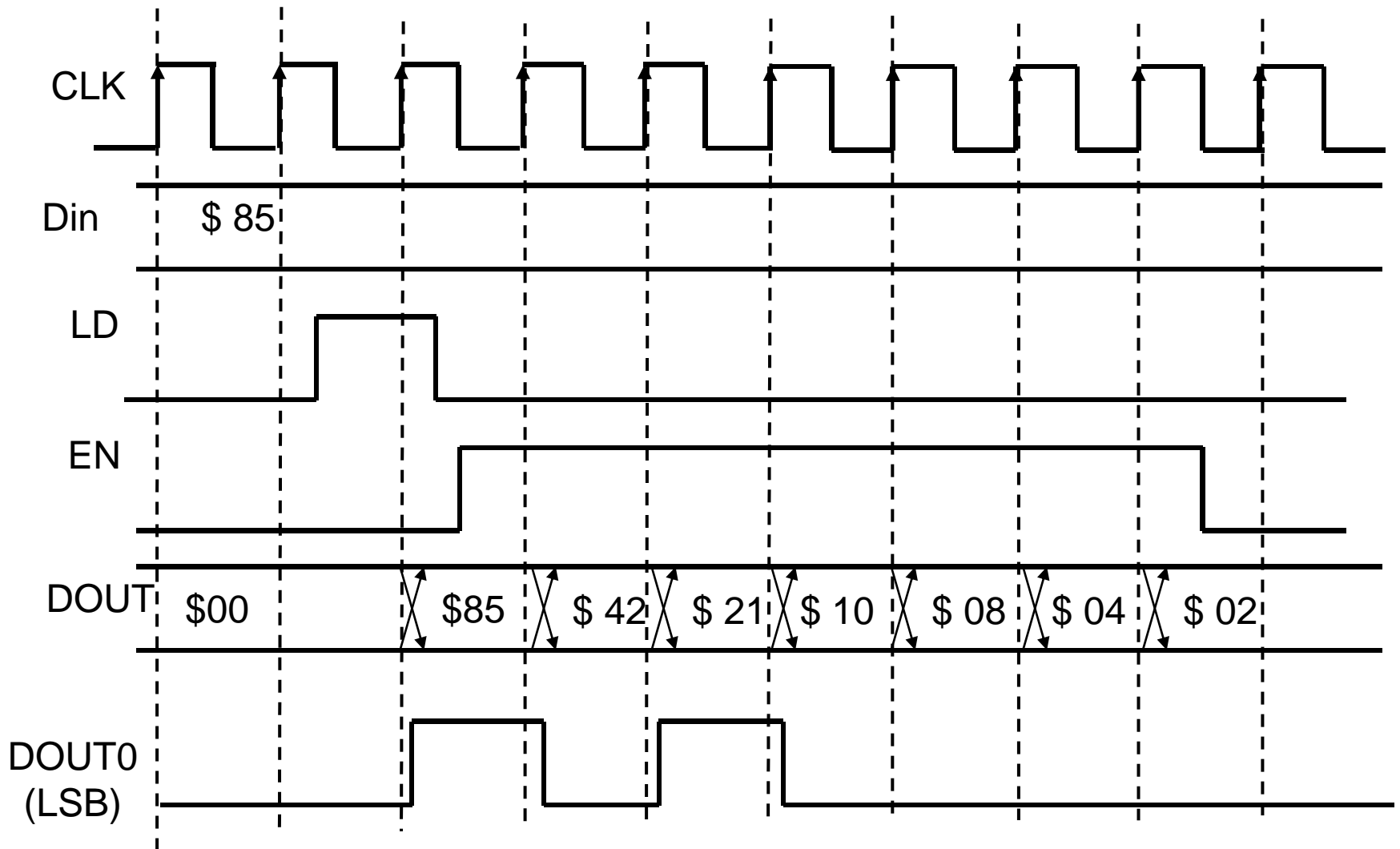
Shift Registers

- Very useful sequential building block. Used to perform either parallel to serial data conversion or serial to parallel data conversion



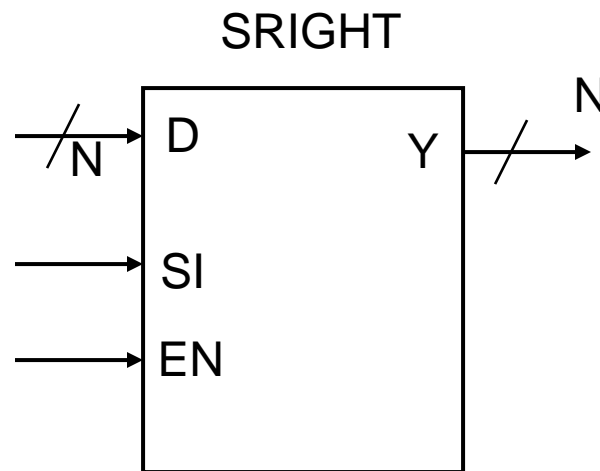
- LD asserted loads register with DIN value
- EN asserted will shift data on next active clock edge
- ACLR is async clear
- SI is serial data in
- Look at LSB of DOUT for serial data out

Shift Register Timing (SI = 0)



Combinational Right Shifter

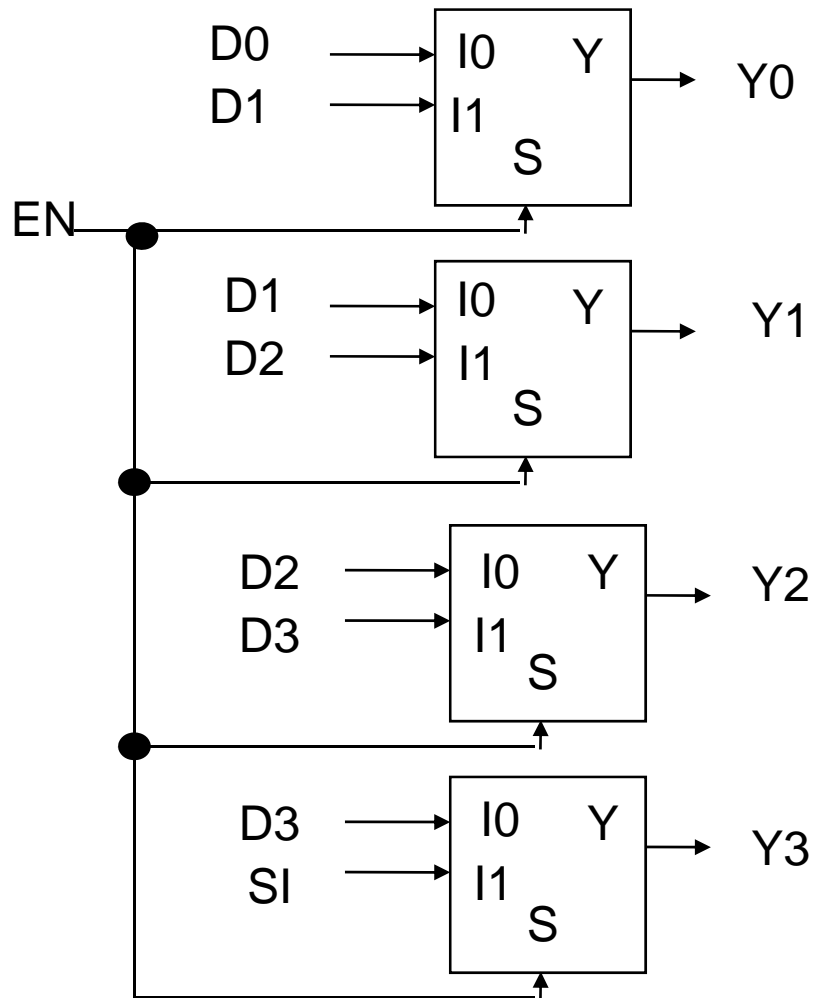
We need a combinational block that can either shift right or pass data unchanged



When $EN = 1$, $Y = D$ shifted right by 1 position.

When $EN=0$, $Y = D$

Combinational Right Shifter – 4 bit



4-bit Combinational RIGHT Shifter Implementation

When $EN = 0$, then:

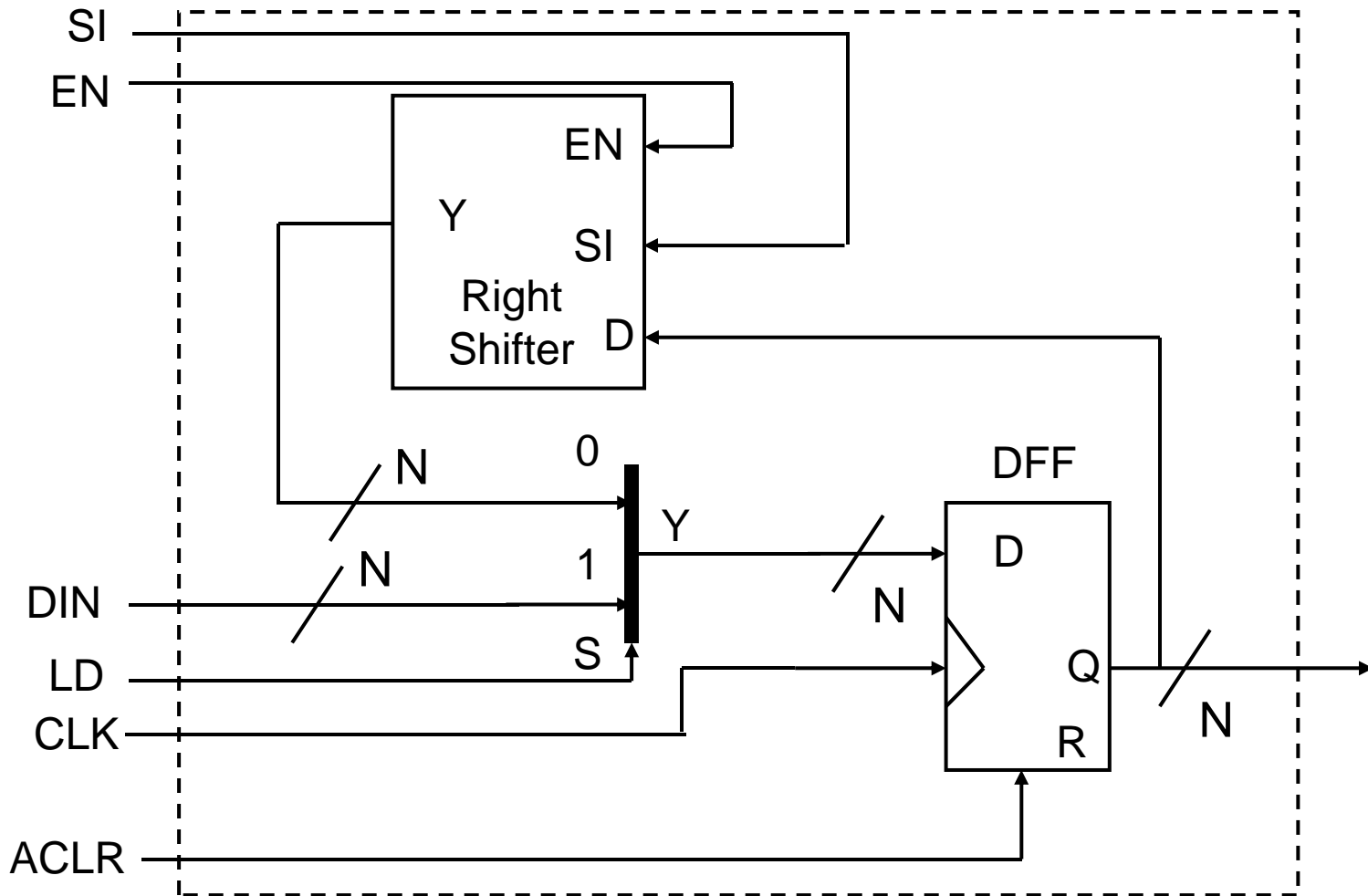
$$Y = D3 D2 D1 D0$$

When $EN = 1$, then:

$$Y = SI D3 D2 D1$$

(right shifted by one position)

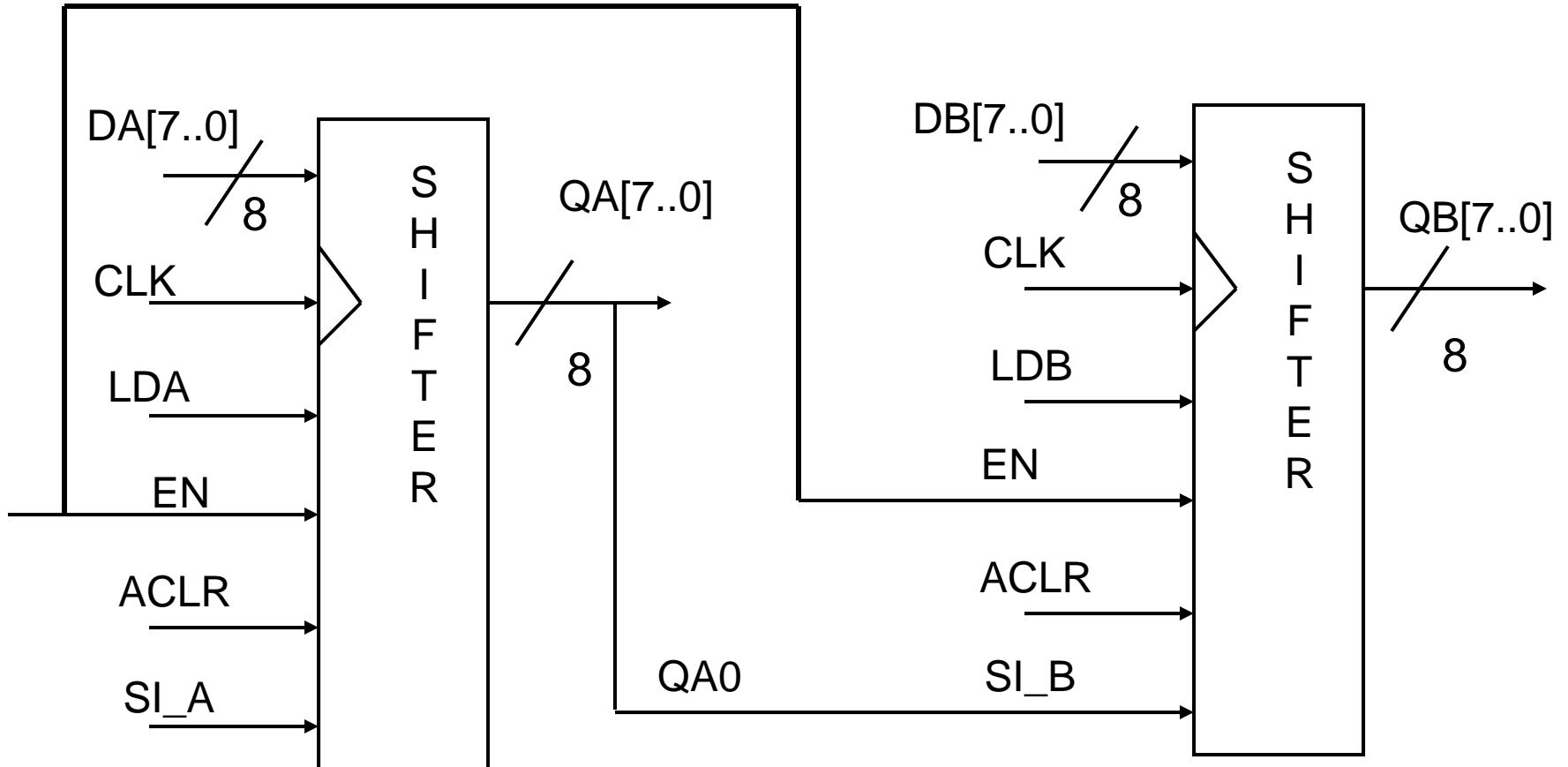
Shift Register (Right shift) Implementation



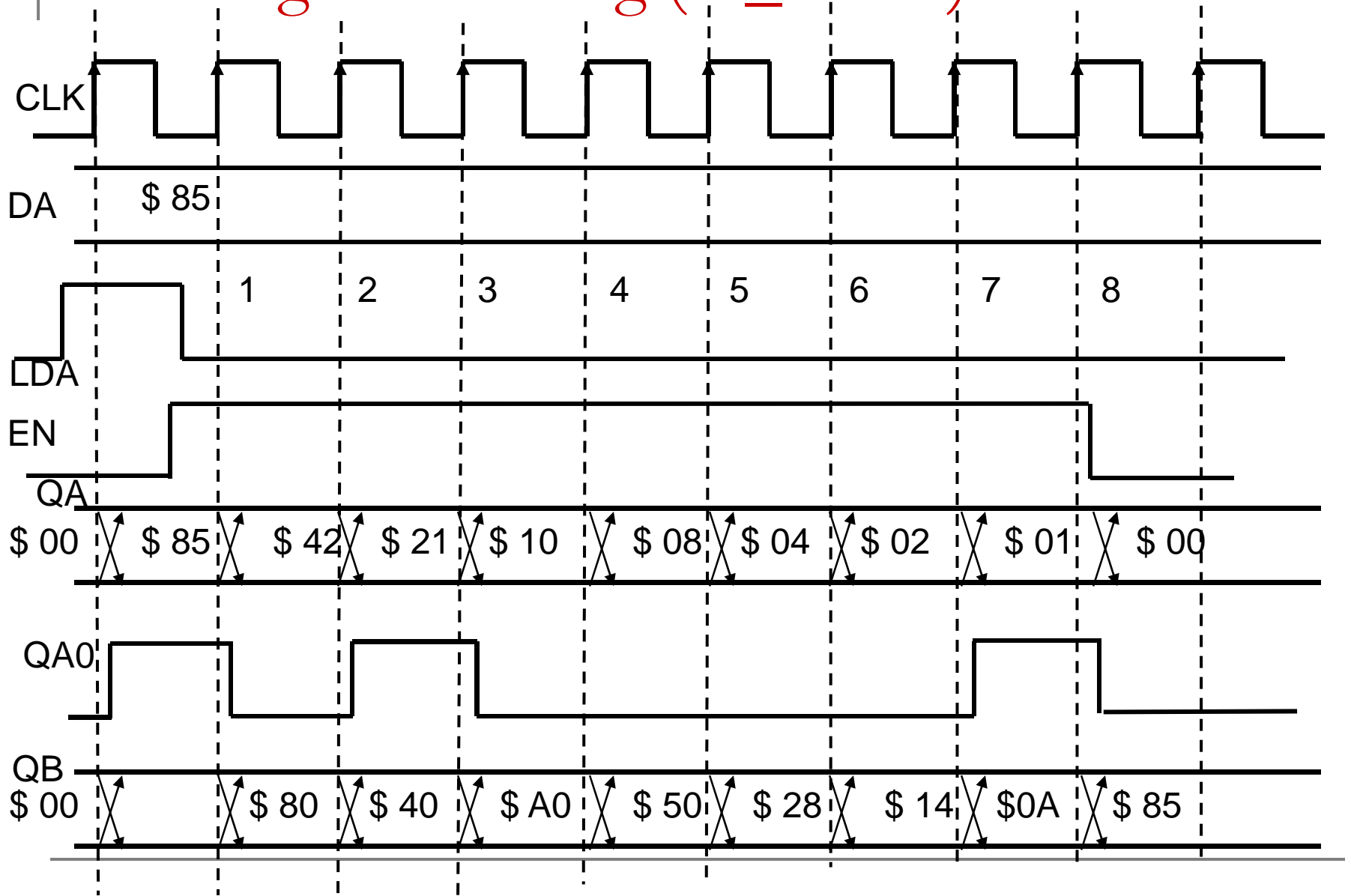
Serial Communication

CPU A

CPU B



Shift Register Timing (SI_A = 0)



Comments on Shift operation

- Took 8 clock cycles to serially send the 8 bits in CPU A to CPU B.
 - Shift Register at CPU A ended up at \$00; Shift Register at CPU B ended up with CPU A value (\$85)
 - Initial contents of CPU B shift register does not matter
 - Data shifted out LSB to MSB from CPUA to CPUB. Note that data enters the MSB at CPUB and progresses toward the LSB.
-

Sequential System Description

- The Q outputs of the flip-flops form a state vector
 - A particular set of outputs is the **Current State (CS)**
 - The state vector that occurs at the next discrete time (clock edge for synchronous designs) is the **Next State (NS)**
 - A sequential circuit described in terms of state is a Finite State Machine (FSM)
 - Not all sequential circuits are described this way; i.e., registers are not described typically as FSMs yet a register is a sequential circuit.
-